Quad-interleaved Block Level Parallel Direct Binary Search Algorithm

Xujie Zhang, Jan P. Allebach; Purdue University; West Lafayette, Indiana/U.S.A

Abstract

Direct Binary Search (DBS), as one of the three categories of halftoning, provides the best, visually pleasing halftoning quality. However, as a sequential algorithm, DBS is most computational complex so it always plays the role of offline algorithm for other halftoning categories (like tone dependent error diffusion, hybrid screen, etc.). Meanwhile, it is seldom directly works as a real-time/online solution for current commercial printers. In this paper, we would like to present a parallel version DBS with same image quality compared with original DBS, which can fit the current Same Instruction Multiple Data (SIMD) system, like General-Purpose Graphics Processing Unit (GP-GPU), and furthermore, there will be the potential that DBS can work on current multi-core system as real-time solution for halftoning.

Introduction

Halftoning is used to render continuous-tone images with output devices that are capable of directly printing or displaying only two or a small number of different gray levels. The three most widely used categories of halftoning methods are pixel method, neighboring method, and iterative method. Usually the halftone output quality and computation timing both increase in the order of pixel, neighboring and iterative. The pixel method is known as screening, or dither. For example, Ulichiney proposed a dither method to generate the blue-noise which is visually pleasant halftone pattern [1] [2] [3]. Pixel method compares or computes on the pixel of the screen and corresponding pixel on the image only, so it is usually computational-efficient. Another pixel method is look-up-table halftoning. Li and Allebach invented a look-up-table halftoning using the correlation information between different graylevels in order to minimize the artifact noise [4]. The neighboring method is well-know as the Floyd-Steinberg error diffusion [5]. It diffuses the error between the halftone output image and the continuous-tone input image to the afterward pixels in the raster order. It is essentially serial algorithm, so error diffusion is more computational inefficient than pixel method. Lots of work has been done to improve the error diffusion algorithm. Kolpatzik and Bouman embedded the Human Visual System model into it [8]. Tone Dependent Error Diffusion [6] was invented by Li and Allebach, which largely eliminated the artifacts caused by the conventional error diffusion. The most visually pleasant halftone pattern can be generated by the iterative method, like Direct Binary Search(DBS) [7]. DBS is computational inefficient, but it is widedly used combined with other halftoning algorithms. Lin and Allebach invented the FM screen by DBS with the stacking constraint [11]. Hybrid screen [9] also use DBS to generate the blue-noise texture in the highlight and shadow regions. Tone Dependent Error Diffusion trained the coefficients and thresholds by the blue-noise DBS pattern in Fourier domain. With the assistance of off-line design by DBS, screen or error diffusion can significantly improve the halftone quality and still remain the same computational complexity.

On the other hand, according the halftone texture, we can also separate the halftoning algorithm as AM, which is also called clustered-dot, and FM, which is also called dispersed-dot. Using the modulation concepts in communication theory. We define the halftoning as AM, when the pattern forms the clustered dots, as the absorptance glows, which in Fourier domain produces a donut-like band-pass green-noise filter. However, FM halftone pattern generates the high-frequency blue noise in Fourier domain. Usually, Ink-jet can produce much more robust single dots, so it use dispersed-dot halftoning more often. However, due to the instability of the single dots produced by electrophotographical printers, they use clustered-dot halftoning more often. Another classification of halftoning according to the texture is periodic or aperiodic. Screen is a periodic halftoning, while error diffusion and DBS are aperiodic. All four combinations (AM/FM, periodic/aperiodic, 2×2) are possible. Clustered halftoning is always used in the electro-photographic printers, because the clustered-dot can provide more stable print and less dot-gain effect. The printing devices which can generate more stable pixels usually employ dispersed halftoning [2]. Clustered periodic halftoning commonly provides more homogeneous structure in midtone region than the dispersed aperiodic halftoning, while the stochastic aperiodic halftoning shows more preferred blue-noise texture in highlight. The other halftoning algorithm may apply hybrid texture, like hybrid screen [9], which has the blue-noise dispersed dot in highlight and shadow region, also enjoys the homogenous clustered periodic texture in the midtone. When the printer's native tones are more than two levels, the common bi-level halftoning we discussed above change to multilevel halftone. Zhang and Allebach developed a periodic clustered-dot multilevel halftoning [12] using hybrid screen to overcome that contouring artifacts in multilevel halftoning.

From the computation device's side, more and more SIMD devices, like GP-GPU, with high computational capacity are available. SIMD device has become a competitive accelerator for computation applications. We take GPU for example. The GP-GPU's rapid increase in both programmability and capability has successfully targeted a wide range of computationally demanding, complex problems. These efforts in general-purpose computing on the GP-GPU and other SIMD devices, has positioned the themselves as compelling alternatives to traditional microprocessors and other dedicated hardwares in high-performance computer systems of the future. With more and more cutting-edge parallel computing systems available on the market, a parallel DBS algorithm will be attractive, if it can offer both (1) as good halftone quality as sequential DBS and (2) less computation latency. Pixel-

level parallel DBS is proposed by Chandu [13] but it is not very friendly to SIMD system because of conflict among neighboring pixels when updating the Cpe look-up-table (LUT). However, previous research by Lieberman and Allebach [14] already proved the block-level DBS offers as good halftone quality as sequential DBS. The authors, in this article, want to propose a block-level parallel DBS that is computational friendly to SIMD processors such as GP-GPU.

In this paper, we propose a novel approach to parallelize DBS. Our objective is: (1)The parallel DBS should have same halftone image quality as original sequential DBS; (2)The parallel DBS should be friendly to SIMD processor, like GP-GPU; (3)It had better that the parallel DBS could keep almost same amount of computations so that it can have a fair comparison with sequential DBS on computational latency. The rest of the paper is organized as: preliminary knowledge of DBS halftoning algorithm and GPU computing programming model; Our block-level parallel DBS algorithm is illustrated in the second section; The third section is the experimental results; The last section is the conclusion.

Preliminary

In this section, we briefly review DBS halftoning algorithm, the HVS model, hybrid screen halftoning algorithm, and the traditional multilevel screen. In this section, we only discuss the monochrome DBS. Also, we focus on the luminance channel of the HVS model. And both DBS and hybrid screen reviewed here are restricted in bi-level.

Direct Binary Search (DBS) and Human Visual System (HVS) model

Direct Binary Search(DBS) is an iterative halftoning algorithm to minimize the perceived difference between the continuous-tone image and the halftone image. As showed in Figure 1:

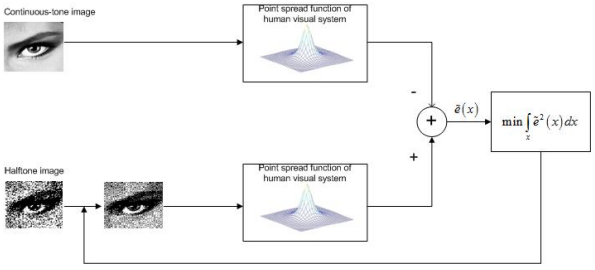


Figure 1. *Objective of Direct Binary Search*

The perceived difference is described by the term, mean square error (MSE) of perceived error between the continuous-tone image and the halftone image. Due to the low-pass characteristic of the human vision, the perceived error is the error between the continuous-tone image and the halftone image filtered by the Human Visual System (HVS) which is essentially a low pass filter.

The spatial frequency response $\bar{H}(\bar{u},\bar{v})$ of the HVS based on Näsänen's modelis given by Equation 1:

$$\bar{H}(\bar{u},\bar{v}) = a\Gamma^b \exp\left[-\frac{\sqrt{\bar{u}^2+\bar{v}^2}}{c\ln(\Gamma)+d}\right] \tag{1}$$

In Equation 1, $\bar{u}$ and $\bar{v}$ are the spatial frequency coordinates in cycles/degree subtended at the retina; Γ is the average luminance of the light reflected from the print in cd/m^2; Constants $a = 131.6$, $b = 0.3188$, $c = 0.525$, $d = 3.91$. If we take the inverse continuous-space Fourier transform (CSFT) of $\bar{H}(\bar{u},\bar{v})$, we get the spatial filter of HVS $\bar{h}(\bar{x},\bar{y})$, where $\bar{x}$ and $\bar{y}$ are in degrees subtended at the retina. Before we invert the HVS from the frequency domain into the spatial domain, we need first to convert $\bar{u}$ and $\bar{v}$ in units of cycles/degree into u and v in units of cycles/inch. Considering that x inches on the paper is viewed as $\bar{x}$ degrees at the retina in a D viewing distance, so

$$\bar{x} = \frac{180}{\pi}\tan^{-1}\left(\frac{180}{\pi D}\right) \approx \frac{180}{\pi D}x, \qquad \text{for } x \ll D \tag{2}$$

Thus, in the spatial frequency domain the relationship between $\bar{u}$ and u is given by

$$\begin{aligned} H(u,v) &= \bar{H}\left(\bar{u}=\frac{\pi D}{180}u, \bar{v}=\frac{\pi D}{180}v\right) \\ &= a\Gamma^b \exp\left[-\frac{\frac{\pi D}{180}\sqrt{\bar{u}^2+\bar{v}^2}}{c\ln(\Gamma)+d}\right] \end{aligned} \tag{3}$$

and in the spatial domain,

$$\begin{aligned} h(x,y) &= \left(\frac{180}{\pi D}\right)^2 \bar{h}\left(\frac{180}{\pi D}x, \frac{180}{\pi D}y\right) \\ &= \left(\frac{180}{\pi D}\right)^2 \bar{h}\left(\bar{x}=\frac{180}{\pi D}x, \bar{y}=\frac{180}{\pi D}y\right) \\ &= \left(\frac{180}{\pi D}\right)^2 \bar{h}(\bar{x},\bar{y}) \end{aligned} \tag{4}$$

On the other hand, we also can have the analytical expression for the HVS filter in the spatial domain [9]:

$$h(x,y) = a\Gamma^b \frac{2\pi k}{\left(k^2+4\pi^2(x^2+y^2)\right)^{\frac{3}{2}}} \tag{5}$$

Let $e[m,n]$ be the error image by:

$$e[m,n] = g[m,n] - f[m,n] \tag{6}$$

where $g[m,n]$ is the output halftone image and $f[m,n]$ is the input continuous-tone image. Then the perceived error image between the halftone image and the continuous-tone image is the error image filtered by HVS:

$$\tilde{e}(x,y) = \sum_m \sum_n e[m,n]\tilde{p}(x-mX, y-nX) \tag{7}$$

where the X is the lattice of addressable points for the printer; $\tilde{p}(x,y) = h(x,y) * *p_{dot}(x,y)$ is the HVS convolved with the printer dot profile function. Assuming that the $h(x,y)$ has much larger extent than printer dot profile function $p_{dot}(x,y)$, we shall have $\tilde{p}(x,y) \approx h(x,y)$.

Therefore, the error metric of DBS as known as the mean square error (MSE) is:

$$E = \int\int |\tilde{e}(x,y)|^2 \, \mathrm{d}x\mathrm{d}y \tag{8}$$

Substitute Equation 7 into Equation 8, we get:

$$E = \sum_m \sum_n e[m,n] c_{\tilde{p}\tilde{e}}[m,n] \tag{9}$$

where $c_{\tilde{p}\tilde{e}}[m,n] = e[m,n] * * c_{\tilde{p}\tilde{p}}[m,n]$. And $c_{\tilde{p}\tilde{p}}[m,n]$ is the autocorrelation function of the HVS on the printer lattice.

The search strategy for the DBS is that it tries to toggle or swap each pixel with its neighbor pixel in a raster order. Toggle or swap is accepted only if it reduces the error metric most. After all pixels have been searched in the image, we call that one iteration finished. After a certain number of iterations has been done or the error metric cannot be reduced in the whole iteration, the DBS algorithm converges. The search strategy of DBS is showed in Figure 2:

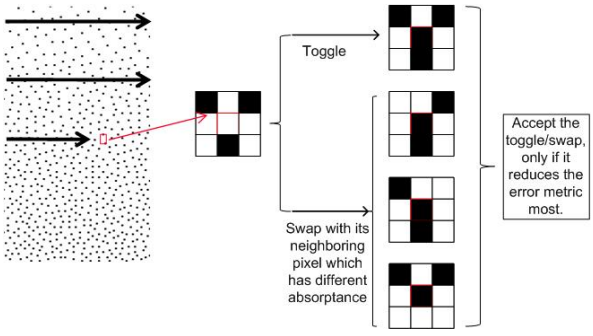


Figure 2. *Search strategy of Direct Binary Search*

If there is a trial toggle on location $\mathbf{m_0} = [m,n]$, the updated trial halftone image $g'[\mathbf{m}]$, the updated error image $e'[\mathbf{m}]$ and $c_{\tilde{p}\tilde{e}}$ are calculated as follows:

$$g'[\mathbf{m}] = g[\mathbf{m}] + a_0 \delta[\mathbf{m} - \mathbf{m_0}] \tag{10}$$

$$e'[\mathbf{m}] = e[\mathbf{m}] + a_0 \delta[\mathbf{m} - \mathbf{m_0}] \tag{11}$$

$$c'_{\tilde{p}\tilde{e}}[\mathbf{m}] = c_{\tilde{p}\tilde{e}}[\mathbf{m}] + a_0 c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{m_0}] \tag{12}$$

where the prime denotes the updated image after trial toggle. $a_0 = -1$ when $g[\mathbf{m_0}]$ is changed from 1 to 0; $a_0 = 1$ when $g[\mathbf{m_0}]$ is changed from 0 to 1. $\delta[\mathbf{m}] = 1$ if $\mathbf{m} = 0$; otherwise $\delta[\mathbf{m}] = 0$.

Substituting Equation 11 and 16 into Equation 9, the change in error ΔE_0 can be expressed as:

$$\Delta E_0 = a_0^2 c_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2a_0 c_{\tilde{p}\tilde{e}}[\mathbf{m_0}] \tag{13}$$

Similarly, according to [10], the trial switch in DBS will have the change in error ΔE_0 as:

$$\begin{aligned}\Delta E = (a_0^2 + a_1^2) c_{\tilde{p}\tilde{p}}[\mathbf{0}] + 2a_0 c_{\tilde{p}\tilde{e}}[\mathbf{m_0}] \\ + 2a_1 c_{\tilde{p}\tilde{e}}[\mathbf{m_1}] + 2a_0 a_1 c_{\tilde{p}\tilde{p}}[\mathbf{m_1} - \mathbf{m_0}]\end{aligned} \tag{14}$$

the updated error image $g'[\mathbf{m}]$ and $c_{\tilde{p}\tilde{e}}$ are:

$$g'[\mathbf{m}] = g[\mathbf{m}] + a_0 \delta[\mathbf{m} - \mathbf{m_0}] + a_1 \delta[\mathbf{m} - \mathbf{m_1}] \tag{15}$$

$$\begin{aligned}c'_{\tilde{p}\tilde{e}}[\mathbf{m}] = c_{\tilde{p}\tilde{e}}[\mathbf{m}] + a_0 c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{m_0}] \\ + a_1 c_{\tilde{p}\tilde{p}}[\mathbf{m} - \mathbf{m_1}]\end{aligned} \tag{16}$$

where a_0 is similar to a_1. $a_1 = -1$ when $g[\mathbf{m_1}]$ is changed from 1 to 0; $a_1 = 1$ when $g[\mathbf{m_1}]$ is changed from 0 to 1.

Therefore, we can see that ΔE is changed according to $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$, while there is either a toggle or switch, $c_{\tilde{p}\tilde{e}}[\mathbf{m}]$ is changed in a very small local area, since $c_{\tilde{p}\tilde{p}}[\mathbf{m}]$'s size is determined by the HVS, which is a small-size local low-pass filter. This conclusion has a great impact on our parallel DBS algorithm that will be illustrated later.

SIMD computing programming model

Again we take GPU as the example of SIMD computing device. The benefit of GPU computing is the massive parallelism computing capability provided by increasing number of number of compute units. Figure 3 shows the SIMD programming model. The compute units, also be called ALUs or cores. The number of cores reach upto hundreds or thousands in one GP-GPU. Every core has is own dedicated registers; A bunch of cores share the local (share) memory; All compute units have the same view on the global memory. The access speed decreases from register, to local memory and to global memory, while the size increases in the same order. From a software programmer's view, every parallelism item works on its own data. If each work-item independently updates the its own data, the the atomic operations, barriers or sequential block can be avoided. Those sequential operations are the most significant bottle-necks for SIMD computing. The relation between CPU and SIMD device (GPU) can be viewed as a master-slave programming model. CPU works as the master, which dispatches the parallel massive work load onto the SIMD device. The SIMD device (GPU) works as massive slaves which finishes their independent jobs. The data transfer from host(CPU) to device(GPU), or from device(GPU) to host(CPU) is another overhead for GPU computing. However, if transfer all the image data once from the host to device, and then dispatches all the parallel work load on the device. After that, the output image data transfers from device to host. This programming model avoids unnecessary data transfer overhead. Our paralle DBS algorithm mapping onto the SIMD device will comply with this model.

Block-level Parallel DBS

According to Lieberman and Allebach [10], block-level DBS offers as good halftone quality as sequential DBS. Therefore, how do we map the block-level DBS onto GPU that shows best fit of the architecture? We come up with the simple block-level DBS.

Bi-interleaved Block-level Parallel DBS

The most straightforward thought is to divide the entire image into several equal size image blocks, and then map each image block one by one to each computing core according to Figure 4. However, because each update, including toggling and swapping of each pixel, needs an update of c_{pe}. Depending on the HVS filter size the developer chooses, each single pixel toggle or switch on the halftone image g arises the change of c_{pe} covers a small area. In our case, the change of c_{pe} is upto 25×25 pixels for a single change. The overlap on the computing of the c_{pe}, showed in Figure 4, among neighboring image blocks would cause conflicts on SIMD computing.

A better way is to have the bi-interleaved block-level DBS. As showed in Figure 5, we first run the block-level DBS in the blocks concurrently marked by index 1. After the DBS finishes on all blocks of index 1, then run the block-level DBS concurrently on the blocks marked by index 2.

However, by careful observation showed in Figure 6 computation conflict on c_{pe} still happens at the corners of these blocks.

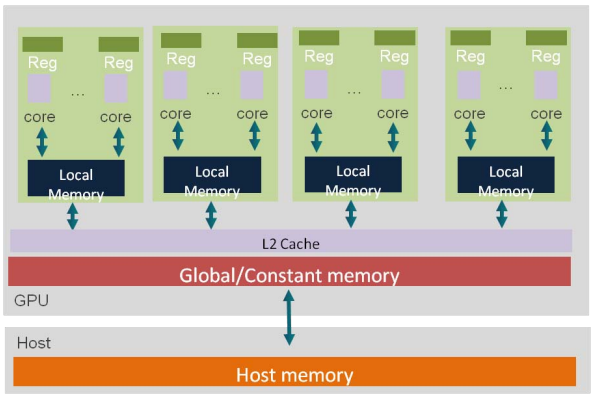


Figure 3. *GPU Architecture*

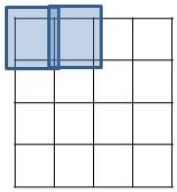


Figure 4. *Block-level parallel DBS*

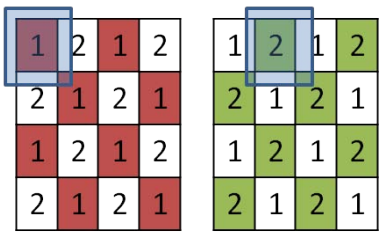


Figure 5. *Bi-interleaved Block-level parallel DBS*

Quad-interleaved Block-level Parallel DBS

The quad-interleaved block-level DBS finally solves the issue of computation conflict on c_{pe}. As showed in Figure 7, as long as the block size is larger than 25×25 (due to the HVS filter size, 25×25 in this paper), the overlap is entirely avoided.

Here we remind ourselves that the DBS is an iterative algorithm. Therefore, we have two choices for the algorithm: (1) Block parallelism as the outer loop; Iteration as the inner loop. It means that the iteration is assigned inside each work-item (2) Iteration as the outer loop; block parallelism as the inner loop. It means that the iteration is controlled by the CPU. Meanwhile, SIMD device runs for one iteration only for each trigger from host (CPU). The first method inevitably causes the boundary artifacts so we choose the second one. The pseudocode is showed as Algorithm 1. i is number of iterations. j is the number of update in each iteration of DBS.

Algorithm 1 Pseudocode for Quad-interleaved Block-level DBS

```
1: i ← 0;
2: while ( j > 0 || i ⩽ threshold ) do
3:     j ← 0;
4:     parallelize the DBS on all block of index 1
5:     update j
6:     parallelize the DBS on all block of index 2
7:     update j
8:     parallelize the DBS on all block of index 3
9:     update j
10:    parallelize the DBS on all block of index 4
11:    update j
12:    i ← i + 1
13: end while
```

Experimental Results

The halftone ramp generated by Quad-interleaved block-level parallel DBS is showed in Figure 8. Compared with the halftone ramp generated by original sequential DBS in Figure 9,

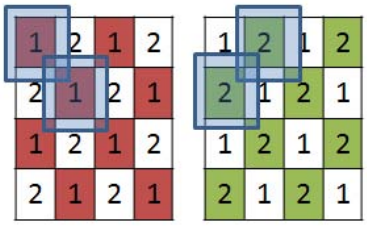


Figure 6. *Bi-interleaved Block-level parallel DBS*

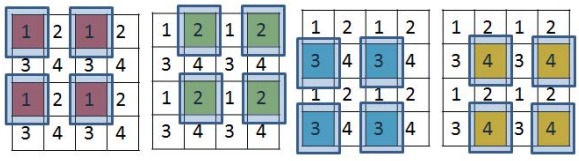


Figure 7. *Quad-interleaved Block-level parallel DBS*

Figure 8 does not show visually unpleasing artifacts. Figure 8 and Figure 9 show almost same halftone quality.

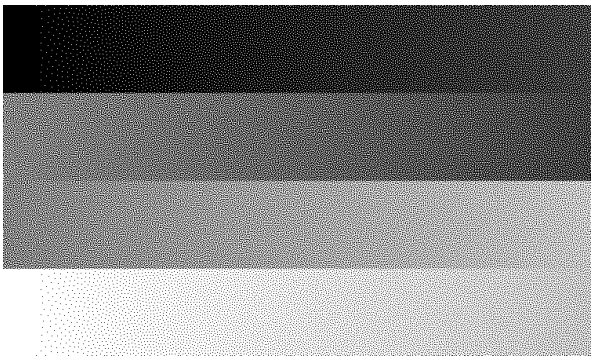


Figure 8. *Halftone ramp generated by quad-interleaved block-level parallel DBS*

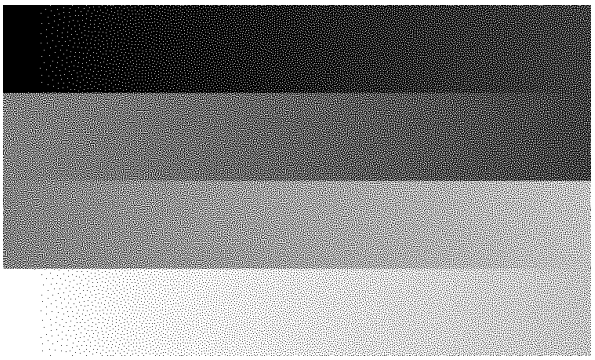


Figure 9. *Halftone ramp generated by original sequential DBS*

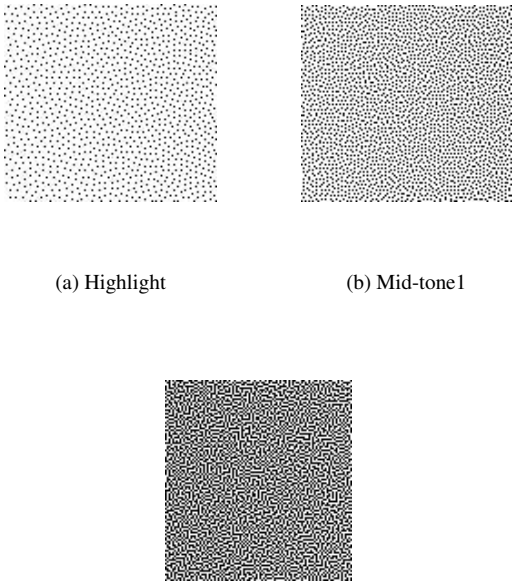
More details are showed in Figure 10 and Figure 11.

We made experiments on the real images. The Figure 12 and Figure 13 show the halftone image quality on person's eyes. There are high-frequency details around the eyes, eye brows, and hairs. Also, there are smooth areas on the skin. There is almost no quality difference between the original sequenctial DBS and the parallel DBS.

The (timing) performance is evaluated as well. Figure 14 shows the performance scores. High score means the executing latency is shorter. The performance is normalized by the GFlops of the computing unit. The parallel DBS is 3 to 5 times faster than the sequential DBS.

Conclusion

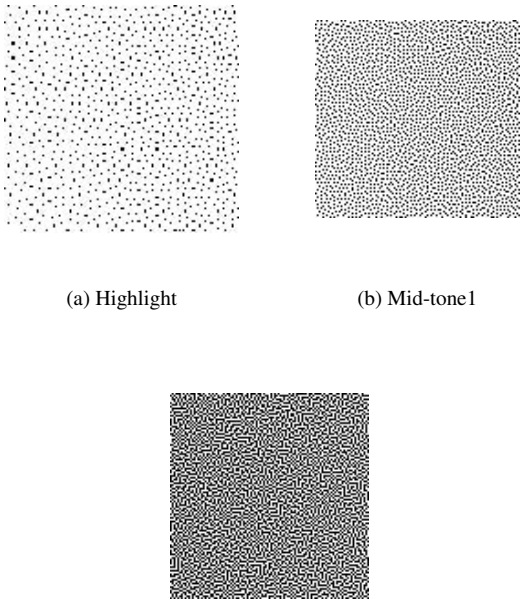
This paper proposes a SIMD device (e.g. GP-GPU) friendly DBS algorithm, named quad-interleaved block-level parallel DBS. With the potential computing performance benefits provided by SIMD device, the halftone image generated by our parallel DBS still remains the same quality compared with original sequential DBS algorithm.



(a) Highlight (b) Mid-tone1

(c) Mid-tone2

Figure 10. *Halftone generated by quad-interleaved block-level parallel DBS*



(a) Highlight (b) Mid-tone1

(c) Mid-tone2

Figure 11. *Halftone generated by original sequential DBS*

References

[1] R. Ulichney, Dithering with Blue Noise, Proc. IEEE, pg. 56-79, vol. 76 (1988).

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

[3] D. Lau and R. Ulichney, Blue-Noise Halftoning for Hexagonal Grids, IEEE Trans. Image Processing, pg. 1270-1284, vol. 15 (2006).

[4] P. Li, and J. P. Allebach, Look-up-table based halftoning algorithm, IEEE Trans. Image Processing, pg. 1593-1603, vol. 9 (2000).
[5] R. W. Floyd, and L. Steinberg, An adaptive algorithm for spatial greyscale, Proc. Soc. Inf. Disp., pg. 75-77, vol. 17 (1976).
[6] P. Li and J. P. Allebach, Tone-dependent error diffusion, IEEE Trans. Image Processing, pg. 201-215, vol. 13 (2004).
[7] D. Lieberman and J. P. Allebach, On the relation between DBS and void and cluster, roc. IS&T, International Conf. Digital Printing Technologies, Toronto, Canada, vol. 14 (1998).
[8] B. W. Kolpatzik, and C. A. Bouman, Optimized error diffusion for image display, Journal of Electronic Imaging, pg. 277-292, vol. 1 (1992).
[9] C. H. Lee and J. P. Allebach, The hybrid screen–improving the breed, IEEE Trans. on Image Processing, pg. 435-450, vol. 19 (2010).
[10] D. Lieberman and J. P. Allebach, A Dual Interpretation for Direct Binary Search and Its Implications for Tone Reproduction and Texture Quality, IEEE Trans. on Image Processing, pg. 1950-1963, vol. 9 (2000).
[11] J. P. Allebach, and Q. Lin, FM screen design using DBS algorithm, Proceedings., International Conference on Image Processing, pg. 549-552, vol. 1 (1996).
[12] X., Zhang, A. Veis, R. Ulichney, and J. P. Allebach, International conference on image processing, pg. 829-832, (2012).
[13] K. Chandu, M. Stanich, B. Trager and C. W. Wu, Circuits and Systems (ISCAS), IEEE International Symposium on, pg. 185-188, (2012).
[14] D. Lieberman, and J. P. Allebach, Efficient Model Based Halftoning Using Direct Binary Search, International Conference on Image Processing, pg. 775-778, (1997).

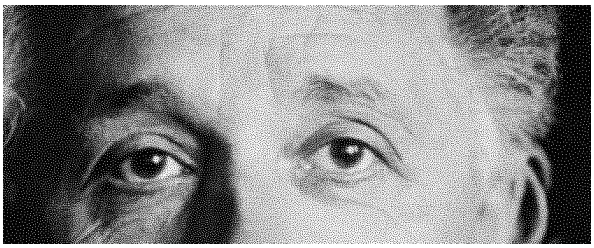


Figure 12. *Quad-interleaved Block-level parallel DBS*

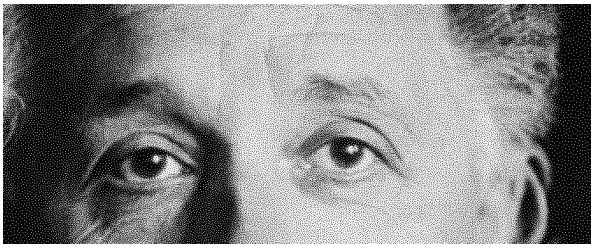


Figure 13. *Original sequential DBS*

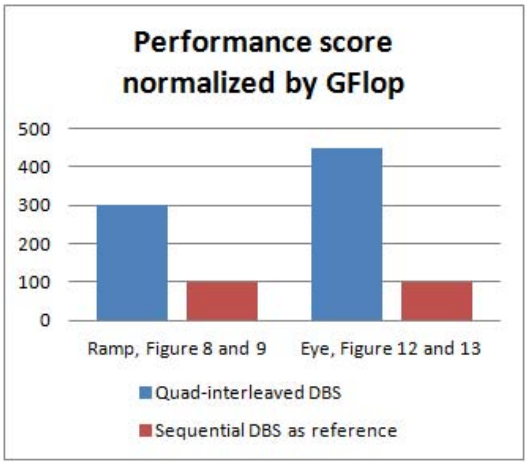


Figure 14. *Quad-interleaved Block-level parallel and sequential DBS performance comparison*

Author Biography

Dr. Xujie Zhang received the B.E degree in Electrical Engineering from Fudan University in China in 2008. He got his PhD degree in Electrical Engineering from Purdue University in US, in 2013. His research topics focus on image processing and halftoning.

Jan P. Allebach received his BSEE from the University of Delaware in 1972 and his Ph.D. from Princeton University in 1976. Since 1983, he has been at Purdue University where he is currently Hewlett-Packard Professor of Electrical and Computer Engineering. His current research interests include image quality, color imaging and color measurement, printer and sensor forensics, and digital publishing.