https://doi.org/10.2352/ISSN.2470-1173.2020.15.COLOR-195

New results for aperiodic, clustered-dot halftoning*

Jiayin Liu [1], *Altyngul Jumabayeva* [1], *Yujian Xu* [1], *Yin Wang* [1], *Tal Frank* [2], *Shani Gat* [2], *Orel Bat Mor* [2], *Yotam Ben-Shoshan* [2], *Robert Ulichney* [3], *and Jan Allebach* [1]; [1] *Purdue University (United States),* [2] *HP Inc. (Israel), and* [3] *HP Labs, HP Inc. (United States)*

Abstract

FM (Frequency modulation) halftoning (aperiodic and dispersed-dots) is increasingly popular with traditional analog offset lithographic printing. There is a desire from customers in the commercial market to use this capability with high-end digital presses based on electrophotographic printing (EP) technologies.

However, the inherent instability of the EP process challenges the achievement of satisfactory print quality with dispersed-dot, aperiodic halftoning. The direct binary search (DBS) algorithm is widely considered to represent the gold standard of dispersed-dot, aperiodic halftone image quality.

In this paper, we continue our previous efforts to adapt DBS from dispersed-dots to clustered-dots to use with the Indigo liquid EP printing technology; we present a new screen design algorithm for aperiodic, clustered-dot halftoning based on Direct Bineary Search. Our screen design has a very good detail rendering capability and very stable halftone frequency overall. The halftone texture after applying screen is even better than directly using our previous work, clustered-dot halftoning.

Introduction

Halftoning is the process of rendering a pattern with a limited number of tone levels to visually match a continuous-tone image. Because the human visual system (HVS) acts like a low-pass filter, the pattern should be visually the same as the original image when viewed by human eyes from an appropriate viewing distance.

Halftoning algorithms can be classified according to whether they produce dispersed-dot textures or clustered-dot textures. Dispersed-dot textures are comprised of isolated printer-addressable dots and clustered-dot textures are comprised of clustered printer-addressable dots. Halftoning algorithms also can be classified as Frequency modulation (FM) or Amplitude modulation (AM). Frequency modulation changes the density of dots: periodic or aperiodic dot spacing, but dot size is fixed. Amplitude modulation generates a periodic or aperiodic grid of dot-clusters: cluster size varies, and the spacing between clusters can be fixed or varying.

Presently, there are two dominant printing technologies, one is the electrophotographic (EP) process with a laser writing system and the other is inkjet (IJ). Clustered-dot textures have been used widely in electrophotographic printers and dispersed-dot textures have been used widely in inkjet printers.

There are three basic architectures for halftoning algorithms: screening, error diffusion, and search-based methods. Our interest in this paper is in the use of a search-based to design halftone screens; so we will not further mention error diffusion. Search-based methods usually are iterative and find the best halftone image by minimizing the perceived error between the continuous-tone and halftone image. The direct binary search (DBS) algorithm [1], [2], [3], [4], first computes the mean-squared error (MSE) between the filtered continuous-tone image and the filtered initial halftone image. Then, the algorithm scans pixel-by-pixel through the halftone image and applies toggling and swapping to minimize the MSE. The algorithm stops when the MSE cannot be further reduced. Screening is the simplest halftoning method to implement[5], [6], [7]. The continuous-tone input image is compared pixel-by-pixel to a matrix of thresholds (the screen) that is periodically tiled over the entire continuous-tone image. A dot is placed at pixels where the value of the continuous-tone image is greater than or equal to the threshold. Otherwise, the pixel is left empty in the halftone image. A screen is made by designing a halftone texture at every gray-level subject to a stacking constraint, and according to a specific sequence of gray-levels. After finishing the design at each gray-level, we update the screen thresholds to reflect the new halftone dots/holes at that gray-level. To prevent the appearance of tilling artifacts in the halftone textures produced by the screen, it is important to incorporate wrap-around properties during the screen design.

In next section, we review the background of aperiodic, clustered-dot halftoning with DBS (CLU-DBS) [8], [9], [10], [11] and the refinement of the halftone texture of CLU-DBS: Multi-Stage Multi-Pass CLU-DBS (MS-MP-CLU-DBS). After that, we present our new screen design based on MS-MP-CLU-DBS. Finally, we discuss how effective line frequency of the halftone textures depends on the parameters of the algorithm and how we calculate the effective line frequency of the halftone textures.

Preliminaries

A. Aperiodic, clustered-dot halftoning with DBS

Throughout this paper, we use $[\boldsymbol{m}] = [m,n]^T$ and $(\boldsymbol{x}) = (x,y)^T$ to represent discrete and continuous spatial coordinates, respectively. $e[\boldsymbol{m}]$ represents the error image between the halftone image $g[\boldsymbol{m}]$ and the continuous-tone image $f[\boldsymbol{m}]$.

CLU-DBS is an aperiodic, clustered-dot halftoning algorithm based on the Direct Binary Search (DBS) algorithm. CLU-DBS renders a continuous-tone image with only clustered dots.

In the DBS algorithm, we generate a dispersed-dot halftone by minimizing the $c_{\tilde{p}\tilde{e}}$ halftone error with swaps and toggles. Rather than using only one filter through out the minimization processes of DBS, we use two low-pass filters for CLU-DBS: $c^i_{\tilde{p}\tilde{p}}[\boldsymbol{m}]$ as the initial filter, will be used in initialization phase and $c^u_{\tilde{p}\tilde{p}}[\boldsymbol{m}]$ as the update filter, will be used in update phase.

Similar to DBS, we first initialize the filtered version of the error image, $c_{\tilde{p}\tilde{e}}[\boldsymbol{m}]$ with the initial filter $c^i_{\tilde{p}\tilde{p}}[\boldsymbol{m}]$:

$$c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}] = e[\boldsymbol{m}] * c^i_{\tilde{p}\tilde{p}}[\boldsymbol{m}] \tag{1}$$

* Research supported by the HP Indigo Division, Rehovot, ISRAEL.

However, the difference is that during the evaluation and update of trial toggle or swap operations, we not longer use $c^i_{\tilde{p}\tilde{p}}[\boldsymbol{m}]$. Instead, we use the update filter for the evaluation and update. For example, it can be shown [3] that a trial toggle at $\boldsymbol{m}_0$ can be accepted if

$$a_0 c_{\tilde{p}\tilde{e}}[\boldsymbol{m}_0] + \frac{1}{2} c^u_{\tilde{p}\tilde{p}}[\boldsymbol{0}] < 0, \tag{2}$$

a_0 is defined to be 1 if $g[\boldsymbol{m}_0] = 0$ before the toggle, and 1 otherwise. With this toggle, $e[\boldsymbol{m}]$, $g[\boldsymbol{m}]$, and $c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}]$ will change as follows:

$$e^{'}[\boldsymbol{m}] = e[\boldsymbol{m}] + a_0 \delta[\boldsymbol{m} - \boldsymbol{m}_0], \tag{3}$$

$$g^{'}[\boldsymbol{m}] = g[\boldsymbol{m}] + a_0 \delta[\boldsymbol{m} - \boldsymbol{m}_0], \tag{4}$$

$$c^{u'}_{\tilde{p}\tilde{e}}[\boldsymbol{m}] = c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}] + a_0 c^u_{\tilde{p}\tilde{p}}[\boldsymbol{m} - \boldsymbol{m}_0]. \tag{5}$$

Here, the $'$ denotes the version subsequent to the toggle; and the symbol without a $'$ denotes the version before the toggle.

The algorithm is said to reached convergence when for all pixels $\mathbf{m}$ in the image, if $g[\boldsymbol{m}] = 0$,

$$c^u_{\tilde{p}\tilde{p}}[\boldsymbol{0}] \geq -\frac{1}{2} a_0 c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}], \tag{6}$$

and if $g[\boldsymbol{m}] = 1$, then

$$c^u_{\tilde{p}\tilde{p}}[\boldsymbol{0}] \leq \frac{1}{2} a_0 c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}]. \tag{7}$$

Most of the CLU-DBS procedure is similar to conventional DBS; but the second filter plays an important role of generating clustered dots.

CLU-DBS cost metric

When we perform CLU-DBS optimization, it can be shown [9], [10] that the cost metric is:

$$\theta = \theta_{homog} - \theta_{clust}, \tag{8}$$

These two terms are important in generating a homogeneous, clustered-dot texture. θ_{homog} is similar to the DBS cost metric; and it leads to a homogeneous distribution of dot-clusters. θ_{homog} is the inner-product of the error $e[\boldsymbol{m}]$ at any stage and $c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}]$, which is the error image filtered with the update filter. θ_{clust} is responsible for clustering. It is the inner-product of the error $e[\boldsymbol{m}]$ at any stage and the function $\Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}]$ that does not change during the iterations.

So, we have

$$\theta_{homog} = \sum_{\boldsymbol{m}} e[\boldsymbol{m}] c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}], \tag{9}$$

$$c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}] = \sum_{\boldsymbol{n}} e[\boldsymbol{n}] c^u_{\tilde{p}\tilde{p}}[\boldsymbol{m} - \boldsymbol{n}], \tag{10}$$

$$\theta_{clust} = 2 \sum_{\boldsymbol{m}} e[\boldsymbol{m}] \Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}], \tag{11}$$

$$\Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}] = \sum_{\boldsymbol{n}} e_0[\boldsymbol{n}] \Delta c_{\tilde{p}\tilde{p}}[\boldsymbol{m} - \boldsymbol{n}], \tag{12}$$

$$e_0[\boldsymbol{n}] = g_0[\boldsymbol{n}] - f[\boldsymbol{n}], \tag{13}$$

where $e_0[\boldsymbol{n}]$ is the difference between the initial halftone $g_0[\boldsymbol{n}]$ and the continuous-tone image $f[\boldsymbol{n}]$. $\Delta c_{\tilde{p}\tilde{p}}[\boldsymbol{n}] = c^i_{\tilde{p}\tilde{p}}[\boldsymbol{n}] - c^u_{\tilde{p}\tilde{p}}[\boldsymbol{n}]$ is the difference between the initialization-filter and update-filter. From Eq. (8), we have that

$$\theta = \sum_{\boldsymbol{m}} e[\boldsymbol{m}] c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}] - 2 \sum_{\boldsymbol{m}} e[\boldsymbol{m}] \Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}], \tag{14}$$

$$\theta = \sum_{\boldsymbol{m}} \sum_{\boldsymbol{n}} e[\boldsymbol{m}] e[\boldsymbol{n}] c^u_{\tilde{p}\tilde{p}}[\boldsymbol{m} - \boldsymbol{n}] - 2 \sum_{\boldsymbol{m}} \sum_{\boldsymbol{n}} e[\boldsymbol{m}] e_0[\boldsymbol{n}] \Delta c_{\tilde{p}\tilde{p}}[\boldsymbol{m} - \boldsymbol{n}], \tag{15}$$

Considering again a trial toggle at $\boldsymbol{m}_0$, as described by Eqs. (3)-(5), we derive the change in the cost metric, as shown in the following equations. First, we consider the change in θ_{homog}. The update to $\theta^{'}_{homog}$ can be expressed as:

$$\theta^{'}_{homog} = \sum_{\boldsymbol{m}} e^{'}[\boldsymbol{m}] c^{u'}_{\tilde{p}\tilde{e}}[\boldsymbol{m}], \tag{16}$$

$$\Delta\theta_{homog} = \theta^{'}_{homog} - \theta_{homog} = \sum_{\boldsymbol{m}} ((e[\boldsymbol{m}] + a_0 \delta[\boldsymbol{m} - \boldsymbol{m}_0]) \cdot (c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}] + a_0 c^u_{\tilde{p}\tilde{p}}[\boldsymbol{m} - \boldsymbol{m}_0]) - e[\boldsymbol{m}] c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}]), \tag{17}$$

$$\Delta\theta_{homog} = \sum_{\boldsymbol{m}} (e[\boldsymbol{m}] \cdot c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}] + a_0 \delta[\boldsymbol{m} - \boldsymbol{m}_0] \cdot c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}] + e[\boldsymbol{m}] \cdot a_0 c^u_{\tilde{p}\tilde{p}}[\boldsymbol{m} - \boldsymbol{m}_0] + a_0 \delta[\boldsymbol{m} - \boldsymbol{m}_0] \cdot a_0 c^u_{\tilde{p}\tilde{p}}[\boldsymbol{m} - \boldsymbol{m}_0] - e[\boldsymbol{m}] \cdot c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}]), \tag{18}$$

$$\Delta\theta_{homog} = \sum_{\boldsymbol{m}} e[\boldsymbol{m}] \cdot c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}] + a_0 c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}_0] + a_0 c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}_0] + a_0^2 c^u_{\tilde{p}\tilde{p}}[\boldsymbol{0}] - \sum_{\boldsymbol{m}} e[\boldsymbol{m}] \cdot c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}], \tag{19}$$

$$\Delta\theta_{homog} = a_0^2 c^u_{\tilde{p}\tilde{p}}[\boldsymbol{0}] + 2 a_0 c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}_0]. \tag{20}$$

The update of $\theta^{'}_{clust}$ can be expressed as:

$$\theta^{'}_{clust} = 2 \sum_{\boldsymbol{m}} e^{'}[\boldsymbol{m}] \Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}], \tag{21}$$

$$\Delta\theta_{clust} = \theta^{'}_{clust} - \theta_{clust} = 2 \sum_{\boldsymbol{m}} (e[\boldsymbol{m}]^{'} - e[\boldsymbol{m}]) \Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}], \tag{22}$$

$$\Delta\theta_{clust} = 2 \sum_{\boldsymbol{m}} (e[\boldsymbol{m}] + a_0 \delta[\boldsymbol{m} - \boldsymbol{m}_0] - e[\boldsymbol{m}]) \Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}], \tag{23}$$

$$\Delta\theta_{clust} = 2 a_0 \Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}_0]. \tag{24}$$

Therefore,

$$\Delta\theta_{toggle} = \Delta\theta_{homog} - \Delta\theta_{clust} = a_0^2 c^u_{\tilde{p}\tilde{p}}[\boldsymbol{0}] + 2 a_0 c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}_0] - 2 a_0 \Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}_0]. \tag{25}$$

Similarly, a trial swap between $\boldsymbol{m}_0$ and $\boldsymbol{m}_1$, when for example, $g[\boldsymbol{m}_0]=0$ and $g[\boldsymbol{m}_1]=1$, can be accepted if

$$a_0(c_{\tilde{p}\tilde{e}}[\boldsymbol{m}_0]-c_{\tilde{p}\tilde{e}}[\boldsymbol{m}_1])+c^u_{\tilde{p}\tilde{p}}[\boldsymbol{0}]-c_{\tilde{p}\tilde{p}}[\boldsymbol{m}_1-\boldsymbol{m}_0]<0, \quad (26)$$

Following an accepted swap, $e[\boldsymbol{m}], g[\boldsymbol{m}]$, and $c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}]$ are updated as follows, where $a_1=-1$ if $g[\boldsymbol{m}_1]$ changes from 1 to 0 and $a_1=1$ if $g[\boldsymbol{m}_1]$ changes from 0 to 1.

$$e^{'}[\boldsymbol{m}]=e[\boldsymbol{m}]+a_0\delta[\boldsymbol{m}-\boldsymbol{m}_0]+a_1\delta[\boldsymbol{m}-\boldsymbol{m}_1], \quad (27)$$

$$g^{'}[\boldsymbol{m}]=g[\boldsymbol{m}]+a_0\delta[\boldsymbol{m}-\boldsymbol{m}_0]+a_1\delta[\boldsymbol{m}-\boldsymbol{m}_1], \quad (28)$$

$$c^{u'}_{\tilde{p}\tilde{e}}[\boldsymbol{m}]=c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}]+a_0c^u_{\tilde{p}\tilde{p}}[\boldsymbol{m}-\boldsymbol{m}_0]+a_1c^u_{\tilde{p}\tilde{p}}[\boldsymbol{m}-\boldsymbol{m}_1], \quad (29)$$

The change in error can be expressed as

$$\begin{aligned}\Delta\theta_{swap}&=\Delta\theta_{homog}-\Delta\theta_{clust}\\&=(a_0^2+a_1^2)c^u_{\tilde{p}\tilde{p}}[\boldsymbol{0}]+2a_0c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}_0]+2a_1c^u_{\tilde{p}\tilde{e}}[\boldsymbol{m}_1]\\&\quad+2a_0a_1c^u_{\tilde{p}\tilde{p}}[\boldsymbol{m}_0-\boldsymbol{m}_1]-2a_0\Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}_0]-2a_1\Delta c_{\tilde{p}\tilde{e}_0}[\boldsymbol{m}_1].\end{aligned} \quad (30)$$

B. Refinement of halftone texture of CLU-DBS

There are several factors which could influence the CLU-DBS halftone texture: the seed halftone, the filter type, the filter scale parameters, the relation between halftone frequency and filter size, Multi-Pass refinement, and Multi-Stage refinement. In this section, we only focus on the seed halftone, Multi-Pass refinement, and Multi-Stage refinement.

Multi-Pass refinement

One-pass refinement is defined as a complete refinement consisting of multiple iterations of the CLU-DBS algorithm, where the number of iterations depends on how quickly the algorithm converges to the point where no more toggles or swaps are accepted. The flowchart of one-pass refinement is shown in Figure 1.

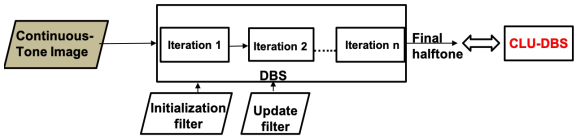


Figure 1: One-pass refinement (the basic CLU-DBS algorithm).

With Multi-Pass refinement, the halftone image is sequentially re-optimized with the original filters. The halftone input to each pass is the output halftone from the previous pass. This leads to a homogeneous texture. The flowchart of Multi-Pass refinement is shown in Figure 2. As shown in Figure 3, the halftone texture after the first pass is still very noisy with a lot of single dots. As the halftone goes through subsequent passes, the texture continues to look much better than it did after the first pass.

Multi-Stage refinement

The **seed halftone** is a homogeneous, dispersed-dot halftone with average absorptance δ. The absorptance δ of the seed halftone is related to the desired analog effective halftone line frequency ρ_a in lpi by the formula $\delta=(\frac{\rho_a}{R})^2$. Here, R is the printer resolution in dots per inch. We use 1625.6 dpi for the Indigo

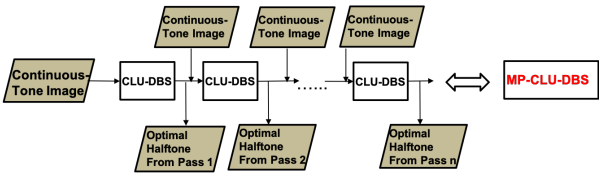


Figure 2: Multi-Pass refinement.

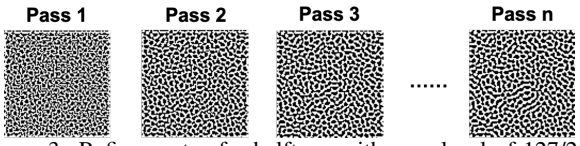


Figure 3: Refinements of a halftone with gray-level of 127/255 with the Multi-Pass option.

press. The purpose of the seed halftone is to set the cluster-centers for development of actual clusters later in the process. The seed halftone can be designed by any dispersed-dot halftoning method, including conventional DBS. We used conventional DBS with a Gaussian filter of σ value 1.3.

Given δ as the input, we generate a random halftone with gray level δ, the same size as the input image. We follow this with the conventional DBS algorithm with a swap-only option to generate a nice homogeneous dispersed-dot halftone, as shown in Figure 4.

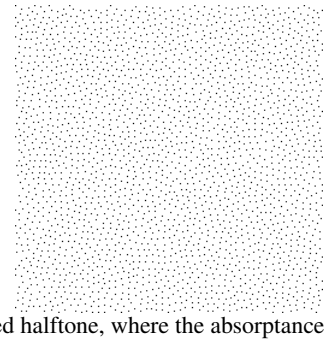


Figure 4: Seed halftone, where the absorptance δ is 7.57/255.

Multi-Stage refinement is the core part of the algorithm to make the halftone texture better. Starting with the seed halftone, we build up the continuous-tone image $f[\boldsymbol{m}]$ stage by stage. The absorptance values for the continuous-tone image in each stage are controlled by an attenuation factor. When we reach the last stage, the absorptance values for the continuous-tone image are the original absorptance values of the input continuous-tone image. The flowchart is shown in Figure 5.

At each stage k, the halftone $g^{(k-1)}[\boldsymbol{m}]$ is optimized with respect to a continuous-tone image $f^{(k)}[\boldsymbol{m}]$ to generate halftone $g^{(k)}[\boldsymbol{m}]$. Note that the continuous-tone image will generally be different from stage to stage. It is defined as an attenuated version of the original input image according to $f^{(k)}[\boldsymbol{m}]=\alpha^{(k)}f[\boldsymbol{m}]$. If total number of stages $K=5$, then the attenuation factors for the stages are $0.2, 0.4, 0.6, 0.8, 1$, respectively. The process of updating the halftone image at each stage is performed using the Multi-Pass refinement described earlier, and shown in Figure 2. After applying MS-MP-CLU-DBS with two Gaussian filters: initial filter $\sigma^i=1.3$ and the update filter $\sigma^u=1.7$, and with 5 stages and 10 passes, the halftone of a folded ramp is shown in Figure 6. There are some artifacts along the edges of the image. These are due to the fact that the version of the algorithm used to generate this

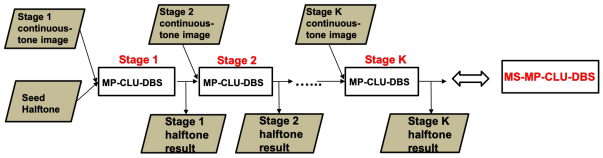


Figure 5: Multi-Stage refinement.

ramp is intended for screen design, and attempts to enforce wrap-around. Those artifacts don't appear when we halftone the ramp with a screen designed by our MS-MP-CLU-DBS algorithm.

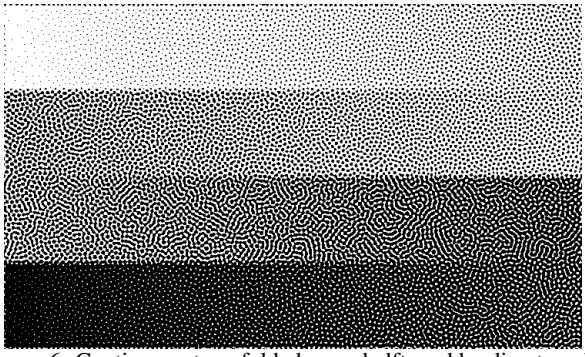


Figure 6: Continuous-tone folded ramp halftoned by direct application of MS-MP-CLU-DBS with 5 stages and 10 passes. The image size is 448×751 pixels.

So far, we have discussed DBS and CLU-DBS in the context of halftoning an arbitrary continuous-tone image $f[\boldsymbol{m}]$. However, for screen design, we are only interested in designing a halftone for a constant image $f[\boldsymbol{m}] \equiv a$ for a given absorptance level $0 \leq a \leq 1$. in the remainder of this paper, we shall only consider the case where $f[\boldsymbol{m}]$ has a constant value.

Screen Design

As discussed in the introduction, the screen is a threshold matrix that can be used to halftone any given continuous-tone image by comparing each pixel with its corresponding element in the screen. There are several methods to design a screen. Our approach is to sequentially design constant-tone halftone textures for each level, subject to a stacking constraint. One option is to design highlights first and then move to shadows. Another option is to design level 0.5 first, and then design highlights and shadows. Due to the stacking constraint, the first method would have a better quality in the highlights, and the second method would have better quality in the midtones.

In our strategy, we design the midtones first, then highlights and shadows. For an 8-bit screen, the sequence is 127, 126, 125, ... , 2, 1, 0, 128, 129, 130, ... , 254, 255. Our design obeys the stacking constraint, which means, for example, that any dots that are in gray-level 126 will stay at the same pixel location as before and appear in gray-level 127. Similarly, any dots that are in gray-level 127 will stay at the same pixel location as before and appear in gray-level 128, and so on. In this paper, we generate a square screen with size 256×256.

Step one: generate a clustered-dot halftone of gray-level 127 using MS-MP-CLU-DBS

We first generate a seed halftone with absorptance δ based on the desired analog halftone line frequency ρ_a and the formula $\delta = (\frac{\rho_a}{R})^2$. Here, we choose the absorptance δ of the seed halftone to be 7.57/255 to achieve to a final halftone frequency of 270 lpi. The seed halftone is shown in Figure 4.

We then use the seed halftone as obtained above and the MS-MP-CLU-DBS algorithm with two Gaussian filters: initial filter $\sigma^i = 1.3$ and the update filter $\sigma^u = 1.7$, and with 5 stages and 10 passes to design a 256×256 halftone patch for the constant-tone image $f[\boldsymbol{m}] = 127/255$. After the MS-MP-CLU-DBS process, we have the best quality aperiodic, clustered-dot halftone for gray-level for $0.5 \approx 127/255$. The halftone patch with gray-level 127/255 is shown in Figure 7.

Step two: add or delete a sufficient number of dots so that the gray-level reaches 0.5 exactly

The actual gray-level of the initial halftone patch designed via MS-MP-CLU-DBS may be lower or higher than 0.5. We want to make the gray-level be exactly 0.5 to perform screen design. To solve this problem, we first calculate the number of dots N_{curr} in the halftone and the difference between N_{curr} and the target number of dots N_{target}, Here, N_{target} is given by product of the halftone patch height and width divided by 2. Then we add or remove a sufficient number of dots one-by-one until the gray-level is exactly 0.5. To add a dot, we scan through the halftone image; and we evaluate trial toggle operations for the change in the total error given by Eq. (25). We find that particular toggle operation that yields the maximum reduction or minimum increase to the total error. Newly added or removed dots will be part of clusters that already exist.

Step three: design highlights

We calculate how many dots N we should remove between each pair of succeeding levels. Here, N is given by the product of the image height and width divided by the number of levels L. For our screen design, $N = 256^2 \div 256 = 256$.

Then, we remove N single dots one-by-one to form a halftone image with gray-level $0.5 - N \div 256^2 \approx 126/255$. In each iteration, to remove a single dot, we scan through the halftone image; and we evaluate trial toggle operations for the change in the total error given by Eq. (25), and find that particular toggle operation that yields the maximum reduction or minimum increase to the total error. Each time after we turn off a single dot, we write the number 127 in the screen matrix in the same location where we turned off that single dot in the halftone image. After we turn off N single dots one-by-one, we repeat this process for levels 125, 124, ... , 0. The two halftones with gray-level 126/255 and 127/255 are shown in Figure 7. The difference between these two halftones patches is shown in Figure 8.

Step four: design shadows

For levels 128, 129, ..., 255, the process is essentially the same as for levels below 127, except that we add N single dots to the halftone image at each level.

For example, to design gray-level 128, in each iteration to add a single dot, we scan through the halftone image at gray-level 127. We evaluate trial toggle operations for the change in the total error, and find that particular toggle operation that yields the maximum reduction or minimum increase to the total error. Each time after we turn on a single dot, we write the number 128 in the screen matrix in the same location where we turn on that single

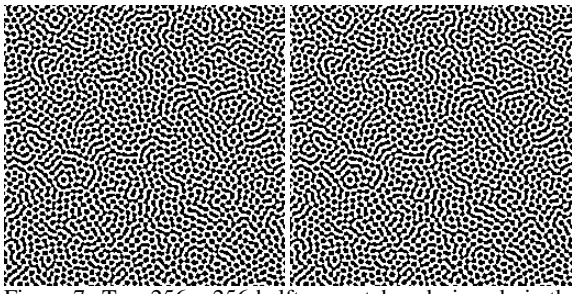


Figure 7: Two 256×256 halftone patches designed via the MS-MP-CLU-DBS algorithm for our halftone screen at (left) gray-level 127/255, and (right) gray-level 126/255.

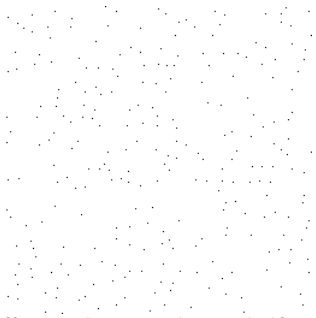


Figure 8: The difference between the two halftone patches shown in Figure 7 for gray-level 127/255 and gray-level 126/255. Starting at level 127/255, we turn off these 256 single dots during the screen design in order to reach gray-level 126/255.

dot in the halftone image. After we turn on N single dots one-by-one, we repeat this process for levels 129, 130, ... , 255. Once the design is finished, we now have an aperiodic, clustered-dot screen matrix.

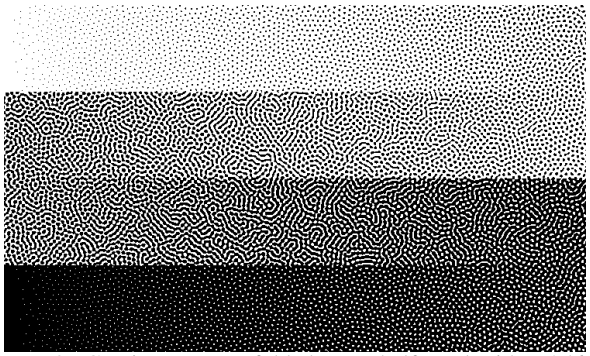


Figure 9: Continuous-tone folded ramp halftoned with a 256×256 screen designed using our MS-MP-CLU-DBS screen design algorithm. The image size is 448×751 pixels.

Results

We used our newly designed screen to halftone the same continous-tone folded ramp that we halftoned in Figure 6 by direct application of the MS-MP-CLU-DBS algorithm. The screen-based result is shown in Figure 9. Except for the boundary artifacts that can be seen in Figure 6, and which, as discussed previously, are spurious, the quality of the two images is very similar. This is in marked contrast to the quality provided by screens that generate aperiodic, dispersed-dot halftones, which is always significantly inferior to the quality yielded by direct application of DBS.

To examine the effect of the sequence of levels used in the screen design, we designed another 256×256 256-level screen starting from level 1/255, rather than level 127/255. The results are shown in Figure 10, where to facilitate comparison of the two different design approaches, we have included the ramp shown in Figure 9 directly above the ramp halftoned with the screen designed starting from level 1/255. The primary difference is that the bottom ramp has a somewhat wormier texture in the third bar.

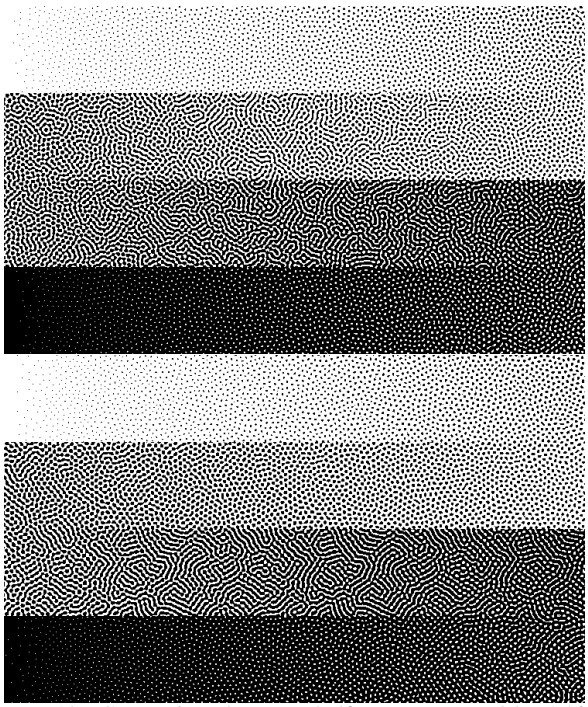


Figure 10: Comparison between halftones generated by 256×256 screens designed using our MS-MP-CLU-DBS algorithm, but with two difference sequences of levels: (Top) starting gray level 127/255, and (Bottom) starting gray level 1/255.

Effective line frequency of halftone textures generated with the MS-MP-CLU-DBS screen

The frequency of a halftone texture is a very important parameter in the printing industry, especially when clustered-dot textures are used for better stability. With periodic, clustered-dot textures, the line frequency is well-defined. However, when the halftone textures are aperiodic, how to define the effective line frequency is not so straightforward. Here, we use the maximum of the Radially Averaged Power Spectrum (RAPS) curve [12], computed from the discrete Fourier transform of the digital halftone. From our previous work, we noted that applying Gaussian filters with various values of σ would cause the frequency of the halftone textures to vary. We tried several Gaussian filters to get a combination for which the generated halftone would have better image quality and reach to desired range of halftone frequencies (260 lpi - 280 lpi). For each σ combination, we generated a screen from it and applied the screen to constant continuous-tone patches to get halftones. We found that applying the combination of $\sigma^i = 1.3$, $\sigma^u = 1.7$ to the screen design would give us a better texture quality and a very stable halftone frequency of 268.21 lpi.

Halftone patches generated by our MS-MP-CLU-DBS screen and their RAPS plots are shown in Figure 11-14. In the RAPS plots, the digital frequency ρ_d is expressed in units of cycles/pixel. We can convert to analog frequency ρ_a in lpi according to the formula $\rho_a = R\rho_d$, where again R is the printer resolution in units of dots per inch (dpi). The effective line frequency over the entire gray-level is shown in Figure 15.

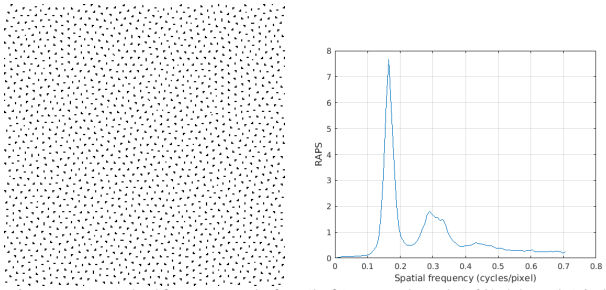


Figure 11: A halftone patch for (left) gray-level 16/255 and (right) its RAPS plot.

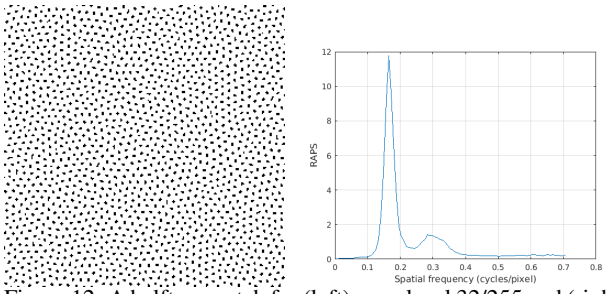


Figure 12: A halftone patch for (left) gray-level 32/255 and (right) its RAPS plot.

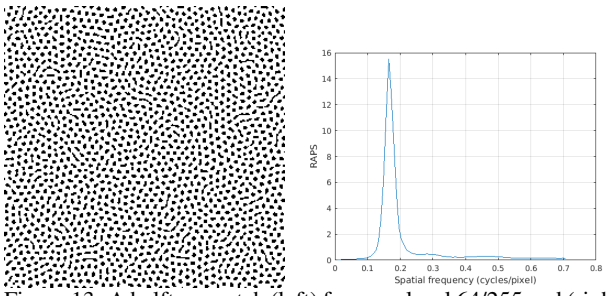


Figure 13: A halftone patch (left) for gray-level 64/255 and (right) its RAPS plot.

Conclusion

We have developed a screen design method based on the MS-MP-CLU-DBS halftoning algorithm to generate aperiodic, clustered-dot halftones to use with printing technologies that are inherently unstable, such as electrophotography. Our screen yields very pleasing, homogeneous halftone textures. Unlike aperiodic, dispersed-dot screens, we see no loss in quality between halftones generated directly by MS-MP-CLU-DBS and halftones generated by application of our screen. We also examine the effective line frequency of the halftone patches generated by our screens. We base our estimate of the effective line frequency on the location of the peak in the radially averaged power spectrum. Except at the very extreme gray levels, we find that the effective line frequency is exceptionally stable across the range of gray levels.

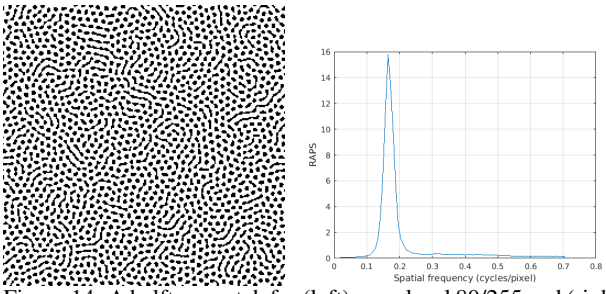


Figure 14: A halftone patch for (left) gray-level 80/255 and (right) its RAPS plot.

References

[1] M. Analoui and J. P. Allebach, "Model-based Halftoning using Direct Binary Search," *Proceedings of SPIE*, vol. 1666, pp. 96-109, 1992.

[2] D. J. Lieberman and J. P. Allebach, "Efficient Model Based Halftoning Using Direct Binary Search," *IEEE International Conference on Image Processing*, 1997.

[3] D. J. Lieberman and J. P. Allebach, "A Dual Interpretation for Direct Binary Search and Its Implications for Tone Reproduction and Texture Quality," *IEEE Transactions on Image Processing*, vol. 9, no. 11, pp. 1950-1963, 2000.

[4] J. P. Allebach, "DBS: retrospective and future directions," *Proceedings of SPIE*, vol. 4300, pp. 358-376, 2001.

[5] D. L. Lau, G. R. Arce, and N. C. Gallagher, "Digital halftoning by means of green-noise masks," *Optical Society of America*, vol. 16, no. 7, pp. 1575-1586, 1999.

[6] D. L. Lau, G. R. Arce, and N. C. Gallagher, "Method and apparatus for producing halftone images using green-noise masks having adjustable coarseness," *U.S. Patent* , 6493112, 2002.

[7] J. P. Allebach, "Random nucleated halftone screen," *Photographic Science and Engineering*, vol. 22, no. 11, pp. 89-91, 1978.

[8] P. Goyal, M. Gupta, C. Staelin, M. Fischer, O. Shacham, and J. P. Allebach, "Clustered-dot halftoning with direct binary search," *IEEE Transactions on Image Processing*, vol. 22, pp. 473-487, 2013.

[9] M. Gupta, C. Staelin, M. Fischer, O. Shacham, R. Jodra, and J. P. Allebach, "Clustered-dot halftoning with direct binary search," *Proceedings of SPIE, Color Imaging XV: Displaying, Processing, Hardcopy and applications*, vol. 7528, pp. 752, 2010.

[10] M. Gupta, "Clustered-dot Halftoning with Direct Binary Search," *Doctoral dissertation, Purdue University*, 2010.

[11] R. Ulichney, "The void-and-cluster method for dither array generation," *Proceedings of SPIE, Human Vision, Visual Processing, and Digital Display IV*, vol. 1913, pp. 332-343, 1993.

[12] R. Ulichney, "Dithering with blue noise," *Proceedings of IEEE*, vol. 76, pp. 56-79, 1988.

[13] W. Xi, T. Frank, Y. Ben-Shoshan, R. Ulichney and J. P. Allebach, "Color CLU-DBS halftoning based on Neugebauer primary area coverage: Improving the breed," *Electronic Imaging, Color Imaging XXII: Displaying, Processing, Hardcopy and applications*, pp. 427, 2018.

[14] R. Ulichney, *"Digital Halftoning,"* MIT Press, 1987.

[15] MATLAB, MATLAB R2017b, *The MathWorks, Inc.*, Natick, Massachusetts, United States.

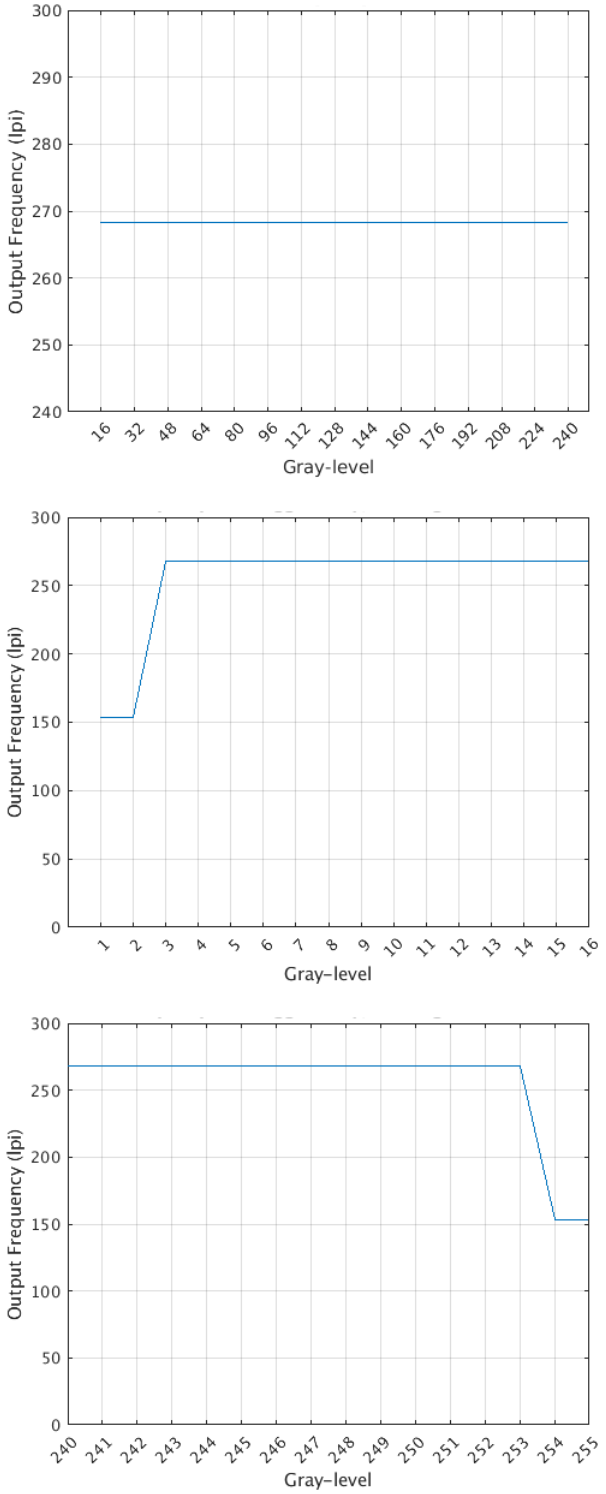


Figure 15: The effective frequency for halftone patches generated using our MS-MP-CLU-DBS screen over the entire gray-level range. (Top) The output frequency for gray-levels 16-240, (Middle) the effective frequency for gray-levels 1-16 , and (Bottom) the effective frequency for gray-levels 240-255.

Author Biography

Jiayin Liu is a Ph.D graduate student majoring electrical engineering at Purdue University. She received B.S. and M.S. from Purdue University in 2015 and 2018, respectively. Her research interests include color management, halftoning, and image processing.

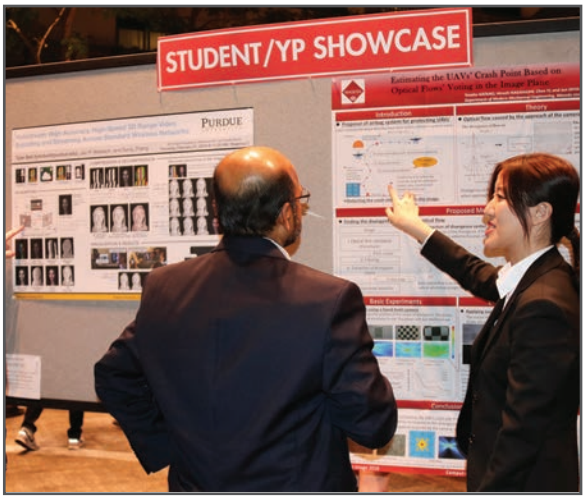
JOIN US AT THE NEXT EI!

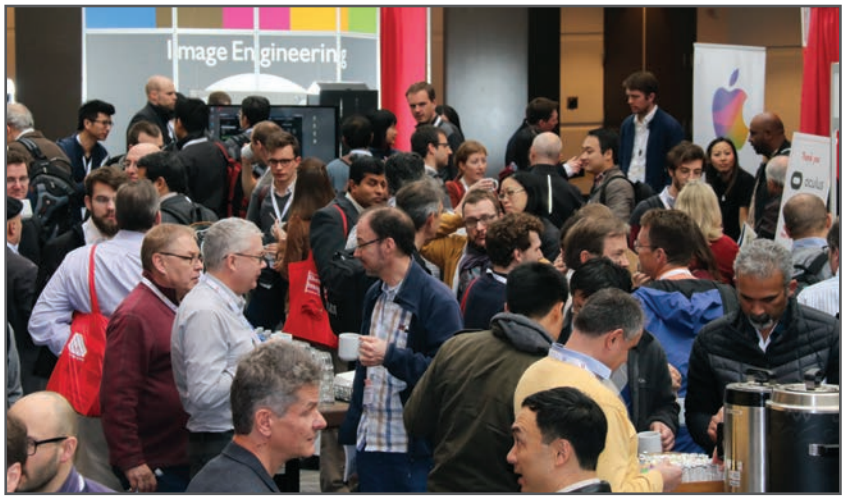
IS&T International Symposium on

Electronic Imaging

SCIENCE AND TECHNOLOGY

Imaging across applications . . . Where industry and academia meet!





• SHORT COURSES • EXHIBITS • DEMONSTRATION SESSION • PLENARY TALKS •
• INTERACTIVE PAPER SESSION • SPECIAL EVENTS • TECHNICAL SESSIONS •

www.electronicimaging.org

IS&T
imaging.org