A New Class of B/W Halftoning Algorithms

Avideh Zakhor, *Member, IEEEE*, Steve Lin, and Farokh Eskafi

Abstract— **We present a new class of dithering algorithms for black and white (b/w) images. The basic idea behind our technique is to divide the image into small blocks and minimize the distortion between the original continuous-tone image and its low-pass-filtered halftone. This corresponds to a quadratic programming problem with linear constraints which is solved via standard optimization techniques. Examples of b/w halftone images using our technique are compared to halftones obtained via existing dithering algorithms.**

I. Introduction

DIGITAL halftoning schemes are important to rendition of continuous-tone images on binary output devices such as displays, workstations, and laser printers. While the pixels in a continuous-tone image take on a continuum of gray levels all the way from black to white, those of binary images are either black or white. If the binary pixels are spaced closely enough, the low-pass filtering of the human visual system results in an illusion of a continuous-tone image. Thus, digital halftoning is a prime example of multidimensional signal representation in which the amplitude resolution of the signal is traded off with its spatial resolution. Other examples of such tradeoffs are shown in [1], [2].

Existing digital halftoning algorithms include globally fixed level thresholding, locally adaptive thresholding, orthographic tone scale creation, clustered or dispersed ordered dithering, white noise dithering, and error diffusion [4], [5]. The metrics used to compare these algorithms are low and high frequency rendition, processing artifacts, and processing complexity. Ordered dither, which is among the most popular of the above halftoning techniques, consists of thresholding samples of the continuous-tone image with a periodic screen, or dither matrix. The optimization of dither matrices has been studied by a number of authors [5]–[7]. The goal in such optimizations is to choose the number and order of the thresholds in the dither matrix in such a way that the resulting dot profile for DC inputs minimizes the amount of distortion energy falling into the viewer's passband. To this end, the low-order Fourier coefficients due to thresholding DC inputs are minimized. A major drawback of this technique, for square screen functions on a rectangularly sampled lattice, is that the fundamental spectral components shift back and forth among the low-order horizontal and vertical coefficients [6]. Specifically, the lowest-order coefficients alternate between zero and nonzero values for adjacent gray level values, resulting in undesirable contour artifacts. Rao *et al.* propose a way of overcoming this problem by changing the orientation and periodicity of the screen function and the sampling lattice [6], [7].

In this paper, we develop another class of digital halftoning algorithms based on binary optimization techniques. Our approach is similar to that of [9], [8] in that it is a least-squares model based technique. Its main difference from [8] and [9] however is that it divides a large optimization problem into a series of smaller ones. The outline of the paper is as follows: Section II includes our basic algorithm for b/w halftoning based on space domain optimization. Examples of this algorithm are included in Section II-B. A variation of our basic algorithm known as the "neighboring" algorithm is included in Section II-C. Section III includes an algorithm based on frequency domain optimization. Section IV includes comparison with existing halftoning techniques. Section V includes a brief discussion of computational complexity, and finally Section VI includes conclusions.

II. Space Domain Optimization

We describe our basic algorithm in Section II-A, present examples of halftone images resulting from our algorithm in Section II-B, and describe the neighboring algorithm in Section II-C.

A. The Algorithm

The most straightforward formulation of the digital halftoning problem can be stated in the following way: Find a binary image such that the distortion between the continuous-tone picture and its perceived bilevel image is minimized [2]. Since the human eye can be modeled as a low-pass filter [10], an acceptable distortion function to minimize is the mean squared error (MSE) between the continuous-tone image and the low-pass version of its halftone. Specifically, if $c(x,y)$ denotes a continuous-tone, continuous-space image and $f(x,y)$ is a bilevel continuous-space signal, and $h(x,y)$ is the low-pass filter model for the human visual characteristics, then the above mentioned MSE can be expressed as

$$MSE = \int_D [f * h - c]^2 dx dy \tag{1}$$

where D denotes the domain of interest, and $*$ denotes convolution. The bilevel continuous space signal $f(x,y)$ is assumed to be piecewise constant and is given by the following expression:

$$f(x,y) = \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{M-1} b(n_1,n_2) rect(x - n_1\Delta, y - n_2\Delta) \tag{2}$$

Manuscript received February 29, 1992; revised March 27, 1993. This work was supported by National Science Foundation PYI award MIP–9 057 466, ONR Young Investigator award N00 014-92-J-1732, Digital Equipment Corporation, and Sun Microsystems. The associate editor coordinating the review of this paper and approving it for publication was Dr. F. Mintzer.

The authors are with the Department of Electrical Engineering and Computer Sciences, University of California, Berkeley, CA 94720.

IEEE Log Number 9210836.

where $M \times M$ is the number of the pixels of the halftone signal, Δ denotes the pixel width, $b(n_1, n_2)$ denotes the binary variable corresponding to the (n_1, n_2)th halftone pixel, and the function $rect(x, y)$ is a rectangular pulse defined by

$$rect(x,y) = \begin{cases} 1 & |x| \le \frac{\Delta}{2} \text{ and } |y| \le \frac{\Delta}{2} \\ 0 & |x| \ge \frac{\Delta}{2} \text{ and } |y| \ge \frac{\Delta}{2}. \end{cases} \quad (3)$$

The above optimization problem corresponds to a quadratic programming one with M^2 variables, and in principle can be solved via a variety of binary optimization techniques. However, for typical images used in practice, the number of variables becomes too large for the computations to remain tractable [2]. On the other hand, since $h(x, y)$ corresponds to the impulse response of a low-pass filter with a finite region of support, we can divide the problem into a large number of smaller-sized problems. This is because the low-pass-filtered version of f at a particular location $(x\prime, y\prime)$ primarily depends on values of f in the immediate neighborhood of $(x\prime, y\prime)$. Specifically, if h is a low-pass filter with bandwidth of W cycles per centimeter, then its impulse response will be approximately $2/W$ centimeters wide. Therefore minimization of MSE shown in (1) over a small area depends on values of c and f at locations which are at most $2/W$ centimeters away from that area. Thus, breaking up the continuous-tone image into small blocks and solving the optimum for each block based on its immediate neighborhood will result in near optimal halftone images.

We further assume that in optimizing the N^2 binary pixels of an $N \times N$ halftone block, the contributions of the pixels surrounding the $N \times N$ block can be discarded altogether. This is clearly an approximation and we will remove it later in Section II-C. Specifically, in Section II-C, we introduce a variation of our algorithm, called the "neighboring" technique which takes the neighboring pixels of an $N \times N$ block into account while optimizing the N^2 binary pixel variables inside the block.

Based on the above discussion, we divide the continuous-tone, continuous-space image into small areas of size $N\Delta \times N\Delta$ and optimize the $N \times N$ halftone pixels of the block centered at location (x_0, y_0), as shown in Fig. 1, by minimizing

$$MSE_{(x_0,y_0)} = \int_{x_0-N\Delta/2}^{x_0+N\Delta/2} \int_{y_0-N\Delta/2}^{y_0+N\Delta/2} [f_N * h - c]^2 dxdy \quad (4)$$

where $f_N(x, y)$ is defined to be

$$f_N(x,y) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} b(n_1,n_2) rect(x - n_1\Delta, y - n_2\Delta) \quad (5)$$

The MSE criterion of (4) can be rewritten as [3]

$$MSE_{(x_0,y_0)} = \int_{x_0-N\Delta/2}^{x_0+N\Delta/2} \int_{y_0-N\Delta/2}^{y_0+N\Delta/2} [(f_N * h)^2 - 2c(f_N * h) + c^2] dxdy \quad (6)$$

Defining $g_r(x, y)$ to be the response of the optical system to the rectangular signal $rect(x, y)$ of (3), the convolution $f_N * h$ can be written as

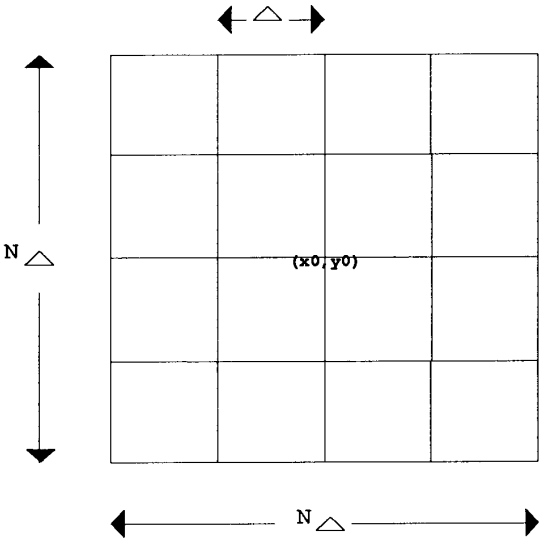


Fig. 1. Pictorial representation of N^2 pixels centered around (x_0, y_0) in an area of $N\Delta \times N\Delta$.

$$(f_N * h)(x,y) = \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{N-1} b(n_1,n_2) g_r(x - n_1\Delta, y - n_2\Delta) \quad (7)$$

Combining (6) and (7), we get

$$MSE_{(x_0,y_0)} = \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} \sum_{l=0}^{N-1} \sum_{m=0}^{N-1} G_{ijlm} b(i,j) b(l,m) + \sum_{i=0}^{N-1} \sum_{j=0}^{N-1} R_{ij} b(i,j) + Q \quad (8)$$

where

$$G_{ijlm} = \int_{D(x_0,y_0)} g_r(x - i\Delta, y - j\Delta) g_r(x - l\Delta, y - m\Delta) dxdy \quad (9)$$

$$R_{ij} = \int_{D(x_0,y_0)} -2c(x,y) g_r(x - i\Delta, y - j\Delta) dxdy \quad (10)$$

$$Q = \int_{D(x_0,y_0)} [c(x,y)]^2 dxdy \quad (11)$$

and $D(x_0, y_0)$ is an area of size $N\Delta \times N\Delta$ centered around (x_0, y_0) as shown in Fig. 1.

Optimization of $MSE_{(x_0,y_0)}$ is a quadratic programming (QP) problem with N^2 variables and, as such, can be solved via a number of binary optimization techniques such as brute force exhaustive search, or the well known branch-and-bound (BB) algorithm. The particular choice of the algorithm is highly dependent on the size of the optimization problem.

Fig. 2. Original 512 × 512 continuous-tone Lena image

B. Experimental Results

We have found numerically that for small values of N, the best choice of optimization algorithm is exhaustive search, while for large values, the branch-and-bound algorithm is more computation efficient. In using the BB algorithm, we have the option of directly applying it to the QP problem at hand, or converting it to an equivalent linear programming (LP) problem with more unknown variables and linear constraints. In our specific problem, it can be shown that the QP problem with N variables can be converted into a LP problem with $\frac{N^2(N^2+1)}{2}$ variables and $N^2(N^2-1)$ linear constraints [11], [12].

To keep the computation tractable, we have chosen $N = 4$, together with exhaustive search algorithm for all the experimental results in this paper. In doing so, we have found that for $N = 4$, the exhaustive search results in faster computation speed as compared to the branch-and-bound algorithm. The particular continuous-tone image we have chosen for our experiments is the 512×512 Lena shown in Fig. 2. The halftone version of Lena using the low-pass triangular filter of the form

$$H_{tri}(\omega_x, \omega_y) = \hat{H}(\omega_x)\hat{H}(\omega_y) \tag{12}$$

where

$$\hat{H}(\omega) = \begin{cases} A[1 - \frac{|\omega|}{2\omega_0}] & |\omega| \leq 2\omega_0 \\ 0 & \text{elsewhere} \end{cases} \tag{13}$$

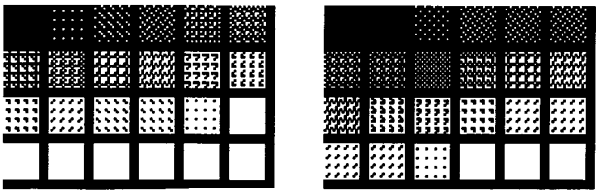
with $A = 1$ and $\omega_0 = 15$ cycles per centimeter is shown in Fig. 3(a). Note that ω, ω_x, and ω_y in (12) and (13) are all in units of cycle per centimeter, and as a result the bandwidth of $H(\omega_x, \omega_y)$ in the two dimensional frequency domain is 30 cycles per centimeter in both x and y directions. Assuming viewing distance of one meter, this is in approximate agreement with the bandwidth of the human eye's modulation transfer function (MTF) as a function of angular frequency in units of cycles per degree [10], [13]. The images in this paper were designed for viewing distance of approximately one meter and resolution of 150 dots per inch. The particular printer we use is Apple laser writer with capability of printing up to 300 dots/inch.

(a)

(b)

Fig. 3. Space domain optimization using the triangular filter of (11) with (a) A= 1. (b) A = 1.1.

As seen in Fig. 3(a), there are large white areas in the forehead, hat, and shoulder indicating that the DC value of the original continuous-tone image has not been preserved. Inserting (13) into (4), it might seem that minimizing the $MSE_{(x_0,y_0)}$ in (4) results in matching the DC values of $f_N * h$ to the DC value of the original continuous space signal $c(x, y)$. In fact, this intuition is only approximately correct because in general minimizing $\int e^2(x)dx$ does not necessarily result in forcing the DC value of $e(x)$ i.e., $\int e(x)dx$ to be zero. Since our optimization problem involves minimizing the integral of $(f_N * h - c)^2$ rather than the integral of $(f_N * h - c)$, we are not explicitly forcing the DC value of $f_N * h$ to match that of c.



(a) (b)

Fig. 4. DC gray bar associated with the filter of 11 and 12 with (a) A= 1, (b) A = 1.1.

The white areas in the forehead and shoulder of Fig. 3(a) can also be explained by considering the "DC gray bar" of the filter in (13) with $A = 1$ shown in Fig. 4(a). The DC gray bar consists of halftone patterns corresponding to 24 equally spaced DC values. As seen, the DC values of the optimized halftone patterns do not exactly increase linearly with the dc value of the continuous space signal. In particular, seven left-most squares of the last row are white, indicating that the optimized pattern for dc values larger than 70% of the full scale have no black dots.

While minimizing $MSE_{(x_0,y_0)}$ in (4) does not assure DC preservation, changing A in (13) will affect the brightness of the final halftone image. Inspection of (4) indicates that increasing A will lower f_N, and $b(n_1, n_2)$ and therefore will make the image darker. Thus, to remove the white areas in the shoulder and forehead, we would have to increase A. As an example setting $A = 1.1$ in (13) results in the halftone image shown in Fig. 3(b). As seen, the white shoulder areas in Fig. 3(a) have disappeared in Fig. 3(b). The DC gray bar of the filter in (13) with $A = 1.1$ is shown in Fig. 4(b). As we expect, comparing the gray bar for $A = 1.1$ with that of $A = 1$, we find the former to be darker than the latter.

An alternative way to ensure that the DC value in each 4×4 block is preserved is to enforce it in the optimization processes as a constraint. This has the added advantage of speeding up the optimization process since it reduces the size of the search space. An example of adding DC constraint to the image of Fig. 3(b) is shown in Fig. 5. As seen, imposing the DC constraint reduces the contouring artifact in Fig. 3(b).

While the frequency response in (12) was primarily chosen for its simplicity and analytical tractability, we have found experimentally that it performs just as well as other filters that are known to model the characteristics of the human eye [10]. Optimal choice of the filter used in the optimization process remains an interesting topic for future work.

C. The Neighboring Algorithm

As it was mentioned earlier, one way to justify the breakup of the optimization problem into a large number of smaller $N \times N$ optimization problems is to assume that the contribution of the pixels surrounding the $N \times N$ block can be discarded altogether. Clearly as N becomes large, the approximation to the large optimization problem becomes more valid. Of course, the major drawback of choosing a large value for N is that the size of the optimization problem becomes so large that it might be intractable.

Fig. 5. Space domain optimization using the triangular filter of (11) and (12) with DC constraints for $A = 1.1$.

One way to circumvent the computational complexity of choosing a large value of N is to take into account the effect of the surrounding region of an $N \times N$ block while optimizing its N^2 binary halftone variables. This way the optimization is carried out over a larger area than $N\Delta \times N\Delta$ at the same time as keeping the number of variables to N^2. We will refer to this variation of our basic algorithm as the "neighboring" algorithm. The neighboring algorithm uses the already halftoned neighboring pixels in optimizing the current $N \times N$ block. Thus, if the optimization is carried out from left to right and top to bottom, then optimization of a typical $N \times N$ block, needs the halftone pattern of the block to its left, the block above it, and the block above and to the left of it. This way, the number of binary variables is still N^2, but the domain of interest is four times larger as before. Specifically, the quantity to be minimized is of the form

$$MSE_{(x_0,y_0)} = \int_{x_0-3N\Delta/2}^{x_0+N\Delta/2} \int_{y_0-3N\Delta/2}^{y_0+N\Delta/2} [f_{3N} * h - c]^2 dxdy \tag{14}$$

where $f_{3N}(x, y)$ is defined to be

$$\begin{aligned} f_{3N}(x,y) = &\sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} b_{unknown}(n_1,n_2) \\ &\cdot rect(x - n_1\Delta, y - n_2\Delta) \\ &+ \sum_{n_1=-N}^{-1}\sum_{n_2=0}^{N-1} b_{known}(n_1,n_2) \\ &\cdot rect(x - n_1\Delta, y - n_2\Delta) \\ &+ \sum_{n_1=0}^{N-1}\sum_{n_2=-N}^{-1} b_{known}(n_1,n_2) \\ &\cdot rect(x - n_1\Delta, y - n_2\Delta) \end{aligned}$$

(a) (b)

Fig. 6. Space domain optimization using the "neighboring" technique and the filter of (11) and (12) with $A = 1$ (a) square optimization blocks and (b) rhombus optimization blocks.

$$+\sum_{n_1=-N}^{-1}\sum_{n_2=-N}^{-1} b_{known}(n_1,n_2) \cdot rect(x-n_1\Delta, y-n_2\Delta). \quad (15)$$

In the above equation, $b_{known}(n_1,n_2)$ denotes the known values of the halftone patterns, above, left, and above/left of the block under consideration. We will refer to optimization of (4) as *local* and that of (14) as *neighboring* optimization.

Similar to the approach in Section II-A, we can write the convolution of f_{3N} with h as

$$(f_{3N} * h)(x,y) = \sum_{n_1=0}^{N-1}\sum_{n_2=0}^{N-1} b_{unknown}(n_1,n_2) g_r(x-n_1\Delta, y-n_2\Delta) + g_{fixed}(x,y) \quad (16)$$

where $g_{fixed}(x,y)$ denotes the contribution due to the known binary pixels b_{known} to the convolution $f_{3N} * h$. Combining (16) and (14), we get

$$MSE_{(x_0,y_0)} = \sum_{i=0}^{N-1}\sum_{j=0}^{N-1}\sum_{l=0}^{N-1}\sum_{m=0}^{N-1} G_{ijlm} b(i,j) b(l,m) + \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} R_{ij} b(i,j) + Q + Q\prime + \sum_{i=0}^{N-1}\sum_{j=0}^{N-1} R_{ij}\prime b(i,j) \quad (17)$$

where G_{ijlm}, R_{ij} and Q are defined in (9), (10), and (11) of Section II-A, except that the integration region is now changed from $D(x_0,y_0)$ to $D\prime(x_0,y_0)$. $Q\prime$ is defined as

$$Q\prime = \int_{D\prime(x_0,y_0)} [g_{fixed}^2(x,y) - 2c(x,y) g_{fixed}] dxdy \quad (18)$$

and $R_{ij}\prime$ is defined as

$$R_{ij}\prime = \int_{D\prime(x_0,y_0)} 2g_r(x-i\Delta, y-j\Delta) g_{fixed}(x,y) dxdy \quad (19)$$

and $D\prime(x_0,y_0)$ is an area of size $2N\Delta \times 2N\Delta$ as defined by the integration limits of (14). Note that the terms Q and $Q\prime$ in the expression for $MSE_{(x_0,y_0)}$ are simply constants and independent of b, and therefore they do not need to be computed for the actual optimization process.

An example of the neighboring optimization algorithm is shown in Fig. 6(a). The filter used in Fig. 6(a) is the triangular filter with $A = 1$. Comparing Fig. 6(a) to the images obtained using our basic algorithm in Fig. 3, we conclude that the neighboring algorithm results in less contouring artifact that the basic algorithm. To improve the cheesecloth appearance of Fig 6(a), we can rotate the optimization blocks of the neighboring algorithm by 45 degrees so that each block is a rhombus rather than square. The motivation behind doing so is to make the periodic artifacts appear at 45 degrees, and therefore less noticeable to the human eye. As seen, the resulting image, shown in Fig. 6(b), exhibits less horizontal and vertical artifacts than Fig. 6(a).

III. Frequency Domain Optimization

In this section, we develop another variation of the halftoning algorithm of the previous section. Specifically, we use a frequency, rather than space domain distortion measure to arrive at the binary optimum solution. In Section III-A we describe our proposed algorithm, and in Section III-B we show examples of halftone images obtained via the proposed algorithm.

A. The Algorithm

A distortion measure which is both tractable from an analytical point of view and is in reasonable agreement with human visual characteristics is the frequency weighted squared error (FWSE) criterion [2]. If $C(k_1, k_2)$ and $B(k_1, k_2)$ denote the discrete Fourier transform (DFT) of a continuous-tone image and its halftone version, then the frequency weighted MSE between the two can be written as

$$FWSE = \sum_{k_1,k_2} |H(k_1, k_2)|^2 |C(k_1, k_2) - B(k_1, k_2)|^2 \quad (20)$$

where $H(k_1, k_2)$ is the weighting function obtained from psychophysical experiments. While the above criterion is closely related to the one in (1), its main advantage lies in its superior image quality, and relative ease of implementation. For instance, the frequency domain optimization does not require numerical integrations of the form in (9). Therefore, unlike the space domain optimization, the function $H(k_1, k_2)$ need not be separable, and can be chosen to be circularly symmetric without a tremendous increase in the computation time.

It can be shown that if the weighting function in (20) is identically one, then optimization of (20) corresponds to fixed binary thresholding. The particular weighting function, $H(k_1, k_2)$ we have chosen corresponds to the first-order low-contrast MTF obtained from psychophysical experiments [13]:

$$H(k_1, k_2) = \begin{cases} 2.2(0.192 + 0.114\tilde{f}_{k_1,k_2}) \\ \quad \cdot \exp\left(-(0.114\tilde{f}_{k_1,k_2})^{1.1}\right) & \text{if } \tilde{f}_{k_1,k_2} > f_{max} \\ 1.0 & \text{otherwise.} \end{cases} \quad (21)$$

$\tilde{f}_{k_1,k_2}$ is the radial spatial frequency in cycles/degree and f_{max} is the frequency at which the exponential peaks. To implement (21) it is necessary to express radial frequencies in terms of the discrete horizontal and vertical frequencies of the discrete Fourier transform.

If Δ denotes the dot pitch of a sampled image and Q the number of horizontal or vertical frequencies, then the horizontal and vertical discrete frequencies are given by the following relations [13]:

$$f_{k_1} = \frac{(k_1 - 1)}{\Delta Q}, \quad f_{k_2} = \frac{(k_2 - 1)}{\Delta Q}. \quad (22)$$

These are converted to radial frequencies and scaled for a particular viewing distance, dis, into cycles/degree.

$$f_{k_1,k_2} = \frac{\pi}{180\, arcsin\left(\frac{1}{\sqrt{1+dis^2}}\right)} \sqrt{f_{k_1}^2 + f_{k_2}^2}. \quad (23)$$

Finally, to account for the anisotropy of the human visual system, the frequencies in (23) are normalized [13].

$$\tilde{f}_{k_1,k_2} = \frac{f_{k_1,k_2}}{s(\theta_{k_1,k_2})}, \quad (24)$$

where $s(\theta_{k_1,k_2})$ is given below.

$$s(\theta_{k_1,k_2}) = \frac{1-w}{2} cos(4\,\theta_{k_1,k_2}) + \frac{1+w}{2}, \quad (25)$$

with w as a symmetry constant and

$$\theta_{k_1,k_2} = arctan\left(\frac{f_{k_1}}{f_{k_2}}\right). \quad (26)$$

Equations (20)–(25) can be used to minimize the visually perceived distortion between an image and its halftone.

Having described the choice of weighting function, we now focus on the computational aspects of optimization. Similar to the previous section, we can reduce the computational intensity of the problem by dividing it into a number of smaller problems. Specifically, if we divide the continuous-tone, continuous-space image into small areas of size $N\Delta \times N\Delta$, then optimization of the $N \times N$ halftone pixels of the block centered at location (x_0, y_0), as shown in Fig. 1, involves minimization of

$$FWSE_{(x_0,y_0)} = \sum_{k_1,k_2=0}^{PN-1} |H(k_1, k_2)|^2 \cdot |C_{(x_0,y_0)}(k_1, k_2) - B_{(x_0,y_0)}(k_1, k_2)|^2 \quad (27)$$

where $C_{(x_0,y_0)}(k_1, k_2)$ is the DFT of $PN \times PN$ samples of the continuous-tone signal centered around (x_0, y_0), and $B_{(x_0,y_0)}(k_1, k_2)$ is the DFT of $PN \times PN$ samples of the bilevel piecewise constant halftone signal, $f_N(x, y)$, in the region $[x_0 - \frac{N\Delta}{2}, x_0 + \frac{N\Delta}{2}] \times [y_0 - \frac{N\Delta}{2}, y_0 + \frac{N\Delta}{2}]$. Thus, each of the N^2 binary pixels of the halftone signal are sampled at $P \times P$ locations. This results in a quadratic programming problem with N^2 variables which we solve via exhaustive search. We will refer to P as the oversampling ratio, and is used to avoid aliasing.

As well as oversampling, we can also pad each of the $PN \times PN$ samples of the contone and halftone images with zeros before taking their DFT. There are two reasons behind this. First, padding may be necessary in order for the FWSE criterion in (27) to correspond to a linear, rather than circular, convolution of the weighting filter h with the difference image $c - b$. Second, and more importantly, without padding, there may be too few samples of the MTF function in the frequency domain to permit an accurate representation of it. Padding increases the number of degrees of freedom in the DFT domain to shape the noise spectrum without introducing excessing complexity in the optimization.

B. Experimental Results

This section compares the results of the new halftoning algorithm with those of existing algorithms. The block size N was chosen to be 4. The halftone pictures in this section were optimized for a minimum viewing distance, dis, of 0.61 meters, and dot pitch of 0.0847 millimeters, corresponding to 300 dots per inch. In order to show the details of the halftone images more clearly, the examples shown in the paper were enlarged and printed at 150 dots per inch. Oversampling ratios, $P = 1$ and 2, were used as well as $0\times$ and $1\times$ zero padding.

We have found experimentally that $0\times$ zero padding for both $P = 1$ and $P = 2$ results in a distinct medium-high frequency noise which tends to obscure detail and gives the image an

Fig. 7. Frequency domain optimization with $P = 1$ and $2\times$ zero padding.

Fig. 8. Frequency domain optimization with $P = 1$ and neighboring algorithm.

overall "dirty" appearance. Fig. 7 shows the effect of $2\times$ zero padding with $P = 1$. The medium-high frequency noise of $0\times$ zero padding has all but disappeared and there is only a hint of high-frequency noise that is only just perceivable. A slight contouring effect is also visible on the shoulder. We have found that increasing the oversampling factor P from one to two in the image of Fig. 7 does not affect its appearance significantly.

C. The Neighboring Algorithm

As alluded to at the end of Section III-A and confirmed in Section III-B, zero padding has a significant effect on the overall appearance of the halftone. But rather than pad each sampled block with zeros before taking their DFT, we can apply the neighboring algorithm described in Section II-C, namely by using the previously halftoned blocks as padding for the current $N \times N$ block. To optimize the $N \times N$ halftone pixels of the block centered at location (x_0, y_0), we must minimize the following metric:

$$FWSE_{(x_0,y_0)} = \sum_{k_1,k_2=0}^{2PN-1} |H(k_1,k_2)|^2 \cdot |C_{(x_0,y_0)}(k_1,k_2) - B_{(x_0,y_0)}(k_1,k_2)|^2 \tag{28}$$

where $C_{(x_0,y_0)}(k_1,k_2)$ and $B_{(x_0,y_0)}(k_1,k_2)$ are the DFT's of $2PN \times 2PN$ samples of the continuous-tone and halftone signals in the region $[x_0 - \frac{3N\Delta}{2}, x_0 + \frac{N\Delta}{2}] \times [y_0 - \frac{N\Delta}{2}, y_0 + \frac{3N\Delta}{2}]$.

Using the parameters chosen in Section III-B—i.e., block size 4, viewing distance 0.61 m, and dot pitch 0.0847 mm—we find that the resulting halftone exhibits many fewer artifacts. In particular, there is little or no evidence of contouring. Unfortunately, the resulting halftone also exhibits a series of horizontal streaks in the region just above her head. Close examination of original image revealed the streaking region to be mostly uniform, indicating a lack of high frequencies. To handle this situation, we adjust the MTF by making it more low-pass. This can be done by increasing the viewing distance parameter. We found that increasing the viewing distance by 25% eliminated the streaking, without excessive blurring. The resulting halftone is shown in Fig. 8. The quality of the halftone image in Fig. 8 is superior to space domain optimization halftones in that it does not suffer from the cheesecloth texture typically seen in space domain optimization halftones.

IV. Comparison With Existing Techniques

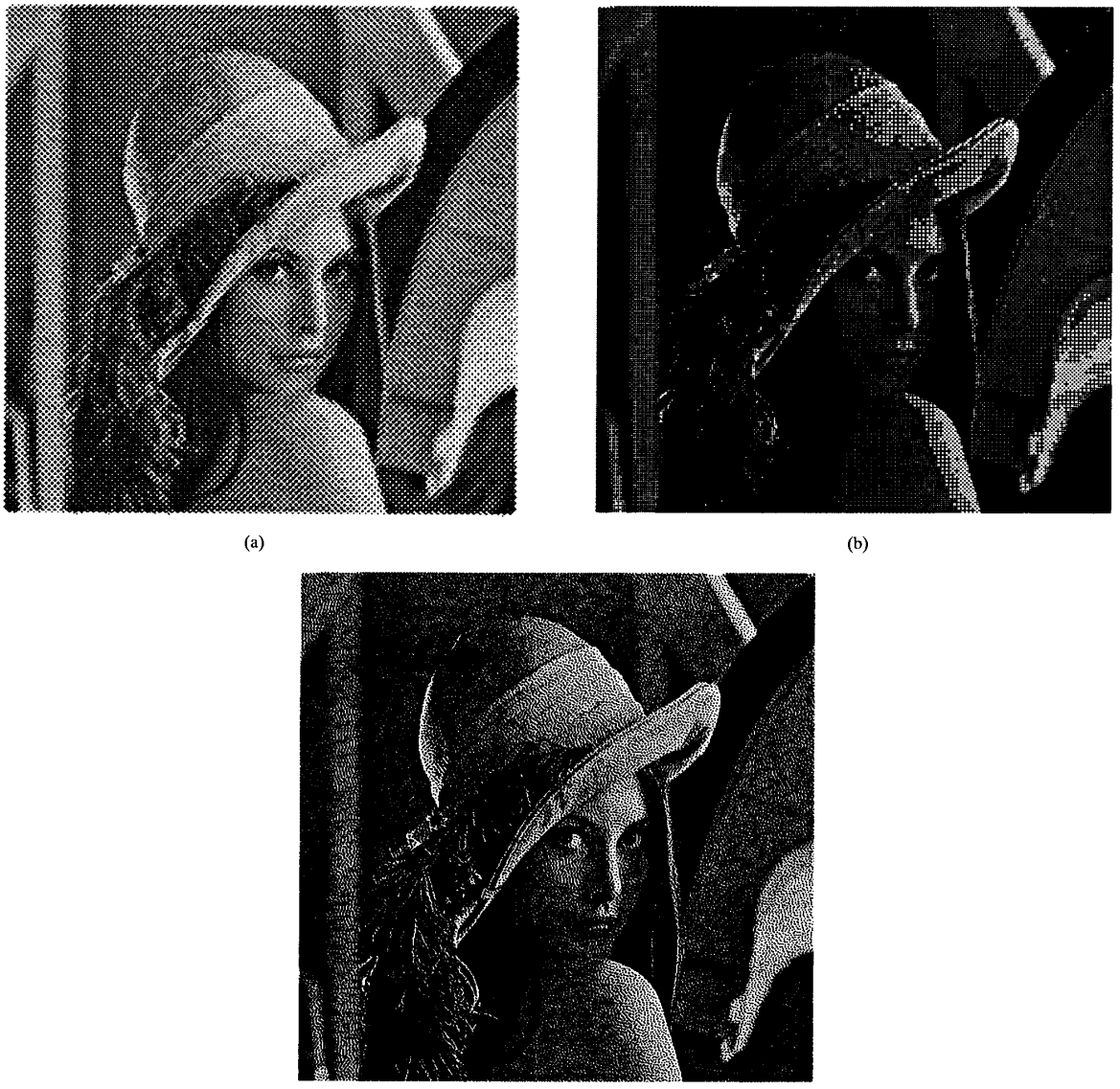
For comparison purposes, the clustered dot dither, dispersed dither, and error diffusion halftoned version of Lena are shown in Figs. 9(a), 9(b), and 9(c), respectively. The threshold matrix for the clustered dot dither of Fig. 9(a) has 18 levels and is given by [14]:

$$\begin{array}{cccccc} & & 11 & 14 & & \\ & 12 & 6 & 2 & 7 & \\ 18 & 15 & 3 & 1 & 4 & 16 \\ & 10 & 8 & 5 & 9 & \\ & & 13 & 17 & & \end{array}$$

The threshold matrix for dispersed dither of Fig. 9(b) has 16 elements and given by [5]:

$$\begin{array}{ccccc} 2 & 16 & 3 & 13 & 2 \\ 10 & 6 & 11 & 7 & 10 \\ 4 & 14 & 1 & 15 & 4 \\ 12 & 8 & 9 & 5 & 12 \\ 2 & 16 & 3 & 13 & 2 \end{array}$$

Jarvis filter with 12 coefficients was used for the error diffusion halftone image of Fig. 9(c) [15]. We have found that with our particular printer, Jarvis filter results in minimum printer artifacts as compared to Floyd and Stienberg and Stucki's filters [8]. While extensive subjective testing is outside the scope of this paper and therefore has not been performed, it



(a) (b)

(c)

Fig. 9. (a) Clustered dither halftoning. (b) Dispersed dither halftoning. (c) Error diffusion halftoning using Jarvis filter.

is the opinion of the authors that the images obtained via our proposed algorithms such as the one shown in Figs. 5, 6, and 8 have less contouring effects than the dispersed dither image, have less periodic artifact and are sharper than clustered dither, and have fewer snake like artifacts than the error diffusion picture. Among Figs. 5, 6(a), 6(b), and 8, the former two have a cheesecloth type horizontal/vertical artifact that the latter one is free of.

V. Computational Complexity

Although we have not optimized the computational speed of our proposed algorithms, it is nevertheless useful to say a few words about the arithmetic count of our proposed algorithms.

A. Non-Neighboring Space Domain Optimization

We now discuss the computation complexity of the space domain optimization algorithm of Section II. The computation cost can be divided into two parts: a) computing the coefficients G_{ijlm} and R_{ij} for $i, j, l, m = 0, \cdots, N-1$ via the integrals shown in (9) and (10); b) carrying out the summation in (8) for all possible combinations of the unknown binary array $b(i, j)$; since the array has N^2 unknown binary variables, the number of combinations of $b(i, j)$ are 2^{N^2}. To begin with, since the computation of G_{ijlm} is a one-time cost and signal-independent, it can be ignored in the run-time computational complexity[1]. On the other hand R_{ij} are signal-dependent:

[1] In spite of this, note that computing G_{ijlm} would tremendously simplify if g_r is separable.

to compute the double integral in (10), we exploit the fact that the continuous time signal $c(x,y)$ is piecewise constant over each pixel; Thus, the double integral can be replaced by a summation with N^2 terms, where each term requires 2 multiplications; this way, the number of multiplications for each (i,j) is $2N^2$ and the number of additions is N^2-1. Since i and j can each range from 0 to $N-1$, there are a total of N^2 values of (i,j) for which R_{ij} needs to be computed. Therefore, computing all the values of R_{ij} requires $2N^4$ adds and $N^2(N^2-1)$ multiplies.

To compute $MSE_{(x_0,y_0)}$ as shown in (8) for each combination of binary pixels of b, we need N^4 adds for the first summation in (8) and N^2 adds for the second summation in (8). Since there are 2^{N^2} combinations of b, once R_{ij} and G_{ijlm} are known, the number of adds for computing $MSE_{(x_0,y_0)}$ is $2^{N^2}(N^4+N^2)$. Including the cost for computing R_{ij}, the total computational cost per $N\times N$ block is $2^{N^2}(N^4+N^2)+2N^4 \approx 2^{N^2}N^4$ adds and $N^2(N^2-1)\approx N^4$ multiplies. Note that once the DC constraint is imposed, the number of combinations of binary variables that need to be searched drops from 2^{N^2} to $\frac{(N^2)!}{p!(N^2-p)!}$ where p is the number of pixels that are pre-determined to be white or black based on the DC constraint. Under these conditions, for $N=4$ and $p=8$, the number of adds is approximately 3.5 million and the number of multiplies is 240.

B. Neighboring Space Domain Optimization

A similar argument can be made for the neighboring version of the space domain optimization algorithm of Section II-C. The computation cost can be divided into two parts: a) computing the coefficients $R_{ij}\prime$ and R_{ij} for $i,j=0,\cdots,N-1$ via the integrals shown in (19) and (10);[2] b) carrying out the summation in (17) for all possible 2^{N^2} combinations of the unknown binary array $b_{unknown}(i,j)$; To compute the double integral in (10), we exploit the fact that the continuous time signal $c(x,y)$ is piecewise constant over each pixel; Thus, the double integral can be replaced by a summation with $4N^2$ terms, where each term requires 2 multiplications; this way, the number of multiplications for each (i,j) is $8N^2$ and the number of additions is $4N^2-1$. Since i and j can each range from 0 to $N-1$, there are a total of N^2 values of (i,j) for which R_{ij} needs to be computed. Therefore, computing all the values of R_{ij} requires $8N^4$ adds and $4N^2(N^2-1)$ multiplies. For similar reasons, computing N^2 values of $R_{ij}\prime$ also requires $8N^4$ adds and $4N^2(N^2-1)$ multiplies.

To compute $MSE_{(x_0,y_0)}$ as shown in equation (17) for each combination of binary pixels in $b_{unknown}$, we need N^4 adds for the first summation in (17) and N^2 adds for each of the second and third summations in (17). Since there are 2^{N^2} combinations of $b_{unknown}$, once R_{ij}, $R_{ij}\prime$ and G_{ijlm} are known, the number of adds for computing $MSE_{(x_0,y_0)}$ is $2^{N^2}(N^4+2N^2)$. Including the cost for computing R_{ij}, and $R_{ij}\prime$, the total computational cost per $N\times N$ block is $2^{N^2}(N^4+2N^2)+16N^4 \approx 2^{N^2}N^4$ adds and $8N^2(N^2-1)\approx 8N^4$ mults. Once the DC constraint is imposed, the number of combinations of binary variables that need to be searched drops from 2^{N^2} to $\frac{(N^2)!}{p!(N^2-p)!}$ where p is the number of pixels that are pre-determined to be white or black based on the DC constraint. Under these conditions, for $N=4$, and $p=8$ the number of adds is approximately 3.7 million and the number of multiplies is 1920.

The run time of the neighboring space domain optimization halftone image of Fig. 6(a) without imposing DC constraint is approximately 6 hours on a DEC 5000 workstation.[3]

C. Frequency Domain Optimization

Since the neighboring version of the frequency domain optimization results in higher quality halftone images than the nonneighboring version, we first discuss the computational complexity of the neighboring algorithm. In the frequency domain optimization algorithm, we first compute the DC in each block in order to determine the number of "on" pixels. If for a specific block, there are p pixels on, then the number of combinations of the block that need to be considered are $\frac{(N^2)!}{p!(N^2-p)!}$. Based on (28), for the neighboring algorithm with $P=1$, for each combination of a block, we need to a) compute two two-dimensional $2N\times 2N$ FFT's, one for $C(k_1,k_2)$ and one for $B(k_1,k_2)$ and b) combine the FFT's as shown in (28) to evaluate the distortion function $FWSE_{(x_0,y_0)}$. To calculate the FFT using row-column decomposition of an $2N\times 2N$ block requires $2N^2\log_2 4N^2$ multiplications and $4N^2\log_2 4N^2$ additions. Once the FFT's are computed, evaluation of the distortion function for $P=1$ requires an additional $8N^2$ multiplications and $8N^2$ additions. As an example, if $N=4$ and $P=1$ in the neighboring algorithm, then we need 512 multiplications and 896 additions for each combination of pixels in a given block. Based on the above discussion, if for example $p=8$ then approximately 7.4 million multiplications and 11.5 million additions are required for a given block in the neighboring frequency domain algorithm. The run time for the halftone picture of Fig.8 on a DEC 5000 workstation is about 12 hours.

For the 4×4 block non-neighboring algorithm, evaluation of the distortion function was optimized by identifying redundant and identity multiplications and additions. This resulted in only 288 multiplications and 456 additions per FFT and evaluation of the distortion function. Therefore, we expect it to be approximately twice as fast as the neighboring frequency domain optimization algorithm.

VI. Conclusions

We demonstrated the feasibility of applying binary optimization techniques to b/w halftoning. These algorithms minimize the distortion between the continuous-tone image and its low-pass-filtered halftone either in space or frequency domain.

[2]Note that in computing R_{ij} in the neighboring algorithm, the integration area in equation (10) should be $D\prime(x_0,y_0)$ rather than $D(x_0,y_0)$.

[3]Note that the run time is a different quantity from the CPU time. By run time, we mean the total amount of time we had to wait for the algorithm to finish provided there is no other computation load on the computer.

While the space and frequency domain distortion criteria are related, the computational properties of their corresponding algorithms are somewhat different. We examined the effect of various optimization parameters on the quality of the halftoned images. Specifically, we found that imposing DC constraint speeds up combinatorial optimization algorithms and results in halftone images with equal or higher quality than the basic algorithm. We also found that the neighboring algorithm reduces the contouring artifact of the basic space and frequency domain optimization algorithms.

Comparing the theoretical and actual computational complexity of the space and frequency domain optimization, we find that the space domain optimization is considerably faster. This can be attributed to fewer multiplications required by the space domain optimization. While the computational complexity of our proposed techniques is higher than traditional schemes such as dispersed dither and error diffusion, it seems likely that today's VLSI technology is capable of implementing our algorithm on an integrated circuit. The regular structure of the algorithm is particularly helpful in such implementations.

Future research should be directed toward enhancing basic distortion model and examination of various optimization filters.

As we mentioned in the introduction, in the last few years there has been a number of new results on model based halftoning [9], [8]. Specifically Pappas and Neuhoff consider the printer model in the halftoning algorithm in [8], and Analoui and Allebach find a near global minimum solution to the larger optimization problem iteratively rather than dividing it into a number of smaller ones [9]. In doing so, they find that the final solution is highly dependent on the initial binary image used in the iterative algorithm; they also develop a "look-up" technique which effectively reduces the CPU time of their global optimization algorithm from about 350 minutes to about 20 minutes. One interesting direction for future research is to see whether the "look-up" approach of Analoui and Allebach in [9] can be used directly or indirectly in our block-based approach. Anoter possibility is to see whether there are intermediate approaches in between global optimization technique in [9] and the approach presented in this paper.

ACKNOWLEDGMENT

The authors would like to express their gratitude to Dr. Yong Liu for generously providing us with his software on optical lithography mask design and optimization, which is the basis of our space domain optimization algorithm. Many thanks go to Frank Yang for writing the program to generate Fig. 6(b). Helpful comments from anonymous reviews are also appreciated.

REFERENCES

[1] A. Zakhor and A.V. Oppenheim, "Reconstruction of two dimensional signals from level crossings," *Proc. IEEE*, vol. 78, pp. 31, Jan. 1990.
[2] D. Anastassiou, "Error diffusion coding for A/D conversion," *IEEE Trans. on Circuits and Systems*, vol. 36, pp. 1175–1186, Sept. 1989.
[3] Y. Liu, "Image synthesis: Binary and phase shift mask design for optical lithography," Ph.D. dissertation, University of California at Berkeley, Dec. 1992.
[4] J. C. Stoffel and J. F. Moreland, "A survey of electronic techniques for pictorial image reproduction," *IEEE Trans. on Communications*, vol. 29, pp. 1898–1925, Dec. 1981.
[5] R. Ulichney, *Digital Halftoning*. Cambridge, MA: MIT Press, 1987.
[6] T. S. Rao and G. R. Arce, "Halftone patterns for arbitrary screen periodicities," *J. Opt. Soc. Am. A*, vol. 5, pp. 1502–1511, Sept. 1988.
[7] T. S. Rao, G. R. Arce, and J. P. Allebach, "Analysis of ordered dither for arbitrary sampling lattices and screen periodicities," *IEEE Trans. on Acoust., Speech, and Sig. Proc.*, vol. 38, pp. 1981–2000, Nov. 1990.
[8] T. Pappas, N. Seshadri, and D. L. Neuhoff, "Model based halftoning," Proceedings of the 7th MDSP workshop, Sept. 1991.
[9] M. Analoui and J. P. Allebach, "Model based halftoning using direct binary search," Proceedings of the SPIE Symposium on Electronic Imaging, Feb. 1992, San Jose, CA.
[10] J. L. Mannos and D. J. Sakrison, "The effects of a visual fidelity criterion on the encoding of images," *IEEE Trans. on Information Theory*, vol. 20, pp. 525–536, July 1974.
[11] Y. Liu and A. Zakhor, "Optimal binary image design based on the branch and bound algorithm," in *Proc. ICASSP 90*, Albuquerque, New Mexico, Apr. 3–6, pp. 1877–1880.
[12] A. Zakhor, F. Eskafi, and S. Lin, "A new class of b/w and color halftoning algorithms," in *Proc. ICASSP 91*, Toronto, Canada, May 1991, pp. 2800–2803.
[13] J. Sullivan, L. Ray, and R. Miller, "Design of minimum visual modulation halftone patterns," *IEEE Transactions on Systems, Man and Cybernatics*, vol. 21, Jan./Feb. 1991, pp. 33–38.
[14] G. Thompson and G. Goertzel, "Digital halftoning for monochrome and color printing," *SPSE 4th International Congress on Advances in non-Impace printing Technologies*, 1988, New Orleans, LA.
[15] G. Goertzel and C. Evangelisti, "Color conversion of digital images," *SPSE 42nd Annual Conference*, Boston, MA 1989.

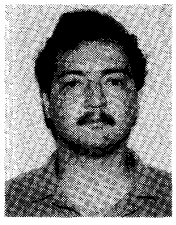
Avideh Zakhor (S'87–M'87) received the B.S. degree from California Institute of Technology, and the S.M. and Ph.D. degrees from MAssachusetts Instutute of Technolgy, all in electrical engineering, in 1983, 1985, and 1987, respectively.

In 1988, she joined the faculty at U.C. Berkeley, where she is currently Assistant Professor in the Department of Electrical Engineering and Computer Sciences. Her research interests are in the general area of signal processing and its applications to images and video, and biomedical data. She has been a consultant to a number of industrial organizations and holds four U.S. patents.

Dr. Zakhor was a General Motors scholar from 1982 to 1983; received the Henry Ford Engineering Award and Caltech Prize in 1983; was a Hertz Fellow from 1984 to 1988; received the Presidential Young Investigators award, IBM junior faculty development award, and Analog Devices junior faculty development award in 1990; and Office of Naval Research Young Investigator Award in 1992. She is currently Associate Editor for IEEE TRANSACTIONS ON IMAGE PROCESSING and a member of the technical committee for multidimensional digital signal processing.

Steve Lin received the B.S. and M.S. degrees in electrical engineering and computer science in 1990 and 1993, respectively, from the University of California, Berkeley, CA. His main research interest was digital image halftoning.

In 1992 he joined a startup company where he applied digital image processing techniques to video telephony. He is currently a software engineer at Cisco Systems in Menlo Park, CA.



Farokh Eskafi received the B.S. and M.S. degrees in 1991 and 1993, respectively, in electrical engineering, from the University of California at Berkeley.

He is currently a Ph.D. student in the Electrical Engineering and Computer Science Department of U.C. Berkeley, and his main research is on IVHS and the communication network needed for support of IVHS implementation.