Hardware-Friendly Descreening

Hasib Siddiqui, Mireille Boutin, and Charles A. Bouman, *Fellow, IEEE*

***Abstract*—Conventional electrophotographic printers tend to produce Moiré artifacts when used for printing images scanned from printed material such as books and magazines. We propose a novel noniterative, nonlinear, and space-variant descreening filter that removes a wide range of Moiré-causing screen frequencies in a scanned document while preserving image sharpness and edge detail. This filter is inspired by Perona-Malik's anisotropic diffusion equation. The amount of diffusion of the image intensity resulting from applying the filter is governed by an edge intensity estimate that is robust under halftone noise. More precisely, the filter extracts a spatial feature vector comprising local intensity gradients estimated from a local window in a presmoothed version of the noisy input image. Tunable nonlinear polynomial functions of this feature vector are then used to perform one iteration of a discrete diffusion controlled by the intensity gradient. The polynomial functions and feature extraction kernels are selected empirically in order to minimize computation while ensuring robust performance across a wide range of test images on a target imaging platform. The algorithm uses integer arithmetic, mostly relying on low-cost bit-wise shift and addition operations, and uses a strictly sequential architecture to provide a cost-effective and robust descreening solution in practical imaging devices including copiers and multifunction printers. We compare the performance of the proposed algorithm to other descreening solutions and demonstrate that the new algorithm improves quality over the existing methods while reducing computation.**

***Index Terms*—Anisotropic diffusion, descreening, halftone, hardware-friendly algorithm, Moiré artifacts, resolution synthesis, SUSAN filter.**

I. Introduction

CLUSTERED dot screening [19] is a halftoning method which involves placing dots of varying sizes at regular grid intervals to simulate different shades of gray. It is the most commonly used halftoning method. Essentially all books, newspapers, and magazines are printed using clustered dot screening. Most electrophotographic multifunction printers (MFPs) also employ clustered dot screening as the default halftoning method for printing. When an MFP is used to scan-and-print a printed original, the periodic clustered dot screens in the scanned original and the printing process can beat against each other giving rise to undesirable artifacts called Moiré patterns [1].

A variety of different methods have been proposed in the literature to avoid Moiré patterns when printing images scanned from printed originals. The proposed techniques fall into two broad categories. The first approach is to employ a stochastic halftoning algorithm for printing scanned halftone content in the scan-to-print pipeline of the MFP [4], [7], [8], [10]. Stochastic halftoning uses a random placement of dots for simulating different gray levels, and is thus more resistant to Moiré resulting from interference of periodic screens [9].

The second approach, which we consider in this paper, is to first process the scanned image with a descreening algorithm that removes the aliasing screen frequencies and to then use the default clustered dot halftone screen in the MFP to print the descreened image. Perhaps the simplest way of suppressing halftone noise is to smooth the entire scanned document with a single low-pass filter; however, this approach also softens the image edges [21]. Important contributions in inverse halftoning and descreening research include wavelet-based descreening [11]–[13], [22], training-based descreening (TBD) [16], segmentation-based descreening [5], gradient-based adaptive filtering [6], inverse halftoning via MAP estimation [18], and inverse halftoning via projection methods [2], [21].

Among the wavelet-based techniques mentioned in the previous paragraph, wavelet-based inverse halftoning via deconvolution (WInHD) [12], [13] is a relatively recent contribution for performing inverse halftoning of binary error-diffused images. The method has been demonstrated to outperform some of the best inverse-halftoning algorithms published in the literature for inverse halftoning binary error diffusion images. However, WInHD is not useful for descreening grayscale scanned images containing clustered-dot halftone screens.

Most of the aforementioned algorithms for inverse halftoning and descreening are ill-suited for hardware implementation. Indeed, in order to be hardware friendly, an algorithm should avoid floating-point arithmetic and should not entail the evaluation of complicated functions (e.g., exponentials). Moreover, it should involve as few as possible fixed-point multiplications and divisions, as these demand a large gate count and a high level of logic on an application specific integrated circuit (ASIC). Large look-up tables (LUTs) should also be avoided as they add to the hardware burden. Finally, and most importantly, the image processing should be *data independent*, in the sense that each pixel should be processed the same way using a single data-flow path so to minimize the chip area and the underlying hardware cost.

In this paper, we develop a hardware-friendly nonlinear filtering technique for descreening together with an optional deblurring of digital scanned documents. The proposed algorithm, which we refer to as hardware-friendly descreening (HFD), functions by first extracting a spatial feature vector comprising the intensity gradients computed at different pixel locations in a small neighborhood of the given pixel. The feature vector is extracted from presmoothed versions of the scanned image. The spatial feature vector is then used in an algebraic framework that depends on a combination of nonlinear

Manuscript received August 01, 2009; revised March 25, 2009. First published October 30, 2009; current version published February 18, 2010. This work was supported by Hewlett-Packard, Inc. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Dimitri Van De Ville.

The authors are with the School of Electrical and Computer Engineering, Purdue University, West Lafayette, IN 47907 USA (e-mail: hsiddiqu@purdue.edu; mboutin@ecn.purdue.edu; bouman@ecn.purdue.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2009.2035238

polynomial functions to compute the local filter coefficients required for directional smoothing. Once the local filter coefficients have been determined, the value of the descreened image at a given pixel location is computed as the weighted average of pixel values in a local window of the scanned image around the given pixel.

We compare HFD to TBD [16] which is an effective way to remove a wide range of clustered dot screen frequencies, as well as error diffusion noise, while preserving edges. Architecturally, HFD and TBD are very dissimilar; however, the two algorithms share one fundamental idea: both methods rely on prefiltered renderings of the noisy scanned image to extract local spatial features that determine the kernel coefficients for nonlinear spatial smoothing. Whereas the training-based strategy uses the spatial feature vector in a preoptimized statistical framework for deciding the local kernel weights, the new HFD algorithm uses preselected polynomial functions to fix the local kernel coefficients.

The HFD filter behaves as a Gaussian filter in smooth regions of the image, effectively suppressing the aliasing halftone screen frequencies, and can function as a sharpening filter in nonsmooth regions, enhancing the quality of edges and detail regions. The descreening algorithm is fast and is designed to satisfy the following criteria for hardware-friendly implementation:

- uses a strictly sequential architecture, avoiding classification-based processing of the input data for which the process pipeline in digital hardware is pixel value dependent;
- avoids division operations and computation of exponentials or other functions which are costly to compute in hardware;
- uses only integer additions, subtractions, multiplications, and bit-wise shift operations for computing the output pixel value;
- uses small local windows to limit the required number of input line buffers in an ASIC;
- relies on symmetric and 2-D separable kernels for reduced computation and, hence, reduced gate count and levels of digital logic.

The spatial processing performed by the proposed HFD algorithm is described with a single nonlinear algebraic equation that takes the spatial feature vector and the local pixel values in the scanned image as the input, and outputs the denoised value of the current pixel. The algebraic form of this filtering algorithm is similar to that of the iterative Perona and Malik anisotropic diffusion equation [14]. In each iteration step, the Perona and Malik filter smooths regions of uniform intensity in the image while preserving large intensity variations due to the image edges. The number of iteration steps required for noise suppression is typically large when the noise variance is high. The novelty of the proposed HFD filter is that it relies on a noise-robust numerical scheme for approximating the local edge intensity, which enables the algorithm to provide robust noise suppression in just one iteration step.

The rest of the paper is organized as follows. In Section II, we review the Perona-Malik anisotropic diffusion equation framework, which will facilitate a more thorough understanding of our following discussion on hardware friendly descreening. The HFD filter is described in Section III, while the experimental results are presented in Section IV. The concluding remarks and directions for future research are discussed in Section V. Note that a preliminary version of this work in the form of a conference paper has been appeared in [15].

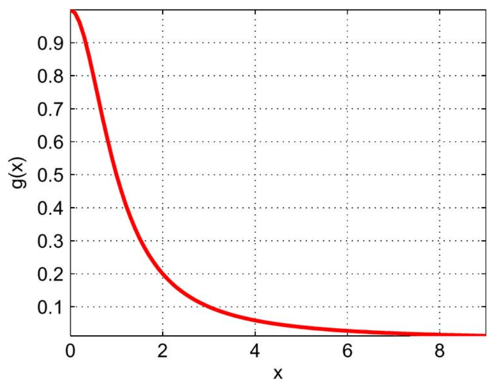


Fig. 1. Edge-stopping function $g(x) = 1/(1 + x^2)$ proposed by Perona and Malik.

II. Background

A. Anisotropic Diffusion

Linear low-pass spatial filters based on Gaussian functions are widely used for noise suppression in image processing applications. It has been shown in the literature [20] that the effect of Gaussian filtering can also be achieved by processing an image according to an isotropic diffusion described by the Laplace equation. Specifically, if $u(x,y,0)$ denotes the initial noisy image at (x,y), the spatial position in a continuous domain, and t the time parameter, then the image $u(x,y,t)$ evolving according to the partial differential equation

$$\frac{\partial u(x,y,t)}{\partial t} = \text{div}\left\{\nabla u(x,y,t)\right\} = \triangle u(x,y,t) \qquad (1)$$

can also be obtained by filtering $u(x,y,0)$ using a Gaussian kernel $G_{\sqrt{t}}(x,y)$ with variance $\sqrt{t}$; that is

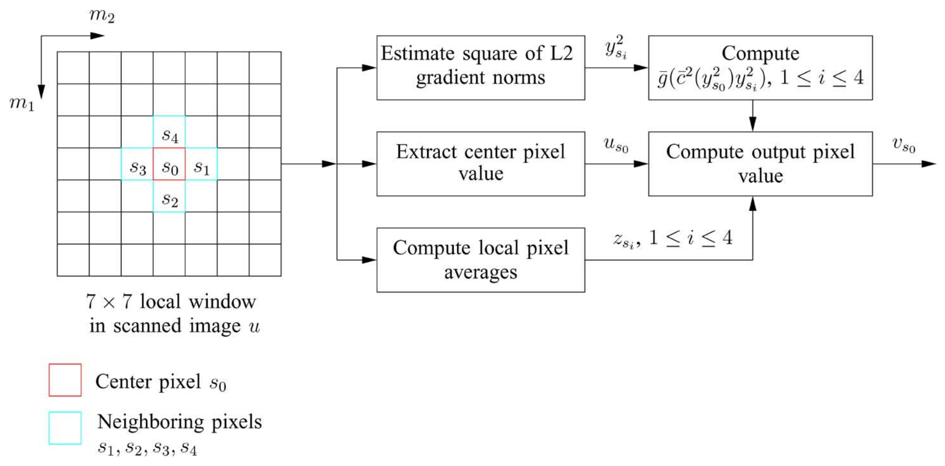
$$u(x,y,t) = \int\int G_{\sqrt{t}}(x-\mu, y-\nu)u(\mu,\nu,0)\mathrm{d}\mu\mathrm{d}\nu \qquad (2)$$

where $\nabla u = (u_x, u_y)$ denotes the gradient of u, $\triangle u = u_{xx} + u_{yy}$ denotes the Laplacian of u, and the kernel $G_{\sqrt{t}}(x,y)$ is given by

$$G_{\sqrt{t}}(x,y) = \frac{1}{\sqrt{2\pi t}}\exp\left(-\frac{x^2+y^2}{2t}\right). \qquad (3)$$

Convolving an image with a Gaussian kernel can provide robust noise suppression. However, it blurs the edges and high frequency spatial detail in the image. In their seminal work on image scale-space representation [14], Perona and Malik introduced a nonlinear anisotropic diffusion process as a means of smoothing noise in degraded images while preserving the edge quality. The continuous form of the Perona-Malik anisotropic diffusion equation is

$$\frac{\partial u(x,y,t)}{\partial t} = \text{div}\left\{g\left(c\left\|\nabla u(x,y,t)\right\|\right)\nabla u\right\} \qquad (4)$$



m_2
m_1
s_4
s_3 s_0 s_1
s_2
7×7 local window in scanned image u
Center pixel s_0
Neighboring pixels s_1, s_2, s_3, s_4
Estimate square of L2 gradient norms
$y^2_{s_i}$
Compute $\bar{g}(\bar{c}^2(y^2_{s_0})y^2_{s_i})$, $1 \le i \le 4$
Extract center pixel value
u_{s_0}
Compute output pixel value
v_{s_0}
Compute local pixel averages
z_{s_i}, $1 \le i \le 4$

Fig. 2. Flow diagram of hardware-friendly descreening (HFD) algorithm. The HFD algorithm computes squares of local gradient magnitudes $y^2_{s_i}$ and local pixel averages z_{s_i} from pixels in the 7 × 7 local window around s_0 in the scanned image u. Using preselected functions $\bar{f}(x)$ and $\bar{g}(x)$, the output descreened pixel value is computed as $v_{s_0} = u_{s_0} + (1/4)\sum_{i=1}^{4} \bar{g}(\bar{c}^2(y^2_{s_0})y^2_{s_i})(z_{s_i} - u_{s_0})$.

where $\|\nabla u\|$ denotes the L_2 norm of the gradient and $\triangle u$ denotes the Laplacian $\triangle u = u_{xx} + u_{yy}$. The function $g(x)$ is typically a nonincreasing function of x and is referred to as the *edge-stopping function*; it encourages diffusion in smooth regions of the image, allowing noise suppression, and discourages diffusion across the edges, preserving the image sharpness and detail. The parameter c, typically referred to as the *contrast parameter*, controls the decay rate of the edge-stopping function. An example of an edge-stopping function, which was proposed by Perona and Malik, is [14]

$$g(x) = \frac{1}{1+x^2}. \tag{5}$$

A plot of $g(x)$ is shown in Fig. 1.

Diffusion is an energy-dissipating process. Specifically, it has been demonstrated in the literature [3], [23] that smoothing an image u according to an isotropic (1) or anisotropic (4) partial differential equation is equivalent to finding the minimum of the energy functional

$$E(u) = \int_{\Omega} \rho\left(\|\nabla u\|\right) d\Omega \tag{6}$$

where Ω is the domain of the image. The function $\rho(x)$ determines the shape of the energy functional and, hence, the behavior of the diffusion process. In general, $\rho(x)$ is related to the edge-stopping function $g(x)$ through the equation [3]

$$g(x) = \frac{\rho'(x)}{x}. \tag{7}$$

In an isotropic diffusion process, the edge-stopping function is unity everywhere, i.e., $g(x) = 1\ \forall x$. Hence, for this case, $\rho(x) = x^2$. In anisotropic diffusion smoothing, it is typically required that, for large values of x, the function $\rho(x)$ increase less rapidly than x^2. Specifically, for the Perona and Malik edge-stopping function given in (5), the function $\rho(x)$ is given by [3]

$$\rho(x) = \frac{1}{2}\log(1+x^2) \tag{8}$$

which increases as $O(\log|x|)$.

B. Discrete Formulation of Anisotropic Diffusion

In order to discretize (4), we sample the signal $u(x,y,t)$ at uniform time intervals τ and uniform space intervals h, both in the x and y directions. We shall assume that the positive direction of x is to the right and the positive direction of y is downward. Let $t_n = n\tau$, $y_i = ih$, and $x_j = jh$, where n, i, and j are non-negative integers, denote the coordinates in time and space where the signal $u(x,y,t)$ is sampled. Let u^n_s denote the intensity value of the image at discrete time index n and at pixel $s = (i,j)$ in the resulting discrete, rectangular, 2-D grid. Denote by $s_0 = (i_0, j_0)$ the current pixel, and let $s_1 = (i_0 - 1, j_0)$, $s_2 = (i_0, j_0 + 1)$, $s_3 = (i_0 + 1, j_0)$, and $s_4 = (i_0, j_0 - 1)$ be the 4 neighboring pixels of s_0. As we show in the Appendix, the Perona-Malik diffusion equation can be discretized as

$$u^{n+1}_{s_0} = u^n_{s_0} + \frac{\tau}{h^2}\sum_{i=1}^{4} g\left(c\mu^n_{\frac{s_i+s_0}{2}}\right)\left(u^n_{s_i} - u^n_{s_0}\right) \tag{9}$$

where $\mu^n_{(s_i+s_0)/2}$ is any approximation of the norm of the gradient of the intensity $\|\nabla u\|$ at time index n halfway between pixel s_i and pixel s_0 that satisfies

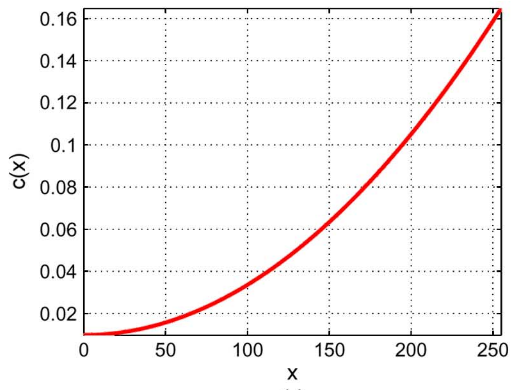
$$\begin{aligned}\mu^n_{\frac{s_3+s_0}{2}} - \|\nabla u\|^n_{\frac{s_3+s_0}{2}} &= \mu^n_{\frac{s_1+s_0}{2}} - \|\nabla u\|^n_{\frac{s_1+s_0}{2}} + O(h^2)\\ \mu^n_{\frac{s_2+s_0}{2}} - \|\nabla u\|^n_{\frac{s_2+s_0}{2}} &= \mu^n_{\frac{s_4+s_0}{2}} - \|\nabla u\|^n_{\frac{s_4+s_0}{2}} + O(h^2)\end{aligned}$$

for all n. For example, one can use the approximations

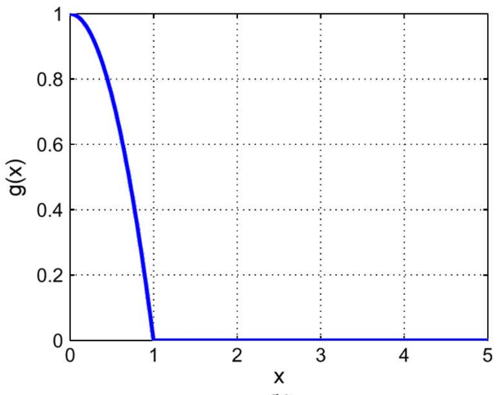
$$\begin{aligned}\mu^n_{\frac{s_1+s_0}{2}} &= \frac{\left\|\left(u^n_{s_0} - u^n_{s_1}, \frac{u^n_{s_2}-u^n_{s_4}}{2}\right)\right\|}{h}\\ &= \|\nabla u\|^n_{\frac{s_1+s_0}{2}} + O(h^2)\end{aligned} \tag{10}$$

$$\begin{aligned}\mu^n_{\frac{s_2+s_0}{2}} &= \frac{\left\|\left(\frac{u^n_{s_3}-u^n_{s_1}}{2}, u^n_{s_2} - u^n_{s_0}\right)\right\|}{h}\\ &= \|\nabla u\|^n_{\frac{s_2+s_0}{2}} + O(h^2)\end{aligned} \tag{11}$$

$$\begin{aligned}\mu^n_{\frac{s_3+s_0}{2}} &= \frac{\left\|\left(u^n_{s_3} - u^n_{s_0}, \frac{u^n_{s_2}-u^n_{s_4}}{2}\right)\right\|}{h}\\ &= \|\nabla u\|^n_{\frac{s_3+s_0}{2}} + O(h^2)\end{aligned} \tag{12}$$



(a)



(b)

Fig. 3. Plots of nonlinear functions (a) $c(x)$ and (b) $g(x)$ used in the hardware-friendly descreening algorithm. The functions $c(x)$ and $g(x)$ are, respectively, monotonically nondecreasing and nonincreasing functions of x.

$$\mu^n_{\frac{s_4+s_0}{2}} = \frac{\left\|\left(\frac{u^n_{s_3}-u^n_{s_1}}{2}, u^n_{s_0} - u^n_{s_4}\right)\right\|}{h} = \|\nabla u\|^n_{\frac{s_4+s_0}{2}} + O(h^2) \tag{13}$$

which meet the aforementioned criteria. In [14], Perona and Malik used an approximation of the above expression, namely

$$\mu^n_{\frac{s_i+s_0}{2}} = \frac{\left|u^n_{s_i} - u^n_{s_0}\right|}{h} \tag{14}$$

which actually corresponds to a slightly different continuous anisotropic diffusion equation, but the resulting diffusion is similar.

Assuming the noise variance is small in comparison with that of the signal components, the overall effect of processing an image according to (9) is to iteratively smooth the image while preserving its edges. For the purposes of illustration, consider setting $\mu^n_{(s_i+s_0)/2}$, for $i = 1,\ldots,4$, as in (10)–(13), respectively, in the discrete formulation of anisotropic diffusion (9). Then the pixel differences $|u^n_{s_i} - u^n_{s_j}|$ will have small values in nonedge regions of the image. Consequently, the values of the $\mu^n_{(s_i+s_0)/2}$ will be small and, in turn, the diffusion coefficients $g(c\mu^n_{(s_i+s_0)/2})$ will have large positive values. As a result, the local image region will be gradually smoothed over multiple iterations of the algorithm. In the vicinity of an edge, when some of the $|u^n_{s_i} - u^n_{s_j}|$ are large, the corresponding diffusion coefficients $g(c\mu^n_{(s_i+s_0)/2})$ are small, which stops diffusion of pixel values across the edge and preserves the local image structure.

However, when the noise variance is large in comparison with the signal components, as is the case when the image is a scanned printed document, a high value of the gradient of the intensity may not be due to the presence of an edge in the initial document but merely to the presence of noise in the image. As a consequence, Moiré patterns are preserved for many iterations of the above described numerical smoothing scheme. Our proposed descreening algorithm, described in the next section, addresses this issue.

III. Hardware Friendly Descreening Algorithm

The proposed HFD algorithm is a modification of the Perona and Malik filter (9) designed to provide robust noise suppression in just one iteration step of anisotropic diffusion. Specifically, for any input pixel u_{s_0}, the value of the output descreened pixel v_{s_0} can be written as

$$v_{s_0} = u_{s_0} + \frac{1}{4}\sum_{i=1}^{4} g\left(c\left(y_{s_0}\right) y_{s_i}\right)\left(z_{s_i} - u_{s_0}\right) \tag{15}$$

where y_{s_i} approximates the norm of the gradient of the image intensity $y_{s_i} \approx \|\nabla u\|_{(s_i+s_0)/2}$, $c(x)$ is a polynomial function used to adjust the contrast parameter as a function of local spatial gradient during diffusion, $g(x)$ is a piecewise polynomial edge-stopping function, and z_{s_i} is a pixel average in a neighborhood of the pixel s_i.

The computations of the y_i's and the z_{s_i}'s, which respectively correspond to different band-pass and low-pass filtered versions of u_s, will be discussed in Section III-B. The nonlinear functions $g(x)$ and $c(x)$ were empirically selected as follows:

$$c(x) = \frac{10}{1024}\left(1 + \frac{x^2}{64^2}\right) \tag{16}$$

and

$$g(x) = \begin{cases} 1 - x^2, & \text{if } 0 \le x < 1 \\ 0, & \text{if } x \ge 1 \end{cases}. \tag{17}$$

The plots of functions $c(x)$ and $g(x)$ are shown in Fig. 3(a) and (b), respectively.

Comparing (9) and (15), we observe the following similarities and differences between the HFD filter and Perona-Malik anisotropic diffusion.

In each iteration step, the Perona-Malik filter computes a weighted sum of center pixel, u_{s_0}, and its four nearest neighbors, u_{s_1}, u_{s_2}, u_{s_3}, and u_{s_4}, for averaging. To achieve the desired amount of screen suppression in just a single iteration step, the HFD filter uses pixels in a larger spatial neighborhood of s_0 for pixel averaging. Thus, pixel values $\{u_{s_i}\}_{i=1}^4$ in (9) are replaced with local spatial pixel averages $\{z_{s_i}\}_{i=1}^4$ in (15). The mean of the four pixel averages, i.e., $\sum_{i=1}^4 z_{s_i}/4$, corresponds to a Gaussian-weighted average of pixels in a rectangular window centered at s_0, optimized to provide a robust suppression of the scanned halftone noise.

In (9), the contrast parameter c is constant and is used to control the trade off between noise suppression and edge preservation in each iteration of Perona-Malik filter. In the HFD filter, we emphasize the role of c by making it a function of local statistics of the image, y_{s_0}. In smooth regions of the image $c(y_{s_0})$ is small, which favors noise suppression, while in edge regions $c(y_{s_0})$ is large, which favors edge preservation.

The HFD filter makes use of noise robust estimates, y_{s_i}, for local intensity gradients, $\mu_{(s_i+s_0)/2}$, to derive weights of the spatially adaptive filter. The original Perona-Malik approximation, $\mu^n_{(s_i+s_0)/2} = |u^n_{s_i} - u^n_{s_0}|$, does not provide a robust measure of edge-strength in a scanned image corrupted with halftone noise.

As in the case of the Perona and Malik diffusion (4), the function $g(x)$ in the HFD algorithm (15) controls the extent of diffusion of the pixel intensity: diffusion is large when $g(c(y_{s_0})y_{s_i})$ is large and vice versa. The spatially-adaptive contrast parameter $c(x)$ is used to distinguish between edge and noise oscillations in the local window. Specifically, when $y_{s_i} > 1/c(y_{s_0})$, where $1 \le i \le 4$, the HFD filter attributes the intensity oscillations to the presence of a local edge in the image and stops diffusion, i.e., $g(c(y_{s_0})y_{s_i}) = 0$. On the other hand, when $y_{s_i} < 1/c(y_{s_0})$, the intensity oscillations are attributed to the image noise; the diffusion coefficients $g(c(y_{s_0})y_{s_i})$ are then positive and the filter performs smoothing. The noise robustness of the resulting numerical scheme ensures that the filter achieves the required amount of noise suppression in just one iteration of diffusion.

Observe that both $c(x)$ and $g(x)$ only depend on the square of x. So, in order to avoid the computation of square roots in the gradient estimates, we define

$$\bar{g}(x) = \begin{cases} 1 - x, & \text{if } 0 \le x \le 1 \\ 0, & x > 1 \end{cases} \tag{18}$$

$$\bar{c}(x) = \frac{10}{1024}\left(1 + \frac{x}{64^2}\right). \tag{19}$$

Our algorithm can then be written as

$$v_{s_0} = u_{s_0} + \frac{1}{4}\sum_{i=1}^{4} \bar{g}\left(\bar{c}^2\left(y^2_{s_0}\right) y^2_{s_i}\right)\left(z_{s_i} - u_{s_0}\right). \tag{20}$$

In order to further reduce computation, we implement the functions $\bar{c}(x)$ and $\bar{g}(x)$ as LUTs. The sample values for $\bar{c}(x)$ are stored in a LUT comprising 128 address locations, with each address location storing 10 bits of integer data. The sample values for the function $\bar{g}(x)$, defined as in (17), are stored in a LUT with 512 9-bit sized integer values.

In the remainder of this section, we first discuss the computation of the square of the local intensity gradient estimates $y^2_{s_i}$ and pixel averages z_{s_i}. We then discuss how modifications to the nonlinear function $g(x)$ enable the HFD filter to deblur the scanned document during descreening.

A. Local Intensity Gradients

The gradient estimation step in the HFD algorithm involves estimating the gradients at the center pixel s_0 and at each of the four neighbors of s_0: s_1, s_2, s_3, and s_4. The supports and coefficients of the derivative estimation kernels are selected empirically based on their observed performance on test halftone images acquired from the target scanner at resolutions between 300 and 600 dpi. In addition, the coefficients of the derivative estimation filters are selected as either powers of two or sums of a few powers of two. This allows us to replace multiplications and divisions involved with kernel convolutions with bit-wise shift operations.

TABLE I
One-Dimensional Filters, h^a_m, h^b_m, g^a_m, and g^b_m Used to Construct Directional Derivative Masks $h^{(i)}_x$ and $h^{(i)}_y$

m	-3	-2	-1	0	1	2	3
h^a_m	1/16	2/16	3/16	4/16	3/16	2/16	1/16
g^a_m	-1/4	-1/4	-2/4	0	2/4	1/4	1/4
h^b_m	0	1/8	2/8	2/8	2/8	1/8	0
g^b_m	0	-1/4	-3/4	0	3/4	1/4	0

The derivative mask is band-pass in the direction of derivative estimation and low-pass in the direction perpendicular to that of derivative estimation. Each 2-D derivative mask is a separable kernel formed by the outer product of a 1-D band-pass filter, g^a_m or g^b_m, and a 1-D low-pass filter, h^a_m or h^b_m. The superscripts a and b are used to denote larger kernels of length 7 and smaller kernels of length 5, respectively. The weights for the 1-D kernels are shown in Table I. The kernel supports are dictated by a compromise between performance and implementation cost of the algorithm in the imaging harware. Large kernel supports are desired to ensure robustness of the edge-intensity estimates to the scanned halftone noise, while small kernel sizes are desired for a low computational cost and for keeping the number of expensive line buffers small in the hardware. In our hardware implementation, the kernel supports are limited to pixels in a 7×7 local window centered at the current pixel. Specifically, we use 7×7 kernels for computing the x- and y- derivatives at s_0; 7×5 kernels for computing the derivatives at the right and left neighbors of s_0, namely, s_1 and s_3; and 5×7 kernels for computing the derivatives at the top and bottom neighbors of s_0, namely, s_2 and s_4. Experimentally, we observed that smaller $5\times$ 5 square kernels do not provide sufficient noise suppression for a robust estimation of the local derivatives at the scanner resolutions of interest.

The 2-D derivative masks, denoted by $h^{(i)}_x$ and $h^{(i)}_y$, respectively, for estimating the local x- and y- directional derivatives at pixel locations s_i can be written in terms of the 1-D kernels as follows:

$$\begin{aligned}
\left(h^{(0)}_x\right)_{m_1,m_2} &= h^a_{m_1} g^a_{m_2}, & \left(h^{(0)}_y\right)_{m_1,m_2} &= g^a_{m_1} h^a_{m_2} \\
\left(h^{(1)}_x\right)_{m_1,m_2} &= h^a_{m_1} g^b_{m_2}, & \left(h^{(1)}_y\right)_{m_1,m_2} &= g^a_{m_1} h^b_{m_2} \\
\left(h^{(2)}_x\right)_{m_1,m_2} &= h^b_{m_1} g^a_{m_2}, & \left(h^{(2)}_y\right)_{m_1,m_2} &= g^b_{m_1} h^a_{m_2} \\
\left(h^{(3)}_x\right)_{m_1,m_2} &= h^a_{m_1} g^b_{m_2}, & \left(h^{(3)}_y\right)_{m_1,m_2} &= g^a_{m_1} h^b_{m_2} \\
\left(h^{(4)}_x\right)_{m_1,m_2} &= h^b_{m_1} g^a_{m_2}, & \left(h^{(4)}_y\right)_{m_1,m_2} &= g^b_{m_1} h^a_{m_2}
\end{aligned}$$

where m_1 indexes the row dimension of the image and points down, while m_2 indexes the column dimension of the image and points to the right. Using the derivative masks $h^{(i)}_x$ and $h^{(i)}_y$, the

x- and y directional derivatives at pixel s_i, where $0 \le i \le 4$, are determined as the following convolutions of the filters and input image u_s

$$(u_x)_{s_i} = \sum_{\boldsymbol{m}} \left(h_x^{(i)}\right)_{\boldsymbol{m}} u_{s_i+\boldsymbol{m}} \tag{21}$$

$$(u_y)_{s_i} = \sum_{\boldsymbol{m}} \left(h_y^{(i)}\right)_{\boldsymbol{m}} u_{s_i+\boldsymbol{m}} \tag{22}$$

where the summation is over the region of support of each derivative mask and $\boldsymbol{m} = (m_1, m_2)$ is a vector representing the 2-D spatial coordinates in the image. Once we have the estimates of the directional derivatives $(u_x)_{s_i}$ and $(u_y)_{s_i}$, the L_2-gradient norms y_{s_i} are estimated using the following equation:

$$y_{s_i}^2 = (u_x)_{s_i}^2 + (u_y)_{s_i}^2. \tag{23}$$

The symmetry and separability of the derivative estimation masks $h_x^{(i)}$ and $h_y^{(i)}$ can be exploited for an efficient computation of the local derivatives. As mentioned above, each derivative mask is separable and can be expressed as an outer product of a 1-D symmetric low-pass filter, h_m^a or h_m^b, and a 1-D anti-symmetric band-pass filter, g_m^a or g_m^b. Thus, each 2-D convolution operation in (21) and (22) can be performed efficiently by first convolving the image with a 1-D filter in the m_1 direction, and then convolving the filtered result with another 1-D filter in the m_2 direction.

Furthermore, as we can observe from Table I, the 1-D kernel coefficients used for gradient estimation are selected as either powers of two or sums of a few powers of two. Thus, the multiplication operations involving the filter coefficients h_m^a, h_m^b, g_m^a, and g_m^b can be replaced with hardware-efficient bit-wise shift and addition operations.

Thus, using the kernel coefficients in Table I and exploiting the separability of the derivative masks, the local directional derivatives $\{(u_x)_{s_i}\}_{i=0}^4$ and $\{(u_y)_{s_i}\}_{i=0}^4$ in the HFD algorithm can be estimated efficiently using only a total of approximately 50 bit-wise shifts and 70 additions per pixel of the input image.

B. Local Pixel Averages

The local pixel averages z_{s_1}, z_{s_2}, z_{s_3}, and z_{s_4} denote average values of pixels in the 7×7 local window centered at s_0 computed in the four different locations about the center pixel. The pixel averages are determined by dividing the local window around s_0 into four overlapping triangular regions and then computing the average of pixel values in each of these four regions. To compute the local pixel averages, we first select a 7 $\times$ 7 low-pass filter, which will be denoted by $h_{\boldsymbol{m}}$, where

$$h_{\boldsymbol{m}} = h_{m_1}^a h_{m_2}^a; |m_1|, |m_2| \le 3 \tag{24}$$

$$= \frac{1}{256} \cdot \begin{bmatrix} 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ 2 & 4 & 6 & 8 & 6 & 4 & 2 \\ 3 & 6 & 9 & 12 & 9 & 6 & 3 \\ 4 & 8 & 12 & 16 & 12 & 8 & 4 \\ 3 & 6 & 9 & 12 & 9 & 6 & 3 \\ 2 & 4 & 6 & 8 & 6 & 4 & 2 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 \end{bmatrix} \tag{25}$$

and h_m^a is the 1-D low-pass filter defined in Table I.

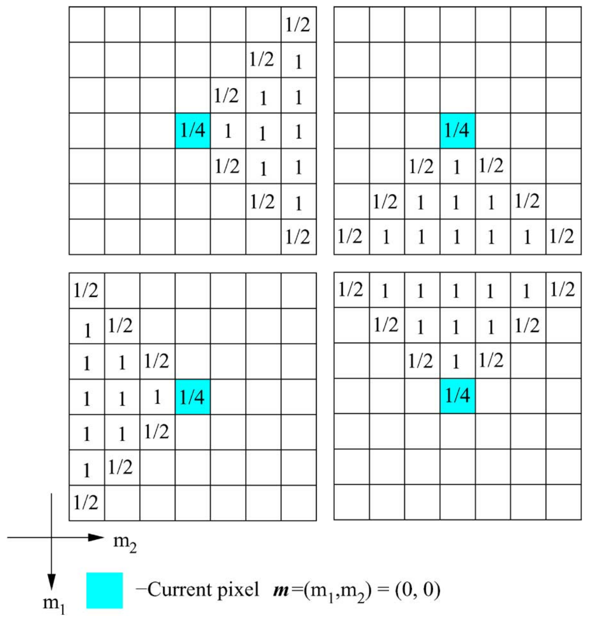


Fig. 4. Two-dimensional weighting masks $w_m^{(i)}$ for computing local pixel averages z_{s_i}. Top left: $w_m^{(1)}$; top right: $w_m^{(2)}$; bottom left: $w_m^{(3)}$; and bottom right: $w_m^{(4)}$.

We next define a set of four 7×7 triangular masks, as shown in Fig. 4. The masks are denoted by $w_{\boldsymbol{m}}^{(i)}$, where $i = 1, 2, 3$, and 4. Using these masks, the directional pixel averages z_{s_i} are computed as

$$z_{s_i} = \frac{\sum_{\boldsymbol{m}} h_{\boldsymbol{m}} w_{\boldsymbol{m}}^{(i)} u_{s_0+\boldsymbol{m}}}{\sum_{\boldsymbol{m}} h_{\boldsymbol{m}} w_{\boldsymbol{m}}^{(i)}}. \tag{26}$$

Notice that the four weighting masks are chosen so that if $|m_1| \le 3$ and $|m_2| \le 3$, then $\sum_{i=1}^4 w_{\boldsymbol{m}}^{(i)} = 1$. Consequently, we have that $\sum_{i=1}^4 w_{\boldsymbol{m}}^{(i)} h_{\boldsymbol{m}} = h_{\boldsymbol{m}}$. Thus, in a smooth region of the image, where the local gradient norms are small, and the value of the function $g(x)$ in (15) is approximately equal to 1, the output of the HFD filter v_{s_0} is given by

$$v_{s_0} = \sum_{i=1}^4 \frac{z_{s_i}}{4} \tag{27}$$

$$= \frac{\sum_{i=1}^4 \sum_{\boldsymbol{m}} h_{\boldsymbol{m}} w_{\boldsymbol{m}}^{(i)} u_{s_0+\boldsymbol{m}}}{4 \sum_{\boldsymbol{m}} h_{\boldsymbol{m}} w_{\boldsymbol{m}}^{(i)}} \tag{28}$$

$$= \frac{\sum_{\boldsymbol{m}} h_{\boldsymbol{m}} u_{s_0+\boldsymbol{m}}}{\sum_{\boldsymbol{m}} h_{\boldsymbol{m}}}. \tag{29}$$

This is equivalent to applying a low-pass blurring filter in smooth regions of the image, which efficiently attenuates the halftone noise. If there is an edge passing through the local window, only those components z_{s_i} that lie on the same side of the edge as the center pixel will contribute to the output pixel value v_{s_0}, thereby preserving the edge strength and the local image structure.

The coefficients of the triangular masks $w_{\boldsymbol{m}}^{(i)}$ and the low-pass filter $h_{\boldsymbol{m}}$ are selected so that multiplication and division operations in (26) can be implemented with fast bit-wise shift and addition operations. The computation of the four pixel averages

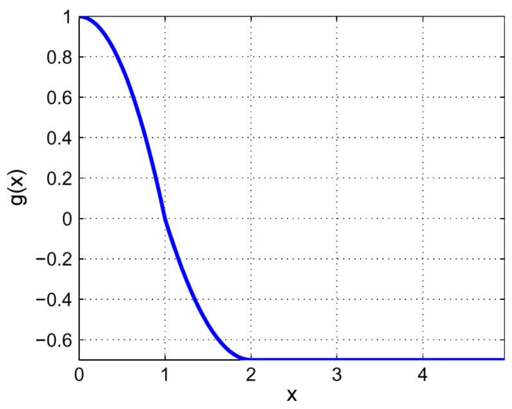


Fig. 5. Plot of nonlinear function $g(x)$, defined in (30), used in the hardware-friendly descreening algorithm for built-in sharpening. The plotted function assumes $\lambda = 0.65$.

z_{s_i}, where $1 \le i \le 4$, in our current HFD implementation requires a total of approximately 80 addition and 60 bit-wise shift operations.

C. Image Sharpening With HFD

The HFD framework can also be exploited for image sharpening by allowing the nonlinear function $g(x)$ in (17) to be negative. An example function $g(x)$ that enables simultaneous deblurring and descreening of scanned documents is

$$g(x) = \begin{cases} 1 - x^2, & \text{if } 0 \le x < 1 \\ -\lambda\left[1 - (2-x)^2\right], & \text{if } 1 \le x < 2 \\ -\lambda, & \text{if } x \ge 2 \end{cases} \tag{30}$$

where $\lambda > 0$ controls the extent of added sharpness. This new function $g(x)$ is plotted in Fig. 5.

As before, for nonedge regions, when the gradient magnitudes $\{y_{s_i}\}_{i=0}^{4}$ (see Section III-A) are small, the values of $g(c(y_{s_0})y_{s_i})$ in (15) are positive, and, therefore, the output pixel is determined as a positively weighted average of neighboring input pixels, which results in blurring of halftone screen. However, when the local gradient magnitudes are high, specifically when $c(y_{s_0})y_{s_i} > 1$, the value of $g(c(y_{s_0})y_{s_i}) < 0$, and, hence, the coefficients of the HFD filter are locally negative, resulting in local contrast enhancement. To see this more clearly, consider the special case where the local edge strength is sufficiently high so that $g(c(y_{s_0})y_{s_i}) = -\lambda$ for $i = 1, 2, 3,$ and 4. The output pixel value v_{s_0}, using (15), is then given by

$$v_{s_0} = u_{s_0} + \lambda\left[u_{s_0} - \sum_{i=1}^{4} \frac{z_{s_i}}{4}\right] \tag{31}$$

$$= u_{s_0} + \lambda\left[u_{s_0} - \frac{\sum_{\boldsymbol{m}} h_{\boldsymbol{m}} u_{s_0+\boldsymbol{m}}}{\sum_{\boldsymbol{m}} h_{\boldsymbol{m}}}\right] \tag{32}$$

which is the familiar equation for the unsharp mask with λ as the sharpness-control parameter. Thus, the proposed HFD filter adapts itself spatially, behaving as a low-pass filter in smooth regions and as an unsharp mask in edge regions. Of course, for the filter to give the desired performance, the edge intensity estimates must be robust to the halftone noise in the image.

Fig. 6. Thumbnails of sample test documents to evaluate the performance of hardware-friendly descreening.

IV. Experimental Results

The performance of the descreening algorithm was evaluated on ten 300-dpi and ten 600-dpi test scanned images. The 300-dpi test images were obtained by scanning printed originals on an HP ScanJet 8250 scanner and an HP LaserJet CM1015 multifunction printer, while the 600-dpi images were obtained by scanning the originals on an HP LaserJet 4730 multifunction printer. The printed originals were obtained from a number of different sources, including various newspapers and magazines, and were representative of a variety of clustered dot halftone screens used in practice. The thumbnails of some of the test scanned documents are shown in Fig. 6.

In the following two sections, we compare the HFD filter to other image descreening algorithms both in terms of resulting image quality and computational resources used by the algorithm. In Section IV-B, we focus on the built-in image sharpening capability of the HFD algorithm.

A. HFD Results: Comparison With Existing Image Denoising Algorithms

We use four different filtering algorithms for performance comparison in this subsection: (1) Space-invariant linear low-pass filter; (2) SUSAN filter [17]; (3) Training-based descreening (TBD) [16]; and (4) HFD algorithm (without sharpening). For linear space-invariant low-pass filtering, we use the filter $h_{\boldsymbol{m}}$, discussed in Section III-B. This filter is approximately Gaussian with $\sigma = 1.7$. For the SUSAN filter, we select: $\text{mask size} = 7 \times 7$, $\sigma_s = 1.7$, and $\sigma_b = 21$ (see [16] for details of these parameters). The TBD algorithm works by making use of a presmoothed image to derive the weights

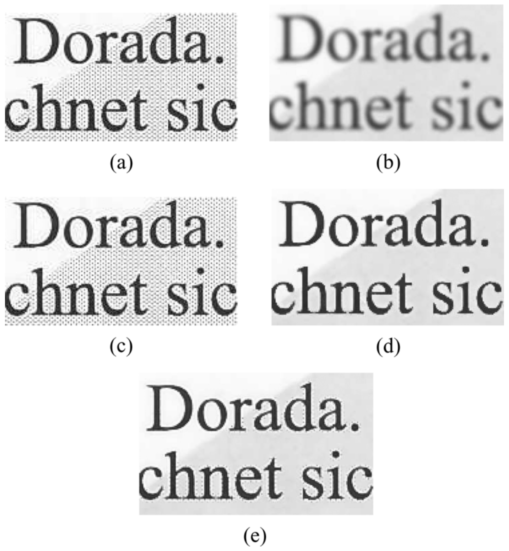


Fig. 7. (a) Scanned image, 300-dpi (Source printer: HP LaserJet 4050, Source scanner: HP ScanJet 8250). Descreened images using (b) Gaussian filter (7 × 7, $\sigma = 1.7$), (c) SUSAN filter (masks size: 7 × 7, $\sigma_s = 1.7, \sigma_b = 21$), (d) TBD, and (e) HFD.

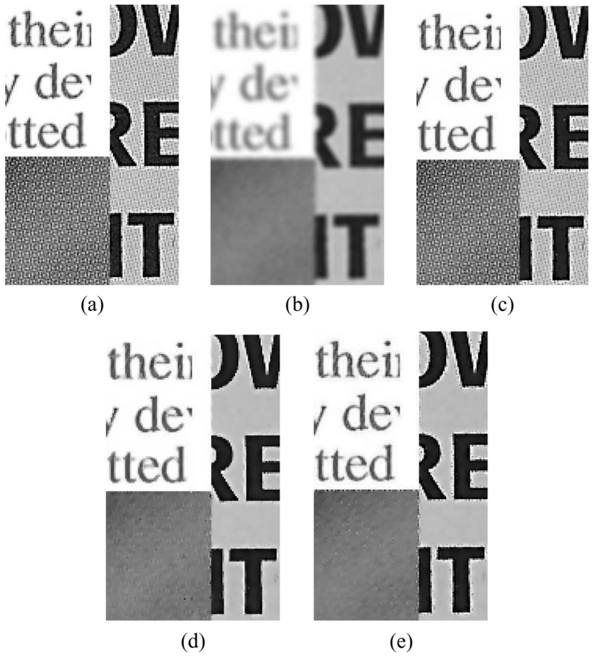


Fig. 8. (a) Scanned image, 300-dpi (Printed original scanned on HP LaserJet CM1015 MFP). Descreened images using (b) Gaussian filter (7 × 7, $\sigma = 1.7$), (c) SUSAN filter (masks size: 7 × 7, $\sigma_s = 1.7$, $\sigma_b = 21$), (d) TBD, and (e) HFD.

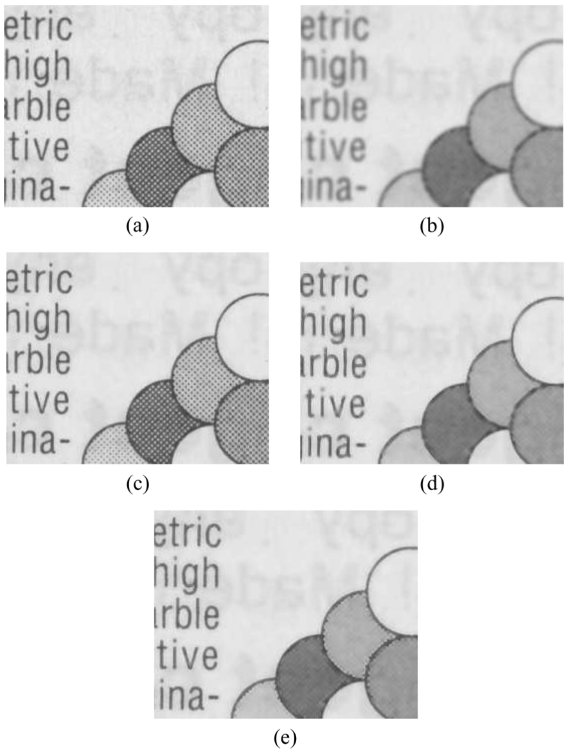


Fig. 9. (a) Scanned image, 600-dpi (Magazine page scanned on HP LaserJet 4730 MFP). Descreened images using (b) Gaussian filter (7 × 7, $\sigma = 1.7$), (c) SUSAN filter (masks size: 7 × 7, $\sigma_s = 1.7$, $\sigma_b = 21$), (d) TBD, and (e) HFD. Observe that the fine details in characters such as "r", "t", and "i" are better preserved in Fig. 9(e) than in Fig. 9(d).

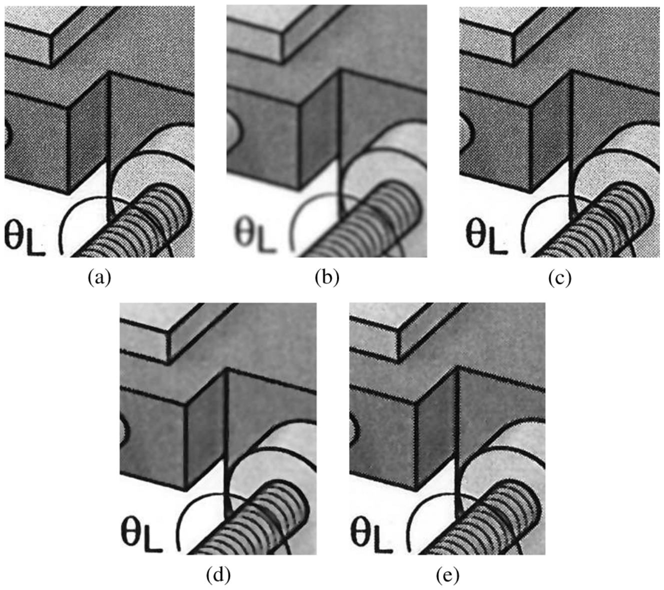


Fig. 10. (a) Scanned image, 600-dpi (Source printer: HP LaserJet 5500, Source scanner: HP ScanJet 8250). Descreened images using (b) Gaussian filter (7 × 7, $\sigma = 1.7$), (c) SUSAN filter (masks size: 7 × 7, $\sigma_s = 1.7, \sigma_b = 21$), (d) TBD, and (e) HFD.

of the SUSAN filter. For presmoothing, TBD uses a nonlinear stochastic predictor, termed as resolution synthesis denoising (RSD), that is described in [16]. The SUSAN filter parameters, namely, mask size, σ_s, and σ_b, are the same as stated above. The classification and filter parameters of the RSD predictor are trained offline and stored in a LUT comprising 8908 address locations, where each address location stores 32-bits of floating point data.

Figs. 7 and 8 show the descreening results for 300-dpi scanned images, while Figs. 9 and 10 show the results for 600-dpi scanned images. In Figs. 7–10(a), we show close-up views from different regions of the unprocessed scanned images in our test image database. As noted in the captions of these figures, the images are obtained by scanning printed test documents, collected from different print sources, on a variety

TABLE II
COMPARISON OF REQUIRED NUMBER OF OPERATIONS PER PIXEL IN TRAINING-BASED DESCREENING (TBD) AND HARDWARE-FRIENDLY DESCREENING (HFD) ALGORITHMS. THE INDICATED OPERATIONS ARE FLOATING-POINT FOR TBD AND FIXED-POINT FOR HFD

Operations	TBD (all float-point ops.)	HFD (all fixed-point ops.)
Additions	603	181
Multiplications	410	18
Divisions	20	0
Bitwise shifts	0	117
Exponentiations	50	0
If-Else comparisons	50	0
Total operations	1133	316

of different scanners. Figs. 7–10(b) show the results of processing the images with the linear space-invariant low-pass filter. While low-pass filtering is very effective in removing the aliasing screen frequencies, it also blurs nonhalftone image structure in the image. Figs. 7–10(c) show the processed results with the edge-preserving SUSAN filter. The SUSAN filter has been demonstrated to work well in the presence of low-variance noise [17], but when the noise strength becomes comparable to the edge strength, as is the case here, the SUSAN filter does not work. The filter processes the halftone noise as if it were high frequency detail in the image, retaining the noise pattern. However, as can be observed from the processed images in Figs. 7–10(d), the TBD algorithm, which relies on a presmoothed image for controlling the weights of the SUSAN filter, produces high quality descreening results. Finally, the descreened images using the proposed HFD algorithm with triangular kernels are presented in Figs. 7–10(e).

For both 300- and 600-dpi scanner resolutions, the HFD algorithm provides robust filtering of the scanned halftone noise. Comparing the output text in Figs. 9(d) and (e), we observe that the HFD algorithm performs slightly better in preserving the sharpness and quality of text than TBD. However, from Figs. 9 and 10, we do observe that TBD may provide slightly improved suppression of the halftone dot structure in smooth regions of the image compared to the HFD filter. The HFD filter tends to leave a thin layer of unsmoothed halftone dots along the image edges. This is because, in the vicinity of the edges, when the local gradient magnitudes y_{s_i} are large, the values of the decaying nonlinear edge-stopping function, $g(c(y_{s_0})y_{s_i})$, are small, which prevents smoothing of the dots and preserves the local halftone texture. However, it should be noticed that the images shown in Figs. 7–10 are zoomed-in views of small portions of full-page documents. When the full-page descreened document is printed at a resolution of 300- or 600-dpi, the residual noise along the edges is typically not objectionable.

To evaluate the performance of the HFD algorithm with varying sizes of scanned halftone dots, we used a 600-dpi scanned document specially designed to contain a variety of clustered-dot halftone screens: 200 lpi, 175 lpi, 150 lpi, 120 lpi, 106 lpi, 85 lpi, 65 lpi, and 45 lpi. Higher screen frequencies correspond to smaller halftone dot sizes and vice versa. The descreening algorithm worked well with all of the above halftone screens. The processed images could be printed on the target electro photographic printer, for which the algorithm was optimized, without any Moiré artifacts.

TABLE III
AVERAGE RUN TIMES TO DESCREEN 300-dpi AND 600-dpi LETTER-SIZED SCANNED DOCUMENTS. THE TRAINING-BASED DESCREENING (TBD) AND HARDWARE-FRIENDLY DESCREENING (HFD) ALGORITHMS WERE RUN ON A LINUX SERVER WITH 3.20 INTEL XEON CPU

Document Resolution	TBD (sec.)	HFD (sec.)
300-dpi	21.5	2.4
600-dpi	73.5	8.3

Table II compares the total number of operations per pixel required by the TBD and HFD algorithms. We see that the HFD algorithm mostly relies on hardware-efficient bit-wise shift and addition operations. The new algorithm completely avoids the use of division and exponentiation operations that demand a large gate count and are expensive to implement in an ASIC chip. Also, unlike the TBD algorithm, HFD avoids the use of an *if-else* construct and processes each pixel the same way using a single data-flow path. The use of a strictly sequential architecture in the design of the new algorithm also helps reduce the chip area and the underlying cost of the imaging hardware.

In a software implementation, HFD algorithm provides a speed-up factor of approximately 9 over TBD. The average times required by the two algorithms on a Linux server with 3.20 Intel Xeon CPU to descreen a letter-sized document scanned at 300-dpi and 600-dpi scanner resolutions are shown in Table III. The run-time of an algorithm shown in the table is determined as the average time required by the algorithm to descreen 20 different scanned documents of a specified resolution.

B. HFD Results: Built-In Image Sharpening

The HFD algorithm can provide image sharpening alongside screen suppression if we allow the nonlinear function $g(x)$ to be negative. Equation (30) is an example of such a function, where the sharpness level can be controlled using the parameter λ.

The built-in sharpening feature of the HFD algorithm is demonstrated in Figs. 11 and 12. Figs. 11 and 12(a) show cropped regions from two different test scanned images. The results in Figs. 11 and 12(b) are produced using training-based descreening. Figs. 11 and 12(c)–(f) show the processed results using HFD with four different values of λ: $\lambda = 0$, $\lambda = 0.5$ $\lambda = 1.0$, and $\lambda = 2.0$. For $\lambda = 0$, the sharpening is turned off, while larger values of λ correspond to increased levels of

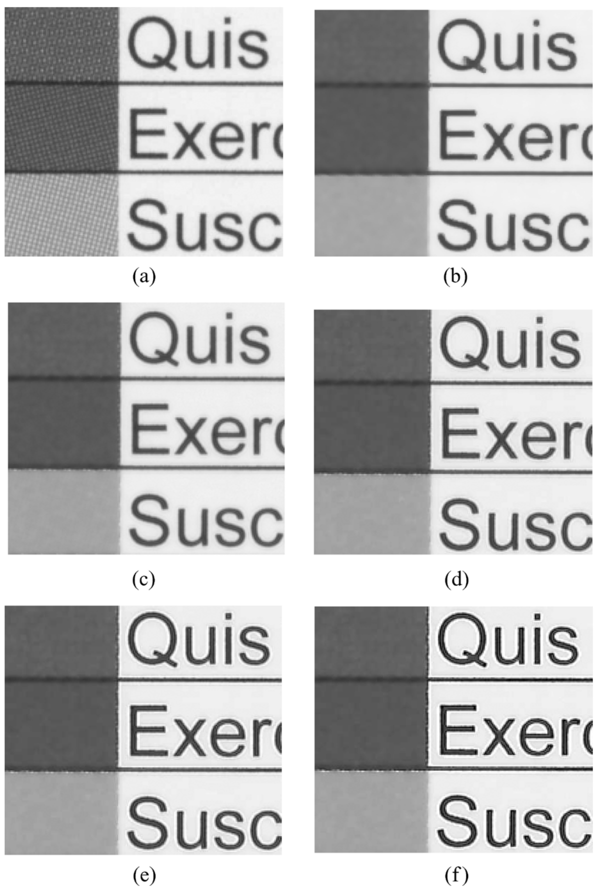


Fig. 11. (a) Scanned image, 600-dpi (Printed original scanned on HP LaserJet 4730 MFP). (b) Descreened image using TBD. Descreened images using HFD with variable levels of sharpening: (c) sharpness level $\lambda = 0$, (d) sharpness level $\lambda = 0.5$, (e) sharpness level $\lambda = 1.0$, and (f) sharpness level $\lambda = 2.0$.

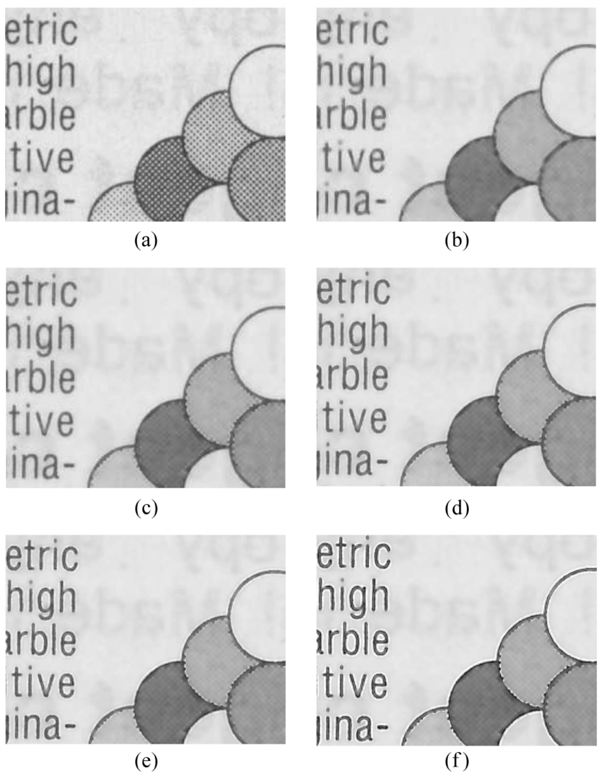


Fig. 12. (a) Scanned image, 600-dpi (magazine page printed on HP LaserJet 4730 MFP). (b) Descreened image using TBD. Descreened images using HFD with variable levels of sharpening: (c) sharpness level $\lambda = 0$, (d) sharpness level $\lambda = 0.5$, (e) sharpness level $\lambda = 1.0$, and (f) sharpness level $\lambda = 2.0$.

sharpening. The sharpening is probably too aggressive when $\lambda = 2.0$, producing aliasing artifacts inside and around the text. However, the results demonstrate that the HFD filter can achieve a high level of sharpening without exaggerating the halftone noise in the scanned image.

V. Conclusions and Future Works

We developed an efficient descreening algorithm targeted for applications such as scan-to-print in a multifunction printer. The algorithm is specifically designed with a hardware-friendly architecture to keep the implementation cost low in a practical imaging system. The experimental results demonstrate the effectiveness of the method for suppressing clustered-dot halftone noise in smooth regions of a scanned document while preserving the edge detail and text quality. The descreening performance of the proposed algorithm depends on a couple of nonlinear polynomial functions that are optimized empirically for a target imaging platform. Our future research will focus on designing a training methodology for automatic optimization of these nonlinear functions based on training image data collected from any given target multifunction printer.

Appendix
Discretization of Perona and Malik Diffusion Equation

In this section, we prove that Perona-Malik's equation

$$\frac{\partial u(x,y,t)}{\partial t} = \operatorname{div}\left(g\left(\|\nabla u\|\right)\nabla u\right) \tag{33}$$

can be discretized as

$$u_{s_0}^{n+1} = u_{s_0}^n + \frac{\tau}{h^2}\sum_{i=1}^{4} g\left(\mu^n_{\frac{s_i+s_0}{2}}\right)\left(u_{s_i}^n - u_{s_0}^n\right) \tag{34}$$

as stated in Section III.

Since τ is the time interval between time index n and $n+1$, one can write

$$\left.\frac{\partial u(x,y,t)}{\partial t}\right|_{s_0} = \frac{u_{s_0}^{n+1} - u_{s_0}^n}{\tau} + O(\tau). \tag{35}$$

Comparing the result with our target (34), one can see that it is sufficient to prove the following equality:

$$\frac{1}{h^2}\sum_{i=1}^{4} g\left(\mu^n_{\frac{s_i+s_0}{2}}\right)\left(u_{s_i}^n - u_{s_0}^n\right) = \operatorname{div}\{g\left(\|\nabla u\|\right)\nabla u\}\Big|_{s_0}^n + O(h). \tag{36}$$

Consider the first term of the left-hand side sum

$$\begin{aligned} g\left(\mu^n_{\frac{s_1+s_0}{2}}\right)(u^n_{s_1}-u^n_{s_0}) &= g\left(\mu^n_{\frac{s_1+s_0}{2}}\right)\left(-h\left.\frac{\partial u}{\partial x}\right|^n_{s_0}+\frac{h^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}+O(h^3)\right)\\ &= g\left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}+\mu^n_{\frac{s_1+s_0}{2}}-\|\nabla u\|^n_{\frac{s_1+s_0}{2}}\right)\\ &\quad\times\left(-h\left.\frac{\partial u}{\partial x}\right|^n_{s_0}+\frac{h^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}+O(h^3)\right)\\ &= \left(g\left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}\right)+\mathcal{E}\right)\\ &\quad\times\left(-h\left.\frac{\partial u}{\partial x}\right|^n_{s_0}+\frac{h^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}+O(h^3)\right) \end{aligned}\tag{37}$$

where the error term $\mathcal{E}$ is

$$\begin{aligned}\mathcal{E} &= \left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}-\mu^n_{\frac{s_1+s_0}{2}}\right)g'(\|\nabla u\|)|^n_{\frac{s_1+s_0}{2}}\\ &\quad+O\left(\left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}-\mu^n_{\frac{s_1+s_0}{2}}\right)^2\right)\\ &= \left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}-\mu^n_{\frac{s_1+s_0}{2}}\right)g'(\|\nabla u\|)|^n_{\frac{s_1+s_0}{2}}+O(h^2)\\ &= \left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}-\mu^n_{\frac{s_1+s_0}{2}}\right)g'(\|\nabla u\|)|^n_{s_0}+O(h^2).\end{aligned}$$

One can write

$$\begin{aligned} g&\left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}\right)\\ &= g(\|\nabla u\|)|^n_{s_0}+\left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}-\|\nabla u\|^n_{s_0}\right)g'(\|\nabla u\|)|^n_{s_0}\\ &\quad+O\left(\left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}-\|\nabla u\|^n_{s_0}\right)^2\right)\\ &= g(\|\nabla u\|)|^n_{s_0}+\left(-\frac{h}{2}\left.\frac{\partial\|\nabla u\|}{\partial x}\right|^n_{s_0}\right)g'(\|\nabla u\|)|^n_{s_0}\\ &\quad+O(h^2)\\ &= g(\|\nabla u\|)|^n_{s_0}-\frac{h}{2}\left.\frac{\partial g(\|\nabla u\|)}{\partial x}\right|^n_{s_0}+O(h^2).\end{aligned}$$

Replacing this last expression into (37), we obtain

$$\begin{aligned} g&\left(\mu^n_{\frac{s_1+s_0}{2}}\right)(u^n_{s_1}-u^n_{s_0})\\ &= \left(g(\|\nabla u\|)|^n_{s_0}-\frac{h}{2}\left.\frac{\partial g(\|\nabla u\|)}{\partial x}\right|^n_{s_0}+\mathcal{E}\right)\\ &\quad\times\left(-h\left.\frac{\partial u}{\partial x}\right|^n_{s_0}+\frac{h^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}+O(h^3)\right)\\ &= g(\|\nabla u\|)|^n_{s_0}\left(-h\left.\frac{\partial u}{\partial x}\right|^n_{s_0}+\frac{h^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}+O(h^3)\right)\\ &\quad+\left(-\frac{h}{2}\left.\frac{\partial g(\|\nabla u\|)}{\partial x}\right|^n_{s_0}\right.\\ &\qquad\left.+\left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}-\mu^n_{\frac{s_1+s_0}{2}}\right)g'(\|\nabla u\|)|^n_{s_0}\right)\\ &\quad\times\left(-h\left.\frac{\partial u}{\partial x}\right|^n_{s_0}+\frac{h^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}+O(h^3)\right)\end{aligned}$$

which can be simplified into

$$\begin{aligned} g&\left(\mu^n_{\frac{s_1+s_0}{2}}\right)(u^n_{s_1}-u^n_{s_0})\\ &= g(\|\nabla u\|)|^n_{s_0}\left(-h\left.\frac{\partial u}{\partial x}\right|^n_{s_0}+\frac{h^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}\right)+\\ &\quad-\frac{h}{2}\left.\frac{\partial u}{\partial x}\right|^n_{s_0}\left(-h\left.\frac{\partial g(\|\nabla u\|)}{\partial x}\right|^n_{s_0}\right.\\ &\qquad\left.+\left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}-\mu^n_{\frac{s_1+s_0}{2}}\right)g'(\|\nabla u\|)|^n_{s_0}\right)\\ &\quad+O(h^3).\end{aligned}\tag{38}$$

In a similar fashion, the third term of the sum can be discretized as

$$\begin{aligned} g&\left(\mu^n_{\frac{s_3+s_0}{2}}\right)(u^n_{s_3}-u^n_{s_0})\\ &= g(\|\nabla u\|)|^n_{s_0}\left(h\left.\frac{\partial u}{\partial x}\right|^n_{s_0}+\frac{h^2}{2}\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}\right)\\ &\quad+\frac{h}{2}\left.\frac{\partial u}{\partial x}\right|^n_{s_0}\left(h\left.\frac{\partial g(\|\nabla u\|)}{\partial x}\right|^n_{s_0}\right.\\ &\qquad\left.+\left(\|\nabla u\|^n_{\frac{s_3+s_0}{2}}-\mu^n_{\frac{s_3+s_0}{2}}\right)g'(\|\nabla u\|)|^n_{s_0}\right)\\ &\quad+O(h^3).\end{aligned}\tag{39}$$

Adding (38) and (39) together, we get

$$\begin{aligned}\sum_{i=1,3}&g\left(\mu^n_{\frac{s_i+s_0}{2}}\right)(u^n_{s_i}-u^n_{s_0})\\ &= \frac{h}{2}\left(\|\nabla u\|^n_{\frac{s_3+s_0}{2}}-\mu^n_{\frac{s_3+s_0}{2}}-\left(\|\nabla u\|^n_{\frac{s_1+s_0}{2}}-\mu^n_{\frac{s_1+s_0}{2}}\right)\right)\\ &\quad\times g'(\|\nabla u\|)|^n_{s_0}\left.\frac{\partial u}{\partial x}\right|^n_{s_0}\\ &\quad+h^2\left(g\left(\|\nabla u\|^n_{s_0}\right)\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}+\left.\frac{\partial g(\|\nabla u\|)}{\partial x}\right|^n_{s_0}\left.\frac{\partial u}{\partial x}\right|^n_{s_0}\right)\\ &\quad+O(h^3).\end{aligned}\tag{40}$$

By assumption, $\|\nabla u\|^n_{(s_3+s_0)/2}-\mu^n_{(s_3+s_0)/2}-(\|\nabla u\|^n_{(s_1+s_0)/2}-\mu^n_{(s_1+s_0)/2})=O(h^2)$, therefore

$$\begin{aligned}\sum_{i=1,3}g\left(\mu^n_{\frac{s_i+s_0}{2}}\right)(u^n_{s_i}-u^n_{s_0}) &= h^2\left(g\left(\|\nabla u\|^n_{s_0}\right)\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}\right.\\ &\quad\left.+\left.\frac{\partial g(\|\nabla u\|)}{\partial x}\right|^n_{s_0}\left.\frac{\partial u}{\partial x}\right|^n_{s_0}\right)+O(h^3).\end{aligned}\tag{41}$$

By symmetry, we also have

$$\begin{aligned}\sum_{i=2,4}g\left(\mu^n_{\frac{s_i+s_0}{2}}\right)(u^n_{s_i}-u^n_{s_0}) &= h^2\left(g\left(\|\nabla u\|^n_{s_0}\right)\left.\frac{\partial^2 u}{\partial y^2}\right|^n_{s_0}\right.\\ &\quad\left.+\left.\frac{\partial g(\|\nabla u\|)}{\partial y}\right|^n_{s_0}\left.\frac{\partial u}{\partial y}\right|^n_{s_0}\right)+O(h^3).\end{aligned}\tag{42}$$

Adding (41) and (42), we obtain

$$\begin{aligned}
&\sum_{i=1}^{4} g\left(\mu^n_{\frac{s_i+s_0}{2}}\right)\left(u^n_{s_i}-u^n_{s_0}\right)\\
&= h^2\left(g\left(\|\nabla u\|^n_{s_0}\right)\left(\left.\frac{\partial^2 u}{\partial x^2}\right|^n_{s_0}+\left.\frac{\partial^2 u}{\partial y^2}\right|^n_{s_0}\right)\right.\\
&\qquad\left.+\left.\frac{\partial g\left(\|\nabla u\|\right)}{\partial x}\right|^n_{s_0}\left.\frac{\partial u}{\partial x}\right|^n_{s_0}+\left.\frac{\partial g\left(\|\nabla u\|\right)}{\partial y}\right|^n_{s_0}\left.\frac{\partial u}{\partial y}\right|^n_{s_0}\right)\\
&\quad + O(h^3)\\
&= h^2 \operatorname{div}\left\{g\left(\|\nabla u\|\right)\nabla u\right\}\Big|^n_{s_0} + O(h^3)
\end{aligned}$$

which demonstrates (36) and thus proves our claim.

ACKNOWLEDGMENT

The authors would like to thank P. Majewicz and B. El-Bakri for their valuable input and comments.

REFERENCES

[1] J. P. Allebach and B. Liu, "Analysis of halftone dot profile and aliasing in the discrete binary representation of images," *J. Opt. Soc. Amer.*, vol. 67, no. 9, pp. 1147–1154, 1977.

[2] M. Analoui and J. Allebach, "New results on reconstruction of continuous-tone from halftone," in *Proc. IEEE Int. Conf. Acoustics, Speech, Signal Processing*, March 23–26, 1992, vol. 3, pp. 313–316.

[3] M. J. Black, G. Sapiro, D. H. Marimont, and D. Heeger, "Robust anisotropic diffusion," *IEEE Trans. Image Process.*, vol. 7, no. 3, pp. 421–432, Mar. 1998.

[4] Z. He and C. A. Bouman, "AM/FM halftoning: Digital halftoning through simultaneous modulation of dot size and dot density," *J. Electron. Imag.*, vol. 13, no. 2, pp. 286–302, Apr. 2004.

[5] A. Jaimes, F. Mintzer, A. R. Rao, and G. Thompson, "Segmentation and automatic descreening of scanned documents," *Proc. SPIE*, vol. 3648, pp. 517–528, 1999.

[6] T. D. Kite, N. D. Venkata, B. L. Evans, and A. C. Bovik, "A fast, high-quality inverse halftoning algorithm for error diffused halftones," *IEEE Trans. Image Process.*, vol. 9, no. 9, pp. 1583–1592, Sep. 2000.

[7] D. Lau, G. R. Arce, and N. Gallagher, "Green-noise digital halftoning," *Proc. IEEE*, vol. 86, no. 12, pp. 2424–2444, Oct. 1998.

[8] D. Lau, G. R. Arce, and N. Gallagher, "Digital halftoning via green-noise masks," *J. Opt. Soc. Amer. A*, vol. 16, no. 7, pp. 1575–1586, 1999.

[9] D. L. Lau, A. Khan, and G. R. Arce, "Stochastic Moiré," in *IS&T Conf. Image Proc., Image Quality and Image Capture Sys. (PICS)*, Montreal, QC, Canada, Apr. 2001, pp. 96–100.

[10] R. Levien, "Output dependent feedback in error diffusion halftoning," in *Proc. IS&T 8th Int. Congr. Advances in Non-Impact Printing Technology*, Oct. 1992, vol. 76, pp. 280–282.

[11] J. Luo, R. de Queiroz, and Z. Fan, "A robust technique for image descreening based on the wavelet transform," *IEEE Trans. Signal Process.*, vol. 46, no. 4, pp. 1179–1184, Apr. 1998.

[12] R. Neelamani, R. Nowak, and R. Baraniuk, "Model-based inverse halftoning with wavelet-vaguelette deconvolution," in *Proc. IEEE Int. Conf. Image Processing*, Vancouver, BC, Canada, Sep. 10–13, 2000, vol. 3, pp. 973–976.

[13] R. Neelamani, R. Nowak, and R. Baraniuk, "WInHD: Wavelet-based inverse halftoning via deconvolution," *IEEE Trans. Image Process.* Sep. 2002 [Online]. Available: http://www.dsp.rice.edu/publications/, submitted for publication

[14] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 629–639, Jul. 1983.

[15] H. Siddiqui, M. Boutin, and C. A. Bouman, "Hardware friendly descreening," presented at the IEEE Int. Conf. Image Processing, San Diego, CA, Oct. 12–15, 2008.

[16] H. Siddiqui and C. A. Bouman, "Training-based descreening," *IEEE Trans. Image Process.*, vol. 16, no. 3, pp. 789–802, Mar. 2007.

[17] S. M. Smith and J. M. Brady, "SUSAN-A new approach to low level image processing," *Int. J. Comput. Vis.*, vol. 23, no. 1, pp. 45–78, 1997.

[18] R. L. Stevenson, "Inverse halftoning via MAP estimation," *IEEE Trans. Image Process.*, vol. 4, no. 4, pp. 486–498, Apr. 1997.

[19] J. C. Stoffel and J. F. Moreland, "A survey of electronic techniques for pictorial reproduction," *IEEE Trans. Commun.*, vol. 29, pp. 1898–1925, 1981.

[20] A. Witkin, "Scale space filtering," presented at the Int. Joint Conf. Artificial Intell., 1983.

[21] P. W. Wong, "Inverse halftoning and kernel estimation for error diffusion," *IEEE Trans. Image Process.*, vol. 4, no. 4, pp. 486–498, Apr. 1995.

[22] Z. Xiong, M. T. Orchard, and K. Ramchandran, "Inverse halftoning using wavelets," *IEEE Trans. Image Process.*, vol. 8, no. 10, pp. 1479–1483, Oct. 1999.

[23] Y. L. You, W. Xu, A. Tannenbaum, and M. Kaveh, "Behavioral analysis of anisotropic diffusion in image processing," *IEEE Trans. Image Process.*, vol. 5, no. 11, pp. 1539–1553, Nov. 1996.

Hasib Siddiqui received the B.S. degree in electrical engineering from NED University, Pakistan, in 2000, and the M.S. and Ph.D. degrees from Purdue University, West Lafayette, IN, in 2003 and 2007, respectively.

He is currently with Qualcomm, Inc., San Diego, CA. His research interests are in image and signal processing.

Mireille Boutin was born in Quebec, Canada. She received the B.Sc. degree in physics-mathematics from the University de Montreal, QC, Canada, and the Ph.D. degree in mathematics from the University of Minnesota, Minneapolis, under the direction of P. Olver.

After a postdoctorate with D. Mumford and David Cooper at Brown University, followed by a postdoctorate at the Max Planck Institute for Mathematics in the Sciences (Leipzig, Germany), she joined Purdue School of Electrical and Computer Engineering, West Lafayette, IN, as an Assistant Professor with a courtesy appointment in the Department of Mathematics. She is the director of the Computational Imaging Laboratory. Her current research interests include image and video processing, object recognition, automatic text translation, portable device applications, and computational mathematics.

Dr. Boutin is an editor of the journal *Applicable Algebra in Engineering, Communication and Computing*. She is the instigator of Project Rhea, a student-driven Purdue-wide collective learning project. She is the co-founder (with Kathryn Leonard) of the Rose Whelan Society, an organization for women graduate students and postdoctorate in mathematics/applied mathematics at Brown University. She is the recipient of the Eta Kappa Nu Outstanding Faculty Award (Fall 2007) and the Wilfred Duke Hesselberth Award for Teaching Excellence (2008). She is a member of SPIE, the AMS, and FoCM.

Charles A. Bouman (S'86–M'89–SM'97–F'01) received the B.S.E.E. degree from the University of Pennsylvania, Philadelphia, in 1981, the M.S. degree from the University of California, Berkeley, in 1982, and the Ph.D. degree in electrical engineering from Princeton University, Princeton, NJ, in 1989.

From 1982 to 1985, he was a full staff member at the Massachusetts Institute of Technology Lincoln Laboratory. In 1989, he joined the faculty of Purdue University, West Lafayette, IN, where he is a Professor with a primary appointment in the School of Electrical and Computer Engineering and a secondary appointment in the School of Biomedical Engineering. Currently, he is Co-Director of Purdue's Magnetic Resonance Imaging Facility located in Purdue's Research Park. His research focuses on the use of statistical image models, multiscale techniques, and fast algorithms in applications including tomographic reconstruction, medical imaging, and document rendering and acquisition.

Prof. Bouman is a Fellow of the IEEE, a Fellow of the American Institute for Medical and Biological Engineering (AIMBE), a Fellow of the society for Imaging Science and Technology (IS&T), a Fellow of the SPIE professional society, a recipient of IS&T's Raymond C. Bowman Award for outstanding contributions to digital imaging education and research, and a University Faculty Scholar of Purdue University. He was the Editor-in-Chief of the IEEE TRANSACTIONS ON IMAGE PROCESSING and a member of the IEEE Biomedical Image and Signal Processing Technical Committee. He has been a member of the Steering Committee for the IEEE TRANSACTIONS ON MEDICAL IMAGING and an Associate Editor for the IEEE TRANSACTIONS ON IMAGE PROCESSING and the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE. He has also been Co-Chair of the 2006 SPIE/IS&T Symposium on Electronic Imaging, Co-Chair of the SPIE/IS&T Conferences on Visual Communications and Image Processing 2000 (VCIP), a Vice President of Publications and a member of the Board of Directors for the IS&T Society, and he is the founder and Co-Chair of the SPIE/IS&T Conference on Computational Imaging.