Efficient Model Based Halftoning Using Direct Binary Search*

David J. Lieberman and Jan P. Allebach
School of Electrical and Computer Engineering
Purdue University
West Lafayette, IN 47907-1285
{dlieberm,allebach}@ecn.purdue.edu

Abstract

The direct binary search (DBS) algorithm is an iterative method which minimizes a metric of error between the grayscale original and halftone image. This is accomplished by adjusting an initial halftone until a local minimum of the metric is achieved at each pixel. The metric incorporates a model for the human visual system (HVS). In general, DBS time complexity and halftone quality depend on three factors: The HVS model parameters, the choice of initial halftone, and the search strategy used to update the halftone. Despite the complexity of the DBS algorithm, it can be implemented with surprising efficiency. We will demonstrate how the algorithm exploits the model for the HVS to efficiently yield very high quality halftones.

1 Introduction

Halftoning algorithms are used to transform continuous-tone grayscale images into binary images. This process is necessary to render the image on a binary display or as printed material generated by either a laser or inkjet printer. Due to the lowpass characteristics of the human visual system (HVS), these rendered images are perceived as exhibiting grayscale. Thus, the objective of every halftoning algorithm is to minimize some measure of the difference between the perceived halftone and perceived continuous-tone image.

A number of halftoning techniques search for a halftone which minimizes a global measure of error based on models of the HVS and/or output device [1, 2, 3, 4, 5]. The halftone which minimizes the error is found iteratively. These algorithms produce high quality halftones, but they require a relatively high level of computation compared to conventional techniques. The algorithm by Carnevali [1] is based on simulated annealing whereas the algorithms designed by Kollias and Anastassiou [2] and Benard [3] are both based on neural networks. Pappas [4] uses a least-squares model based technique which is similar to our approach. However, our work focuses on computational efficiency.

*Research supported by the Hewlett-Packard Company.

The DBS algorithm proceeds by transforming an initial halftone into one which locally minimizes the visual model response to the error image. Since this minimum is local, it is not unique and a given continuous-tone image and fixed cost function can be used to generate different halftones. We have found that the most appropriate technique for generating an initial halftone is to employ a screen function designed using DBS [6]. In this paper, we investigate the effect of the HVS model and search strategy on halftone quality and computational complexity. To further reduce complexity, we develop a technique that approximates the results of a filtering operation faster than conventional FFT based approaches. In addition, DBS is modified to correct for tone bias. We begin with an overview of DBS in Sec. 2. In Sec. 3, we investigate methods for improving halftone quality. Computational issues are investigated in Sec. 4. Results are in Sec. 5.

2 Overview of DBS

Direct binary search (DBS) is a recursive search heuristic which uses a HVS model to minimize a measure of the perceived error image: the difference between the perceived halftone and the perceived continuous-tone image. We model the lowpass characteristics of the HVS as a linear shift-invariant filtering operation. We use a luminance spatial frequency response function proposed by Näsänen [7] to compute a point spread function $\tilde{p}(\mathbf{x})$. We use f, g, and e to denote the grayscale, binary, and error images, respectively. The perceived versions of these images $\tilde{f}$, $\tilde{g}$, and $\tilde{e}$ are computed as

$$\tilde{f}(\mathbf{x}) = \sum_{n} f[\mathbf{n}]\tilde{p}(\mathbf{x}-\mathbf{X}\mathbf{n}), \quad (1)$$

$$\tilde{g}(\mathbf{x}) = \sum_{n} g[\mathbf{n}]\tilde{p}(\mathbf{x}-\mathbf{X}\mathbf{n}), \quad (2)$$

$$\tilde{e}(\mathbf{x}) = \sum_{n} e[\mathbf{n}]\tilde{p}(\mathbf{x}-\mathbf{X}\mathbf{n}), \quad (3)$$

where $[\mathbf{n}] = [m,n]^T$ and $(\mathbf{x}) = (x,y)^T$ represent discrete and continuous spatial coordinates, $e[\mathbf{n}] = f[\mathbf{n}] - g[\mathbf{n}]$ is the error image, and $\mathbf{X}$ is the periodicity matrix corresponding to the lattice of addressable points of the output device. The

0-8186-8183-7/97 $10.00 © 1997 IEEE

error measure minimized by DBS is the total squared perceived error given by

$$E = \int_{\mathbf{x}} |\tilde{e}(\mathbf{x})|^2. \tag{4}$$

Although our error metric operates upon a continuous-space version of the perceived error image $\tilde{e}(\mathbf{x})$, we may evaluate this measure by accurately computing the values of certain correlation functions at discrete points only. We may substitute (3) into (4) to express the initial error as

$$\begin{aligned} E &= \sum_n \sum_m e[\mathbf{n}] e[\mathbf{m}] c_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{n}], \\ &= \sum_n e[\mathbf{n}] c_{\tilde{p}\tilde{e}}[\mathbf{n}], \end{aligned} \tag{5}$$

where $c_{\tilde{p}\tilde{p}}[\mathbf{m}]$ is the autocorrelation of $\tilde{p}(\mathbf{x})$ and $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ is the cross correlation function between $\tilde{p}(\mathbf{x})$ and the perceived error image $\tilde{e}(\mathbf{x})$.

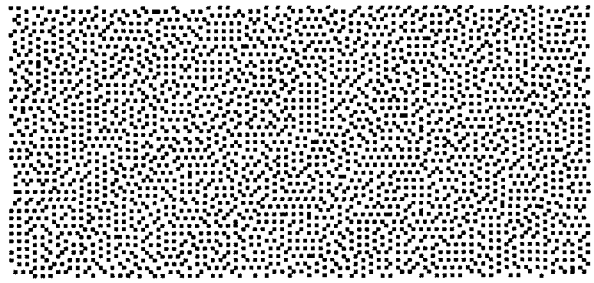
Pixels of the halftone image are processed by evaluating the impact of a trial pixel change on the error E. The pixel changes considered are restricted to toggling the pixel or swapping its binary state with one of its neighbors. If the trial evaluation reduces error E, we may choose to accept the change. Processing every pixel once and accepting every change that reduces the error will not guarantee convergence. The number of iterations required is image dependent.

3 Improving DBS Quality

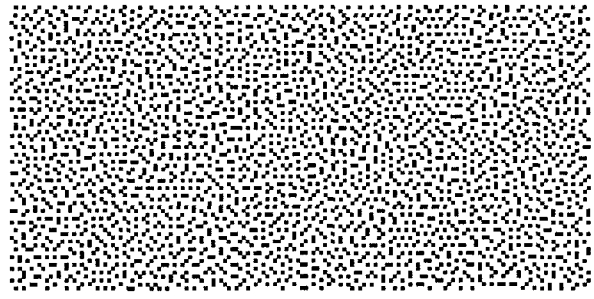
The value of the HVS scale (distance $\times$ resolution) parameter has a significant influence on the textures generated by DBS. The scale parameter of the visual model accounts for the expected viewing distance and the resolution of the display device. The cutoff frequency of the HVS spatial frequency response is a decreasing function of scale. Consequently, using a small scale value results in halftones which are very smooth, but which exhibit low variety. A large scale value will cause DBS to generate coarser textures with more variety as illustrated in Fig. 1.

In Fig. 1, we show two DBS halftones corresponding to a uniform input image with a greylevel of 0.25 in units of absorptance. A small scale parameter encourages DBS to generate textures which exhibit a checkerboard pattern as shown in Fig. 1(a). This pattern is only disrupted by occasional shifts and diagonal textures. A large scale parameter encourages DBS to generate noisier halftones without repetitive texture as in Fig. 1(b).

We choose a moderate scale value (10in.$\times$300dpi.), then we modify the DBS algorithm to improve tone accuracy. We compute the mapping from grayscale tone to the average tone of the binary image. The inverse mapping is used as a tone correction curve. In this way, we may coax DBS into generating halftones which exhibit negligibly small tone error. The corrected and uncorrected DBS tone error curves are plotted in Fig. 2.



(a) DBS Halftone using scale parameter 6in.$\times$300dpi.



(b) DBS Halftone using scale parameter 12in.$\times$300dpi.

Figure 1: DBS halftone textures printed at 50dpi.

4 Improving DBS Efficiency

Our most effective technique for reducing computational complexity is an efficient procedure for evaluating the effect of trial changes [5]. Consider the effect on the perceived halftone $\tilde{g}(\mathbf{x})$ of a trial change in the state of two halftone pixels at positions $\mathbf{m}_0$ and $\mathbf{m}_1$. The new perceived halftone $\tilde{g}'(\mathbf{x})$ and error image $\tilde{e}'(\mathbf{x})$ are

$$\tilde{g}'(\mathbf{x}) = \tilde{g}(\mathbf{x}) + a_0 \tilde{p}(\mathbf{x}-\mathbf{X}\mathbf{m}_0) + a_1 \tilde{p}(\mathbf{x}-\mathbf{X}\mathbf{m}_1), \tag{6}$$

$$\tilde{e}'(\mathbf{x}) = \tilde{e}(\mathbf{x}) - a_0 \tilde{p}(\mathbf{x}-\mathbf{X}\mathbf{m}_0) - a_1 \tilde{p}(\mathbf{x}-\mathbf{X}\mathbf{m}_1), \tag{7}$$

where $a_0 = -1$ if $g[\mathbf{m}_0] = 1$, $a_0 = 1$ if $g[\mathbf{m}_0] = 0$. We set $a_1 = 0$ to toggle one pixel, and $a_1 = -a_0$ to swap the binary states of two pixels. We substitute (7) into (4), to express the change in error $\triangle E$ as

$$\begin{aligned} \triangle E &= (a_0^2 + a_1^2) c_{\tilde{p}\tilde{p}}[\mathbf{0}] - 2a_0 c_{\tilde{p}\tilde{e}}[\mathbf{m}_0] - \\ &\quad 2a_1 c_{\tilde{p}\tilde{e}}[\mathbf{m}_1] + 2a_0 a_1 c_{\tilde{p}\tilde{p}}[\mathbf{m}_1 - \mathbf{m}_0]. \end{aligned} \tag{8}$$

$\triangle E$ can be evaluated with a few table lookups and additions. If a trial change is accepted, the halftone pixel is toggled or swapped and the $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ look up table (LUT) must

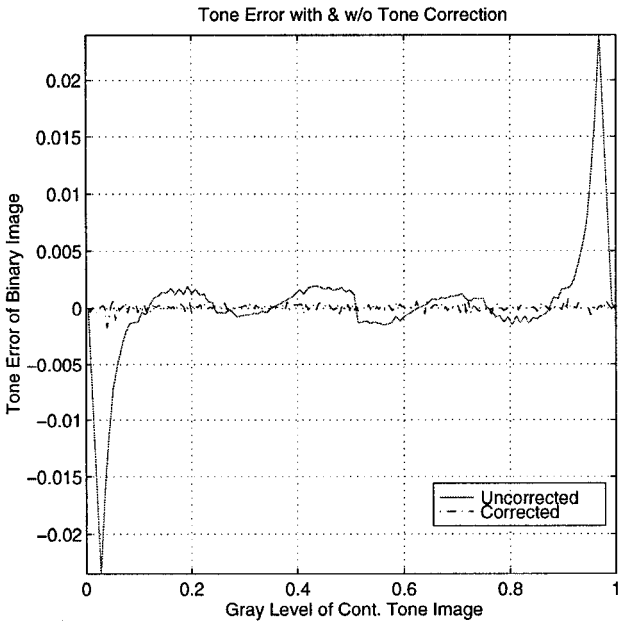


Figure 2: Corrected and uncorrected DBS tone error curves.

be updated to $c'_{\tilde{p}\tilde{e}}[\mathbf{m}]$ as follows

$$\begin{aligned} c'_{\tilde{p}\tilde{e}}[\mathbf{m}] &= c_{\tilde{p}\tilde{e}}[\mathbf{m}] - a_0 c_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_0] - \\ &\quad a_1 c_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_1]. \end{aligned} \tag{9}$$

Since $\tilde{p}(\mathbf{x})$ has support over an N^2 region of pixels, the support of $c_{\tilde{p}\tilde{p}}[\mathbf{m}]$ is limited to $(2N-1)\times(2N-1)$ pixels. Thus, the update given by (9), requires roughly four times more computation than the direct approach update. The fast evaluation implementation is effective because in a typical halftoning application, roughly 400 trial evaluations are computed for each accepted change. Once the LUTs are initialized, halftones can be processed efficiently. However, the $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ LUT has the same dimensions as the image, and its initialization can consume two-thirds of the total DBS time complexity.

Our next goal is to reduce the complexity of the $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ initialization. This initialization is computed as

$$c_{\tilde{p}\tilde{e}}[\mathbf{m}] = \sum_{\mathbf{n}} c_{\tilde{p}\tilde{p}}[\mathbf{n}-\mathbf{m}]e[\mathbf{n}]. \tag{10}$$

We have established that if we accept a small variation in $\tilde{p}(\mathbf{x})$, we can initialize $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ several times faster than conventional block based FFT approaches. We begin by observing that any constant-valued $2D$ rectangular kernel R of arbitrary size, can be used to filter images with only 5 additions per output point. For each row in the image f, we compute the sum over the columns. Then we repeat the procedure for each column. The result is an *integrated* image, $Integ[m,n] = \sum_{[i,j]\le[m,n]} f[i,j]$ (To ensure clarity, we will occasionaly employ discrete spatial coordinates $[m,n]$ in our notation). If R has dimensions $2H+1$ by $2W+1$, then the filtered output $Filt[m,n]$ is

$$\begin{aligned} Filt[m,n] &= Integ[m+H,n+W] - \\ &\quad Integ[m+H,n-W-1] - \\ &\quad Integ[m-H-1,n+W] + \\ &\quad Integ[m-H-1,n-W-1]. \end{aligned} \tag{11}$$

We use three $2D$ rectangular functions R_1, R_2, and R_3, and one constant k to replace $\tilde{p}(\mathbf{x})$. The substitution $\hat{\tilde{p}}[m,n]$ and corresponding autocorrelation function $\hat{c}_{\tilde{p}\tilde{p}}[m,n]$ are computed as

$$\hat{\tilde{p}}[m,n] = kR_1 * R_2 + (1-k)R_3, \tag{12}$$

$$\begin{aligned} \hat{c}_{\tilde{p}\tilde{p}}[m,n] &= k^2 R_1 * R_1 * R_2 * R_2 + \\ &\quad 2k(1-k)R_1 * R_2 * R_3 + \\ &\quad (1-k)^2 R_3 * R_3, \end{aligned} \tag{13}$$

where $*$ indicates $2D$ discrete convolution.

Note that $\tilde{p}[m,n]$, $c_{\tilde{p}\tilde{p}}[m,n]$, and all three $2D$ rectangular functions are normalized so they sum to one. To insure $\hat{\tilde{p}}[m,n]$ is non-negative, we require the value of k to be within the unit interval. We perform an exhaustive search (over the realistic range of values) for the best combination of $2D$ rectangular functions to match $c_{\tilde{p}\tilde{p}}[m,n]$. For each combination, we choose the value of k which minimizes the squared error ε, between the genuine $c_{\tilde{p}\tilde{p}}[m,n]$ and $\hat{c}_{\tilde{p}\tilde{p}}[m,n]$ given by

$$\varepsilon = \sum_{m,n} (c_{\tilde{p}\tilde{p}}[m,n] - \hat{c}_{\tilde{p}\tilde{p}}[m,n])^2. \tag{14}$$

We can substitute (13) into (14) to obtain a forth order polynomial in k. Therefore, we may take its derivative, and solve the resulting cubic polynomial for the value of k in the unit interval which minimizes ε. Once the three rectangular functions and the value of k have been selected, we can compute $\hat{c}_{\tilde{p}\tilde{e}}[m,n]$. First, we compute Integ; using Integ, compute filtered outputs Filt_1 and Filt_3 using the dimensions of the rectangular functions R_1 and R_3, respectively. Then, Filt_1 and Filt_3, are integrated to produce Integ_1 and Integ_3, respectively. Next, we use Integ_3 and the dimensions of R_3 to compute Filt_{33}, the output from filtering the error image with R_3 twice. Using similar notation, the process of computing $\hat{c}_{\tilde{p}\tilde{e}}[\mathbf{m}]$ is illustrated in Fig. 3.

As the algorithm proceeds, we allocate memory in blocks; Each block is roughly the size of the grayscale image. If memory is managed carefully, the peak demand for memory is four blocks. We could use less memory, but this would prohibit us from computing multiple filtering and

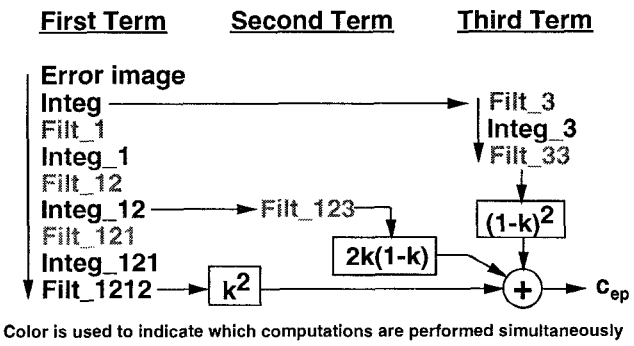


Figure 3: Procedure for computing $\hat{c}_{\bar{p}\bar{e}}$.

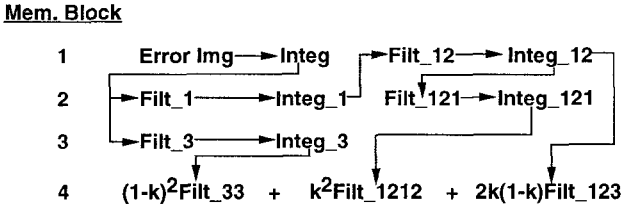


Figure 4: Memory management computing $\hat{c}_{\bar{p}\bar{e}}$.

integrating operations simultaneously. This would significantly increase time complexity. Our memory management scheme is illustrated in Fig. 4.

Our last technique for reducing time complexity involves modifying the search strategy. The most obvious approach for running DBS is to process pixels in raster-scan order (from left to right, and top to bottom). We could use (8) to compute the effect of a trial change, and accept every change which reduces the time complexity. We would then iterate until no changes are accepted in an entire iteration. This approach is called the greedy strategy. Using the greedy strategy, the vast majority of accepted changes yield a negligibly small reduction in error. Consequently, to achieve a significant improvement in the halftone quality, we are required to accept a large number of changes. Another problem is that some portions of a halftone will converge to a local minimum of the cost function more rapidly than other portions. The computation required to continue processing these areas is wasted.

Our solution is a block based strategy. First, the halftone under test is tessellated into non-overlapping blocks. During an iteration, each block is processed as follows: The effect of trial toggles and swaps are computed for each pixel in the block. Only the trial change corresponding to the largest reduction in the cost function is accepted. Then, the required update is performed. Since each change accepted is one which produces the largest local improvement in the halftone, the halftone converges to a local minimum of the cost function more rapidly. If no changes are accepted in this block for two consecutive iterations, the block will not receive further processing. The block based strategy efficiently exploits our ability to make fast evaluations and reduces the number of required pixel changes by roughly 90%. In addition, since we omit processing blocks that have already converged, we avoid wasted computation.

5 Results

We have improved the quality and reduced the computational complexity of DBS halftoning. The block based strategy runs 10 times faster than the greedy strategy. Using our approximation approach, the MSE between $\hat{c}_{\bar{p}\bar{e}}$ and $c_{\bar{p}\bar{e}}$ is only 0.5%, and $\hat{c}_{\bar{p}\bar{e}}[\mathbf{m}]$ is computed 6 times faster than an optimized block based FFT approach. Using an HP755 workstation, the processing time required for a 1024×1024 grayscale image has been reduced from 25 min. to 75 seconds. DBS has been modified to achieve high tone accuracy, and generates visually pleasing textures.

References

[1] L. C. P. Carnevali and S. Patarnello, "Image processing by simulated annealing," *IBM J. Res. Develop*, vol. 29, pp. 569 – 579, November 1985.

[2] S. Kollias and D. Anastassiou, "Image processing by simulated annealing," *IBM J. Res. Develop*, vol. 29, pp. 569 – 579, November 1985.

[3] T. Bernard, "From $\sum - \triangle$ modulation to digital halftoning of images," *Proc. of 1991 International Conf. on Acoustic, Speech, and Sig. Proc.*, May 14-17 1991, Toronto, Canada, pp. 2805 – 2808.

[4] T. Pappas and D. Neuhoff, "Least-squares model-based halftoning," *Proc. SPIE/IS&T Symp. on Elec. Imag. Sci. and Tech.*, Feb. 1992, San Jose, CA, pp. 165 – 176.

[5] M. Analoui and J. P. Allebach, "Model-based halftoning using direct binary search," *Proc. of SPIE/IS&T Symp. on Electronic Imaging Science and Tech.*, February 1992, San Jose, CA, pp. 96 – 108.

[6] J. P. Allebach and Q. Lin, "Fm screen design using dbs algorithm," *Proc. of 1996 IEEE International Conference on Image Proc.*, 16-19 Sept. 1996, Lausanne, Switzerland.

[7] R. Nasanen, "Visibility of halftone dot textures," *IEEE Trans. Syst. Man. Cyb.*, vol. 14, no. 6, pp. 920 – 924, 1984.