A High Quality, Fast Inverse Halftoning Algorithm for Error Diffused Halftones

Thomas D. Kite, Niranjan Damera-Venkata, Brian L. Evans, and Alan C. Bovik
Laboratory for Image and Video Engineering
Department of Electrical and Computer Engineering
The University of Texas at Austin, Austin, TX 78712-1084 USA
E-mail: {tom,damera,bevans,bovik}@vision.ece.utexas.edu

Abstract— **We present an inverse halftoning algorithm for error diffused halftones. At each pixel, the algorithm applies a separable 7×7 FIR filter parameterized by the computed local horizontal and vertical gradients. All operations are entirely local; only 7 rows of image storage and fewer than 300 operations per pixel are required. The algorithm can be easily implemented in embedded software or hardware. We compare our algorithm with previously reported approaches, and show that it delivers comparable PSNR and subjective quality at a fraction of the computation and memory requirements. A C implementation of the algorithm is available at** http://www.ece.utexas.edu/~bevans/projects/inverseHalftoning.html.

I. Introduction

Inverse halftoning algorithms recover grayscale images from halftones. This is useful when a halftone is the only available version of an image, and enhancement, compression, or some other manipulation of the image is required. Apart from very simple operations, such as cropping and rotation through multiples of 90°, halftones cannot be manipulated without causing severe image degradation. They are also difficult to compress, either losslessly or lossily; grayscale images, on the other hand, can be compressed efficiently [1], [2]. The ability to generate an inverse halftone allows a wide range of operations to be performed.

Several inverse halftoning methods have been described in the literature. Screened halftones and error diffused halftones have greatly differing artifacts, and must be dealt with individually. We focus on error diffusion. Published inverse halftoning methods for error diffused halftones include vector quantization [2], projection onto convex sets [3], nonlinear permutation filtering [4], MAP projection [5], wavelets [6], and Bayesian schemes [7]. Many methods show good results, but several are iterative, requiring large amounts of computation and memory. Most also make heavy use of floating-point arithmetic.

In this paper, we present a single-pass scheme with the lowest computation and memory requirements that produces results comparable to those seen in the literature. Our scheme consists of multiscale directional gradient estimation followed by adaptive lowpass filtering. Most of the processing is accomplished with integer additions, and only seven image rows are kept in memory at one time.

This research was supported in part by a grant from Hewlett-Packard Laboratories, Palo Alto, CA, and an NSF CAREER Award under Grant MIP-9702707.

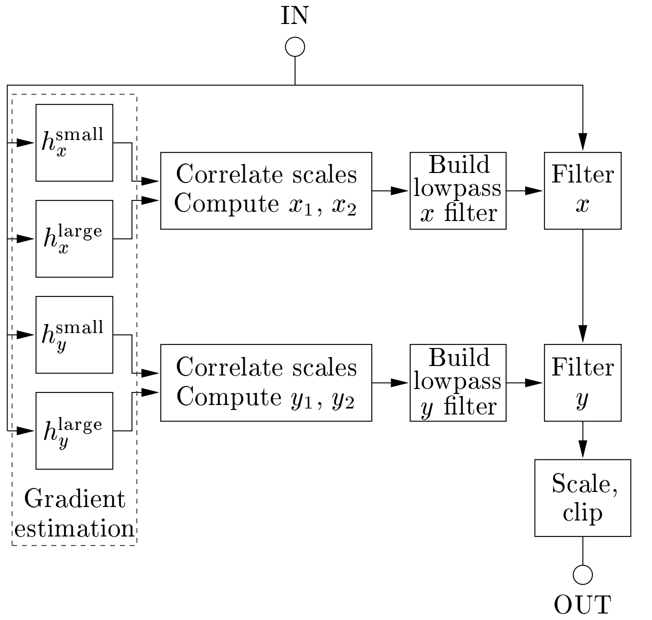


Figure 1: Block diagram of the inverse halftoning algorithm. Outputs of the four gradient estimation filters are correlated across scales and conditioned to produce filter parameters x_1, y_1, from which x_2, y_2 are computed. The x and y filters are constructed and applied separably.

II. Background

Halftones have a very low signal-to-noise ratio (SNR) relative to the original image, because of the one-bit wordlength. In error diffused halftones, most of the quantization noise power falls at high spatial frequencies. This is known as the *blue noise* characteristic, and is proposed as optimal in [8]. An inverse halftoning scheme should remove as much of this noise as possible, while retaining important image features. Using a linear lowpass filter gives poor results, since it is impossible to find a filter that produces adequately smooth regions and sharp edges simultaneously.

The algorithm described here is a form of *anisotropic diffusion*, a tool introduced by Perona and Malik principally to implement robust multi-scale edge detection [9]. Anisotropic diffusion estimates image gradients to compute a diffusion coefficient that governs smoothing. A non-linear relationship between the estimated gradient and the diffusion coefficient encourages smoothing inside regions, but not between them. To perform inverse halftoning, we estimate gradients from the halftone, and derive *control functions* that vary the cutoff frequency of a smoothing filter. A block diagram of the algorithm is shown in Figure 1.

Error diffusion is equivalent to a two-dimensional form of delta-sigma modulation [10], and can be viewed as spatially-interactive wordlength reduction. Inverse halftoning is therefore spatially-interactive wordlength expansion; the increase in wordlength is achieved by averaging over a neighborhood of pixels. We vary the trade-off between spatial resolution (detail) and grayscale resolution (wordlength) according to the local image gradient. The gradient estimation and filtering is conducted separably in the horizontal (x) and vertical (y) directions, allowing smoothing to occur parallel to an edge. This increases the wordlength, without blurring the edge.

III. Smoothing filter design

The smoothing filter, which recovers the inverse halftone from the halftone, must satisfy the following criteria:

- Small extent, FIR
- Simple to generate
- Separable
- Cutoff frequency determined by a single parameter
- Frequency response tailored for halftones

An FIR filter is guaranteed to be stable, and its output can be computed quickly when its extent is small. Computation is reduced by making the filter simple to generate on the fly. By making the filter separable, it can be designed, constructed and applied independently in each direction, thereby further reducing execution time. We require the cutoff frequency of the filter in each direction to be determined by one parameter, namely, the control function.

The frequency response of the filter is constrained to account for the particular characteristics of halftones. Strong idle tones (also known as "worms") are often present, and should be suppressed in the inverse halftone, else they will lead to undesirable texture. In Floyd-Steinberg error diffusion [11], idle tones are particularly likely to occur at (f_N, f_N), $(f_N, 0)$, and, to a lesser extent, $(0, f_N)$ [12], where f_N refers to the Nyquist frequency, and (f_h, f_v) denotes horizontal and vertical spatial frequency, respectively. We place a zero at f_N in the lowpass filter to suppress these tones. Halftones produced using Jarvis error diffusion [13] are less likely to contain these tones [12].

We require the gain of the filter to be unity at DC, to preserve the image mean (brightness). We use a symmetric filter for linear phase; it is well-known that this is critical for good performance of image processing filters [14]. Two parameters are free to determine the filter response. We constrain the maximum passband ripple to ensure that the inverse halftone is a faithful reproduction of the original image. A filter with an excessively peaked passband produces falsely sharpened images. We found empirically that restricting the ripple to $\pm 7\%$ (± 0.59 dB) produced high quality images. The maximum stopband gain was specified as 0.05 (-26.0 dB), so that the total noise power in the filter output decreases monotonically as the cutoff frequency of the filter is lowered. If the maximum stopband gain is not specified, it is possible to design a filter a whose cutoff frequency is lower than that of filter b, yet whose output has a higher noise power. This produces poor inverse halftones, since the reduction of quantization noise is no longer inversely proportional to the local image gradient.

The class of one-dimensional filters satisfying the criteria of unity gain at DC and a zero at f_N has the form

$$[\, x_2 - x_1 + 2,\ x_2,\ x_1,\ 4,\ x_1,\ x_2,\ x_2 - x_1 + 2 \,]\,, \qquad (1)$$

where x_1 and x_2 are chosen so that the filter satisfies the passband and stopband specifications. A scaling factor of $1/(4x_2 + 8)$ is applied to achieve unity gain at DC. We refer to this class of filters as the *one-dimensional prototype* class. We construct two filters from the class at each pixel of the input image, one for each of the x and y directions.

We designed ten lowpass filters that met the specifications using the sequential quadratic programming (SQP) algorithm in the Matlab optimization toolbox. This algorithm varies parameters (in this case, x_1 and x_2) to minimize a cost function, subject to a constraint. We used the passband ripple as the constraint, and the maximum gain in the stopband as the cost function. These definitions lead to equiripple filters in principle, and near-equiripple filters in practice. We then found a cubic polynomial fitting the designed values of x_2 to the corresponding values of x_1, so that only one parameter is needed to specify the filter. The fit is given by (the y relation is analogous):

$$x_2 = 0.4631x_1^3 - 2.426x_1^2 + 4.660x_1 - 3.612\,. \qquad (2)$$

The result is a filter whose cutoff frequency can be varied continuously between $0.07 f_N$ and $0.50 f_N$ by changing x_1.

IV. Gradient estimator design

Discrete difference gradient estimators are not robust to noise. Catté, Lions, Morel and Coll [15] address this by smoothing the gradient estimate with a Gaussian lowpass filter, which is known to be optimal in this regard [16]. However, the high noise power at high frequencies and strong idle tones characteristic of error diffused halftones necessitate a different filter. We use a lowpass filter with the characteristics described in Section III, which performs better than the Gaussian. To improve robustness to noise further, we estimate gradients at two scales and correlate the results. Large, sharp edges appear across scales, whereas noise does not [17]. We found that gradient estimation at two scales gave the best performance for the test images used. The filter specifications are as follows:

- Line zeros at $(-, 0)$, $(f_N, -)$, and $(-, f_N)$
- Maximum stopband gain of 0.03
- Peak passband gain of 1
- Narrowest possible passband for a given filter size

The specifications on the line zeros and the maximum stopband gain arise from the considerations of Section III. The peak passband gain is defined to be unity, so the filter output range is known. The filter passband is made as narrow as possible to best distinguish between the two scales.

Each filter is separable. In the direction in which gradients are estimated, the filter is bandpass, with zeros at DC and the Nyquist frequency. The free parameters are chosen to give the narrowest passband possible, subject to the maximum stopband gain being 0.030. In the direction perpendicular to the direction of gradient estimation, the filter is lowpass, with the lowest possible cutoff frequency for the filter size to maximize noise rejection.

Since the peak passband gain of the filters is known, we can find fast integer implementations. We scaled each filter by a power of two and rounded the coefficients to fit into one byte. Since the halftone is binary, only integer additions are needed to compute the output of each filter. We denote the four filters h_x^{small}, h_y^{small}, h_x^{large}, and h_y^{large}, where the superscripts 'small' and 'large' refer to the scale.

V. Correlation across scales

At each pixel of the input image, we filter the halftone with the four gradient estimators to produce outputs e_x^{small}, e_y^{small}, e_x^{large}, and e_y^{large}. To correlate the gradients across scales, we compute the control functions according to

$$e_x^{\text{comp}} = \left| e_x^{\text{small}} \times e_x^{\text{large}} \times e_x^{\text{large}} \right|^{1/3} , \tag{3}$$

where $|\cdot|$ denotes absolute value. (The expression for the y control function is analogous.) We weight the large-scale gradient more heavily than the small-scale gradient to suppress small-scale noise. This produces slightly smoother, better quality inverse halftones than equal weighting. Since each gradient estimator is linear, its output is proportional to its input. The product in (3) is therefore proportional to the cube of the gradient. We find its cube root so that the control function varies linearly with the gradient.

We quantify the accuracy of the gradient images obtained from the halftone by computing their signal-to-noise ratio (SNR) relative to the gradients obtained from the grayscale image. We found that the small-scale images, e_x^{small} and e_y^{small}, have an average SNR of approximately 2.9 dB. The large amount of quantization noise in the small-scale images computed from the halftone leads to the low SNR figure; however, the images are sharp. The large-scale images, e_x^{large} and e_y^{large}, have an average SNR of approximately 11 dB. However, they are not as sharp as the small-scale images. The control functions have an average SNR of approximately 8.1 dB, an improvement of more than 5 dB over the small-scale figure. Furthermore, they are sharp. By correlating across scales, we obtain most of the noise rejection of the large-scale gradient image, while retaining the sharpness of the small-scale image.

VI. Construction of the inverse halftone

The control functions, e_x^{comp} and e_y^{comp}, are used to determine the cutoff frequencies of the lowpass filter in the x and y directions independently. We require a relation between e_x^{comp} and x_1. To reduce computation, we use a linear relation (the y relation is analogous):

$$x_1 = a + b\, e_x^{\text{comp}} . \tag{4}$$

We determined values for a and b by varying them while monitoring the visual quality of test images. The best results were achieved when $a = 3.33$ and $b = -5.7$.

Once x_1 has been computed, we derive x_2 using (2), and construct the prototype filter according to (1), ignoring for the moment the factor of $1/(4x_2 + 8)$. Each coefficient is a floating-point number in the approximate range $(-0.5, 4)$. We scale each coefficient by the factor 1024 (2^{10}), and convert it to an integer by discarding the fractional part. This results in at most a 13-bit signed integer, apart from the fixed central coefficient, which is 14-bit. By using 13-bit coefficients, the filtered result has a wordlength less than or equal to 32 bits, which is a common integer wordlength for general purpose hardware. The coefficient quantization has no measurable effect on the final results.

The x and y prototype filters are applied separably to the 7×7 neighborhood centered on the current pixel. (At the boundaries of the image, three pixels are replicated by mirroring to simplify the filtering.) Applying the filters separably obviates the need to construct the equivalent two-dimensional filter, saving 42 integer multiplications per pixel over a non-separable implementation.

The filtered output pixel is converted to a `float` and scaled. The scaling simultaneously accounts for the ignored factor $1/(4x_2 + 8)$ from (1) (and the corresponding factor from the y filter), the scaling factor used in converting the filter coefficients to integers, and the requirement that the output pixels be in the range $(0, 255)$. Clipping enforces this range, before the pixel is rounded to the nearest integer and converted to an `unsigned char` (single byte).

VII. Computation and memory requirements

The following operations are required per pixel:

- 303 increments (`++`)
- 30–226 integer additions
- 7 integer multiplications
- 34 floating-point additions
- 19 floating-point multiplications
- 5 floating-point divisions

The number of integer additions depends on the image. A halftone composed solely of black pixels would require 30 integer additions per pixel, whereas an all-white halftone would require 226. A typical mid-gray image requires approximately 128 integer additions. We list the increment operator separately, because some hardware can perform this operation with zero time penalty. The number of floating-point operations, particularly divisions, has been minimized for speed. For an image of size 512×512 pixels, the entire inverse halftoning process takes 2.9 seconds to execute on a 167 MHz Sun UltraSparc 2 machine, and 6.8 seconds on a Sparc 10.

Execution proceeds in raster fashion, one row at a time. Seven image rows are required for the filters; they are kept in the *image storage area*, a pre-allocated array of memory

Algorithm, Reference	Memory usage	Complexity	PSNR (dB)	
			lena	*peppers*
POCS [3]	$8N^2$	High	30.4	–
Bayes [7]	$8N^2$	High	–	–
Kernel est. [5]	$8N^2$	Med.	32.0	30.2
Wavelet [6]	$36N^2$	Med.	31.5	30.4
Proposed	$7N$	Low	31.3	31.4

TABLE I: Comparison of inverse halftoning schemes. Memory requirements are estimated in bytes, assuming an image size of $N \times N$ pixels. Computational complexity is estimated from algorithm information given in the cited papers. "Low" complexity denotes fewer than 500 operations per pixel, "medium" denotes 500–2000 operations per pixel, and "high" denotes more than 2000 operations per pixel. PSNR figures are taken directly from the publications, where available.

of size $7\,(c+6)$ bytes, where c is the number of image columns. (There are 6 more columns in the storage area than in the image itself, because of the mirroring extension of 3 pixels at the image boundaries.) The image pixels themselves take up one byte each. For an image of size 512×512 pixels, 3626 bytes of memory are allocated for image storage.

After an entire row has been inverse halftoned, rows 2–7 of the image storage area are moved upwards into the positions occupied by rows 1–6, and a new image row is written into the row 7 position. If circular buffering were available (as on programmable digital signal processors), the block move could be avoided. However, the time penalty due to the move is small, because of the small block size, and because only one shift is needed for each row.

VIII. Results

Figures 2(a) and 2(b) show the original *lena* and *peppers* images, respectively. Figures 2(c) and 2(d) show the corresponding Floyd-Steinberg halftones. Figures 2(e) and 2(f) show the inverse halftones computed using the proposed algorithm. Both inverse halftones display a range of sharp edges and smooth regions. The edges are particularly sharp, and close inspection shows them to be sharper and more realistic than those produced by the wavelet scheme described in [6].

Table I shows that the proposed algorithm uses by far the least memory of any scheme for reasonable image sizes, since it is the only scheme whose memory requirement increases linearly with N, rather than quadratically. Furthermore, it does not store copies of the image, as iterative schemes do. The computational complexity of the proposed algorithm is also considerably lower than the other schemes, all of which make heavy use of floating-point arithmetic. Nevertheless, the PSNR achieved for the standard images is comparable to the best schemes. (The large improvement in PSNR for the *peppers* image is due in part to an error in the original image. This error was corrected for this work, and was reported to the authors of [6].)

IX. Conclusion

We have presented a fast inverse halftoning method that produces results comparable to other methods reported in the literature. When compared to existing methods, our scheme has the lowest computation and memory requirements for the same visual quality. Its simple implementation, using only local operations, makes it attractive for practical applications and low-cost devices such as facsimile machines. The small wordlength, integer additions required for the gradient estimation make the algorithm ideally suited for efficient parallel implementations on the Intel MMX architecture.

Currently, we are exploring visual quality measures designed specifically for inverse halftoning schemes, in a manner similar to that proposed for halftoning schemes in [10]. Our results indicate that PSNR is particularly inappropriate for inverse halftones.

References

[1] D. Neuhoff and T. Pappas, "Perceptual coding of images for halftone display," *IEEE Trans. Image Processing*, vol. 3, pp. 1–13, Jan. 1994.

[2] M. Ting and E. Riskin, "Error-diffused image compression using a binary-to-grayscale decoder and predictive pruned tree-structured vector quantization," *IEEE Trans. Image Processing*, vol. 3, pp. 854–858, Nov. 1994.

[3] S. Hein and A. Zakhor, "Halftone to continuous-tone conversion of error-diffusion coded images," *IEEE Trans. Image Processing*, vol. 4, pp. 208–216, Feb. 1995.

[4] Y. Kim, G. Arce, and N. Grabowski, "Inverse halftoning using binary permutation filters," *IEEE Trans. Image Processing*, vol. 4, pp. 1296–1311, Sept. 1995.

[5] P. Wong, "Inverse halftoning and kernel estimation for error diffusion," *IEEE Trans. Image Processing*, vol. 4, pp. 486–498, Apr. 1995.

[6] Z. Xiong, M. Orchard, and K. Ramchandran, "Inverse halftoning using wavelets," *Proc. IEEE Conf. Image Processing*, pp. 569–572, Sept. 1996.

[7] R. Stevenson, "Inverse halftoning via MAP estimation," *IEEE Trans. Image Processing*, vol. 6, pp. 574–583, Apr. 1997.

[8] R. Ulichney, *Digital Halftoning*. Cambridge, MA: MIT Press, 1987.

[9] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 12, pp. 629–639, July 1990.

[10] T. Kite, B. L. Evans, A. C. Bovik, and T. Sculley, "Digital halftoning as 2-D delta-sigma modulation," *Proc. IEEE Conf. Image Processing*, vol. 1, pp. 799–802, Oct. 1997.

[11] R. Floyd and L. Steinberg, "An adaptive algorithm for spatial grayscale," *Proc. Soc. Image Display*, vol. 17, no. 2, pp. 75–77, 1976.

[12] Z. Fan and R. Eschbach, "Limit cycle behavior of error diffusion," *Proc. IEEE Conf. Image Processing*, vol. 2, pp. 1041–1045, Nov. 1994.

[13] J. Jarvis, C. Judice, and W. Ninke, "A survey of techniques for the display of continuous tone pictures on bilevel displays," *Computer Graphics and Image Processing*, vol. 5, pp. 13–40, 1976.

[14] T. Huang, J. Burnett, and A. Deczky, "The importance of phase in image processing filters," *IEEE Trans. Acoustics, Speech, and Signal Processing*, vol. 23, pp. 529–542, Dec. 1975.

[15] F. Catté, P.-L. Lions, J.-M. Morel, and T. Coll, "Image selective smoothing and edge detection by nonlinear diffusion," *SIAM J. Numerical Analysis*, vol. 29, pp. 182–193, Feb. 1992.

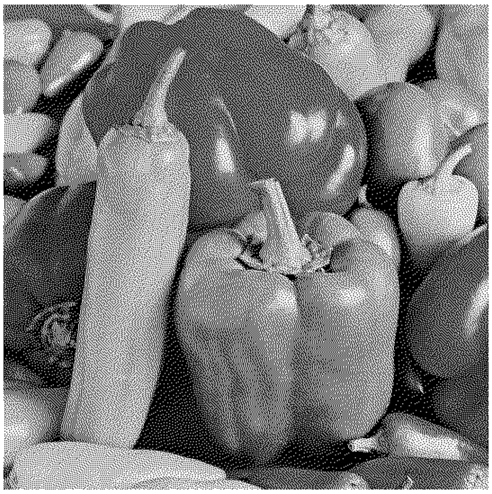
[16] D. Marr and E. Hildreth, "The theory of edge detection," *Proc. Royal Society of London. Series B: Biological Sciences*, vol. 207, pp. 187–217, 1980.

[17] S. Mallat and S. Zhong, "Characterization of signals from multiscale edges," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, pp. 710–732, July 1992.

(a) Original *lena* image.

(b) Original *peppers* image.

(c) Floyd-Steinberg halftone.



(d) Floyd-Steinberg halftone.

(e) Inverse halftone. PSNR = 31.34 dB.

(f) Inverse halftone. PSNR = 31.43 dB.

FIGURE 2: Inverse halftoning results of proposed algorithm. Images are 512×512 pixels in size.