

# HARDWARE-FRIENDLY DESCREENING

Hasib Siddiqui, Mireille Boutin, and Charles A. Bouman

School of Electrical and Computer Engineering  
Purdue University, West Lafayette, IN 47907-1285, USA

## ABSTRACT

Conventional electrophotographic printers tend to produce Moiré artifacts when used for printing images scanned from printed material such as books and magazines. Inspired by anisotropic diffusion, we propose a novel non-iterative, non-linear, 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. The amount of diffusion of the image intensity resulting from applying the filter is governed by an estimate of the gradient 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 pre-smoothed version of the noisy input image. Tunable non-linear polynomial functions of this feature vector are then used to perform one iteration of a discrete diffusion controlled by the intensity gradient. 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**— Halftone, Moiré artifacts, descreening, anisotropic diffusion.

## 1. INTRODUCTION

Clustered dot screening [1] 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 screen in the scanned original and the screen used in the printing process can beat against each other giving rise to undesirable artifacts called Moiré patterns [2].

One approach to avoid Moiré artifacts is to first process the scanned image with a descreening algorithm that efficiently removes the aliasing halftone screen frequencies. Important contributions in the descreening work include wavelet-based descreening [3], training-based descreening [4], gradient-based adaptive filtering [5], inverse halftoning via MAP estimation [6], and inverse halftoning via projection methods [7].

Unfortunately, most of the above mentioned methods 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 fixed-point multiplications and divisions as possible since these demand a large gate

count on an application specific integrated circuit (ASIC). Large look-up tables (LUTs) should also be avoided. 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 propose a novel non-linear descreening filter, referred to herein as hardware friendly descreening (HFD), that meets the criteria for hardware efficient implementation stated in the previous paragraph. The space-variant filtering procedure is described by an algebraic equation, which was originally inspired by the Perona and Malik discrete formulation of anisotropic diffusion [8]. The Perona and Malik anisotropic diffusion filter is iterative. In each iteration step, the 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 intensity gradients, 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 2, we review the Perona-Malik anisotropic diffusion equation framework. The HFD filter is described in Section 3, while the experimental results are presented in Section 4. Finally, the concluding remarks are presented in Section 5.

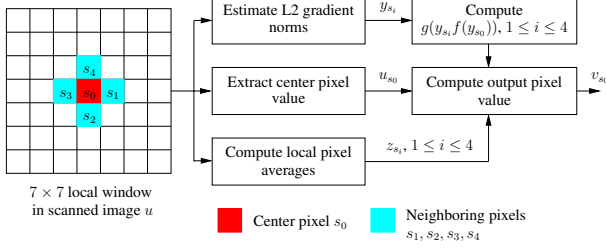
## 2. ANISOTROPIC DIFFUSION

In their seminal work on image scale-space representation [8], Perona and Malik introduced the non-linear anisotropic diffusion process as a means of smoothing noise in degraded images while preserving the edge quality. If  $u_s^n$  denotes the intensity value of the image at discrete time index  $n$  and at pixel  $s = (i, j)$  in a discrete, 2-D grid, one can discretize the Perona-Malik equation as follows:

$$u_{s_o}^{n+1} = u_{s_o}^n + \frac{\lambda}{4} \sum_{i=1}^4 g(|u_{s_i}^n - u_{s_o}^n|) (u_{s_i}^n - u_{s_o}^n), \quad (1)$$

where  $s_0 = (i_0, j_0)$  denotes the current pixel;  $s_1 = (i_0, j_0 + 1)$ ,  $s_2 = (i_0 + 1, j_0)$ ,  $s_3 = (i_0, j_0 - 1)$ , and  $s_4 = (i_0 - 1, j_0)$  denote the 4 neighboring pixels of  $s_0$ ; and  $\lambda \in [0, 1]$  is a constant that controls the diffusion rate. The function  $g(x)$  is typically a non-increasing, non-negative function of  $x$  and is referred to as the edge-stopping function [8].

The overall effect of the Perona and Malik algorithm (1) is to iteratively smooth the image while preserving its edges. Assuming the noise variance is small in comparison with that of the signal components, the absolute pixel difference  $|u_{s_i}^n - u_{s_o}^n|$  will have a small value in non-edge regions of the image. Consequently, the diffusion



**Fig. 1.** Flow diagram of hardware friendly descreening (HFD) algorithm. The HFD algorithm computes local gradient magnitudes  $y_{s_i}$  and local pixel averages  $z_{s_i}$  from pixels in the  $7 \times 7$  local window around  $s_0$  in the scanned image  $u$ . Using pre-selected non-linear functions  $f(x)$  and  $g(x)$ , the output descreened pixel value is computed as  $v_{s_0} = u_{s_0} + \frac{1}{4} \sum_{i=1}^4 g(y_{s_i} f(y_{s_0}))(z_{s_i} - u_{s_0})$ .

coefficients  $g(|u_{s_i}^n - u_{s_0}^n|)$  will have large positive values, and the local image region will be gradually smoothed over multiple iterations of the algorithm. In the vicinity of an edge, when  $|u_{s_i}^n - u_{s_0}^n|$  is large,  $g(|u_{s_i}^n - u_{s_0}^n|)$  is small, which stops diffusion of pixel values across the edge and preserves the local image structure.

### 3. HARDWARE FRIENDLY DESCREENING ALGORITHM

Fig. 1 shows the flow diagram of the HFD algorithm, where  $u$  denotes the input scanned image,  $s_0$  denotes the center pixel in the  $7 \times 7$  local window, and  $s_i$ , where  $1 \leq i \leq 4$ , denotes each of the four neighboring pixels of  $s_0$ . The input pixel value at pixel location  $s_i$  is denoted by  $u_{s_i}$ .

The algorithm works by extracting a noise-robust spatial feature vector that characterizes the edge strength and edge orientation in the neighborhood of the center pixel. The spatial feature vector comprises five components:  $y_{s_0}, y_{s_1}, \dots, y_{s_4}$ , where  $y_{s_i}$  represents the  $L_2$  norm of the gradient vector evaluated at pixel  $s_i$ . The square of the  $L_2$  norm is defined as

$$y_{s_i}^2 = (u_x)_{s_i}^2 + (u_y)_{s_i}^2, \quad (2)$$

where  $(u_x)_{s_i}$  and  $(u_y)_{s_i}$ , respectively, denote the directional derivatives of  $u$  at  $s_i$  computed in the  $x$ - and  $y$ -directions.

Next, the HFD filter calculates local pixel averages  $z_{s_1}, z_{s_2}, z_{s_3}$ , and  $z_{s_4}$ , which, respectively, denote the average value of pixels located to the east, south, west, and north of the center pixel  $s_0$ . The computation of  $z_{s_i}$  will be described in Section 3.2.

Once the values of  $y_{s_i}$  and  $z_{s_i}$  have been determined, the hardware friendly descreening algorithm computes the value of the output pixel using the following equation:

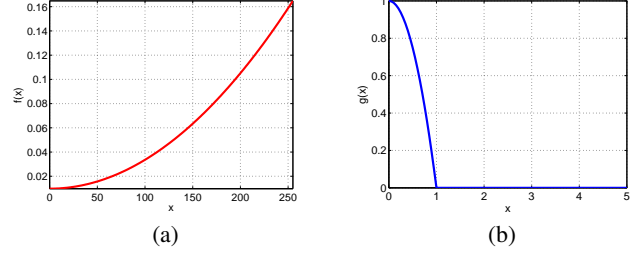
$$v_{s_0} = u_{s_0} + \frac{1}{4} \sum_{i=1}^4 g(y_{s_i} f(y_{s_0}))(z_{s_i} - u_{s_0}), \quad (3)$$

where  $g(x)$  and  $f(x)$  are non-linear functions of  $x$ . For the descreening results presented in this paper, these functions were empirically selected as follows:

$$f(x) = \frac{10}{1024} \left( 1 + \frac{x^2}{64^2} \right) \quad (4)$$

and

$$g(x) = \begin{cases} 1 - x^2, & \text{if } 0 \leq x < 1, \\ 0, & \text{if } x \geq 1. \end{cases} \quad (5)$$



**Fig. 2.** Plots of non-linear functions (a)  $f(x)$  and (b)  $g(x)$  used in the hardware friendly descreening algorithm. The functions  $f(x)$  and  $g(x)$  are, respectively, monotonically non-decreasing and non-increasing functions of  $x$ .

The plots of functions  $f(x)$  and  $g(x)$  are shown in Figs. 2 (a) and (b), respectively.

As in the case of the Perona-Malik equation (1), the function  $g(\cdot)$  in the HFD algorithm (3) controls the extent of diffusion of the pixel intensity: diffusion is large when  $g(y_{s_i} f(y_{s_0}))$  is large and vice versa.

The function  $f(\cdot)$  is used to compute a spatially adaptive threshold that distinguishes between edge and noise oscillations in the local window. Specifically, when  $y_{s_i} > 1/f(y_{s_0})$ , where  $1 \leq i \leq 4$ , the HFD filter attributes the intensity oscillations to the presence of a local edge in the image and stops diffusion, i.e.  $g(y_{s_i} f(y_{s_0})) = 0$ . On the other hand, when  $y_{s_i} < 1/f(y_{s_0})$ , the intensity oscillations are attributed to the image noise; the diffusion coefficients  $g(y_{s_i} f(y_{s_0}))$  are then positive and the filter performs smoothing.

The discrete formulation of Perona and Malik (1) uses absolute differences between adjacent and neighboring pixel values, i.e.,  $|u_{s_i} - u_{s_0}|$ , where  $i = 1, 2, 3$ , and  $4$ , for estimating the local gradient magnitudes. The proposed HFD algorithm (3) replaces the absolute pixel differences with noise-robust gradient estimates  $y_{s_i}$ , which expedites diffusion and, hence, smoothing in non-edge regions of the image.

Finally, as we shall see in Sec. 3.2, the use of pixel averages  $z_{s_i}$  instead of the original pixel values  $u_{s_i}$  in the HFD filter equation (3) is to ensure that the required amount of noise reduction is achieved in just one iteration of the HFD algorithm.

#### 3.1. Computation of Local Gradients

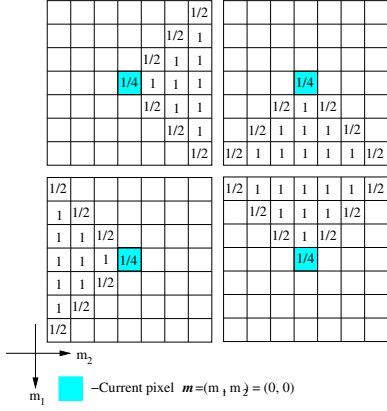
The gradient estimation step in the proposed hardware friendly descreening algorithm involves estimating the gradients at the center pixel  $s_0$  and at each of the four neighbors of  $s_0$ :  $s_1, s_2, s_3$ , and  $s_4$ . Each 2-D derivative mask is a separable kernel formed by the outer product of a 1-D band-pass filter,  $g_m^a$  or  $g_m^b$ , and a 1-D low-pass filter,  $h_m^a$  or  $h_m^b$ . The derivative mask is band-pass in the direction of derivative estimation and low-pass in the direction perpendicular to that of derivative estimation. The 1-D kernels are selected empirically to provide the required amount of noise suppression in the presence of scanned halftone noise. The weights for the 1-D kernels are shown in Table 1.

The 2-D derivative masks, denoted by  $h_x^{(i)}$  and  $h_y^{(i)}$ , 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:

$$\begin{aligned} (h_x^{(0)})_{m_1, m_2} &= h_{m_1}^a g_{m_2}^a, |m_1| \leq 3 \text{ and } |m_2| \leq 3; \\ (h_y^{(0)})_{m_1, m_2} &= g_{m_1}^a h_{m_2}^a, |m_1| \leq 3 \text{ and } |m_2| \leq 3; \end{aligned}$$

$m$	-3	-2	-1	0	1	2	3
$h_m^a$	1/16	2/16	3/16	4/16	3/16	2/16	1/16
$g_m^a$	-1/4	-1/4	-2/4	0	2/4	1/4	1/4
$h_m^b$	0	1/8	2/8	2/8	2/8	1/8	0
$h_m^b$	0	-1/4	-3/4	0	3/4	1/4	0

**Table 1.** One-dimensional filters,  $h_m^a$ ,  $h_m^b$ ,  $g_m^a$ , and  $g_m^b$  used to construct directional derivative masks  $h_x^{(i)}$  and  $h_y^{(i)}$ .



**Fig. 3.** 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:  $w_m^{(4)}$ .

$$\begin{aligned}
(h_x^{(1)})_{m_1, m_2} &= h_{m_1}^a g_{m_2}^b, |m_1| \leq 3 \text{ and } |m_2| \leq 2; \\
(h_y^{(1)})_{m_1, m_2} &= g_{m_1}^a h_{m_2}^b, |m_1| \leq 3 \text{ and } |m_2| \leq 2; \\
(h_x^{(2)})_{m_1, m_2} &= h_{m_1}^b g_{m_2}^a, |m_1| \leq 2 \text{ and } |m_2| \leq 3; \\
(h_y^{(2)})_{m_1, m_2} &= g_{m_1}^b h_{m_2}^a, |m_1| \leq 2 \text{ and } |m_2| \leq 3; \\
(h_x^{(3)})_{m_1, m_2} &= h_{m_1}^a g_{m_2}^b, |m_1| \leq 3 \text{ and } |m_2| \leq 2; \\
(h_y^{(3)})_{m_1, m_2} &= g_{m_1}^a h_{m_2}^b, |m_1| \leq 3 \text{ and } |m_2| \leq 2; \\
(h_x^{(4)})_{m_1, m_2} &= h_{m_1}^b g_{m_2}^a, |m_1| \leq 2 \text{ and } |m_2| \leq 3; \\
(h_y^{(4)})_{m_1, m_2} &= g_{m_1}^b h_{m_2}^a, |m_1| \leq 2 \text{ and } |m_2| \leq 3;
\end{aligned}$$

where  $m_1$  and  $m_2$  index the row and column dimensions of the image respectively.

Using the derivative masks  $h_x^{(i)}$  and  $h_y^{(i)}$ , the  $x$ - and  $y$ -directional derivatives at pixel locations  $s_i$  are determined as the following convolutions of the filters and input image  $u_s$ :

$$(u_x)_{s_i} = \sum_{\mathbf{m}} (h_x^{(i)})_{\mathbf{m}} u_{s_i + \mathbf{m}}, \quad (6)$$

$$(u_y)_{s_i} = \sum_{\mathbf{m}} (h_y^{(i)})_{\mathbf{m}} u_{s_i + \mathbf{m}}, \quad (7)$$

where the summation is over the region of support of each derivative mask, and  $\mathbf{m} = (m_1, m_2)$  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 (2).

### 3.2. Computation of 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  $7 \times 7$  local window centered at  $s_0$  computed

in four different directions, namely, east, south, west, and north, respectively from 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  $h_{\mathbf{m}}$ , where  $\mathbf{m} = (m_1, m_2)$  represents the discrete spatial coordinates in the 2-D image. The low-pass filter is selected empirically to provide robust suppression of screen frequencies in the scanned image that could interfere with the print screen in the target MFP to produce aliasing artifacts. The low-pass filter  $h_{\mathbf{m}}$  is separable and can be written as the outer product of two 1-D kernels. Specifically,  $h_{\mathbf{m}}$  is defined as

$$h_{\mathbf{m}} = h_{m_1}^a h_{m_2}^a; |m_1|, |m_2| \leq 3 \quad (8)$$

$$= \frac{1}{256} \begin{bmatrix} 1 & 2 & 3 & 4 & 3 & 2 & 1 \\ 2 & 4 & 6 & 8 & 6 & 4 & 2 \\ 3 & 6 & 9 & 12 & 9 & 6 & 3 \\ 4 & 8 & 12 & 16 & 12 & 8 & 4 \\ 3 & 6 & 9 & 12 & 9 & 6 & 3 \\ 2 & 4 & 6 & 8 & 6 & 4 & 2 \\ 1 & 2 & 3 & 4 & 3 & 2 & 1 \end{bmatrix}, \quad (9)$$

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

$$z_{s_i} = \frac{\sum_{\mathbf{m}} h_{\mathbf{m}} w_{\mathbf{m}}^{(i)} u_{s_0 + \mathbf{m}}}{\sum_{\mathbf{m}} h_{\mathbf{m}} w_{\mathbf{m}}^{(i)}}, \quad (10)$$

where  $|m_1|, |m_2| \leq 3$  and  $\sum_{\mathbf{m}} h_{\mathbf{m}} w_{\mathbf{m}}^{(i)} = \frac{1}{4} h_{\mathbf{m}} \forall i$ .

It can be seen from Fig. 3 that the sum of the four weighting masks, i.e.  $\sum_{i=1}^4 w_{\mathbf{m}}^{(i)}$ , produces a  $7 \times 7$  boxcar filter, consequently yielding  $\sum_{i=1}^4 w_{\mathbf{m}}^{(i)} h_{\mathbf{m}} = h_{\mathbf{m}}$ . Thus, in a smooth region of the image, where the local gradient norms are small, and the value of the function  $g(x)$  in (3) is approximately equal to 1, the output of the HFD filter  $v_{s_0}$  is given by

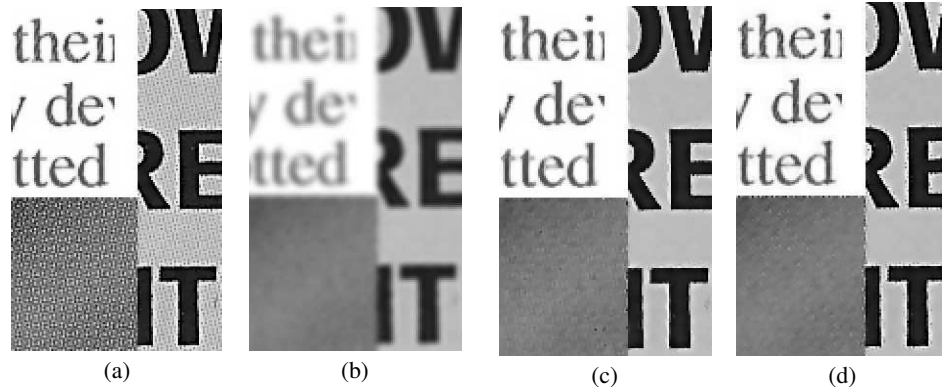
$$v_{s_0} = \sum_{i=1}^4 \frac{z_{s_i}}{4} = \frac{\sum_{i=1}^4 \sum_{\mathbf{m}} h_{\mathbf{m}} w_{\mathbf{m}}^{(i)} u_{s_0 + \mathbf{m}}}{4 \sum_{\mathbf{m}} h_{\mathbf{m}} w_{\mathbf{m}}^{(i)}}, \quad (11)$$

$$= \frac{\sum_{\mathbf{m}} h_{\mathbf{m}} u_{s_0 + \mathbf{m}}}{\sum_{\mathbf{m}} h_{\mathbf{m}}}. \quad (12)$$

This is equivalent to applying a low-pass blurring filter  $h_{\mathbf{m}}$  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.

## 4. EXPERIMENTAL RESULTS

The performance of the descreening algorithm was evaluated on ten 300-dpi scanned documents, which were obtained by scanning printed originals on an HP ScanJet 8250 scanner and an HP LaserJet CM1015 MFP. 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.



**Fig. 4.** (a) Scanned image, 300-dpi (Source printer: HP LaserJet 4050, Source scanner: HP ScanJet 8250). Descreened images using (b) Gaussian filter ( $7 \times 7, \sigma = 1.7$ ), (c) training-based descreening, and (d) hardware friendly descreening.

Operations	TBD (float-point)	HFD (fixed-point)
Additions	603	181
Multiplications	410	18
Divisions	20	0
Bitwise shifts	0	117
Exponentiations	50	0

**Table 2.** Comparison of required number of operations per pixel in training-based descreening (TBD) and hardware friendly descreening (HFD) algorithms.

We use 3 different filtering algorithms for performance comparison in this section: (1) Space-invariant linear low-pass filter; (2) Training-based descreening [4]; and (3) HFD algorithm.

For linear space-invariant low-pass filtering, we use the filter  $h_m$ , discussed in Section 3.2. This filter is approximately Gaussian with  $\sigma = 1.7$ .

The parameters of the training-based descreening (TBD) algorithm are optimized as in [4]. TBD requires a floating-point buffer of size approximately 10K to store the trainable descreening parameters.

The non-linear functions  $f(x)$  and  $g(x)$  in the hardware friendly descreening algorithm are implemented as LUTs to reduce computation. The sample values for  $f(x)$ , defined in (4), are stored in a LUT with 128 10-bit sized integer values, while the sample values for  $g(x)$ , defined in (5), are stored in a LUT with 512 9-bit sized integer values.

Fig. 4 shows the descreening results. Fig. 4 (a) shows a zoomed-in view of a portion of a digital scanned document. Fig. 4 (b) shows the processed result with the linear space-invariant low-pass filter. While low-pass filtering is very effective in removing the aliasing screen frequencies, it also blurs non-half-tone detail in the image. The output of training-based descreening, shown in Fig. 4 (c), displays substantial improvement in image quality over that of linear low-pass blur. The descreened image with the proposed HFD algorithm is presented in Fig. 4 (d).

Table 2 compares the computational complexity of training-based descreening to that of the proposed HFD algorithm. We conclude that the HFD filter delivers an image quality comparable to training-based descreening, but at a considerably lower cost.

## 5. CONCLUSIONS

The descreening algorithm developed in this research was primarily designed to provide a robust and cost-effective solution for halftone screen removal in scanned halftone images. The algorithm uses integer arithmetic, mostly relying on low-cost bitwise shift and addition operations, and uses a strictly sequential architecture to allow a hardware efficient implementation. The proposed filter is non-linear and space-variant. The local filter weights are determined through the use of non-linear polynomial functions and spatial feature vectors extracted from pre-smoothed versions of the noisy scanned image.

## 6. REFERENCES

- [1] J. C. Stoffel and J. F. Moreland, "A survey of electronic techniques for pictorial reproduction," *IEEE Trans. on Communications*, vol. 29, pp. 1898–1925, 1981.
- [2] J. P. Allebach and B. Liu, "Analysis of halftone dot profile and aliasing in the discrete binary representation of images," *J. Optical Society. America*, vol. 67, no. 9, pp. 1147–1154, 1977.
- [3] J. Luo, R. de Queiroz, and Z. Fan, "A robust technique for image descreening based on the wavelet transform," *IEEE Trans. on Signal Processing*, vol. 46, no. 4, pp. 1179–1184, April 1998.
- [4] Hasib Siddiqui and Charles A. Bouman, "Training-based descreening," *IEEE Trans. on Image Processing*, vol. 16, no. 3, pp. 789–802, March 2007.
- [5] 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. on Image Processing*, vol. 9, no. 9, pp. 1583–1592, September 2000.
- [6] R. L. Stevenson, "Inverse halftoning via MAP estimation," *IEEE Trans. on Image Processing*, vol. 4, no. 4, pp. 486–498, April 1997.
- [7] P. W. Wong, "Inverse halftoning and kernel estimation for error diffusion," *IEEE Trans. on Image Processing*, vol. 4, no. 4, pp. 486–498, April 1995.
- [8] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, pp. 629–639, July 1983.