Clustered-Dot Halftoning With Direct Binary Search

Puneet Goyal, Madhur Gupta, Carl Staelin, Mani Fischer,
Omri Shacham, and Jan P. Allebach, *Fellow, IEEE*

***Abstract*— In this paper, we present a new algorithm for aperiodic clustered-dot halftoning based on direct binary search (DBS). The DBS optimization framework has been modified for designing clustered-dot texture, by using filters with different sizes in the initialization and update steps of the algorithm. Following an intuitive explanation of how the clustered-dot texture results from this modified framework, we derive a closed-form cost metric which, when minimized, equivalently generates stochastic clustered-dot texture. An analysis of the cost metric and its influence on the texture quality is presented, which is followed by a modification to the cost metric to reduce computational cost and to make it more suitable for screen design.**

***Index Terms*— Clustered-dot, digital halftoning, direct binary search.**

I. Introduction

HALFTONING is the process of transforming a continuous-tone image into an image with a limited number of tone levels. This problem is important for reproducing continuous-tone grayscale or color images in situations where only a limited number of output states are possible. The output image produced by the halftoning process is called a halftone. The most common application of halftoning is in printing images on paper, where tone levels of halftone represent the presence or absence of a dot (in the binary case), or the size or density of a dot (in the multi-level case).

The objective of halftoning is that the halftone, when rendered, should be visually the same as the original image. While there will be microscopic differences between the two, the differences should be minimal when observed by a human viewer from a normal viewing distance. The human eye, being limited in its capacity to resolve dots placed very close together, cannot distinguish between the original image and the halftone, if the differences between them are of high spatial frequency content. However, low frequency differences are visible to the human eye. Ulichney [1], [2] demonstrates through many examples that halftones containing energy predominantly in the high frequency region appear pleasing because the low frequency noise is absent. That is why acceptable halftones have minimal energy content in the low-frequency region.

One classification of halftones is based on the size and distribution of dots [3]. The dispersed-dot halftones are those which represent any gray level by modulating the density of dots, while keeping their size fixed. The periodic, clustered-dot halftones are those which render a gray level by modulating the size of dots, keeping their distances fixed on a regular lattice. There is also another type of halftone composed of stochastically distributed clustered dots [4]. The spectral characteristics of the above-mentioned types of halftones can be studied by computing the Radially Averaged Power Spectrum (RAPS) [1]. Figure 1 shows examples of these three types of halftones and their corresponding RAPS curves.

Even though all the above-mentioned types of halftones can be designed to have spectral energy outside the visible low-frequency region, the most appropriate type must be chosen based on properties of the printer. For example, printers using inkjet technology can form stable isolated dots, while printers using electrophotographic (EP) technology cannot render isolated dots stably. Therefore, dispersed-dot halftones can be printed using inkjet printers; but printing dispersed-dot halftones using EP printers leads to many artifacts such as banding in midtones and dot drop-out in highlights. Therefore, clustered-dot halftones are preferred for EP printers [5], [6]. Of the two types of clustered-dot halftones, periodic and stochastic, periodic halftones are quite smooth but they also suffer from a serious drawback of periodic moiré resulting from periodic interference between halftones of different color channels. Although Amidror *et al.* [7] show that this problem can be reduced by offsetting the color screens by appropriate angles with respect to each other, determining angular offsets that will yield acceptable quality becomes increasingly challenging with the increase in the number of primary colorants of the printer (e.g. in hexachrome printing). Stochastic, clustered-dot halftones provide an alternative solution because they are free from the problem of periodic interference [8]. Periodic halftones can be used in conjunction with stochastic halftones to get a good combination of smoothness and freedom from periodic moiré. Therefore, in this paper we consider how to generate aperiodic, clustered-dot halftones.

Many schemes have been proposed in the literature for producing aperiodic, clustered-dot halftones. One of them is the output-dependent feedback in error diffusion by

Manuscript received August 5, 2011; revised July 3, 2012; accepted August 28, 2012. Date of publication September 13, 2012; date of current version January 8, 2013. This work was supported by the Hewlett-Packard Company. This work was completed in part while P. Goyal and M. Gupta were Ph.D. students with Purdue University, and while C. Staelin and O. Shacham were with the Hewlett-Packard Company. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Vishal Monga.

P. Goyal is with Graphic Era University, Dehradun 248002, India (e-mail: pgoyal@alumni.purdue.edu).

M. Gupta is with AT&T Services, Inc., San Ramon, CA 94583 USA (e-mail: mgupta.iitd@gmail.com).

C. Staelin is with the Google Advanced Technology Center, Technion City 31905, Israel (e-mail: carl.staelin@gmail.com).

M. Fischer is with Hewlett-Packard Laboratories Israel, Technion City 32000, Israel (e-mail: mani.fischer@hp.com).

O. Shacham is with HP Indigo Ltd., Ness Ziona 61999, Israel (e-mail: shacham.omri@gmail.com).

J. Allebach is with Purdue University, West Lafayette, IN 47907 USA (e-mail: allebach@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.2012.2218821

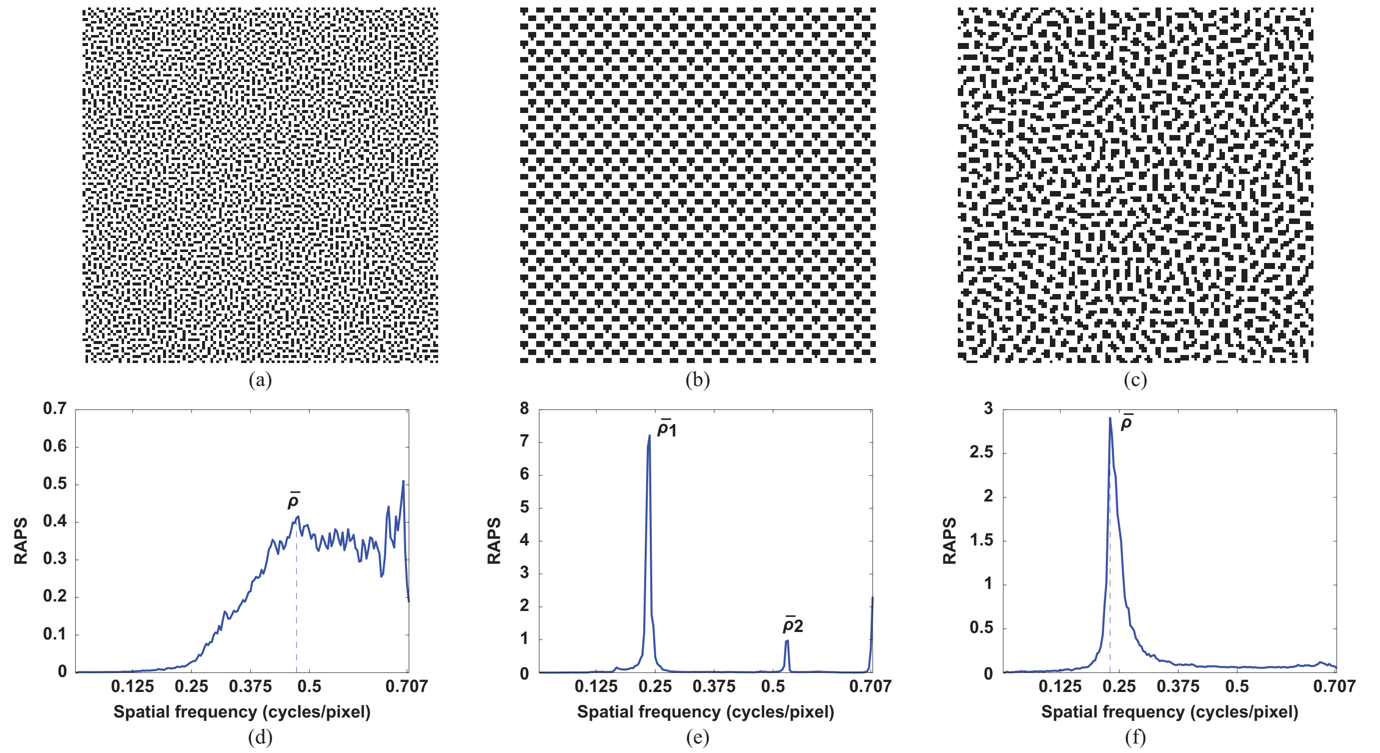


Fig. 1. Examples of (a) aperiodic dispersed-dot, (b) periodic clustered-dot, and (c) aperiodic clustered-dot halftones with their (d)–(f) spectral characteristics in the form of radially averaged power spectrum (RAPS) curves. In (d) and (f), the principal frequency $\bar{\rho}$ defined by [1] and [2] is shown.

Levien [9], [10]. Like basic error diffusion, this algorithm works by diffusing the quantization error from the current pixel to future neighboring pixels. But it is different in the sense that while traditional error diffusion techniques such as that proposed by Floyd and Steinberg [11] produce dispersed-dot halftones, Levien's technique produces clustered-dot halftones by encouraging cluster formation wherever the error exceeds a threshold. This is done by modulating the threshold with the weighted output from previous neighboring pixels. However, Levien's method generates structured worm-like artifacts in midtones. Lau *et al.* [4] randomized the feedback coefficients to break-up the structured artifacts; but this, in turn, introduces noise in the texture. Li and Allebach [12] included a dot activation map along with Levien's output-dependent feedback to modify the threshold. This gives more control over dot size and dot shape, and also reduces the midtone artifacts. Cooper [13] also invented a method for stochastic, clustered-dot halftoning using a difference weighting function. He proposed two functions with different extents and suggested that the difference between these functions can be used to establish a growth rule for new dots in the neighborhood of existing cluster centers. Another development in stochastic clustering is the AM/FM halftoning method proposed by He and Bouman [14]. This algorithm uses error diffusion operating on the output of a dot-density LUT to determine cluster location, and directly sets dot size based on the output of a dot-size LUT. Both LUTs are driven by the pixel values from the continuous-tone image.

Apart from the above-mentioned schemes based on error diffusion, there are also methods for stochastic, clustered-dot screen design. One of them is a spatial statistics-based iterative halftone manipulation proposed by Lau *et al.* [15], [16]. Lau *et al.* experimentally determined a model for the spatial distribution of minority pixels in stochastic, clustered-dot halftones, and iteratively manipulated a given halftone to attain the desired statistics.

Earlier, Allebach [17] and Wang [18] invented stochastic, clustered-dot halftoning systems based on first identifying locations of the set of stochastically distributed black and white cluster centers and then establishing a growth sequence for adding pixels about the cluster centers. Ostromoukhov [19] proposed a method for designing a stochastic, clustered-dot dither matrix by first obtaining cluster centers by the use of space-filling curves, and then assigning threshold values based on an iso-intensity map. Work by Abe [20] involved construction of an aperiodic, clustered-dot screen via simulated annealing. The cluster size can be varied by adjusting the weight parameter of the cost function. Damera-Venkata and Lin [21], [22] devised a simple and efficient method for design of stochastic, clustered-dot halftone screens by the use of linear filters. These filters have a radial profile resembling the radial cross-section of a donut. Dots are grown at those locations where the filtered output is minimum. This results in halftones with the desired spatial characteristics.

Our method for stochastic, clustered-dot halftoning is based on the Direct Binary Search (DBS) algorithm [23], [24] originally designed for dispersed-dot halftoning. DBS employs toggle and swap operations to transform any given halftone into a homogeneous halftone [23]. We modify the conventional DBS procedure by using different filters in the initialization and update phases. In the modified set-up, which we call CLU-DBS, toggle and swap operations yield a stochastic, clustered-dot halftone.

In the current section, we introduced the topic of the paper and prior work, along with motivations and approach of our work. In Sec. II we present the background of DBS to explain our new technique. Section III includes a description of the new halftoning algorithm, accompanied with an intuitive explanation of why it works. In Sec. IV, we introduce a cost metric which guides the optimization of a given halftone into a stochastic, clustered-dot halftone. The method of optimizing a halftone via minimization of this cost metric is equivalent to executing DBS on the halftone with modified initialization- and update-filters.

II. Preliminaries: Direct Binary Search

Direct binary search is an iterative halftone manipulation-based algorithm that achieves a homogeneous, dispersed-dot texture by minimizing a perceptual-error-based cost metric [23]. In this section, we present a brief analysis of DBS [24]. Throughout this paper, we use $[\mathbf{m}] = [m, n]^T$ and $(\mathbf{x}) = (x, y)^T$ to represent discrete and continuous spatial coordinates, respectively.

The perceptual characteristics of a human viewer can be modeled as a low-pass, linear shift-invariant filtering operation. The model for the human visual system (HVS) point spread function (PSF) is denoted by $\tilde{p}_{hvs}(\mathbf{x})$. Given a digital halftone $g[\mathbf{m}]$, the perceived halftone $\tilde{g}(\mathbf{x})$ is computed by convolving the digital halftone $g[\mathbf{m}]$ with $\tilde{p}(\mathbf{x})$.

$$\tilde{g}(\mathbf{x}) = \sum_{m} g[\mathbf{m}]\tilde{p}(\mathbf{x} - \mathbf{X}\mathbf{m}) \tag{1}$$

where $\mathbf{X}$ is the periodicity matrix with its columns representing a basis for the printer-addressable dot lattice, and $\tilde{p}(\mathbf{x}) = p_{hvs}(\mathbf{x}) * *p_{dot}(\mathbf{x})$ embodies the cascaded effect of printer dot rendering and the human perception model. In this paper, we assume that the extent of $p_{hvs}(\mathbf{x})$ is much greater than that of $p_{dot}(\mathbf{x})$, in which case $\tilde{p}(\mathbf{x}) \approx p_{hvs}(\mathbf{x})$, and simply refer to $\tilde{p}(\mathbf{x})$ as the HVS PSF.

The HVS PSF $\tilde{p}(\mathbf{x})$ is based on the contrast sensitivity function of the HVS. The spatial frequency response of the luminance model is given by an exponential function proposed by Näsänen [25]. The cost metric, however, depends on $\tilde{p}(\mathbf{x})$ only through its autocorrelation function $c_{\tilde{p}\tilde{p}}(\mathbf{x})$, evaluated on the printer lattice [24]. Assuming a printer with resolution R dots/inch (dpi), an expected viewing distance D (in), and a rectangular lattice of addressable points, we find that

$$c_{\tilde{p}\tilde{p}}[\mathbf{m}] = \frac{1}{D^2}\int_{\mathbf{y}} h(\mathbf{y})h\left(\mathbf{y} + \frac{\mathbf{m}}{S}\right)d\mathbf{y} \tag{2}$$

where h is the inverse Fourier transform of the contrast sensitivity function evaluated at the retina in units of radians, and $S \equiv RD$ is the scale parameter.

A. Error Metric

We use $f[\mathbf{m}]$ to denote the original discrete-space continuous-tone image. Each pixel of $f[\mathbf{m}]$ takes on values of absorptance between 0 and 1, whereas each pixel of the halftone $g[\mathbf{m}]$ takes on values 0 (white) or 1 (black) only. The error metric minimized in conventional DBS is denoted by ϕ and it is defined as

$$\phi = \int |\tilde{e}(\mathbf{x})|^2\,d\mathbf{x} \tag{3a}$$

$$\text{where}\quad \tilde{e}(\mathbf{x}) = \sum_{\mathbf{m}} e[\mathbf{m}]\tilde{p}(\mathbf{x} - \mathbf{X}\mathbf{m}) \tag{3b}$$

$$\text{and}\quad e[\mathbf{m}] = g[\mathbf{m}] - f[\mathbf{m}]. \tag{3c}$$

After simplification, the conventional DBS error metric ϕ can also be expressed as

$$\phi = \sum_{\mathbf{m}}\sum_{\mathbf{n}} e[\mathbf{m}]e[\mathbf{n}]c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{n}] \tag{4a}$$

$$= \sum_{\mathbf{m}} e[\mathbf{m}]c_{\tilde{p}\tilde{e}}[\mathbf{m}] \tag{4b}$$

where

$$c_{\tilde{p}\tilde{e}}[\mathbf{m}] = \sum_{\mathbf{n}} e[\mathbf{n}]c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{n}]. \tag{5}$$

B. Analyzing the Effect of Halftone Changes

Prior to optimizing the halftone given as input to the algorithm, the $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ LUT is initialized as shown in (5) [24]. During optimization, the halftone image is iteratively scanned in raster order, from left to right and top to bottom. At each pixel, we evaluate the effect of toggling its state or swapping its value with other pixels in a 3×3 neighborhood centered at that pixel location. If any toggle or swap decreases the cost metric, that change which decreases the cost metric the most is accepted. An iteration is complete when every pixel in the image has been visited. When no change is accepted during an iteration, the algorithm is said to have converged to a local minimum of the cost metric. The number of accepted changes is usually a very small fraction of the number of trial evaluations. So it is desirable to minimize the computational cost of evaluating the trial changes.

When a pixel $\mathbf{m}_0$ is considered for toggle, a_0 is defined to be 1 if $g[\mathbf{m}_0] = 0$ before the toggle, and -1 otherwise. When $\mathbf{m}_0$ is considered for a swap with $\mathbf{m}_1$ where, for example, $g[\mathbf{m}_0] = 0$ and $g[\mathbf{m}_1] = 1$, then $a_0 = 1$ and $a_1 = -1$. The effect of a trial toggle operation and a trial swap operation is given by [24]

$$\Delta\phi_{tog} = a_0^2 c_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2a_0 c_{\tilde{p}\tilde{e}}[\mathbf{m}_0] \tag{6}$$

$$\begin{aligned}\Delta\phi_{swap} &= ({a_0}^2 + {a_1}^2)c_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2a_0 c_{\tilde{p}\tilde{e}}[\mathbf{m}_0] \\ &\quad + 2a_1 c_{\tilde{p}\tilde{e}}[\mathbf{m}_1] + 2a_0 a_1 c_{\tilde{p}\tilde{p}}[\mathbf{m}_1 - \mathbf{m}_0].\end{aligned} \tag{7}$$

Substituting ${a_0}^2 = 1$ into (6), it follows that a toggle at $\mathbf{m}_0$ may be considered acceptable if

$$a_0 c_{\tilde{p}\tilde{e}}[\mathbf{m}_0] + \frac{1}{2}c_{\tilde{p}\tilde{p}}[\mathbf{0}] < 0 \tag{8}$$

or in general, if $|c_{\tilde{p}\tilde{e}}[\mathbf{m}_0]| > \frac{1}{2}c_{\tilde{p}\tilde{p}}[\mathbf{0}]$. If a toggle is accepted, then $g[\mathbf{m}]$ and $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ are updated as follows:

$$g'[\mathbf{m}] = g[\mathbf{m}] + a_0\delta[\mathbf{m} - \mathbf{m}_0] \tag{9}$$

$$c'_{\tilde{p}\tilde{e}}[\mathbf{m}] = c_{\tilde{p}\tilde{e}}[\mathbf{m}] + a_0 c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{m}_0]. \tag{10}$$

It follows that updating $c_{\tilde{p}\tilde{e}}[\mathbf{m}_0]$ according to the toggle will result in its absolute value getting closer to zero. This implies that when toggle operations converge, we shall have for every pixel $\mathbf{m}$ in the image

$$|c_{\tilde{p}\tilde{e}}[\mathbf{m}]| < \frac{1}{2}c_{\tilde{p}\tilde{p}}[\mathbf{0}]. \tag{11}$$

On the other hand, by substituting $a_0{}^2 = a_1{}^2 = 1$, $a_1 = -a_0$, and $a_0 a_1 = -1$ into (7), it follows that a swap may be considered acceptable if

$$a_0(c_{\tilde{p}\tilde{e}}[\mathbf{m}_0]-c_{\tilde{p}\tilde{e}}[\mathbf{m}_1])+(c_{\tilde{p}\tilde{p}}[\mathbf{0}]-c_{\tilde{p}\tilde{p}}[\mathbf{m}_1-\mathbf{m}_0]) < 0 \tag{12}$$

or in general, if $|c_{\tilde{p}\tilde{e}}[\mathbf{m}_1]-c_{\tilde{p}\tilde{e}}[\mathbf{m}_0]| > c_{\tilde{p}\tilde{p}}[\mathbf{0}]-c_{\tilde{p}\tilde{p}}[\mathbf{m}_1-\mathbf{m}_0]$. In other words, if the local gradient in $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ exceeds the gradient in $c_{\tilde{p}\tilde{p}}[\mathbf{m}]$ measured at its center. If a swap is accepted, then $g[\mathbf{m}]$ and $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ are updated as follows:

$$\begin{aligned} g'[\mathbf{m}] &= g[\mathbf{m}] + a_0\delta[\mathbf{m}-\mathbf{m}_0] + a_1\delta[\mathbf{m}-\mathbf{m}_1] && (13)\\ &= g[\mathbf{m}] + a_0(\delta[\mathbf{m}-\mathbf{m}_0] - \delta[\mathbf{m}-\mathbf{m}_1]) && (14)\\ c'_{\tilde{p}\tilde{e}}[\mathbf{m}] &= c_{\tilde{p}\tilde{e}}[\mathbf{m}]+a_0 c_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_0]+a_1 c_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_1] && (15)\\ &= c_{\tilde{p}\tilde{e}}[\mathbf{m}]+a_0(c_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_0]-c_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_1]). && (16)\end{aligned}$$

Hence, updating $c_{\tilde{p}\tilde{e}}[\mathbf{m}_0]$ and $c_{\tilde{p}\tilde{e}}[\mathbf{m}_1]$ according to the swap will reduce the local gradient in $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ at $\mathbf{m} = \mathbf{m}_0$ and $\mathbf{m} = \mathbf{m}_1$. In general, this implies that swap operations tend to reduce the local gradient in $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ such that at convergence

$$|c_{\tilde{p}\tilde{e}}[\mathbf{m}_0] - c_{\tilde{p}\tilde{e}}[\mathbf{m}_1]| < c_{\tilde{p}\tilde{p}}[\mathbf{0}] - c_{\tilde{p}\tilde{p}}[\mathbf{m}_0 - \mathbf{m}_1] \tag{17}$$

for any pair of pixels $\mathbf{m}_0$ and $\mathbf{m}_1$ in a 3×3 neighborhood. In this sense, a swap operation is more powerful than a toggle because whereas a toggle only reduces the absolute value of $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$, a swap further improves halftone homogeneity by removing any drastic local variation in the perceived halftone.

Figure 2 illustrates the essential elements of both the DBS and the proposed CLU-DBS halftoning algorithms. It shows (a) the continuous-tone input image, (b) the random halftone that initializes the algorithm, (c) the HVS function used to compute filtered halftone error, and (d) the filtered version of initial halftone error containing peaks and valleys corresponding to high and low dot density regions in the halftone. Figure 3 shows that DBS minimizes the magnitude of $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ and simultaneously turns the initial random halftone into a homogeneous pattern, as discussed in this section. Figure 4 compares halftones resulting from DBS optimization by employing (a) only toggle operations and (b) both toggle and swap operations. The halftone optimized by both toggle and swap operations appears more smooth than the halftone optimized by only toggle operations.

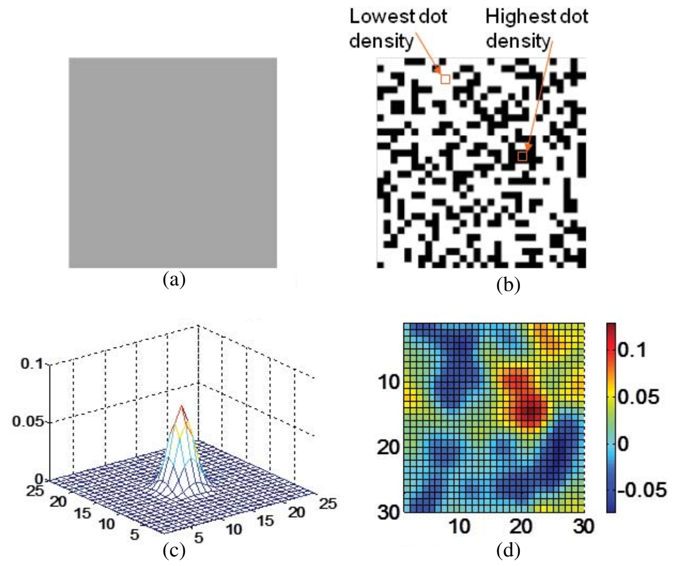


Fig. 2. Images illustrating the essential elements of the DBS and CLU-DBS halftoning algorithms. (a) Continuous-tone input image $f[\mathbf{m}]$ with absorptance 0.30 and (b) initial halftone image $g[\mathbf{m}]$ with average absorptance 0.30. Pixels corresponding to highest and lowest dot density are marked in the halftone. (c) HVS filter autocorrelation function $c_{\tilde{p}\tilde{p}}[\mathbf{m}]$ based on Näsänen's model with $R = 300$ dpi and $D = 10$ in. (d) Filtered error image $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ obtained by filtering the error image $e[\mathbf{m}] = g[\mathbf{m}] - f[\mathbf{m}]$ with $c_{\tilde{p}\tilde{p}}[\mathbf{m}]$.

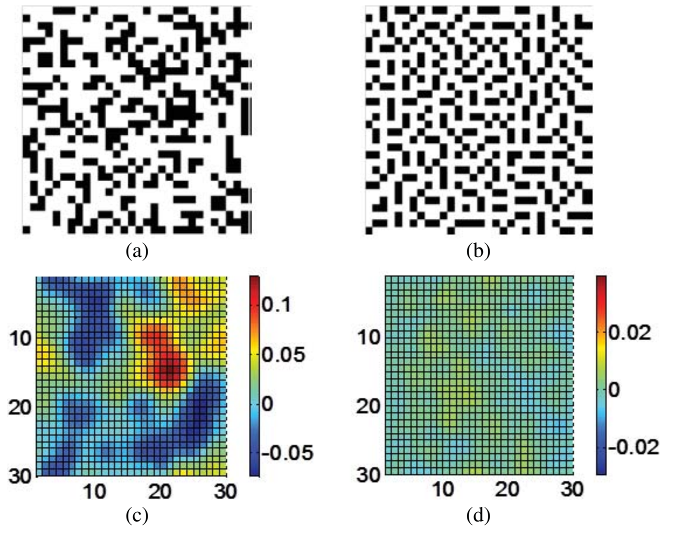
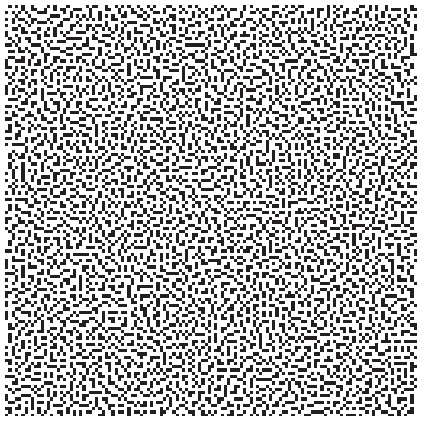


Fig. 3. Example showing how DBS optimizes a white noise halftone pattern into a homogeneous, dispersed-dot pattern by reducing the magnitude of $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ and its local gradient. The HVS filter autocorrelation function $c_{\tilde{p}\tilde{p}}[\mathbf{m}]$ is based on Näsänen's model with $R = 300$ dpi and $D = 10$ in. (a) Initial halftone. (b) Final halftone. (c) Initial filtered error $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$. (d) Final filtered error $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$. Note the difference in the amplitude scales for (c) and (d).

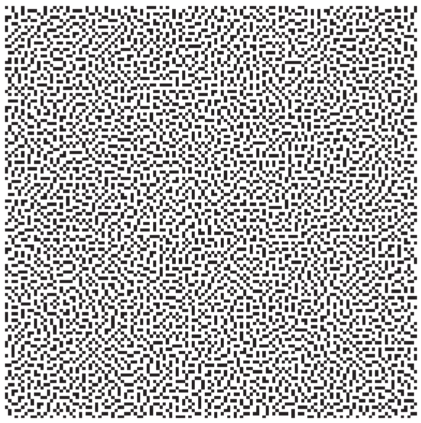
III. Aperiodic, Clustered-Dot Halftoning With DBS

Aperiodic or stochastic, clustered-dot halftoning refers to the rendering of a continuous-tone image with stochastically distributed clustered dots. We discussed in Sec. II how conventional DBS yields a stochastic, *dispersed-dot* texture by minimizing the $c_{\tilde{p}\tilde{p}}$-filtered halftone error through toggles and swaps. Here, we describe an algorithm, based on DBS, that generates stochastic, *clustered-dot* texture by minimizing the filtered halftone error through toggles and swaps. First, we describe the algorithm, and then we provide an intuitive explanation for how it generates a clustered-dot texture.

As discussed in Sec. II, DBS optimization works in two steps. The first step involves initialization of the filtered halftone error $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ (5); and the second step involves manipulation of the halftone accompanied by a corresponding update of $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ ((6), (7), (9), (10), (13)-(16)). The function used to



(a)



(b)

Fig. 4. Halftones optimized by DBS using toggle-only and toggle-and-swap operations. Halftone (b) looks more homogeneous than halftone (a). The HVS filter autocorrelation function used is based on Näsänen's model with $R = 300$ dpi and $D = 10$ inches. (a) DBS (toggle only). (b) DBS (toggle-and-swap).

filter the error image $e[\mathbf{m}]$ is the autocorrelation function of the HVS PSF. The algorithm CLU-DBS is also executed in two steps: the first step consisting of initialization of a filtered halftone error, and the second step consisting of updates to the halftone and the filtered error. However, in CLU-DBS we introduce the use of two separate low-pass filters to the conventional DBS framework. We denote the two filters as $c^{i}_{\tilde{p}\tilde{p}}[\mathbf{m}]$ and $c^{u}_{\tilde{p}\tilde{p}}[\mathbf{m}]$, corresponding to the initialization step and the update step, respectively. The notation for the filtered version of the halftone error remains the same, i.e. $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$. Given this notation, the initialization step in CLU-DBS can be represented as follows:

$$c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] = e[\mathbf{m}] ** c^{i}_{\tilde{p}\tilde{p}}[\mathbf{m}] \tag{18}$$

where $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ denotes the initial filtered version of the error image, distinct from its subsequent versions obtained upon accepting halftone changes. The evaluation and update rules in CLU-DBS differ from those in conventional DBS ((6)-(16)) only in the sense that the filter used for evaluating trial halftone changes and updating the filtered error corresponding to accepted halftone changes is different from the filter used for the initialization step. Analogous to the DBS criteria for evaluating toggle and swap operations ((8) and

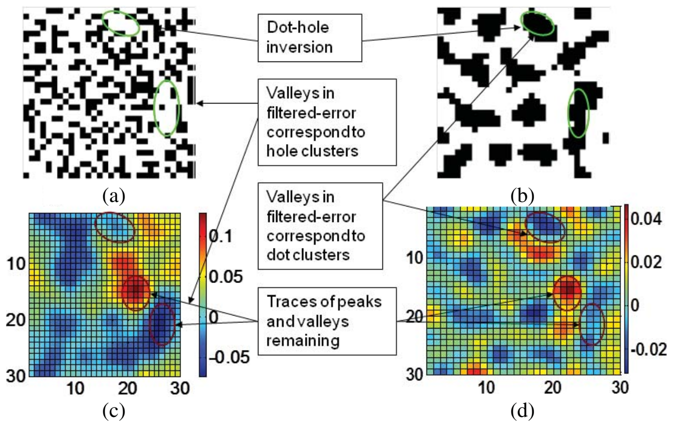


Fig. 5. Optimization of a random initial halftone using the new CLU-DBS halftoning scheme with toggle operations. The initialization- and update-filters, $c^{i}_{\tilde{p}\tilde{p}}[\mathbf{m}]$ and $c^{u}_{\tilde{p}\tilde{p}}[\mathbf{m}]$ are HVS filter autocorrelation functions based on Näsänen's model with same resolution 300 dpi, and different viewing distances of 10 and 24 in, respectively. (a) Initial halftone. (b) Final halftone. (c) Initial filtered error $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$. (d) Final filtered error $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$.

(12)), in each iteration of the CLU-DBS update-step, a trial toggle at $\mathbf{m}_0$ may be considered acceptable if

$$a_0 c_{\tilde{p}\tilde{e}}[\mathbf{m}_0] + \frac{1}{2} c^{u}_{\tilde{p}\tilde{p}}[\mathbf{0}] < 0 \tag{19}$$

and a trial swap between $\mathbf{m}_0$ and $\mathbf{m}_1$, where, for example, $g[\mathbf{m}_0] = 0$ and $g[\mathbf{m}_1] = 1$, may be considered acceptable if

$$a_0(c_{\tilde{p}\tilde{e}}[\mathbf{m}_0] - c_{\tilde{p}\tilde{e}}[\mathbf{m}_1]) + (c^{u}_{\tilde{p}\tilde{p}}[\mathbf{0}] - c^{u}_{\tilde{p}\tilde{p}}[\mathbf{m}_1 - \mathbf{m}_0]) < 0. \tag{20}$$

After a halftone change is accepted, $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ is updated accordingly. As is the case with conventional DBS, among the possible changes for a fixed 3×3 set of pixels centered at $\mathbf{m}_0$ that evaluate the left hand side expression of (19) or (20) to a negative value, only that one which evaluates to the most negative value will actually be accepted. Following a toggle at $\mathbf{m}_0$ according to (19), $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ is updated as

$$c'_{\tilde{p}\tilde{e}}[\mathbf{m}] = c_{\tilde{p}\tilde{e}}[\mathbf{m}] + a_0 c^{u}_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{m}_0]. \tag{21}$$

Following a swap between $\mathbf{m}_0$ and $\mathbf{m}_1$ according to (20), $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ is updated as

$$c'_{\tilde{p}\tilde{e}}[\mathbf{m}] = c_{\tilde{p}\tilde{e}}[\mathbf{m}] + a_0(c^{u}_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{m}_0] - c^{u}_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{m}_1]). \tag{22}$$

Halftone changes are continually performed according to the criteria given in (19) and (20), and $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ is updated according to (21) and (22), until the process converges. The algorithm is said to have reached convergence with respect to toggles when, for all pixels $\mathbf{m}$ in the image,

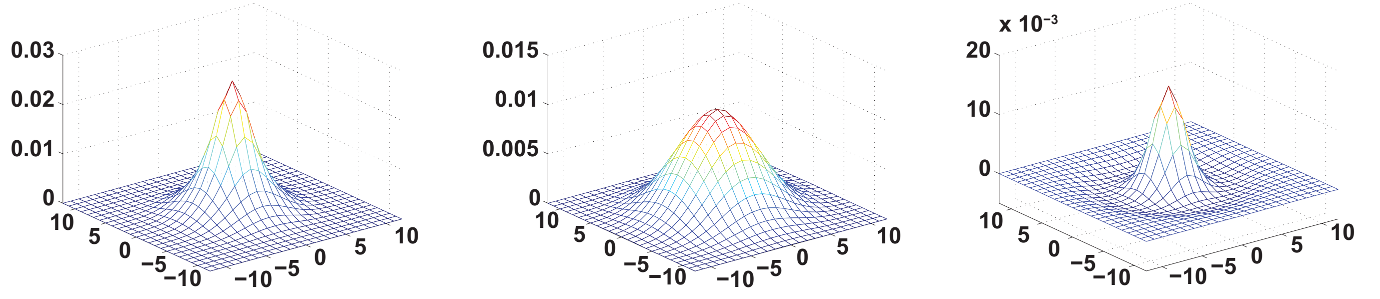
$$c_{\tilde{p}\tilde{e}}[\mathbf{m}] \geq -\frac{1}{2} c^{u}_{\tilde{p}\tilde{p}}[\mathbf{0}] \text{ if } g[\mathbf{m}] = 0 \tag{23a}$$

and

$$c_{\tilde{p}\tilde{e}}[\mathbf{m}] \leq \frac{1}{2} c^{u}_{\tilde{p}\tilde{p}}[\mathbf{0}] \text{ if } g[\mathbf{m}] = 1. \tag{23b}$$

The algorithm is said to have reached convergence with respect to swaps when, for every pixel $\mathbf{m}_0$ in the image and any $\mathbf{m}_1$ within the 3×3 neighborhood of $\mathbf{m}_0$,

$$c_{\tilde{p}\tilde{e}}[\mathbf{m}_1] - c_{\tilde{p}\tilde{e}}[\mathbf{m}_0] < c^{u}_{\tilde{p}\tilde{p}}[\mathbf{0}] - c^{u}_{\tilde{p}\tilde{p}}[\mathbf{m}_1 - \mathbf{m}_0] \quad \text{if } g[\mathbf{m}_0] = 0 \text{ and } g[\mathbf{m}_1] = 1. \tag{24}$$



(a) (b) (c)

Fig. 6. Filters used in the initialization and update steps of CLU-DBS to generate halftones shown in Fig 5. (a) Initialization filter: $c^i_{\tilde{p}\tilde{p}}$. (b) Update filter: $c^u_{\tilde{p}\tilde{p}}$. (c) Difference: $c^i_{\tilde{p}\tilde{p}} - c^u_{\tilde{p}\tilde{p}}$. Both filters (a) and (b) are HVS filter autocorrelation functions based on Näsänen's model with (a) $R = 300$ dpi, $D = 10$ in. (b) $R = 300$ dpi, $D = 24$ in.

It can be noted that there is a great degree of similarity between (19)-(24) and (6)-(17). However, because the update-filter is different from that used for initialization, there are certain differences which lead to the formation of stochastically distributed clusters with CLU-DBS. An intuitive reasoning behind the cluster formation with CLU-DBS is given here.

Consider, for example, the random halftone of Fig. 5(a) (size 30×30) as the initial halftone for a constant-tone image with absorptance 0.30. The objective is to transform this initial halftone into an aperiodic, clustered-dot halftone. Figure 5(c) shows the $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ image corresponding to the initial halftone error. Because $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ is computed by filtering the initial halftone error with a low-pass filter $c^i_{\tilde{p}\tilde{p}}[\mathbf{m}]$, it captures the dot density information from that halftone. It can be seen that $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ contains a peak in regions where the halftone dot density is high, and a valley in regions where the halftone dot density is low. The magnitudes of peaks and valleys are shown in the side color bar in Fig. 5(c). One implication of the use of an update-filter different from that used for initialization is that the values of $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ do not truly represent a filtered version of the halftone error at any stage in the optimization process, except at initialization when $c_{\tilde{p}\tilde{e}}[\mathbf{m}] \equiv c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$.

Figures 6(a) and 6(b) show the initialization- and update-filters, respectively, used in this example. The update-filter is chosen to have a larger scale parameter $S = RD$ (see (2)) than the initialization-filter. Since both filters are normalized to sum to 1, the larger scale parameter of the update-filter causes it to be wider and flatter than the initialization-filter. Figure 6(c) shows the difference $c^i_{\tilde{p}\tilde{p}}[\mathbf{m}] - c^u_{\tilde{p}\tilde{p}}[\mathbf{m}]$ between the two filters. It shows that for any such combination of filters, the difference is maximum at the center and decays with increasing distance from the center, becoming slightly negative outside the main lobe. Moreover, it can be inferred that as the difference between the sizes of the two filters increases, the difference in their peak values also increases. This difference, as we shall see later, plays a vital role in our algorithm. Note that it is not necessary to use Näsänen's HVS model for the filters, one can use other low-pass filters such as the Gaussian filter and the raised cosine filter. In fact, we primarily use Gaussian filters in our later experiments to design better quality halftones. However, regardless of the type of filter used, the principles governing CLU-DBS halftoning remain the same.

Next, we explain how the toggle and swap operations of CLU-DBS in the dual-filter setting transform the halftone into a clustered-dot pattern.

A. Effect of a Toggle Operation

First, we consider the effect of a toggle operation. As explained earlier, $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ shows a peak wherever the density of dots in the initial halftone is high, and a valley wherever the density of dots is low. From (19), it is evident that a toggle may be acceptable at $\mathbf{m}_0$ if $g[\mathbf{m}_0] = 1$ and $c_{\tilde{p}\tilde{e}}[\mathbf{m}_0] > \frac{1}{2}c^u_{\tilde{p}\tilde{p}}[\mathbf{0}]$, or if $g[\mathbf{m}_0] = 0$ and $c_{\tilde{p}\tilde{e}}[\mathbf{m}_0] < -\frac{1}{2}c^u_{\tilde{p}\tilde{p}}[\mathbf{0}]$. If this toggle is performed, an update of $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ according to (21) will reduce the magnitude of the valley or peak at $\mathbf{m}_0$. This phenomenon is similar to DBS. However, the point of interest here is the effect of using a filter with larger size for update than that used for initialization. Let us analyze the change in value of $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ in CLU-DBS following a toggle at $\mathbf{m}_0$ where $g[\mathbf{m}_0] = 1$ before the toggle, and hence $a_0 = -1$. Initially,

$$\begin{aligned} c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] &= \sum_{\mathbf{n}} e_0[\mathbf{n}] c^i_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{n}] \\ &= \sum_{\mathbf{n}\neq\mathbf{m}_0} e_0[\mathbf{n}] c^i_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{n}] + e_0[\mathbf{m}_0] c^i_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_0]. \quad (25) \end{aligned}$$

The update equation for $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ at an arbitrary pixel $\mathbf{m}$ corresponding to a toggle at $\mathbf{m}_0$ is obtained by substituting $a_0 = -1$ in (21) as follows:

$$\begin{aligned} c'_{\tilde{p}\tilde{e}}[\mathbf{m}] &= c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] - c^u_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_0] \\ &= \sum_{\mathbf{n}\neq\mathbf{m}_0} e_0[\mathbf{n}] c^i_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{n}] + e_0[\mathbf{m}_0] c^i_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_0] \\ &\quad - c^u_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_0]. \quad (26) \end{aligned}$$

In particular, the update-equation for $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}_0]$ corresponding to a toggle at $\mathbf{m}_0$ is obtained by substituting $\mathbf{m} = \mathbf{m}_0$ in (26) as follows:

$$\begin{aligned} c'_{\tilde{p}\tilde{e}}[\mathbf{m}_0] &= c_{\tilde{p}\tilde{e}_0}[\mathbf{m}_0] - c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] \\ &= \sum_{\mathbf{n}\neq\mathbf{m}_0} e_0[\mathbf{n}] c^i_{\tilde{p}\tilde{p}}[\mathbf{m}_0-\mathbf{n}] + e_0[\mathbf{m}_0] c^i_{\tilde{p}\tilde{p}}[\mathbf{0}] - c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] \end{aligned} \quad (27)$$

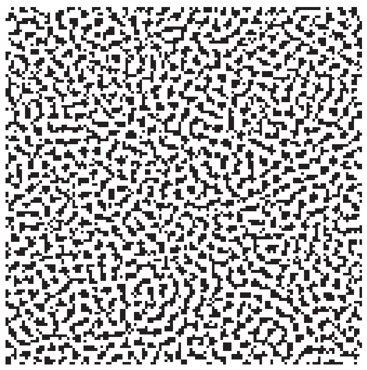
where $0 \leq e_0[\mathbf{m}_0] \leq 1$. The second term of (25) suggests that the contribution of the initial dot at $\mathbf{m}_0$ to the initial filtered error $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ can be characterized as a distribution of the error at $\mathbf{m}_0$ to its neighborhood[1] weighted by the values of filter $c^i_{\tilde{p}\tilde{p}}[\mathbf{m}]$. Given that the update is being performed with a filter that is flatter than the initialization-filter, a residual effect of the original dot at $\mathbf{m}_0$ remains in the updated filtered error $c'_{\tilde{p}\tilde{e}}[\mathbf{m}]$ in the neighborhood of $\mathbf{m}_0$, as suggested by (26). The residual effect is most conspicuous at $\mathbf{m}_0$ where the filters are centered, and where the difference in filter values is maximum, as observed from (27). In short, the reduction in magnitude of $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ around $\mathbf{m}_0$ is not commensurate with the initial contribution of the dot at $\mathbf{m}_0$. Thus, every time a toggle is performed to reduce the magnitude of $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ at a peak or valley, a residual influence of the initial dot or hole there inevitably remains. Consequently, a homogeneous, dispersed-dot halftone does not solve the optimization problem being attempted by CLU-DBS. As the optimization progresses, gradually all the dots lying in a peak region of $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$, and all the holes lying in a valley region of $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ are toggled in an attempt to reduce the magnitude of peaks and valleys, until the toggle convergence condition (19) is reached. The toggle convergence condition, in the CLU-DBS context, simply implies that there are no more dots/holes available in a peak/valley region that could be toggled to reduce the magnitude of that peak/valley. Hence, the pattern of dots and holes that results is a stochastic, clustered-dot pattern.

In Fig. 5(b), which shows the final halftone obtained by performing toggles on the initial halftone of Fig. 5(a), we see the formation of clusters and voids comprising a stochastic, clustered-dot pattern. Figure 5(d) shows the $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ corresponding to the final halftone. It can be observed that clusters in the final halftone correspond to valleys in the final $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$, and voids correspond to peaks. It is interesting to observe that the locations and shapes of clusters in the final halftone depend on the dot distribution in the initial halftone. Regions of low dot density in the initial halftone consist of clusters in the final halftone, and regions of high dot density in the initial halftone consist of voids in the final halftone. As explained above, this phenomenon is a consequence of the incomplete neutralization in the final $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ of the peaks and valleys of the initial $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$. In later sections, this phenomenon is referred to as the *inversion phenomenon*.

B. Effect of a Swap Operation

Now, we analyze the effect of a swap operation on clustering. In Fig. 7(a), we observe that clusters obtained as a result of performing only toggle operations are not compact. There are isolated dots near the edges of clusters, and isolated holes near the edges of voids. This non-compactness adds undesired high frequency noise to the halftone, which leads to unstable printing. We noted in the analysis of the effect of toggles on $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ that a cluster in the halftone corresponds to a residual valley in $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ and a void corresponds to a residual peak. Since $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ is a continuous[2] surface, the transition from peak to valley has to pass through zero. We refer to this transition region between a peak and a valley where $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ values are in the range $[-\frac{1}{2}c^u_{\tilde{p}\tilde{p}}[\mathbf{0}], \frac{1}{2}c^u_{\tilde{p}\tilde{p}}[\mathbf{0}]]$ as the *zero region*. In that region, the halftone may contain a dot or a hole without violating any of the toggle convergence conditions. This ambiguity about the zero region is the cause of isolated dots or holes that remain there after toggle-only optimization. On the other hand, a halftone optimized with a combination of both toggle and swap operations is mostly free from this defect, as can be seen in Fig. 7(b). Swaps help to merge these isolated objects into neighboring clusters or voids. How this happens is explained below.

Consider an isolated dot at $\mathbf{m}_0$ in the neighborhood of a cluster. Let $\mathbf{m}_1$ be a hole that separates the dot at $\mathbf{m}_0$ from the cluster. It was stated in the discussion of (20) that a swap may be accepted if the local gradient in the filtered error value is greater than the gradient in the filter value at the origin, and the effect of a swap on the filtered error is to reduce the magnitude of that gradient. This is similar to (12) of conventional DBS. The only difference here is that the threshold for the gradient is defined in terms of the update-filter $c^u_{\tilde{p}\tilde{p}}[\mathbf{m}]$. Since $\mathbf{m}_1$ lies between a valley (cluster) and a zero region at $\mathbf{m}_0$, $c_{\tilde{p}\tilde{e}}[\mathbf{m}_1]$ is less than $c_{\tilde{p}\tilde{e}}[\mathbf{m}_0]$ due to the assumed continuity of the


(a)

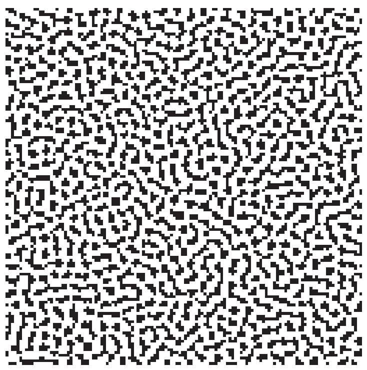

(b)

Fig. 7. Clustering attainable through use of (a) toggle operations only and (b) both toggle and swap operations. The input image absorptance is 0.30, and the initialization- and update-filters are Gaussian filters with standard deviation values $\sigma^i = 1.5$ pixels and $\sigma^u = 2.0$ pixels, respectively.

[1]The neighborhood of a pixel includes the pixel itself and pixels within close proximity.

[2]In the strict mathematical sense, $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ is not continuous, because it is defined on a discrete space. But it is continuous in the sense that unlike a binary halftone, it can assume any real value between -1 and 1, and being a low-pass filtered version of halftone error, it has relatively slow variation.

filtered error surface. Also, since $c^u_{\tilde{p}\tilde{p}}[\mathbf{m}]$ is a wide filter, the threshold $c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] - c^u_{\tilde{p}\tilde{p}}[\mathbf{m}_1 - \mathbf{m}_0]$ is quite small. For both these reasons, the condition specified in (20) is much more likely to be satisfied than the corresponding condition (12). Therefore, the likelihood of an isolated dot at $\mathbf{m}_0$ merging into the neighboring cluster by swapping with $\mathbf{m}_1$ is higher with CLU-DBS than with conventional DBS.

From the above discussion on the comparative effect of swaps and toggles, we have seen that the condition for convergence with swaps is stronger than that with toggles, if the initial halftone has the desired number of dots. Experimentally also, it is found that swap-only CLU-DBS generates a halftone of the same quality as the toggle-and-swap CLU-DBS, but with possibly a larger number of iterations, because a single swap results in a smaller change to the filtered halftone image than does a single toggle.

IV. CLU-DBS Cost Metric

In Sec. III, we discussed intuitively why CLU-DBS generates clustered-dot texture. In this section, we examine the exact cost function that is minimized in the course of optimization and discuss its implications for halftone texture.

When we perform CLU-DBS optimization, the cost metric that is minimized can be expressed as follows[3]:

$$\theta = \theta_{homog} + \theta_{clust} + \theta_{const} \tag{28}$$

where

$$\theta_{homog} = \phi^u \tag{29}$$

$$\theta_{clust} = 2\sum_{\mathbf{m}} e[\mathbf{m}]\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \tag{30}$$

$$\theta_{const} = \phi^u_0 - \phi^i_0. \tag{31}$$

Here ϕ^u represents the conventional DBS cost metric computed with the update-filter $c^u_{\tilde{p}\tilde{p}}[\mathbf{m}]$, and ϕ^i_0 and ϕ^u_0 represent conventional DBS cost metric values computed with the initialization- and update-filters, respectively, for the initial error $e_0[\mathbf{m}]$ (see (4a)). The function

$$\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] = \sum_{\mathbf{n}} e_0[\mathbf{n}]\Delta c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{n}] \tag{32}$$

is the cross-correlation between $e_0[\mathbf{m}]$ and $\Delta c_{\tilde{p}\tilde{p}}[\mathbf{m}] = c^i_{\tilde{p}\tilde{p}}[\mathbf{m}] - c^u_{\tilde{p}\tilde{p}}[\mathbf{m}]$, the difference between the initialization- and update-filters. It can also be expressed as follows:

$$\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] = c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}] - c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}]. \tag{33}$$

Figure 6 shows an example of these filters and their difference. The role of the above-mentioned error terms is discussed below.

Of the three error terms, θ_{const} is dependent only on the initial halftone and therefore remains constant throughout the optimization. It does not play any role in controlling the texture. Hence, we drop this term from the total cost metric, and consider the cost metric to be the sum of the two other terms: $\breve{\theta} = \theta_{homog} + \theta_{clust}$. These two terms play complementary roles in generating a homogeneous, clustered-dot texture. The term θ_{homog} is similar to the conventional DBS cost metric. Hence, it leads to a homogeneous distribution of dots. The term θ_{clust}, as defined in (30), represents the inner-product of the error $e[\mathbf{m}]$ at any stage and the function $\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ that does not change during the iterations. The term θ_{clust} is responsible for clustering. How $\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ controls clustering is explained next.

[3]See proof in the Appendix.

Figure 6 shows that $\Delta c_{\tilde{p}\tilde{p}}[\mathbf{m}]$ can be considered as a bandpass smoothing filter. Hence, $\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ can be considered a smoothed version of the initial error. So it captures low and high dot density information from the initial error in the form of peaks and valleys. The term θ_{clust} can be minimized if we minimize pixel-wise the inner-product of $\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ and $e[\mathbf{m}]$. The value of $e[\mathbf{m}]$ is negative where the halftone pixel value is 0 (hole) and positive where it is 1 (dot). Minimization of θ_{clust}, therefore, leads to a pattern of clustered-dots and holes corresponding to locations of valleys and peaks, respectively, in the $\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ profile. This formally explains the inversion phenomenon observed in Sec. III, where it was described intuitively as a consequence of the difference in shape between the two filters. Thus, the combination of θ_{homog} and θ_{clust} as a cost metric leads to a homogeneous distribution of clustered dots.

Although we started with an ad-hoc method to generate aperiodic, clustered-dot texture, we arrived at a dual-component cost function. One component of this cost function contributes to homogeneity, and the other contributes to clustering. Note that the use of a dual-component cost function is not a new concept. Lau *et al.* [15] and Damera-Venkata *et al.* [21] both suggested the use of a Gaussian filter, in addition to their spatial statistics-based clustering functions, for penalizing non-homogeneity in cluster distribution and cluster size. This further confirms our interpretation of how the cost function for our new method generates aperiodic, clustered-dot textures.

The preceding interpretation of the two components of the cost metric is also supported by empirical data. Figure 8 shows the process whereby a given initial halftone is optimized with CLU-DBS. The first and the last images in the sequence are the initial and the final halftones, respectively. All the other images are intermediate halftones generated in the course of the optimization. For each intermediate halftone and the final halftone, the value of the cost metric $\breve{\theta} = \theta_{homog} + \theta_{clust}$, the iteration index, and the change in value of both components of the cost metric with respect to the previous intermediate halftone, are shown. It can be inferred from the data in Fig. 8 that initially when the halftone is highly structured, the cost of inhomogeneity is very high. Hence, there is a great potential for decrease in the value of θ_{homog}. Consequently, in the early iterations, θ_{homog} plays the main role in deciding upon the halftone changes. Later, when sufficient dispersedness is introduced in the halftone, the potential for reduction in θ_{homog} becomes comparatively less than that for θ_{clust}, which in turn, tends to clump dots and holes together in the regions of valleys and peaks, respectively, of $\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$, as indicated by (30). Hence in the later iterations, θ_{clust} plays the decisive role in making the halftone changes that form clusters and voids.

Figure 9 gives a more complete picture of how the values of the cost metric and its two components change as a function of the iteration index. Here a white noise initial halftone with average absorptance 0.25 is optimized with Gaussian filters

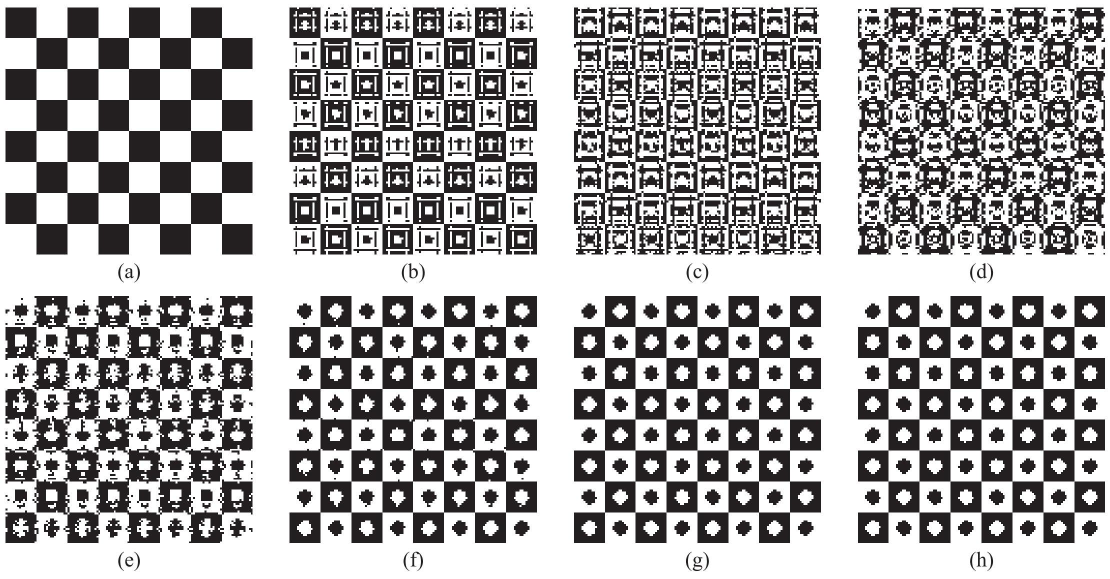


Fig. 8. (b)–(g) Sequence of intermediate halftones obtained during the optimization of the (a) initial halftone leading to the (h) final halftone. The input image has a constant tone of 0.5. The initialization- and update-filters used are Näsänen's HVS model-based autocorrelation functions with a resolution of 300 dpi, and viewing distances of 6 and 16 in, respectively. The optimization cost metric is $\breve{\theta} = \theta_{homog} + \theta_{clust}$; the constant term θ_{const} is excluded. Note that $\Delta\theta_{homog}$ plays the main role in earlier iterations bringing in dispersedness to the halftone. Later $\Delta\theta_{clust}$ dominates $\Delta\theta_{homog}$, leading to cluster formation. (a) $\breve{\theta}^{(0)} = 1,804.9$. (b) $\breve{\theta}^{(4)} = 154.8$; $\Delta\theta_{homog} = -1,112.0$; $\Delta\theta_{clust} = -538.3$. (c) $\breve{\theta}^{(8)} = -517.0$; $\Delta\theta_{homog} = -187.1$; $\Delta\theta_{clust} = -484.7$. (d) $\breve{\theta}^{(12)} = -736.3$; $\Delta\theta_{homog} = 97.6$; $\Delta\theta_{clust} = -316.9$. (e) $\breve{\theta}^{(16)} = -852.9$; $\Delta\theta_{homog} = 18.5$; $\Delta\theta_{clust} = -135.1$. (f) $\breve{\theta}^{(20)} = -888.5$; $\Delta\theta_{homog} = 6.7$; $\Delta\theta_{clust} = -42.3$. (g) $\breve{\theta}^{(24)} = -890.7$; $\Delta\theta_{homog} = -4.3$; $\Delta\theta_{clust} = 2.1$. (h) $\breve{\theta}^{(26)} = -890.7$; $\Delta\theta_{homog} = 0.001$; $\Delta\theta_{clust} = -0.007$.

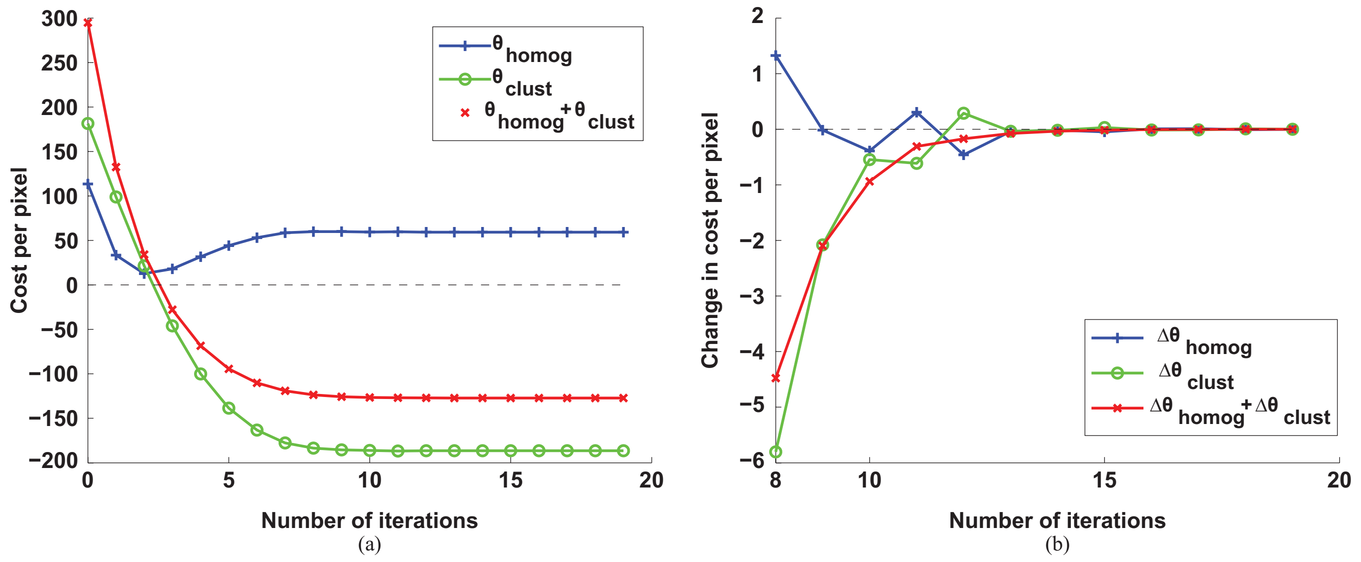


Fig. 9. Graphs showing (a) value and (b) change in value, of the cost metric $\breve{\theta}$ as a function of iteration for halftoning a constant image with absorptance 0.25. The ordinate in (a) represents the value of the cost metric normalized by image size (128 × 128) and the ordinate in (b) represents the change in this value in the corresponding iteration. Note that in (b) abscissa is clipped from the left for better visibility. The filters used are Gaussian functions with standard deviations $\sigma^{i} = 1.5$ pixels and $\sigma^{u} = 2.0$ pixels. Note that the value of the cost metric becomes negative as the optimization progresses because the term θ_{const} is not included.

having standard deviations 1.5 pixels and 2.0 pixels for the initialization and update steps, respectively. As observed in Fig. 9(a), the value of θ_{homog} at first decreases to improve the homogeneity of the initial halftone, but is then forced to increase because of the large potential for decrease in the value of θ_{clust}. Decreasing the value of θ_{clust} results in clustering. However, when the desired level of clustering is attained, the value of θ_{homog} decreases slightly to rectify the remnant non-homogeneities in clustering which could not be accomplished with θ_{clust}. These minute variations in θ_{homog} and θ_{clust} can be seen more clearly in Fig. 9(b) which plots the *change* in the value of cost metric and its two components as a function of iteration. A positive value for $\Delta\theta_{homog}$ indicates a decrease in homogeneity, and a negative value indicates an improvement. The relationship between $\Delta\theta_{clust}$ and clustering is similar. A positive value for $\Delta\theta_{homog}$ in Fig. 9(b) is justified

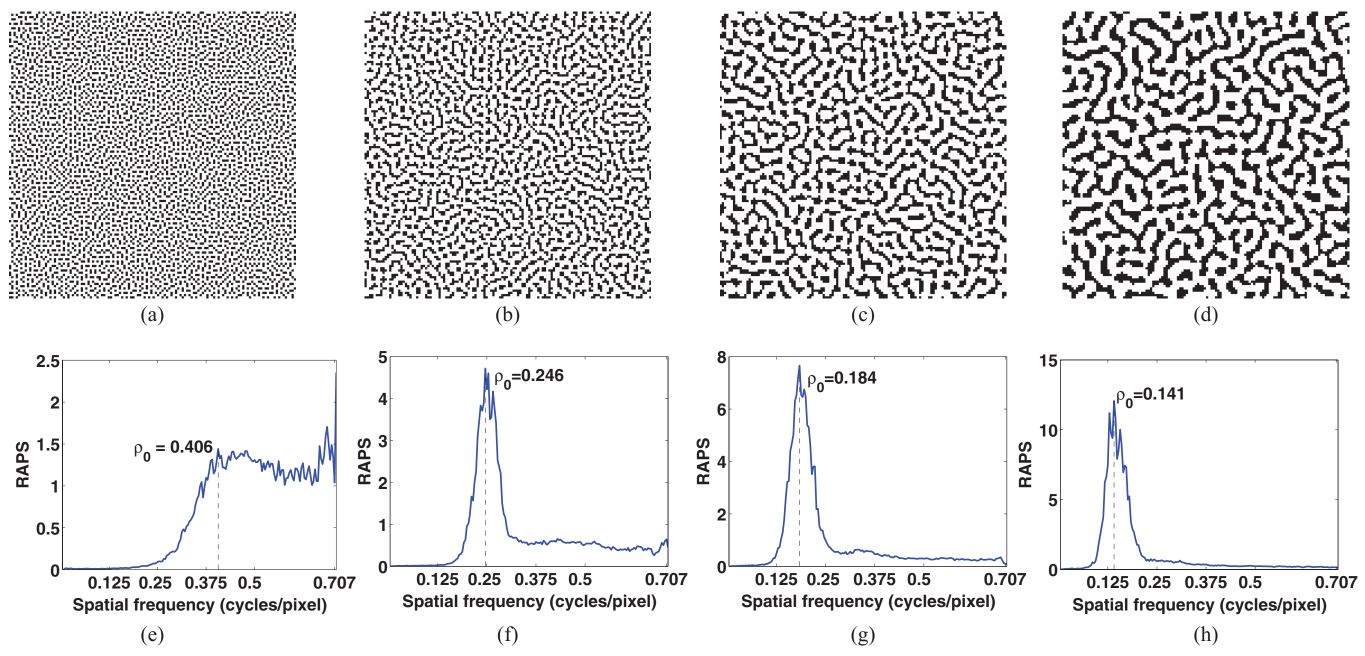


Fig. 10. Halftone textures for absorptance 0.30 and RAPS curves corresponding to different size Gaussian filters with standard deviations $\sigma^i = 1.5$ pixels and different values for σ^u. (a) and (e) $\sigma^u = 1.5$ pixels (conventional DBS), (b) and (f) $\sigma^u = 1.8$ pixels, (c) and (g) $\sigma^u = 2.2$ pixels, and (d) and (h) $\sigma^u = 2.8$ pixels. The parameter $\bar{\rho}$ denotes the principal frequency as defined by [2] and [15]. Note that as the difference between σ^u and σ^i increases beyond zero, the halftone patches exhibit clustering; and the halftone texture becomes increasingly coarse. Correspondingly, the radially averaged power spectra exhibit a decreasing principal frequency and a roll-off in amplitude beyond the principal frequency.

by a negative value for $\Delta\theta_{clust}$ if the total change is still negative. The graphs further show that convergence effectively takes place in about 10 iterations, which is comparable to conventional DBS.

A. Relation Between Cost Metric and Halftone Coarseness

The term θ_{clust} can be analyzed from another viewpoint to see how it influences cluster formation. One of its most important effects on halftone texture is in determining coarseness. The coarseness of clusters depends on the spread of $\Delta c_{\tilde{p}\tilde{p}}[\mathbf{m}]$, which in turn depends on the difference between the sizes of $c^i_{\tilde{p}\tilde{p}}[\mathbf{m}]$ and $c^u_{\tilde{p}\tilde{p}}[\mathbf{m}]$. The greater the difference between these filter sizes, the greater the average cluster size. This is because a greater difference between the filter sizes causes $\Delta c_{\tilde{p}\tilde{p}}[\mathbf{m}]$ to have a smaller peak and a wider spread, causing the formation of large valleys and peaks in the term $\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$. Hence, the resulting clusters and voids are also large. Conversely, the clusters and voids are small if the difference between the filter sizes is small. In the extreme case, when the sizes of both filters are the same, $\Delta c_{\tilde{p}\tilde{p}}[\mathbf{m}] = 0$, resulting in the formation of dispersed-dot texture (conventional DBS). Figure 10 illustrates these relationships.

B. Modifying the Cost Metric to Prevent Inversion

Another aspect of how θ_{clust} controls clustering is seen in the correlation of the locations of clusters and voids in the final halftone with the locations of low and high dot density regions, respectively, in the initial halftone. This inversion phenomenon was informally explained in Sec. III, and demonstrated in Fig. 5. As explained earlier, $\breve{\theta}$ is minimized when θ_{clust} is minimized, and θ_{clust} is minimized when the final halftone develops clusters and voids in the regions of the valleys and peaks, respectively, of the $\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ profile, thus necessitating the inversion.

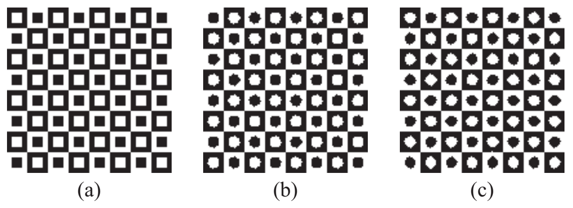


Fig. 11. Example showing the impact of changing the sign of θ_{clust} in the cost metric. (a) Input image has a constant absorptance of 0.50; the filters used are Näsänen's HVS model-based filters with a resolution 300 dpi, and viewing distances of 6 and 16 in, respectively. (b) Optimization with $\breve{\theta}_+ = \theta_{homog} + \theta_{clust}$ causes inversion in the final halftone with respect to the initial halftone (a), whereas (c) optimization with $\breve{\theta}_- = \theta_{homog} - \theta_{clust}$ does not cause inversion.

However, if we change the sign of θ_{clust} in the expression for $\breve{\theta}$, then $\breve{\theta}$ will be minimized when θ_{clust} is maximized. Minimizing the cost metric $\breve{\theta}_- = \theta_{homog} - \theta_{clust}$ will then generate clusters and voids matching the $\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ profile, without inversion. In that situation, clusters and voids in the final halftone will develop in the high and low dot density regions, respectively, of the initial halftone. Figure 11 illustrates the impact of changing the sign of θ_{clust} on cluster formation. Figure 11(a) shows the initial halftone that is input to CLU-DBS. Figure 11(b) shows the halftone that results when the cost function is $\breve{\theta}_+ = \theta_{homog} + \theta_{clust}$; and Fig. 11(c) shows the halftone that results when the cost function is $\breve{\theta}_- = \theta_{homog} - \theta_{clust}$. The inversion with respect

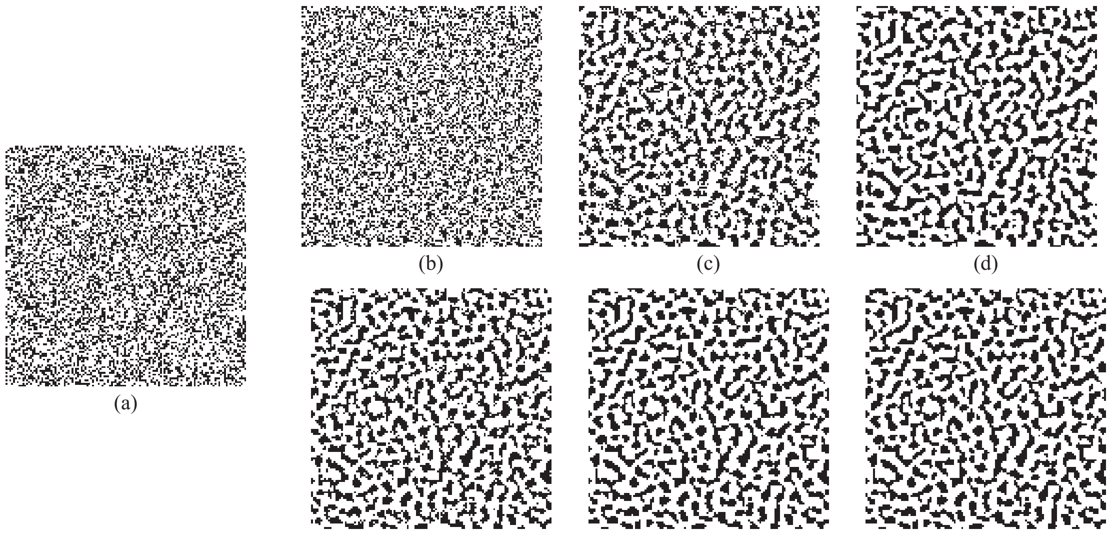


Fig. 12. Development of texture in (a) initial halftone as a function of number of iterations with the (b)–(d) cost metric $\breve{\theta}_+ = \theta_{homog} + \theta_{clust}$, and with the (e)–(g) cost metric $\breve{\theta}_- = \theta_{homog} - \theta_{clust}$. Halftones in (d) and (g) are the final halftones of the respective optimizations. The halftones (b) and (e), and (c) and (f) are the intermediate halftones obtained after four and eight iterations, respectively. The input image has a constant absorptance of 0.30 and size 128×128. The initialization- and update-filters used are Gaussian filters with $\sigma^i = 1.5$ pixels and $\sigma^u = 3.5$ pixels.

Fig. 13. Halftones optimized with the two error metrics $\breve{\theta}_+$ and $\breve{\theta}_-$ as shown in Fig. 12 are superposed to clearly demonstrate the development of complementary cluster distributions by the two approaches. White pixel: no dot in either of the halftones. Gray pixel: a dot in only one of the halftones. Black pixel: a dot in both the halftones.

TABLE I

HALFTONE CHANGE STATISTICS FOR THE OPTIMIZATION METRICS $\theta_{homog} + \theta_{clust}$ AND $\theta_{homog} - \theta_{clust}$ WITH A WHITE NOISE INITIAL HALFTONE FOR INPUT ABSORPTANCE 0.30

Statistics	**A:** $\theta_{homog} + \theta_{clust}$	**B:** $\theta_{homog} - \theta_{clust}$	Percent Reduction of B Over A
Total number of trial changes per pixel	72.3	44.7	38.2%
Total number of accepted changes per pixel	0.429	0.164	61.8%
Total number of iterations	17	12	29.4%

to Fig. 11(a) that is present in Fig. 11(b) has been eliminated in Fig. 11(c).

Figure 12 illustrates the same phenomenon when CLU-DBS is initialized with a binary white noise image with average absorptance 0.30. It shows the intermediate images that result with the two cost functions $\breve{\theta}_+ = \theta_{homog} + \theta_{clust}$ and $\breve{\theta}_- = \theta_{homog} - \theta_{clust}$ after 4 and 8 iterations, as well as the final halftone images obtained at convergence. Comparing the evolution of CLU-DBS under the two cost functions, we see that the final halftone output for $\breve{\theta}_+$ is approximately inverted with respect to that for $\breve{\theta}_-$. In addition, the process converges more quickly with $\breve{\theta}_-$ than with $\breve{\theta}_+$. Figure 13 shows a superposition of the final halftones resulting from optimization with $\breve{\theta}_+$ and $\breve{\theta}_-$. A white pixel indicates that there is no dot in either of the halftones. A gray pixel indicates that there is a dot in only one of the halftones. A black pixel indicates that there are dots in both halftones. The absence of black pixels indicates that the clusters are completely dot-off-dot. Table I presents statistics for the CLU-DBS process used to generate the halftone images shown in Fig. 12 under the two cost functions. We see that changing the sign of the second term in the cost function reduces the total number of trial changes by 38.2%, the total number of accepted changes by 61.8%, and the number of iterations required for convergence by 29.4%. Considering the computational complexity of CLU-DBS, these are very significant gains.

V. CONCLUSION

In this paper, we presented a new algorithm for stochastic, clustered-dot halftoning with DBS, that we call CLU-DBS. The DBS framework, consisting of initialization with the filtered halftone error and subsequent updates of the halftone with toggles and swaps to minimize a cost metric, has been modified by introducing the use of different filters in the initialization and update steps. First, the reason why clustered-dot textures result with this modification was discussed qualitatively.

Then, a closed-form expression for the cost metric minimized by CLU-DBS was derived. Minimization of this cost metric with a given initial halftone results in the same output halftone as the algorithm based on using different filters in the conventional DBS framework. The cost metric consists of the sum of three terms. The first two terms account for homogeneity and clusteredness of the halftone texture. The third term depends only on the initial halftone, and therefore does not impact the course of the CLU-DBS algorithm, as it attempts to improve the quality of the halftone texture.

By changing the sign of the term associated with clustering, we showed that it is possible to eliminate the *inversion* phenomenon, whereby areas in the initial halftone image with a low density of dots become clusters in the final halftone, and vice versa. This change leads to much more rapid convergence of the CLU-DBS algorithm. It also makes CLU-DBS better suited for screen design based on individual design of the binary textures corresponding to each gray-level, subject to the stacking constraint.

In this paper, we have only discussed the generation of CLU-DBS patterns for constant-tone patches. The process that we described can also be used as the basis for design of a monochrome screen for halftoning a spatially varying image. By designing a separate screen for each colorant plane, this process can further support the halftoning of color images. The screens may be designed jointly to minimize the appearance of stochastic moiré due to the superposition of more than one color plane. However, the success of such an approach depends on being able to maintain accurate registration among the individual color planes, which is something that not all marking engines can do. Finally, it is also possible to explicitly halftone a spatially varying image using CLU-DBS. However, direct application of the process described in this paper may not result in the best texture quality. These topics will be the subject of forthcoming papers.

Appendix
Derivation of the CLU-DBS Cost Metric (28)–(32)

In this appendix, we begin by deriving an expression for the cost metric for conventional DBS that will be used in the derivation of the cost metric for CLU-DBS. We then proceed to obtain the cost metric for CLU-DBS. The effect of a swap operation on the conventional DBS error metric ϕ, and the functions $g[\mathbf{m}]$ and $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ are described in (7), (13), and (15), respectively. Letting $[\mathbf{m}] = [\mathbf{m}_1]$ in (10), which describes the change in $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ because of a toggle at pixel $[\mathbf{m}_0]$, we get

$$c'_{\tilde{p}\tilde{e}}[\mathbf{m}_1] = c_{\tilde{p}\tilde{e}}[\mathbf{m}_1] + a_0 c_{\tilde{p}\tilde{p}}[\mathbf{m}_1 - \mathbf{m}_0]. \quad \text{(A1)}$$

Combining (A1) with (7), the change in ϕ due to swap between pixels $[\mathbf{m}_0]$ and $[\mathbf{m}_1]$ can be expressed as

$$\begin{aligned}\Delta\phi_{swap} = &\left(a_0^2 c_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2a_0 c_{\tilde{p}\tilde{e}}[\mathbf{m}_0]\right) \\ &+ a_1{}^2 c_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2a_1 c'_{\tilde{p}\tilde{e}}[\mathbf{m}_1]. \quad \text{(A2)}\end{aligned}$$

Comparing (A2), (13), and (15) with (6), (9), and (10), respectively, we may without loss of generality consider a swap operation to be a combination of two toggle operations in the discussion that follows.

Consider the effect on the conventional DBS error metric ϕ of a trial toggle at pixel $[\mathbf{m}_k]$ in the halftone image. According to (6), the change in ϕ due to this trial toggle is given by

$$\Delta\phi_k \triangleq \phi_{k+1} - \phi_k = a_k^2 c_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2a_k c_{\tilde{p}\tilde{e}_k}[\mathbf{m}_k] \quad \text{(A3)}$$

where ϕ_k denotes the value of the conventional DBS cost metric after k updates, and $c_{\tilde{p}\tilde{e}_k}[\mathbf{m}]$ denotes the filtered-error image after k updates. The term a_k is defined to be 1 if $g[\mathbf{m_k}] = 0$ before the toggle, and 0 otherwise.

If the trial toggle is accepted, which means $\Delta\phi_k < 0$, then $g[\mathbf{m}]$ and $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ are updated as follows:

$$g_{k+1}[\mathbf{m}] = g_k[\mathbf{m}] + a_k\delta[\mathbf{m} - \mathbf{m}_k] \quad \text{(A4)}$$

$$c_{\tilde{p}\tilde{e}_{k+1}}[\mathbf{m}] = c_{\tilde{p}\tilde{e}_k}[\mathbf{m}] + a_k c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{m}_k]. \quad \text{(A5)}$$

Lemma 1.1: For conventional DBS, $\forall k \geq 1$,

$$e_k[\mathbf{m}] = e_0[\mathbf{m}] + \sum_{i=0}^{k-1} a_i\delta[\mathbf{m} - \mathbf{m}_i] \quad \text{(A6)}$$

$$c_{\tilde{p}\tilde{e}_k}[\mathbf{m}] = c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{j=0}^{k-1} a_j c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{m}_j] \quad \text{(A7)}$$

$$\begin{aligned}\phi_k &\triangleq \sum_{\mathbf{m}} e_k[\mathbf{m}]\, c_{\tilde{p}\tilde{e}_k}[\mathbf{m}] \\ &= \sum_{\mathbf{m}} e_0[\mathbf{m}]\, c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\ &\quad +2\sum_{\mathbf{m}} (e_k[\mathbf{m}] - e_0[\mathbf{m}])\, c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\ &\quad +\sum_{i=0}^{k-1} a_i^2 c_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2\sum_{i=1}^{k-1}\sum_{j=0}^{i-1} a_i a_j c_{\tilde{p}\tilde{p}}[\mathbf{m}_i - \mathbf{m}_j].\end{aligned} \quad \text{(A8)}$$

Proof: (A6): Using (3c) and (A4), we get

$$\begin{aligned}e_k[\mathbf{m}] &\triangleq g_k[\mathbf{m}] - f[\mathbf{m}] \\ &= g_{k-1}[\mathbf{m}] + a_{k-1}\delta\left[\mathbf{m} - \mathbf{m}_{k-1}\right] - f[\mathbf{m}] \\ &= e_{k-1}[\mathbf{m}] + a_{k-1}\delta\left[\mathbf{m} - \mathbf{m}_{k-1}\right] \\ &= e_0[\mathbf{m}] + \sum_{i=0}^{k-1} a_i\delta[\mathbf{m} - \mathbf{m}_i].\end{aligned}$$

(A7): Combining (5) with (A6), we get

$$\begin{aligned}c_{\tilde{p}\tilde{e}_k}[\mathbf{m}] &\triangleq \sum_{\mathbf{n}} e_k[\mathbf{n}]\, c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{n}] \\ &= \sum_{\mathbf{n}} e_0[\mathbf{n}]\, c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{n}] \\ &\quad + \sum_{\mathbf{n}}\sum_{j=0}^{k-1} a_j\delta\left[\mathbf{n} - \mathbf{m}_j\right] c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{n}] \\ &= c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{j=0}^{k-1} a_j c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{m}_j].\end{aligned}$$

(A8): Substituting (A6) and (A7) in (4b), we get

$$\begin{aligned}
\phi_k &\triangleq \sum_{\mathbf{m}} e_k[\mathbf{m}]\, c_{\tilde{p}\tilde{e}_k}[\mathbf{m}] \\
&= \sum_{\mathbf{m}} \left(e_0[\mathbf{m}] + \sum_{i=0}^{k-1} a_i \delta[\mathbf{m}-\mathbf{m}_i] \right) \\
&\quad \times \left(c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{j=0}^{k-1} a_j c_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_j] \right) \\
&= \sum_{\mathbf{m}} e_0[\mathbf{m}]\, c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{\mathbf{m}} \sum_{i=0}^{k-1} a_i \delta[\mathbf{m}-\mathbf{m}_i]\, c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\
&\quad + \sum_{\mathbf{m}} \sum_{j=0}^{k-1} a_j c_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_j] e_0[\mathbf{m}] \\
&\quad + \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_i a_j c_{\tilde{p}\tilde{p}}[\mathbf{m}_i-\mathbf{m}_j] \\
&= \sum_{\mathbf{m}} e_0[\mathbf{m}]\, c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{\mathbf{m}} \sum_{i=0}^{k-1} a_i \delta[\mathbf{m}-\mathbf{m}_i]\, c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\
&\quad + \sum_{\mathbf{m}} \sum_{\mathbf{n}} \sum_{j=0}^{k-1} a_j \delta\left[\mathbf{n}-\mathbf{m}_j\right] c_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{n}] e_0[\mathbf{m}] \\
&\quad + \sum_{i=0}^{k-1} \sum_{j=0}^{k-1} a_i a_j c_{\tilde{p}\tilde{p}}[\mathbf{m}_i-\mathbf{m}_j] \\
&= \sum_{\mathbf{m}} e_0[\mathbf{m}]\, c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{\mathbf{m}} \sum_{i=0}^{k-1} a_i \delta[\mathbf{m}-\mathbf{m}_i]\, c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\
&\quad + \sum_{\mathbf{n}} \sum_{j=0}^{k-1} a_j \delta\left[\mathbf{n}-\mathbf{m}_j\right] c_{\tilde{p}\tilde{e}_0}[\mathbf{n}] \\
&\quad + \sum_{i=0}^{k-1} a_i^2 c_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2\sum_{i=1}^{k-1} \sum_{j=0}^{i-1} a_i a_j c_{\tilde{p}\tilde{p}}[\mathbf{m}_i-\mathbf{m}_j]. \qquad \text{(A9)}
\end{aligned}$$

Noting that the second and third terms on the right side of the equation above are the same and applying (A6) to those terms, we obtain

$$\begin{aligned}
\phi_k &= \sum_{\mathbf{m}} e_0[\mathbf{m}]\, c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + 2\sum_{\mathbf{m}} \left(e_k[\mathbf{m}] - e_0[\mathbf{m}]\right) c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\
&\quad + \sum_{i=0}^{k-1} a_i^2 c_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2\sum_{i=1}^{k-1} \sum_{j=0}^{i-1} a_i a_j c_{\tilde{p}\tilde{p}}[\mathbf{m}_i-\mathbf{m}_j]. \qquad \text{(A10)}
\end{aligned}$$

■

In CLU-DBS, we use different filters $c^i_{\tilde{p}\tilde{p}}[\mathbf{m}]$ and $c^u_{\tilde{p}\tilde{p}}[\mathbf{m}]$ in the initialization and update steps, respectively. Therefore, the cost metric θ minimized by the CLU-DBS algorithm is different from the cost metric ϕ that is minimized by the conventional DBS algorithm. In the initialization step, we get $c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ using $c^i_{\tilde{p}\tilde{p}}[\mathbf{m}]$ to filter the initial halftone error $e_0[\mathbf{m}]$ as follows

$$c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] = c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}] = \sum_{\mathbf{n}} e_0[\mathbf{n}]\, c^i_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{n}]. \qquad \text{(A11)}$$

Also, prior to any updates, the CLU-DBS cost metric can be expressed as

$$\theta_0 = \sum_{\mathbf{m}} e_0[\mathbf{m}]\, c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}] = \phi^i_0. \qquad \text{(A12)}$$

The symbol ϕ^i_0 denotes the conventional DBS cost metric computed with the initialization filter. During the CLU-DBS optimization, the change in the value of its cost metric θ due to a trial toggle at a pixel $[\mathbf{m}_k]$ in the halftone is evaluated according to

$$\Delta\theta_k = a_k^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2a_k c_{\tilde{p}\tilde{e}_k}[\mathbf{m}_k]. \qquad \text{(A13)}$$

This expression is the same as (6) except that we have added the subscript k to denote the fact that the toggle under consideration is the k-th one in a sequence of $k-1$ changes that have already been accepted. In addition, we have added the superscript u to the symbol for the filter. If this trial toggle is accepted, which means $\Delta\theta_k < 0$, then $g[\mathbf{m}]$ is updated according to (A4) and $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ is updated according to

$$c_{\tilde{p}\tilde{e}_{k+1}}[\mathbf{m}] = c_{\tilde{p}\tilde{e}_k}[\mathbf{m}] + a_k c^u_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_k] \qquad \text{(A14)}$$

which is the modified version of (A7). Combining (A11) with (A14), $c_{\tilde{p}\tilde{e}_{k+1}}[\mathbf{m}]$ after $k+1$ updates can also be expressed as follows:

$$\begin{aligned}
c_{\tilde{p}\tilde{e}_{k+1}}[\mathbf{m}] &= c_{\tilde{p}\tilde{e}_{k-1}}[\mathbf{m}] + a_{k-1} c^u_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_{k-1}] \\
&\quad + a_k c^u_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_k] \\
&= c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{i=0}^{k} a_i c^u_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_i] \\
&= c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{i=0}^{k} a_i c^u_{\tilde{p}\tilde{p}}[\mathbf{m}-\mathbf{m}_i]. \qquad \text{(A15)}
\end{aligned}$$

To derive the CLU-DBS cost metric θ given by (28)-(32), we combine (28)-(31) and restate the result as the theorem below.

Theorem 1.1: $\forall k \geq 1$,

$$\theta_k \triangleq \theta_{k-1} + \Delta\theta_{k-1} = \phi^u_k + 2\sum_{\mathbf{m}} e_k[\mathbf{m}]\, \Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \left(\phi^u_0 - \phi^i_0\right) \qquad \text{(A16)}$$

where $e_k[\mathbf{m}]$ and $\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ are given by (A6) and (33), respectively. Here we have made it explicit that the cost metric is being evaluated after k changes.

Proof: We use the principle of mathematical induction.

Base Case: Substituting $k = 0$ in (A13) and noting the equivalence stated in (A11), we get

$$\theta_1 \triangleq \theta_0 + \Delta\theta_0 = \theta_0 + a_0^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2a_0 c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}_0]. \qquad \text{(A17)}$$

Using (A12) and (A6) with $k = 1$, we can write

$$\theta_1 = \phi^i_0 + 2\sum_{\mathbf{m}} \left(\left(e_1[\mathbf{m}] - e_0[\mathbf{m}]\right) c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \right) + a_0^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}]. \qquad \text{(A18)}$$

Adding and subtracting the term $2\sum_{\mathbf{m}} e_1[\mathbf{m}]\, c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$, we get

$$\begin{aligned}
\theta_1 &= \phi^i_0 + 2\sum_{\mathbf{m}} e_1[\mathbf{m}]\, c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}] - 2\sum_{\mathbf{m}} e_0[\mathbf{m}]\, c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\
&\quad + 2\sum_{\mathbf{m}} e_1[\mathbf{m}] \left(c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}] - c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}]\right) + a_0^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}].
\end{aligned}$$

Substituting in the second term on the right side of the above equation for $e_1[\mathbf{m}]$ from (A6) with $k=1$, applying (4b) to the third term, and applying (33) to the fourth term yields

$$\begin{aligned}\theta_1 &= \phi_0^i + 2\sum_{\mathbf{m}} \left(e_0[\mathbf{m}] + a_0\delta[\mathbf{m}-\mathbf{m}_0]\right) c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\ &\quad -2\phi_0^i + 2\sum_{\mathbf{m}} e_1[\mathbf{m}]\,\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + a_0^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] \\ &= \phi_0^u + 2a_0 c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}_0] + 2\sum_{\mathbf{m}} e_1[\mathbf{m}]\,\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\ &\quad +(\phi_0^u - \phi_0^i) + a_0^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}].\end{aligned}$$

Applying (A3) with $k=0$ and the update filter, we get

$$\theta_1 = \phi_1^u + 2\sum_{\mathbf{m}} e_1[\mathbf{m}]\,\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \left(\phi_0^u - \phi_0^i\right) \quad \text{(A19)}$$

which is the statement of the theorem for the base case.

Induction Hypothesis: Assume that the statement of the theorem holds for $k = K \geq 1$. Thus

$$\theta_K = \phi_K^u + 2\sum_{\mathbf{m}} e_K[\mathbf{m}]\,\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \left(\phi_0^u - \phi_0^i\right). \quad \text{(A20)}$$

Induction Step: Using the induction hypothesis (A20) and (A13) for $k=K$, we obtain

$$\begin{aligned}\theta_{K+1} &\triangleq \theta_K + \Delta\theta_K \\ &= \phi_K^u + 2\sum_{\mathbf{m}} e_K[\mathbf{m}]\,\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \left(\phi_0^u - \phi_0^i\right) \\ &\quad + a_K^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2a_K c_{\tilde{p}\tilde{e}_K}[\mathbf{m}_K]. \qquad \text{(A21)}\end{aligned}$$

Applying (A8) with $k=K$ to the first term on the right side of (A21), and applying (A15) to the last term with index of summation j, $k=K-1$, and $\mathbf{m}=\mathbf{m}_K$, we get

$$\begin{aligned}\theta_{K+1} &= \sum_{\mathbf{m}} e_0[\mathbf{m}]\,c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + 2\sum_{i=1}^{K-1}\sum_{j=0}^{i-1} a_i a_j c^u_{\tilde{p}\tilde{p}}[\mathbf{m}_i - \mathbf{m}_j] \\ &\quad +2\sum_{\mathbf{m}} \left(e_K[\mathbf{m}] - e_0[\mathbf{m}]\right) c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{i=0}^{K-1} a_i^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] \\ &\quad +2\sum_{\mathbf{m}} e_K[\mathbf{m}]\,\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \left(\phi_0^u - \phi_0^i\right) + a_K^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] \\ &\quad +2a_K\left(c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}_K] + \sum_{j=0}^{K-1} a_j c^u_{\tilde{p}\tilde{p}}[\mathbf{m}_K - \mathbf{m}_j]\right). \qquad \text{(A22)}\end{aligned}$$

Adding $2\sum_{\mathbf{m}} a_K\delta[\mathbf{m}-\mathbf{m}_K]\,c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ to the third term of (A22); then combining the negative of this term with the fifth and the eighth terms of (A22), while noting that $2a_K c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}_K]$ may be expressed as $2\sum_{\mathbf{m}} a_K\delta[\mathbf{m}-\mathbf{m}_K]\,c^i_{\tilde{p}\tilde{e}_0}[\mathbf{m}]$ yields the third and the fifth terms of (A23). Finally, combining the fourth and the seventh terms of (A22), and combining the second and the eighth terms of (A22) yields the second and the fourth terms of (A23), respectively

$$\begin{aligned}\theta_{K+1} &= \sum_{\mathbf{m}} e_0[\mathbf{m}]\,c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{i=0}^{K} a_i^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] \\ &\quad +2\sum_{\mathbf{m}} \left(e_K[\mathbf{m}] + a_K\delta[\mathbf{m}-\mathbf{m}_K] - e_0[\mathbf{m}]\right) c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\ &\quad +2\sum_{i=1}^{K}\sum_{j=0}^{i-1} a_i a_j c^u_{\tilde{p}\tilde{p}}[\mathbf{m}_i - \mathbf{m}_j] + \left(\phi_0^u - \phi_0^i\right) \\ &\quad +2\sum_{\mathbf{m}} \left(e_K[\mathbf{m}] + a_K\delta[\mathbf{m}-\mathbf{m}_K]\right)\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}]. \qquad \text{(A23)}\end{aligned}$$

Substituting $e_{K+1}[\mathbf{m}]$ for $(e_K[\mathbf{m}] + a_K\delta[\mathbf{m}-\mathbf{m}_K])$ in the third and the last terms above, we get

$$\begin{aligned}\theta_{K+1} &= \sum_{\mathbf{m}} e_0[\mathbf{m}]\,c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \sum_{i=0}^{K} a_i^2 c^u_{\tilde{p}\tilde{p}}[\mathbf{0}] \\ &\quad +2\sum_{\mathbf{m}} \left(e_{K+1}[\mathbf{m}] - e_0[\mathbf{m}]\right) c^u_{\tilde{p}\tilde{e}_0}[\mathbf{m}] \\ &\quad +2\sum_{i=1}^{K}\sum_{j=0}^{i-1} a_i a_j c^u_{\tilde{p}\tilde{p}}[\mathbf{m}_i - \mathbf{m}_j] + \left(\phi_0^u - \phi_0^i\right) \\ &\quad +2\sum_{\mathbf{m}} e_{K+1}[\mathbf{m}]\,\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}].\end{aligned}$$

Applying (A8) with $k=K+1$ and the update filter, we get

$$\theta_{K+1} = \phi_{K+1}^u + 2\sum_{\mathbf{m}} e_{K+1}[\mathbf{m}]\,\Delta c_{\tilde{p}\tilde{e}_0}[\mathbf{m}] + \left(\phi_0^u - \phi_0^i\right). \qquad \text{(A24)}$$

This shows that the theorem statement holds for $k = K+1$, thus completing the proof of Theorem 1.1. ■

Acknowledgment

The authors would like to thank R. Ulichney for many helpful discussions.

References

[1] R. Ulichney, "Dithering with blue noise," *Proc. IEEE*, vol. 76, no. 1, pp. 56–79, Jan. 1988.

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

[3] D. L. Lau and G. R. Arce, *Modern Digital Halftoning*, 1st ed. Boca Raton, FL: CRC Press, 2001.

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

[5] B. E. Cooper and D. L. Lau, "Evaluation of green noise masks for electrophotographic halftoning," *Proc. SPIE, Human Vis. Electron. Imag.*, vol. 3959, no. 1, pp. 613–624, Jan. 2000.

[6] J. Stoffel and J. Moreland, "A survey of electronic techniques for pictorial image reproduction," *IEEE Trans. Commun.*, vol. 29, no. 12, pp. 1898–1925, Dec. 1981.

[7] I. Amidror, R. D. Hersch, and V. Ostromoukhov, "Spectral analysis and minimization of moiré patterns in color separation," *J. Electron. Imag.*, vol. 3, no. 3, pp. 295–317, Jul. 1994.

[8] D. L. Lau, A. M. Khan, and G. R. Arce, "Minimizing stochastic moiré in frequency-modulated halftones by means of green-noise masks," *J. Opt. Soc. Amer. A*, vol. 19, no. 11, pp. 2203–2217, 2002.

[9] R. Levien, "Output dependent feedback in error diffusion," in *Proc. IS&T's 46th Annu. Conf. Springfield, VA*, 1993, pp. 115–118.

[10] R. L. Levien, "Photographic image reproduction device using digital halftoning to screen images allowing adjustable coarseness," U.S. Patent 5 055 942, Oct. 8, 1991.
[11] R. Floyd and L. Steinberg, "An adaptive algorithm for spatial grayscale," *J. Soc. Inf. Display*, vol. 17, no. 2, pp. 75–77, 1976.
[12] P. Li and J. P. Allebach, "Clustered-minority-pixel error diffusion," *J. Opt. Soc. Amer. A*, vol. 21, no. 7, pp. 1148–1160, 2004.
[13] B. E. Cooper, "Method for halftoning using a difference weighting function," U.S. Patent 6 710 778, Mar. 23, 2004.
[14] 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, 2004.
[15] D. L. Lau, G. R. Arce, and N. C. Gallagher, "Digital halftoning by means of green-noise masks," *J. Opt. Soc. Amer. A*, vol. 16, no. 7, pp. 1575–1586, 1999.
[16] D. L. Lau and G. R. Arce, "Method and apparatus for producing halftone images using green-noise masks having adjustable coarseness," U.S. Patent 6 493 112, Dec. 10, 2002.
[17] J. P. Allebach, "Random nucleated halftone screen," *Photogr. Sci. Eng.*, vol. 22, no. 11, pp. 89–91, Mar.–Apr. 1978.
[18] S. G. Wang, "Stochastically clustered dot halftoning system," U.S. Patent 5 859 955, Jan. 12, 1999.
[19] V. Ostromoukhov and R. D. Hersch, "Stochastic clustered-dot dithering," *J. Electron. Imag.*, vol. 8, no. 4, pp. 439–445, 1999.
[20] Y. Abe, "Digital halftoning with optimized dither array," in *Proc. IEEE Int. Symp. Circuits Syst.*, Sydney, Australia, May 2001, pp. 517–520.
[21] N. Damera-Venkata and Q. Lin, "AM-FM screen design using Donut filters," *Proc. SPIE Process. Hardcopy Appl.*, vol. 5293, pp. 469–480, Jan. 2004.
[22] N. Damera-Venkata, "Dither matrix generation," U.S. Patent 7 420 709, Sep. 2, 2008.
[23] M. Analoui and J. P. Allebach, "Model-based halftoning using direct binary search," *Proc. SPIE Human Vis. Visual Process. Digit. Display III*, vol. 1666, pp. 96–108, Mar. 1992.
[24] D. J. Lieberman and J. P. Allebach, "A dual interpretation for direct binary search and its implications for tone reproduction and texture quality," *IEEE Trans. Image Process.*, vol. 9, no. 11, pp. 1950–1963, Nov. 2000.
[25] R. Nasanen, "Visibility of halftone dot textures," *IEEE Trans. Syst. Man Cybern.*, vol. 14, no. 6, pp. 920–924, Dec. 1984.

Puneet Goyal received the B.Tech. and M.Tech. degrees in computer science and engineering from the Indian Institute of Technology Delhi, New Delhi, India, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette, IN, in 2006 and 2010, respectively.

He was a Senior Member of Technical Staff with AT&T Labs, San Ramon, CA, until 2011. He is currently an Associate Professor with the Graphic Era University, Dehradun, India. His current research interests include halftoning, image forensics, and optimization algorithms.

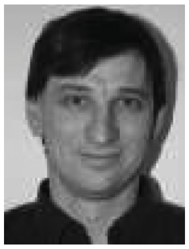
Madhur Gupta received the B.Tech. degree in computer science and engineering from the Indian Institute of Technology Delhi, New Delhi, India, and the Ph.D. degree in electrical and computer engineering from Purdue University, West Lafayette IN, in 2005 and 2010, respectively.

He is currently a Senior Member of Technical Staff with AT&T Labs, San Ramon CA. His current research interests include halftoning, image enhancement, electronic imaging systems, and optimization algorithms.

Carl Staelin received the Ph.D. degree in CS from Princeton University, Princeton, NJ, in 1991, with research on high-performance file system design.

He is currently with the Google Advanced Technology Center, Technion City, Israel. He was the Chief Technologist with HP Labs Israel, Haifa, Israel, where he was involved in digital commercial print, automatic image analysis and enhancement, and enterprise IT management. His current research interests include storage systems, machine learning, image analysis and processing, document and information management, and performance analysis.

Dr. Staelin is an Associate Editor of the *Journal of Electronic Imaging*.



Mani Fischer received the B.Sc. degree in computer science, the M.Sc. degree in mechanical engineering, and the Ph.D. degree in automatic control from Israel Institute of Technology, Technion, Israel, in 1986, 1988, and 1992, respectively.

He joined ACSTECH80 in 2001 as the Vice President of R&D. He then joined HP Labs, Haifa, Israel, in 2002, as a Principal Researcher, engaged in imaging, printing, and analytics. He holds nine U.S. patents and has 30 U.S. patents pending.

Omri Shacham received the B.Sc. degree in physics with a minor in CS in 2004 and the Master's degree in electroptics in 2006.

He specializes in image processing and worked at HP Indigo Ltd., Rehovot, Israel, for many years. He is currently a Consultant in Tel-Aviv, Israel.

Jan P. Allebach (F'91) is currently the Hewlett-Packard Distinguished Professor of electrical and computer engineering with Purdue University, West Lafayette, IN. His current research interests include image rendering, image quality, color imaging and color measurement, document aesthetics, and printer forensics.

Dr. Allebach was a recipient of the Senior (Best Paper) Award from the IEEE Signal Processing Society, the Bowman Award and the Itek Best Paper Award from IS&T, was recipient of the Electronic Imaging Scientist of the Year from IS&T and SPIE, was recipient of the Honorary Member of IS&T, the highest award that IS&T bestows, five teaching awards from Purdue University, and separate awards for team leadership, mentoring, and research. He is a fellow of the Society for Imaging Science and Technology (IS&T) and SPIE. He was a Distinguished or Visiting Lecturer of IS&T and the IEEE Signal Processing Society.