

Linear Pixel Shuffling (I): New Paradigms for New Printers

Peter G. Anderson, Jonathan Arney, and Kevin Ayer
Rochester Institute of Technology
Rochester, NY, USA

Abstract

Linear Pixel Shuffling (LPS) is a novel order for image pixel processing which provides opportunities for construction of dot placement algorithms for high-resolution printers through micro-clumping and the formation of pseudo clustered dots.

We present the details of LPS, how to program using it, several of its properties and applications, especially for electrophotographic (EP) imaging.

1. Fibonacci numbers, the golden mean, and applications

The well-known Fibonacci sequence

$$\mathcal{F} : 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, \cdots \tag{1}$$

where each number is the sum of the previous two numbers has the formal, recursive definition

$$\begin{aligned} F_0 &= 0, \;\; F_1 = 1 \\ F_n &= F_{n-1} + F_{n-2} \;\text{ for } n \geq 2 \end{aligned} \tag{2}$$

An alternative definition is

$$F_n = \frac{\alpha^n - \beta^n}{\sqrt{5}} \tag{3}$$

where

$$\alpha, \beta = \frac{1 \pm \sqrt{5}}{2} \tag{4}$$

and

$$\alpha = \lim_{n \to \infty} \frac{F_{n+1}}{F_n} \tag{5}$$

$\alpha \approx 1.618$ is the *golden mean*.

A convenient formulation of the Fibonacci numbers and the golden mean is by means of a matrix. This matrix formulation is particularly useful for our generalizations below.

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix}^n \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} F_{n+1} \\ F_n \end{bmatrix} \tag{6}$$

The golden mean appears in an eigenvector:

$$\begin{bmatrix} 1 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ 1 \end{bmatrix} = \begin{bmatrix} \alpha + 1 \\ \alpha \end{bmatrix} = \alpha \begin{bmatrix} \alpha \\ 1 \end{bmatrix} \tag{7}$$

1.1. One-dimensional blue noise

The *Fibonacci shuffle* is the permutation of $\{0, 1, \cdots, F_n - 1\}$ formed by multiples of a Fibonacci number modulo the following Fibonacci number. Here, for example, is the first full period of the sequence of multiples of $F_6 = 8$ modulo $F_7 = 13$:

$$0, 8, 3, 11, 6, 1, 9, 4, 12, 7, 2, 10, 5 \tag{8}$$

This sequence has several useful properties:

- The period consisting of the first F_n values contains all of the numbers $\{0,\ 1,\ 2,\ \cdots, F_{n+1} - 1\}$, because two successive Fibonacci numbers have no common factors greater than one (they are "relatively prime").
- Numbers close in the shuffle are not close in value. The Fibonacci shuffle was originally developed for a simple approach to progressive rendering of computer graphics (fractals and ray tracing). The idea is to render the scan lines in the order they appear in the shuffle list. $F_{16} = 987$ is a convenient number of scan lines to use for modern computer monitors.
- The sequence has excellent "blue noise" properties. Figure 1 shows the power spectrum of the multiples of $F_{15} = 610$ modulo $F_{16} = 987$.

We can construct a sequence of real numbers in the range [0,1] by

$$s_k = \{k\alpha\} \tag{9}$$

where the brackets denote the fractional part:

$$\{x\} = x - [x] \tag{10}$$

which is also referred to as "x mod 1." The first 13 elements of this sequence are

$$\begin{aligned} &0, 0.618, 0.236, 0.854, 0.472, 0.090, 0.708, \\ &\quad 0.326, 0.944, 0.562, 0.180, 0.798, 0.416 \end{aligned} \tag{11}$$

The power spectrum of $\{\{k\alpha\} \mid 0 \leq k < 987\}$ is nearly identical to Figure 1, because the two sequences are very close to multiples of each other.

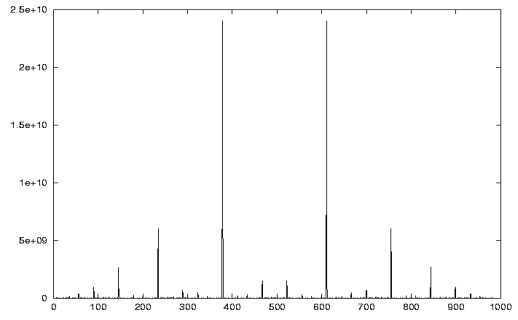

Figure 1: *Power spectrum of the Fibonacci shuffle consisting of multiples of $F_{15} = 610$ modulo $F_{16} = 987$. (In all displayed power spectra, the DC coefficient has been set to zero.)*

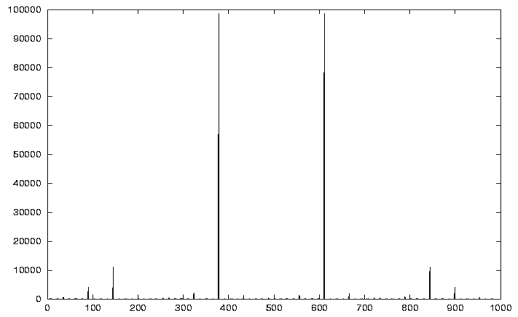

Figure 2: *Power spectrum of $\{\{k\alpha\} \mid 0 \le k < 987\}$ thresholded at 0.5.*

The sequence $\{k\alpha\}$ may be used as a one-dimensional blue noise mask (for example, for dithering audio signals). The power spectrum of the sequence $\{\{k\alpha\} \mid 0 \le k < 987\}$ thresholded at 0.5 is shown in Figure 2.

Another application for the sequence $s_k = \{k\alpha\}$ is Monte Carlo integration:

$$\int_0^1 f(x)dx \approx \frac{1}{N}\sum_{k=0}^{N-1} f(s_k) \tag{12}$$

The values of $\{k\alpha\}$ are very evenly spread throughout [0,1]. For any given number N of points, the gaps between pairs of the s_k take on at most three different values, and these are three numbers in golden mean geometric progression. Each new number s_N subdivides one of the largest gaps in golden mean proportions. For Monte Carlo integration, this is far superior to a white noise pseudo random sequence.

2. Two dimensional generalizations

The Fibonacci sequence is defined by a *second order linear recurrence*, meaning that each term is a linear combination of the two preceding terms. Our "two-dimensional generalization" to the preceding material uses sequences defined by third order linear recurrences. The Tribonacci sequence uses

$$\begin{aligned} T_0 &= 0,\ T_1 = T_2 = 1 \\ T_n &= T_{n-1} + T_{n-2} + T_{n-3} \ \text{ for } n \ge 3 \end{aligned} \tag{13}$$

The first few terms are

$$\mathcal{T} : 0, 1, 1, 2, 4, 7, 13, 24, 44, 81, 149, 274, 504 \tag{14}$$

Another sequence, we call $\mathcal{G}$, is defined by

$$\begin{aligned} G_0 &= 0,\ G_1 = G_2 = 1 \\ G_n &= G_{n-1} + G_{n-3} \ \text{ for } n \ge 3 \end{aligned} \tag{15}$$

The first few terms are

$$\mathcal{G} : 0, 1, 1, 1, 2, 3, 4, 6, 9, 13, 19, 28, 41, 60, 88 \tag{16}$$

Like the Fibonacci numbers, $\mathcal{T}$ and $\mathcal{G}$ are conveniently described by matrix equations:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^n \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} T_{n+1} \\ T_n \\ T_{n-1} \end{bmatrix} \tag{17}$$

and

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}^n \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} G_{n+1} \\ G_n \\ G_{n-1} \end{bmatrix} \tag{18}$$

We get two-dimensional analogues to the golden mean via eigenvectors:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \tau_1 \\ \tau_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \tau_1 + \tau_2 + 1 \\ \tau_1 \\ \tau_2 \end{bmatrix} = \tau_2 \begin{bmatrix} \tau_1 \\ \tau_2 \\ 1 \end{bmatrix} \tag{19}$$

and

$$\begin{bmatrix} 1 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \gamma_1 + 1 \\ \gamma_1 \\ \gamma_2 \end{bmatrix} = \gamma_2 \begin{bmatrix} \gamma_1 \\ \gamma_2 \\ 1 \end{bmatrix} \tag{20}$$

Eq. (19) implies

$$\tau_2^3 = \tau_2^2 + \tau_2 + 1 \tag{21}$$

and Eq. (20) implies

$$\gamma_2^3 = \gamma_2^2 + 1 \tag{22}$$

The vectors $\vec{\tau} = (\tau_1, \tau_2)$ and $\vec{\gamma} = (\gamma_1, \gamma_2)$ provide methods for uniformly sampling points in $[0, 1]^2$ for Monte Carlo integration, generalizing Eq. (9):

$$\vec{s}_k = \{k\vec{\tau}\} = (\{\tau_1\}, \{\tau_2\}) \tag{23}$$

The matrix formulations (Equations (6), (17), and (18)) all use invertible matrices. This allows us to use F_n, G_n, and T_n with negative values of n, which we exploit in Section 2.3.

```
 0 60 32  4 64 36  8 68 40 12 72 44 16
41 13 73 45 17 77 49 21 81 53 25 85 57
82 54 26 86 58 30  2 62 34  6 66 38 10
35  7 67 39 11 71 43 15 75 47 19 79 51
76 48 20 80 52 24 84 56 28  0 60 32  4
29  1 61 33  5 65 37  9 69 41 13 73 45
70 42 14 74 46 18 78 50 22 82 54 26 86
23 83 55 27 87 59 31  3 63 35  7 67 39
64 36  8 68 40 12 72 44 16 76 48 20 80
17 77 49 21 81 53 25 85 57 29  1 61 33
58 30  2 62 34  6 66 38 10 70 42 14 74
11 71 43 15 75 47 19 79 51 23 83 55 27
52 24 84 56 28  0 60 32  4 64 36  8 68
```

Figure 3: *Mask table defined using three consecutive $\mathcal{G}$-numbers: $M_{pq} = (41p + 60q)\%88$.*

2.1. Halftone masks

$\mathcal{T}$ and $\mathcal{G}$ can be used to construct two-dimensional tables that we call "linear pixel shuffling" (LPS), which generalize the one-dimensional Fibonacci shuffle. Let the parameters A, B, and C be three successive numbers in either $\mathcal{T}$ or $\mathcal{G}$. Then the mask M is defined by

$$M_{pq} = (pA + qB)\%C \tag{24}$$

where % denotes the remainder operation. Figure 3 shows a portion of M using $A = G_{12} = 41$, $B = G_{13} = 60$, $C = G_{14} = 88$. Similar to the Fibonacci shuffle, values numerically close are physically distant in M. This property makes these masks excellent candidates for Bayer-type thresholding masks for digital halftoning. Figures 4 and 5 show images of constant gray levels of 25% and 50% quantized to bi-level using masks with $\mathcal{G}$ and $\mathcal{T}$ parameters, respectively. These masks can be made as large as desired, but there is no storage penalty for large masks. The masks are remainders of addition tables of linear progressions, so each entry can be computed directly from the entry above it or to its left using a single addition and a conditional subtraction—this is at most as computationally expensive as a table lookup, but the table is absent. The important property of the three parameters A, B, and C, is their ratios, so masks can be constructed, guided by the parameters of $\mathcal{G}$ and $\mathcal{T}$ numbers, with any desired range of values.

Analogous to the sequence $s_k = \{k\alpha\}$, we can define a two-dimensional mask of values in [0,1] defined by:

$$M_{pq} = \{p\gamma_1 + q\gamma_2\} \tag{25}$$

2.2. Spectral properties of M and their images

Figure 6 shows the power spectra of the 50% black, 64×64 image shown in Figure 4. Figure 7 shows the power spectrum of the thresholding mask used to create that image.

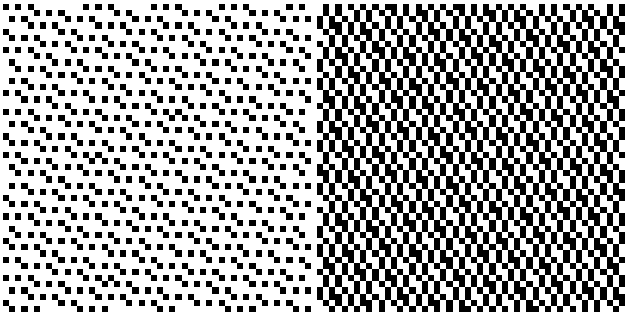

Figure 4: *Dot patterns produced by using the mask M with parameters $A = T_{13} = 927$, $B = T_{14} = 1705$, $C = T_{15} = 3136$, as a threshold mask at levels 25% and 50%.*

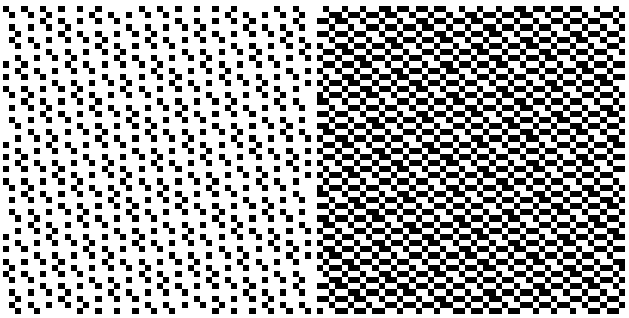

Figure 5: *Dot patterns produced by using the mask M with parameters $A = G_{21} = 1873$, $B = G_{22} = 2745$, $C = G_{23} = 4023$, as a threshold mask at levels 25% and 50%.*

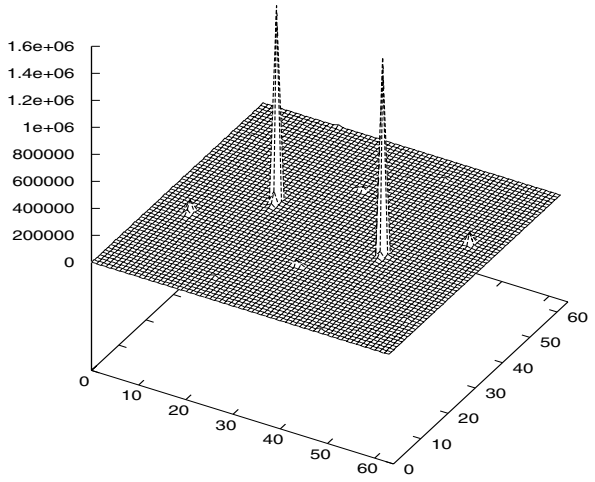

Figure 6: *Power spectrum of the 50% black image of Figure 4.*

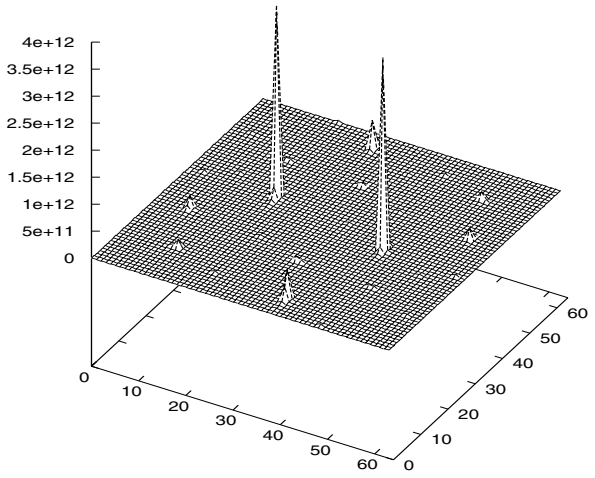

Figure 7: *Power spectrum of the mask M with parameters $A = G_{21} = 1873$, $B = G_{22} = 2745$, $C = G_{23} = 4023$.*

2.3. Processing in LPS order

Figure 3 provides directions for processing the pixels of an image, namely: process the pixels (p, q) such that $M_{pq} = 0$, then those such that $M_{pq} = 1$, and so on. The following algorithm will do this.

```
for ( i = 0; i < G_n; i++ ) {
    for ( j = 0; j < G_n; j++ ) {
        [p]   [G_{3-n}  G_{n-3}] [i]
        [q] = [G_{2-n}  G_{n-2}] [j]
        process pixel (p, q)
    }
}
```

It is straightforward to see that the coordinates (p, q) satisfy $M_{pq} = i$; see [1].

When we use the LPS order to progressively render an image we find it convenient to use "fat pixels," that is, the first several pixels we determine and render can be surrounded by a square of pixels the same color. This serves to fill the screen rapidly, showing a low resolution image. As more and more pixels are computed and rendered, the size of the squares can be decreased to avoid overwriting any previously rendered pixel (of course we overwrite the surrounding squares). Typically, we begin with 7×7 squares. If the image dimensions are $G_n \times G_n$, then we will switch to 5×5 squares when we reach $i = G_{n-10}$ (i is the outer loop induction variable in the above program), switch to 3×3 squares when $i = G_{n-7}$, and 1×1 squares (isolated pixels) when $i = G_{n-5}$. For example, for an image of size $G_{20} \times G_{20} = 872 \times 872$, we start with 7×7 squares, and reduce to 5×5 squares at $i = 19$, 3×3 squares at $i = 60$, and isolated pixels at $i = 129$. Figure 8 shows a sample image at 2%, 7%, 15%, and 100% rendering using the LPS order combined with fat pixels.

The fat pixel idea, as illustrated here, extends to many areas of image processing. In Section 3, we discuss an error diffusion algorithm that takes advatage of the uniform properties of LPS pixel visitation. We note, for example, that a vast number of the pixels in the 2% rendered picture are already correct, and do not need to be rendered, stored, or transmitted. That is, we have an algorithm that is a two-dimesional analog of run-length compression. We have investigated this algorithm, and it is competitive with run-length in compression ratios, but it has the special feature: a prefix of an image's compressed representation is a lossy compression of the entire image, rather than a lossless compression of a portion of the image.

Other applications for LPS are Hough transforms and approximations to morphological transforms.

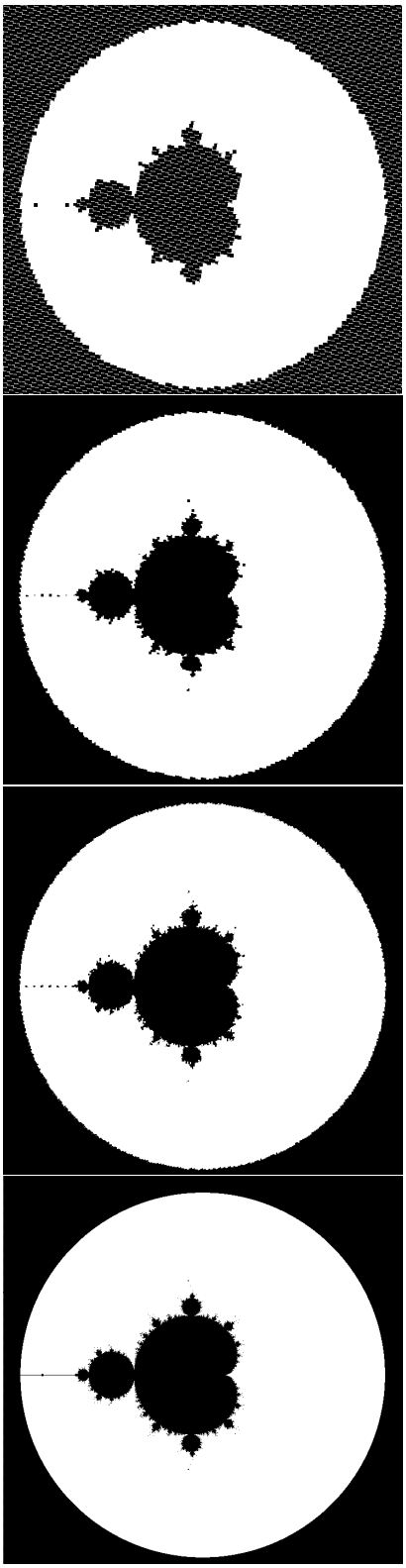

Figure 8: *The Mandelbrot set, partially rendered using fat pixels of sizes:* 7×7 *(2% rendered),* 5×5 *(7%),* 3×3 *(15%), and* 1×1 *(100%).*

3. Electrophotographic dot gain

Ink dots on EP paper are approximately small circles, not squares. To assure possible 100% ink coverage, the dots' diameters must be at least $D\sqrt{2}$ inches, where D is the number of dots per inch. The *ink dot gain* resulting from this is at least 1.57 (the area of the ink circle as a multiple of the nominal area of a pixel).

Another phenomenon, known as *optical dot gain*, provides a gray halo around ink dots, because light photons entering paper near an ink boundary are diffused within the paper and trapped by the ink.

Consequently, a single black pixel has the printed blackness effect of 2.0–2.5 its nominal value (see [2] and [3]), and printed images are much darker than the nominal pixel coverage suggests. An image with 50% black pixels arranged in a checkerboard alternating pattern will be totally black.

One method to correct for this dot gain is to scale the thresholding matrix, which can be done by a lookup table or a calculation. Linear pixel shuffling provides an opportunity for more delicately responding to this dot gain phenomenon via a method of error diffusion (introduced in [1] and [4]) in which the image pixels are visited and quantized in the LPS visitation order described above. This permits the quantization error to be divided among unquantized pixels onmidirectionally. The following pseudocode shows a generic error diffusion algorithm to convert an input image with pixel values $0 \leq g_{pq} \leq 1$ to a bi-level output image with pixel values $b_{pq} \in \{0, 1\}$ (here 0 denotes white and 1 denotes black). If the output pixel is $b_{pq} = 1$, then the error is computed using a black overshoot value of dg, the dot gain parameter (in 2.0–2.5).

```
for each image pixel (p, q) {
    if (g_pq > 0.5)
        then b_pg = 1
        else b_pg = 0
    error = dg * b_pq − g_pq
    distribute error among unprocessed neighbors
}
```

We suggest that the pixels (p, q) be chosen in LPS order.

4. Electrophotographic precision and stability

High resolution (1,200 dpi or more) EP printers cannot reliably print isolated black dots. They can, however, place clusters of black dots with high positioning accuracy. Halftone al-

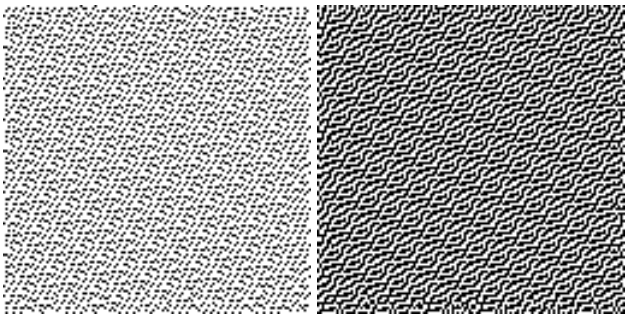

Figure 9: *Patches at gray level 64 (25%) and 128 (50%) converted to black and white using linear pixel shuffling error diffusion.*

gorithms need to take this phenomenon into account, converting continuous tones scales to clustered dots or little lines and curves. The linear pixel shuffling error diffusion algorithm can produce bilevel images with lines in the microstructure. Figure 9 shows two images this algorithm produces using the following error diffusion kernel:

$$\begin{bmatrix} 0 & 1 & 1 & 1 & 0 \\ 1 & 2 & 3 & 2 & 1 \\ 1 & 3 & \mathbf{P} & 3 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ 0 & 1 & 1 & 1 & 0 \end{bmatrix} \tag{26}$$

The error from quantizing the pixel **P** is distributed among its unquantized neighbors according to the weights indicated. We have investigated several other kernels, and the textures can be even more striking. This technique can provide a wide variety of special halftoning effects and work well with the peculiarities of modern printers.

The LPS error diffusion process can also quantize small line segments in various orientations rather than simply individual pixels. This can force black dot clustering and hence EP image stability. Similarly, the LPS masks M can also be modified to use small constant lines and similar figures yet retain the LPS number patterns and their desirable blue noise properties. This is work in progress.

References

[1] Peter G. Anderson. Error diffusion using linear pixel shuffling. In *Proceeding of Image Processing, Image Quality, and Image Capture Systems Conference (PICS 2000)*, Springfield, VA, 2000. The Society for Imaging Science & Technology.

[2] Jonathan Arney, Peter G. Anderson, Kevin Ayer, and Prashant Mehta. Linear pixel shuffling (ii): An experimental analysis of tone and spacial characteristics. In *Proceedings of The 16th International Congress on Digital Printing Technologies (NIP 16)*, Springfield, VA, 2000. The Society for Imaging Science & Technology.

[3] Jonathan Arney, Peter G. Anderson, Kevin Ayer, and Prashant Mehta. The MTF of printing systems. In *Proceedings of The 16th International Congress on Digital Printing Technologies (NIP 16)*, Springfield, VA, 2000. The Society for Imaging Science & Technology.

[4] John Szybist and Peter G. Anderson. Digital halftoning using error diffusion and linear pixel shuffling. In Fredric T. Howard, editor, *Applications of Fibonacci Numbers*, Boston, MA, 1999. Klewer Academic Publishers.

This research was supported by a grant from Hewlett-Packard.