Implementation of Halftone Blending in Dual-Mode Halftoning to Improve Print Quality for Electrophotographic Printers

Seong Jun Park[a], Mark Shaw[b], George Kerby[b], Terry Nelson[b], Di-Yuan Tzeng[b], Victor Loewen[b], Kurt Bengtson[b], and Jan P. Allebach[c];
[a] Texas Instruments Inc., Dallas, Texas [b] Hewlett-Packard Company, Boise, Idaho [c] School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana

Abstract

In this paper, we consider a dual-mode halftoning system for electrophotographic printers; a low frequency halftoning for smooth regions and a high frequency halftoning for detail regions. This screen-switching approach has a stable halftone pattern in smooth areas and preserves fine rendering in detail areas, but undesirable jaggies may occur along the boundaries where the low and high frequency screens meet. To reduce the jaggies, we propose a new approach to blend two halftone patterns based on a transition region built around boundary pixels. In addition, we provide an implementation methodology which requires simple structure, minimal memory, and small processing time.

Introduction

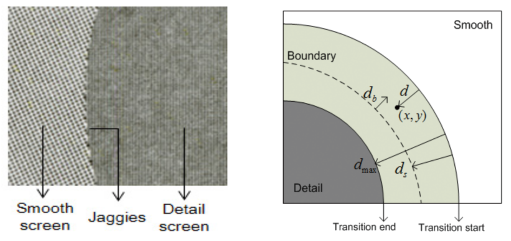
Periodic, clustered-dot halftone patterns generated by screening with a threshold matrix [1–3] are generally considered to be the preferred choice for printing with electrophotographic (EP) printers. This approach has the benefits of yielding more stable printed patterns than aperiodic, dispersed-dot halftoning methods, and is very computationally efficient, since the halftoning is performed on a pixel-by-pixel basis with simple comparison operations. However, there is a tradeoff with periodic, clustered-dot halftones between using a coarser screen to yield more stable halftone patterns and using a finer screen to yield better rendering of detail.

One way to take advantage of both aspects of this tradeoff is to switch screens depending on the local content in the page to be rendered. This screen-switching technique has a stable halftone pattern that is artifact-free in smooth areas and preserves detail rendering in detail or texture areas. This idea has been suggested by previous researchers such as Huang and Bhattacharjya [4] or Daly and Feng [5]. However, one factor that limits the quality of the rendered image is the appearance of artifacts at the boundaries between the smooth and detail halftones. Switching between periodic, clustered-dot halftones with two different frequencies may give rise to an appearance similar to the "jaggies" along boundaries between smooth and detail areas. Figure 1(a) shows an example of such a boundary artifact. To overcome this disadvantage, we developed a novel algorithm called *seamless halftoning* [6] that establishes a transition region along the boundaries between the two types of halftones. The two halftone patterns are blended in an appropriate manner in this transition region.

After a brief review of the algorithm introduced in [6], this paper focuses on the architecture and efficient implementation strategies for the seamless halftoning, which are required to make it feasible to use halftone blending in the hardware imaging pipeline of a printer. The halftone blending algorithm may be implemented on a pixel-by-pixel basis with minimal memory. However, computing the distance d illustrated in Fig. 1(b) from the start of the transition region on the smooth side to the current pixel (x,y) is a significant challenge. We first describe an efficient strategy based on a sliding window of pre-computed distance factors. However, if the desired width of the transition region exceeds the number of lines of image data that can be stored locally, this approach cannot be used. As an alternative, we describe a multi-resolution approach in which a lower resolution version of the local region in the page to be rendered, is used to gain access to a larger spatial neighborhood centered at pixel (x,y). Finally, we describe the generation of the lookup table (LUT) for the boundary blending that contains the pair of gray levels of the smooth and detail screens, as a function of the input gray level a and the distance d to the object boundary.

Seamless halftoning

The basic idea of the seamless halftoning in [6] is that the halftone pattern is produced by blending the two halftone textures that result from the smooth and detail halftone screens in the transition region as illustrated in Figure 1(b).



(a) Boundary artifact caused by object oriented halftoning.

(b) Transition region for blending halftone textures from smooth and detail screens.

Figure 1: Boundary artifact and seamless halftoning approach.

The principal operation of halftone blending is performed by the following equation.

$$g(x,y) = \max\left\{h^{(S)}(x,y), h^{(D)}(x,y)\right\}. \tag{1}$$

At each pixel, there are a smooth halftone texture $h^{(S)}(x,y)$ and a detail halftone texture $h^{(D)}(x,y)$, which are the outputs of a smooth part gray level $a^{(S)}$ and a detail part gray level $a^{(D)}$, respectively. We choose one of them by (1). In fact, $a^{(S)}$ and $a^{(D)}$ are not equivalent to the real input gray level a from the continuous-tone image. Rather $(a^{(S)}, a^{(D)})$ must be determined by a blending LUT based on the gray level a from the continuous-tone image and the distance parameter d of Fig. 1(b). The blending LUT is predetermined by an off-line training process before real time printing. Once the LUT is determined, we can perform the seamless halftoning process in printing according to Fig. 2. When printing, we have the information of the current pixel position (x,y) and the pixel value at this position from the input continuous-tone image. From these two data, we get a and d. The outputs of modules [Get $a^{(S)}$] and [Get $a^{(D)}$] are determined under the following conditions. First, $a^{(S)} = 0$ and $a^{(D)} = a$ when (x,y) belongs to a detail object; and $a^{(S)} = a$ and $a^{(D)} = 0$ when (x,y) belongs to a smooth object. Second, in the transition region, $a^{(S)}$ and $a^{(D)}$ are determined by the LUT. The screening process uses both a smooth screening threshold $t^{(S)}(x,y)$ and a detail screening threshold $t^{(D)}(x,y)$. Lastly, we choose one of the smooth halftone $h^{(S)}(x,y)$ and the detail halftone $h^{(D)}(x,y)$ by (1). After saving the result to a page buffer, we repeat this process at the next pixel until the last pixel is processed. This flow diagram shows that the seamless halftoning algorithm is well-suited to the embedded printer hardware in that it follows a pixel-by-pixel process just like the normal screening process.

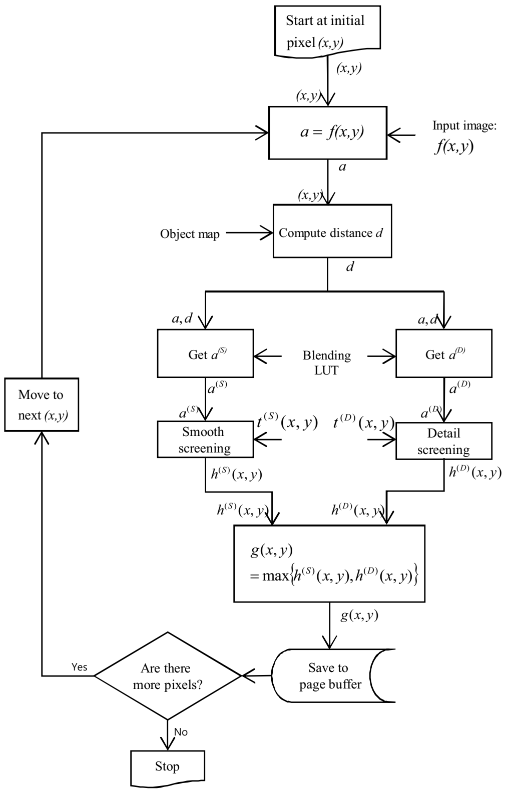


Figure 2: Seamless halftoning flow diagram.

Distance Measurement in the Transition Region

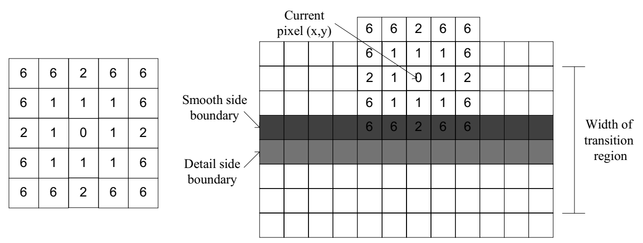
We explained the two key factors of seamless halftoning in the above section: the smooth and detail part gray levels $a^{(S)}$ and $a^{(D)}$, and the distance d from the current pixel (x,y) to the start of the transition region on the smooth side. In this section, we would like to discuss the distance d. Before applying seamless halftoning at a pixel position (x,y) in the transition region, we need to know where are the boundary pixels in the neighborhood of (x,y). Once we get the boundary pixel positions, we are able to decide the boundary blending strength according to the distance to the boundary pixels. Since we are using dual-mode halftoning, we assume we already have an object map $o(x,y)$ that determines for each pixel position whether it is in a smooth or detail area. We apply the Sobel operator to this object map to get the boundary pixel position map $b(x,y)$. Because $o(x,y)$ has only two pixel value levels (smooth and detail indicators), boundary pixels result when the pixel values change in the Sobel window.

The next step is how to compute d. We have the input image $f(x,y)$, the object map $o(x,y)$, and the boundary map $b(x,y)$. Although we do not know where the transition region starts, we know the boundary position from $b(x,y)$. Our strategy is to compute d_b the distance between the nearest boundary point and (x,y) (see Fig. 1(b)), and to find d based on $o(x,y)$ and d_b. We consider that the set of possible d values is $\{0,1,2,\cdots,d_{\max}-1\}$ i.e. at the transition start location, $d=0$ and at the transition end location, $d = d_{\max}-1$. Since boundary pixels always exist on both the smooth and detail sides of the boundary as shown in Fig. 3(b), if we define d_s to be the distance between the smooth side boundary pixel and the transition start, we have $d_s = d_{\max}/2-1$. And the distance between the detail side boundary pixel and the transition start is $d_s+1 = d_{\max}/2$.

The most commonly used way to compute the distance between two pixels (x_1,y_1) and (x_2,y_2) is the Euclidean distance defined by $\sqrt{(x_1-x_2)^2+(y_1-y_2)^2}$. In this paper, instead of directly computing Euclidean distance, we use a pre-computed distance window, W, because it is better suited to hardware implementation. We use W to compute d_b. Since d_b cannot be greater than d_s, we set the size of the window to be $\{2d_s+1\}\times\{2d_s+1\}$. The window is centered at (x,y); and each entry value of W is the Euclidean distance from (x,y). Since we deal with only integers for computing d, since $d_b \le d_s$, each entry of W is defined as follows,

$$w_{i,j} = \begin{cases} \left[\sqrt{(i-d_s)^2+(j-d_s)^2}\right]_R, \text{if} \sqrt{(i-d_s)^2+(j-d_s)^2} \le d_s \\ d_{\max}, \qquad\qquad\qquad\qquad\qquad\qquad \text{else} \end{cases}. \tag{2}$$

The operation $[z]_R$ denotes rounding of the argument z to the nearest integer. In Fig. 3(a) we illustrate an example for the case $d_{\max} = 6$, for which the corresponding size of W is 5×5.



(a) Distance window W showing weights.

(b) Application of window W to compute d_b by choosing minimum value that overlaps a boundary pixel.

Figure 3: Use of distance window to efficiently compute parameter d_b.

Figure 3(b) shows how we apply the distance window to compute d_b. We superimpose W on the boundary map $b(x,y)$. We find the entries of W which overlap the boundary pixel. The value for d_b is chosen as the minimum value among the entries of W that overlap the boundary pixel. The next step is the process for choosing d

based on d_b and the object map $o(x,y)$. The parameter d_b has one value among $0, 1, 2, \cdots, d_s$, or $d_{\max}$. The map $o(x,y)$ has a value corresponding to either a smooth object (S) or a detail object (D). We build a distance LUT with d_b and $o(x,y)$ as inputs and d as the output. If $d_b = d_{\max}$, we consider that the current pixel (x,y) does not belong to the transition region. Thereby we do either smooth-only or detail-only halftoning. If $0 \leq d_b < d_{\max}$, we choose d according to the value of d_b; so that d increases from 0 at the smooth side of the transition region start to $d_{\max}$.

Low Resolution Approach

In the previous section, we described an approach to computing the distance d_b from the current pixel (x,y) to the nearest boundary pixel that is based on superimposing a window W of precomputed distances on the object map. The window is centered at the current pixel, and has dimensions equal to the width of the transition region. This method is computationally efficient and hardware-friendly. But, in a stripe-based implementation, it would be expected to require storage of a number of rows of the object map equal to the width of the transition region.

In this section, we outline a method that uses a low-resolution (LR) version of the object map to estimate the distance d_b. This approach can support a much wider transition region with much more modest memory requirements. The concept is similar to that employed by Bernal et al [7] to estimate the width of a character stroke or line containing the current pixel. In our case, 1 pixel in the low resolution object map (LROM) is obtained from the average value of 8×8 full-resolution pixels. The original boundary position can be approximated based on the LROM pixel values.

The LR approach needs additional information called the LRBM (low resolution boundary map). The LRBM is obtained by applying the Sobel operator to the LROM. To find boundary pixels, three kinds of blocks in the LROM can be used: processing blocks, boundary blocks, and reference blocks. The processing block is the block which includes the current processing pixel. In Fig. 4, B5 is the processing block. The boundary block is the block determined by the LRBM. The dark gray blocks in Fig. 4 are the boundary blocks. We set the boundary search range to be 3×3 in the LROM; so a boundary block must be one of B1 through B9 if one exists. If not, we skip the blending algorithm and move to the next pixel. Reference blocks are dependent on the boundary block positions. Figure 4 shows the boundary and reference block pairs. For instance, the corresponding reference block of B6 is R6 and the corresponding reference block of B7 is R7.

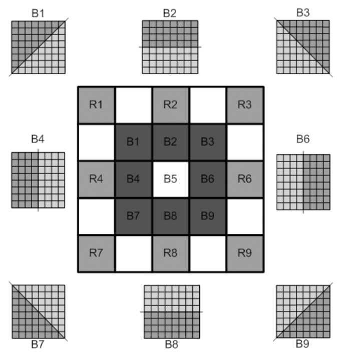


Figure 4: Boundary/reference block pair and boundary direction.

We approximate two factors from the LR images: boundary direction and boundary position. The boundary direction is determined by the boundary blocks. Figure 4 also shows the boundary direction according to the boundary blocks. This means that we assume the boundary direction is horizontal when the boundary block is B2 or B8, vertical when the boundary block is B4 or B6, and diagonal when the boundary block is B1, B3, B7, or B9.

The boundary position can be approximated by boundary blocks, as well. Once a boundary block is specified in Fig. 4, we choose the processing block and reference block. Denote the processing block value by P, the boundary block value by B, and the reference block value by R. If a boundary pixel exists in the boundary block, it can be assumed that the processing block and the reference block are different types of objects in the LROM. In addition, considering that the LROM is based on the average value of full resolution pixels, it can be assumed that the boundary block has an intermediate value between P and R, i.e. $P \leq B \leq R$ or $R \leq B \leq P$.

Now we define the proportion γ.

$$\gamma = \left| \frac{R-B}{R-P} \right|. \tag{3}$$

Because $P \leq B \leq R$ or $R \leq B \leq P$, γ has a range $0 \leq \gamma \leq 1$. If γ is close to 0, this means γ is closer to R than to P. This leads to the conclusion that the real boundary position can be considered to be close to the processing block. On the contrary, if γ is close to 1, then the real boundary position is considered to be close to the reference block.

Based on the concepts introduced above, if there is an object boundary in the vicinity of the current pixel (x,y), we can estimate its angle relative to the current pixel, quantized to eight directions, and its distance from the current pixel based on the relative difference between the values of the processing block, the boundary block, and the reference block. We use this information to estimate the distance d_b from the current pixel to the nearest boundary pixel in the high-resolution object map.

Design of Blending Lookup Table

Briefly reviewing the paper [6] which describes the basic idea of seamless halftoning, the gray level a results in absorptance $T(a)$ after printing with the smooth screen, and the distance d determines the ratio $\rho(d)$ between the smooth part gray level $a^{(S)}$ and the detail part gray level $a^{(D)}$. When a and d are given, we select $(a^{(S)}, a^{(D)})$ and apply (1) to each pixel in the transition region so as to simultaneously satisfy T and ρ as nearly as possible. The main issue is how we select $(a^{(S)}, a^{(D)})$ so that blending pattern is as close to the original T and ρ as possible when it is printed. What we can do is to choose $(a^{(S)}, a^{(D)})$ to have the minimum error between the ideal $T(a)$ and the real $T(a^{(S)}, a^{(D)})$ and the minimum error between the ideal $\rho(d)$ and the real $a^{(D)}/a^{(S)}$. Therefore, it is necessary to design and print a training set of $(a^{(S)}, a^{(D)})$ combinations before online printing and to find the best blending pair based on a and d. This can be provided in a table form with input (a,d) and output $(a^{(S)}, a^{(D)})$. This is how the blending LUT is constructed.

We define a cost function $C = \Phi + \Psi$; and for each pair (a,d), we find the pair $(a^{(S)}, a^{(D)})$ that minimizes the cost function. Here, Φ is the *absorptance error cost* and Ψ is the *ratio error cost*. The solution can be expressed as

$$\widetilde{\theta}(a,d) = \underset{\theta}{\operatorname{argmin}} \, \{\Phi(\theta|a,d) + \Psi(\theta|a,d)\}, \tag{4}$$

where $\theta = (a^{(S)}, a^{(D)})$ represents the blending pair. In this manner, we determine the optimal blending pairs $(a^{(S)}, a^{(D)})$ for all input pairs (a,d), and store them in a blending LUT to be used during the online printing process.

To characterize the absorptance response curves for the smooth halftone screen and the blended halftone screens, we designed sets of patch sheets, as illustrated in Fig. 5. Each patch corresponds to an input gray value a for the smooth screen (Fig. 5(a)), or a pair of input gray values $(a^{(S)}, a^{(D)})$ for the blended smooth and detail halftone screens (Fig. 5(b)). We print each sheet and measure the average absorptance of each patch on the sheet. After measuring the absorptance of these patches, we use a full search to find the blending pair that minimizes the cost function among the blending pair candidates $(a^{(S)}, a^{(D)})$.

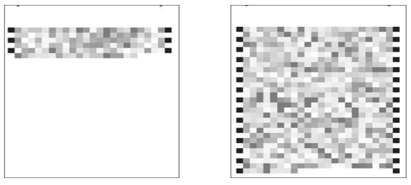

(a) Smooth halftone only.
(b) Blended smooth and detail halftones.

Figure 5: Patch sheets.

We base our measurement of the printed absorptance errors on the device-independent 1976 *CIE* $L^*a^*b^*$ uniform color space [8]. We represent our absorptance values $T(a)$ as the l_2 norm distance in $\Delta E^{(76)}{}_{a^*b^*}$ units between the color value of the printed patch and the color value of the white paper. Denote the distance for the smooth halftoning patch in Fig. 5(a) as $T_s(a)$, and the distance for the blending halftoning patch in Fig. 5(b) as $T_b(a^{(S)}, a^{(D)})$. The absorptance error cost function Φ is then defined as the absolute difference between the smooth patch absorptance and the blending patch absorptance shown in (5).

$$\Phi = \Phi(a^{(S)}, a^{(D)}|a) = \left| T_s(a) - T_b(a^{(S)}, a^{(D)}) \right|. \tag{5}$$

For the ratio error cost Ψ, we want to minimize the difference between the target ratio $\rho(d)$ and the actual ratio $a^{(D)}/a^{(S)}$. However, we cannot simply choose $\Psi = \left|\rho(d) - a^{(D)}/a^{(S)}\right|$, because this form for the cost function would not reflect the nonlinear ratio difference correctly. When $\rho(d) < 1$, it would be less sensitive to the difference; and when $\rho(d) > 1$, it would be more sensitive to the difference. Consequently, we define the ratio error cost as

$$\Psi = \Psi\left(a^{(S)}, a^{(D)}|d\right) = \log\left(\max\left[\frac{a^{(D)}/a^{(S)}}{\rho(d)}, \frac{\rho(d)}{a^{(D)}/a^{(S)}}\right]\right). \tag{6}$$

Note that Ψ is always greater than 0; and the best case is $\Psi = 0$, which means $a^{(D)}/a^{(S)} = \rho(d)$. The cost Ψ increases as the difference between $\rho(d)$ and $a^{(D)}/a^{(S)}$ increases. In addition, the log operator limits the influence of very large values of the ratio $a^{(D)}/a^{(S)}$, such as $a^{(D)}/a^{(S)} = 255/1$.

By combining (5) and (6), we obtain the final cost function in the form shown in (4). Now that we have the absorptance error cost Φ and the ratio error cost Ψ, we can find the pair $(a^{(S)}, a^{(D)})$ that minimizes the cost function (4) given (a,d); and we store the result to the blending LUT. We repeat this process over the range of input levels $a = 0,1,2,\cdots,255$ and distances $d = 0,1,2,\cdots,d_{\max}-1$.

Conclusion

In this paper, we have discussed an algorithm and implementation scheme to improve print quality for EP printers. We first introduced a seamless halftoning approach to remove undesirable boundary artifacts. Next we dealt with implementation issues, and proposed a hardware friendly structure and low memory strategy that is useful to characterize the boundary transition region when the transition region is relatively wide. Last, we proposed a LUT-based boundary blending technique based on an off-line training set.

Acknowledgment

This work was supported by the Hewlett-Packard Company.

References

[1] F. A. Baqai and J. P. Allebach, "Computer-aided design of clustered-dot color screens based on a human visual system model," *Proc. IEEE*, vol. 90, pp. 104–122, Aug. 2002.

[2] G.-Y. Lin and J. P. Allebach, "Generating stochastic dispersed and periodic clustered textures using a composite hybrid screen," *IEEE Transactions on Image Processing*, vol. 15, pp. 3746–3758, 2006.

[3] T. M. Holladay, "Electronic halftoning screening," *U.S Patent 813,559, assigned to Xerox Corportation*, 1980.

[4] J. Huang and A. Bhattacharjya, "An adaptive halftone algorithm for composite documents," *Proceedings of SPIE, Color Imaging: Processing, Hardcopy and Applications IX*, vol. 5293, pp. 425–433, 2004.

[5] S. Daly and X. Feng, "Methods and systems for adaptive dither structures," *U.S. patent 5,017,4360, assigned to Sharp Laboratories of America, Inc.*, 2004.

[6] S. J. Park, M. Shaw, G. Kerby, T. Nelson, D. Y. Tzeng, V. Loewen, K. Bengtson, and J. P. Allebach, "Halftone blending between smooth and detail screens to improve print quality with electrophotographic printers," *Proceedings of SPIE, Color Imaging XVII: Displaying, Processing, Hardcopy, and Applications*, vol. 8292, Jan. 2012.

[7] E. Bernal, J. P. Allebach, and J. Trask, "Model-based memory-efficient algorithm for compensation of toner overdevelopment in electrophographic printers," *Journal of Imaging Science and Technology*, vol. 52, pp. 060 504–1 – 060 504–15, Nov 2008.

[8] G. Wyszecki and W. S. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley Interscience, Aug 2000.

Author Biography

Seong Jun Park received the B.S and M.S degrees in Radio Science and Engineering from Korea University, Seoul, Korea, in 1998 and 2000, respectively, and the Ph.D degree from the School of Electrical and Computer Engineering, Purdue University, West Lafayette, Indiana, 2011. He is currently an image processing systems engineer with Texas Instruments, Inc. His current research interests include color image processing for cameras and printers.