DIGITAL HALFTONING USING ERROR DIFFUSION AND LINEAR PIXEL SHUFFLING

John Szybist and Peter G. Anderson

INTRODUCTION

Digital halftoning is used to render continuous gray-scale images to a device, such as a laser printer or video display, that is only capable of binary output (black and white). Other applications include image transmission and storage where the gray-scale to binary data reduction translates to reduced transmission and storage costs. The idea is that the local density of dots mimics the original gray-scale and appears nearly the same to a viewer. This method works because the human visual system acts like an averaging or low-pass filtering device when presented with the relatively high frequency dot pattern. The challenge is to optimize the appearance by minimizing side effects such as undesirable texture artifacts and false contours, that can be produced by the pattern. The general issues that must be addressed are those concerned with the original image relative to dynamic range and spatial resolution of the gray-scale and those concerned with the visibility of the halftone pattern.

Digital halftoning techniques receive as input an image represented as an $m \times n$ array A of multi-level values (e.g., 256 levels of gray), and produce an $m \times n$ array B of bi-level values (black and white) that modulate the bi-level imaging system. "Good" algorithms do this in such a fashion that the average gray level in the neighborhood of B_{ij} approximates the value of A_{ij} at a reasonable viewing distance (generally considered 18 inches for 300 dots per inch).

This paper is in final form and no version of it will be submitted for publication elsewhere.

F. T. Howard (ed.), *Applications of Fibonacci Numbers*

Digital halftone algorithms can be categorized as either "point" or "neighborhood" [6]. A point algorithm renders B_{ij} using information from A at A_{ij} only. Neighborhood algorithms use information from A_{ij} and pixels in some neighborhood of A_{ij} to render B_{ij}. The selection between these two classes of algorithms will usually trade off rendering speed (point algorithms) and resulting image quality (neighborhood algorithms). Point algorithms quite often show contouring (especially if they support too few gray levels), degrade contrast, and have quite prominent texture artifacts.

ERROR DIFFUSION

Error diffusion is a neighborhood method where the value of each input pixel, A_{ij}, is compared to a (usually) fixed threshold, θ, to determine the value of the corresponding output pixel, B_{ij} and the difference ("error") between A_{ij} and B_{ij} is diffused to some as yet unprocessed pixels in a small neighborhood of A. We suppose that the input pixel values are integers in the range 0-255, expressing a range of 256 gray values from white to black, and that the quantized output pixel values are either 0 or 255 (white or black, respectively). In this case $\theta = 128$. The following pseudo code expresses the general error diffusion algorithm:

```
for every (i, j) pixel location in the image
    {Quantize pixel (i, j)}
    if A_ij ≥ θ then B_ij←255
      else B_ij←0

    {Determine and disperse the quantization error}
    E←B_ij − A_ij
    divide up E, and add its portions
    to unprocessed neighbors of A_ij
```

In most error diffusion algorithms, the order of pixel processing corresponding to the code step

```
for every (i, j) pixel location in the image
```

generally takes the form of the following raster ordering:

```
for i←0, step 1, while i < max.i
    for j←0, step 1, while j < max.j
        process pixed (i, j)
```

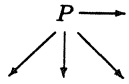
One common variation on the left-to-right, row-by-row ordering is *serpentine* or *bostrouphedon* (as the ox plows) order, in which the processing order for lines alternates between left-to-right

and right-to-left.

The literature has many references to variations on error diffusion algorithms that analyze the size of the neighborhood and the amount of error diffused to each neighbor (denoted as the diffusion kernel or mask). The classic diffusion mask developed by Floyd and Steinberg [4] will be used to benchmark our development:

$$D = \frac{1}{16}\begin{bmatrix} & P & 7 \\ 3 & 5 & 1 \end{bmatrix}$$

P represents the pixel A_{ij} under consideration. This mask diffuses the error from P to pixels yet to be rendered when scanning in a top-to-bottom, left-to-right, raster fashion:



Specifically, the quantization error, $E = B_{ij} - A_{ij}$, is dispersed by adding $\frac{7}{16}E$ to $A_{i,j+1}$, $\frac{3}{16}E$ to $A_{i+1,j-1}$, $\frac{5}{16}E$ to $A_{i+1,j}$ and $\frac{1}{16}E$ to $A_{i+1,j+1}$. The pixel to the left of P, $A_{i,j-1}$ has already been quantized, as have all the pixels in the rows above P. It is possible, and not at all damaging, if the values of unquantized A_{ij}'s go outside the range 0-255. Thus, the error is pushed down and to the right—like snow before a plow, it eventually exceeds the capacity (i.e., θ), and becomes exposed in clumps. This results in the "worm" artifacts; see the hat in Fig. 1.

Error diffusion uses the error to bring the average gray level in the neighborhood of B_{ij} closer to the value of A_{ij}. Some problems evident in Figure 1 such as the "tearing" pattern at midtone and the "worm" effects at other tones will be addressed by our LPS error diffusion. The "tearing" evident down the center of Figure 1 is caused by the checkerboard pattern that Floyd-Steinberg generates at the neighborhood of 50% gray values and, to a lesser extent, around 25% and 75% gray.

LINEAR PIXEL SHUFFLING

Anderson [2, 3] developed a method, called *Linear Pixel Shuffling* (LPS), of visiting pixels in a smooth manner all over an image. This method extends to higher dimensions the one-dimensional permutation of the integers, $0, \cdots, F_n - 1$ given by the rule $v_k = kF_{n-1} \% F_n$ ("%" denoting remainder). For $n = 8$, this permutation of $0, \cdots, 20$ is

$$0, 13, 5, 18, 10, 2, 15, 7, 20, 12, 4, 17, 9, 1, 14, 6, 19, 11, 3, 16, 8.$$

The special property of this permutation—namely, that sequentially near values are numerically distant $\left(|p-q| \text{ is small} \Leftrightarrow |v_p - v_q| \text{ is large}\right)$ was exploited in [1]. For the two-dimensional generalization of this permutation, we utilize a Fibonacci-like sequence, G_n, third-order linear recurrence defined as follows:

$$G_0 = 0$$

$$G_1 = G_2 = 1$$

$$G_n = G_{n-1} + G_{n-3} \text{ for } n \geq 3.$$

This sequence begins:

$$0, 1, 1, 1, 2, 3, 4, 6, 9, 13, 19, 28, 41, 60, 88, 129, 189, 277, 406, 595, 872, 1278, 1873, 2745, \cdots$$

This sequence provides the parameters for LPS as follows. Suppose the image is contained within a $G_n \times G_n$ square. We introduce the LPS shuffling matrix

$$S = \begin{pmatrix} G_{n-2}^2 + G_{n-1}^2 & G_{n-1} \\ G_{n-1}^2 & G_n - G_{n-2} \end{pmatrix}$$

and specify the pixel visitation order for error diffusion using a change of basis:

```
for i←0, step 1, while i < G_n
        for j←0, step 1, while j < G_n

                {Perform matrix-vector multiplication}
                process the pixel at location S[i j]'
```

For convenience, we describe the image as a $G_n \times G_n$ square. In practice, we must include the image in $G_n \times G_n$ square, and ignore the pixels in that square that are outside the image. This is reminiscent of the common computing practice of using a power of 2 for sizes of the data structures.

To describe this order of pixel processing, we introduce an auxiliary $G_n \times G_n$ array, T, of integers in the range $0, \cdots, G_n - 1$, with the rule

$$T_{ij} = (iG_{n-2} + jG_{n-1}) \% G_n.$$

This is illustrated in Fig. 3 for $n = 10$, and $G_n = 19$. The specifically useful properties of T are:

- Numerically close values are geometrically dispersed; or, equivalently, neighboring values are numerically distant. The 0's in T are widely distributed; the 1's are also, and they are not near the 0's; and so on.
- T is specified by a linear rule, so that the neighborhoods of each entry are arithmetic translates (mod G_n) of neighborhoods of any other entry.
- T gives the LPS order of pixel visitation: the (i, j)-positions such that $T_{ij} = 0$ are visited first, then the positions with $T_{ij} = 1$, and so on. (The order in which the 0 positions are visited is irrelevant for the present application. This could be an opportunity to exploit parallelism, by processing all the 0 positions simultaneously, followed by the 1's, etc.)

The areas outlined in the center of Fig. 3 are discussed in the following section. That the matrix-vector multiplication indicated in the above pseudocode performs the desired transformations follows from these observations (developed in [3]):

- If $[p\ q]' = S[0\ 1]'$, then $T_{pq} = 0$.
- If $[p\ q]' = S[1\ 0]'$, then $T_{pq} = 1$.
- So, by linearity, if $[p\ q]' = S[i\ j]'$, then $T_{pq} = i$.

LPS ERROR DIFFUSION

The LPS halftoning algorithm scans the image using the LPS order described in the previous section, and diffuses error to unprocessed pixels in all directions using an algorithm similar to Floyd and Steinberg's. We use the following diffusion mask:

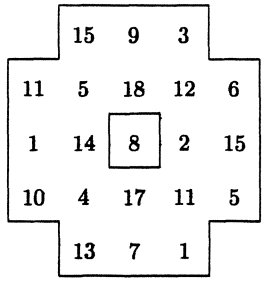
$$D = \frac{1}{32}\begin{bmatrix} & 1 & 1 & 1 & \\ 1 & 2 & 3 & 2 & 1 \\ 1 & 3 & P & 3 & 1 \\ 1 & 2 & 3 & 2 & 1 \\ & 1 & 1 & 1 & \end{bmatrix}$$

Our LPS error diffusion algorithm takes the form:

```
for i←0, step 1, while i < G_n
    Determine the unprocessed pixels at positions with label i
    for j←0, step 1, while j < G_n
        {Determine the next pixel to process}
        [p q]'←S[i j]'
        {Note: T_pq = i}

        {Quantize and distribute the error}
        if A_pq > θ then B_pq←255
            else B_pq←0
        distribute the quantization error, E←B_pq − A_pq
        over unprocessed pixels in the neighborhood of A_pq
```

The LPS scanning order allows us easily to determine which of the pixels in the 5×5 error diffusion kernel D have already been processed. Because the LPS rule is a simple linear rule, the neighborhood of every pixel labeled i is identical (see Fig. 3). The pixel labels in a kernel centered at $T_{pq} = i$ are a translation by $+i \pmod{G_n}$ of the labels in the kernels centered at $T_{00} = 0$. The following neighborhood diagram was excised from Fig. 3:



In the neighborhood of a $T_{pq} = 8$ we have 9 already processed pixels (those with labels less than eight, namely, 3, 5, 6, 1, 2, 4, 5, 7, and 1) and 11 unprocessed pixels (those with labels greater than 8, namely, 15, 9, 11, 18, 12, 14, 15, 10, 17, 11, and 13).

We easily keep the description of the error diffusion neighborhood up to date, for each new i in the program—this i is controlled by the outer loop, so there is minor impact on program performance. Donald Knuth employed a similar idea in his "dot diffusion" algorithm [5].

DISCUSSION

Figs. 1 and 2, *Woman With a French Horn*[1] is a scene with several natural and man-made objects and their textures. A principal advantage of Floyd-Steinberg error diffusion is its enhancement of edges and preservation of high frequency image information (here in the woman's hair and sweater). These images show that our LPS error diffusion appears to do admirably in this aspect as well.

Differences between the two error diffusion algorithms are also evident in two artificial images, *arctangent* (Figs. 4 and 5) and *ripples* (Figs. 6 and 7). Arctangent was constructed so that the gray values range fill the range from 0 to 255; the pixel at position (p, q) has a gray value proportion to $\text{atan}\left(\frac{p}{q}\right)$. This provides a ramping from the two extreme colors in the top and left edges, exposing directional artifacts of an algorithm, and a range of gray-scale gradients. Fig. 5 illustrates that LPS error diffusion appears to introduce some undesired texture artifacts into the image. The Floyd-Steinberg rendering of arctangent clearly shows artifacts that LPS overcomes; namely, tearing at the 50% gray level, worms, and a delayed introduction of black pixels in the very light gray region. LPS, unfortunately, introduces some evident "grain" and undesired stripes. (Choices of error diffusion algorithms are, unfortunately, image dependent.)

Ripples was constructed so that its pixel values filled a midrange of values, 0-127. The pixel at position (p, q) has a gray value proportional to $\cos(\alpha(p^2 + q^2))$, where α was chosen to make the maximum frequency in the image barely visible. The ripples appear in a range of orientations to expose any directional artifacts in the algorithms. Here, we are quite pleased with the performance of LPS error diffusion. The high frequency information shows clearly in this experiment.

---

[1]Image courtest of Heidelberger Druckmaschinen A.G.

As we see it, the primary deficiency in the LPS error diffusion algorithm is a small reduction in spatial resolution and the introduction of some low frequency texture in the halftoned image. This deficiency is an appropriate area for further investigation. Several *ad hoc* attempts, using alternative diffusion masks, increased the spatial resolution and lowered the texture pattern, but the cost was the appearance of false contours, which was original mask nicely avoided.

We experimented with masks such as the following,

$$D = \frac{1}{44}\begin{bmatrix} & 0 & 1 & 0 & \\ 0 & 3 & 7 & 3 & 0 \\ 1 & 7 & P & 7 & 1 \\ 0 & 3 & 7 & 3 & 0 \\ & 0 & 1 & 0 & \end{bmatrix}$$

but undesirable artifacts and image degradation were still evident.

Our algorithm requires more storage than Floyd-Steinberg, because the error needs to be saved for a longer period of time—until all pixels have been rendered. We used a data structure with a size equivalent to the size of the entire image to store our error. Error storage for the Floyd-Steinberg technique is merely the size of a single scan line.

Our algorithm has significant advantages over Floyd-Steinberg relative to the worm and tearing artifacts, and LPS compares favorably overall. Further studies will include a more strict quantization of the image transformation, psychophysical experiments, the texture problem, and applications in color halftoning.

REFERENCES

[1] Anderson, Peter G. "A Fibonacci-based pseudo-random number generator. Fibonacci Numbers and Their Applications, Volume 4. Edited by G.E. Bergum, A.F. Horadam and A.N. Philippou. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1990: pp. 1-8.

[2] Anderson, Peter G. "Multidimensional golden means. Fibonacci Numbers and Their Applications, Volume 5. Edited by G.E. Bergum, A.F. Horadam and A.N. Philippou. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1992: pp. 1-10.

[3] Anderson, Peter G. "Advances in linear pixel shuffling. Fibonacci Numbers and Their Applications, Volume 6. Edited by G.E. Bergum, A.F. Horadam and A.N. Philippou. Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994: pp. 1-22.

[4] Floyd, R.W. and Steinberg, L. "An adaptive algorithm for spatial gray scale." *In SID International Symposium Digest of Technical Papers*, Society for Information Display (1974): pp. 36-37.

[5] Knuth, Donald E. "Digital halftones by dot diffusion." *ACM Transactions on Graphics*, Vol. *6.4* (1987): pp. 245-273.

[6] Ulichney, Robert Digital Halftoning. The MIT Press, 1987.

AMS Classification Numbers: 68U99, 11B39

Figure 1: Floyd-Steinberg error diffusion. "Worm" artifacts appear in the hat, "tearing" in the background.

Figure 2: LPS error diffusion

0	13	7	1	14	8	2	15	9	3	16	10	4	17	11	5	18	12	6
9	3	16	10	4	17	11	5	18	12	6	0	13	7	1	14	8	2	15
18	12	6	0	13	7	1	14	8	2	15	9	3	16	10	4	17	11	5
8	2	15	9	3	16	10	4	17	11	5	18	12	6	0	13	7	1	14
17	11	5	18	12	6	0	13	7	1	14	8	2	15	9	3	16	10	4
7	1	14	8	2	15	9	3	16	10	4	17	11	5	18	12	6	0	13
16	10	4	17	11	5	18	12	6	0	13	7	1	14	8	2	15	9	3
6	0	13	7	1	14	8	2	15	9	3	16	10	4	17	11	5	18	12
15	9	3	16	10	4	17	11	5	18	12	6	0	13	7	1	14	8	2
5	18	12	6	0	13	7	1	14	8	2	15	9	3	16	10	4	17	11
14	8	2	15	9	3	16	10	4	17	11	5	18	12	6	0	13	7	1
4	17	11	5	18	12	6	0	13	7	1	14	8	2	15	9	3	16	10
13	7	1	14	8	2	15	9	3	16	10	4	17	11	5	18	12	6	0
3	16	10	4	17	11	5	18	12	6	0	13	7	1	14	8	2	15	9
12	6	0	13	7	1	14	8	2	15	9	3	16	10	4	17	11	5	18
2	15	9	3	16	10	4	17	11	5	18	12	6	0	13	7	1	14	8
11	5	18	12	6	0	13	7	1	14	8	2	15	9	3	16	10	4	17
1	14	8	2	15	9	3	16	10	4	17	11	5	18	12	6	0	13	7
10	4	17	11	5	18	12	6	0	13	7	1	14	8	2	15	9	3	16

Figure 3: The numbers in this table satisfy $T_{ij} - (9i + 13j)\%19$. The parameters (9, 13, 19) are a consecutive triple in the sequence G_n. The pixels in a 19×19 image would be processed, in LPS order, as shown here; namely, all 19 positions labeled 0 are processed first, then those labeled 1, and so on. The outlined section centered at $\boxed{8}$ pictures the error diffusion neighborhood (the region wherein we disperse the quantization error) for the pixel in the center.

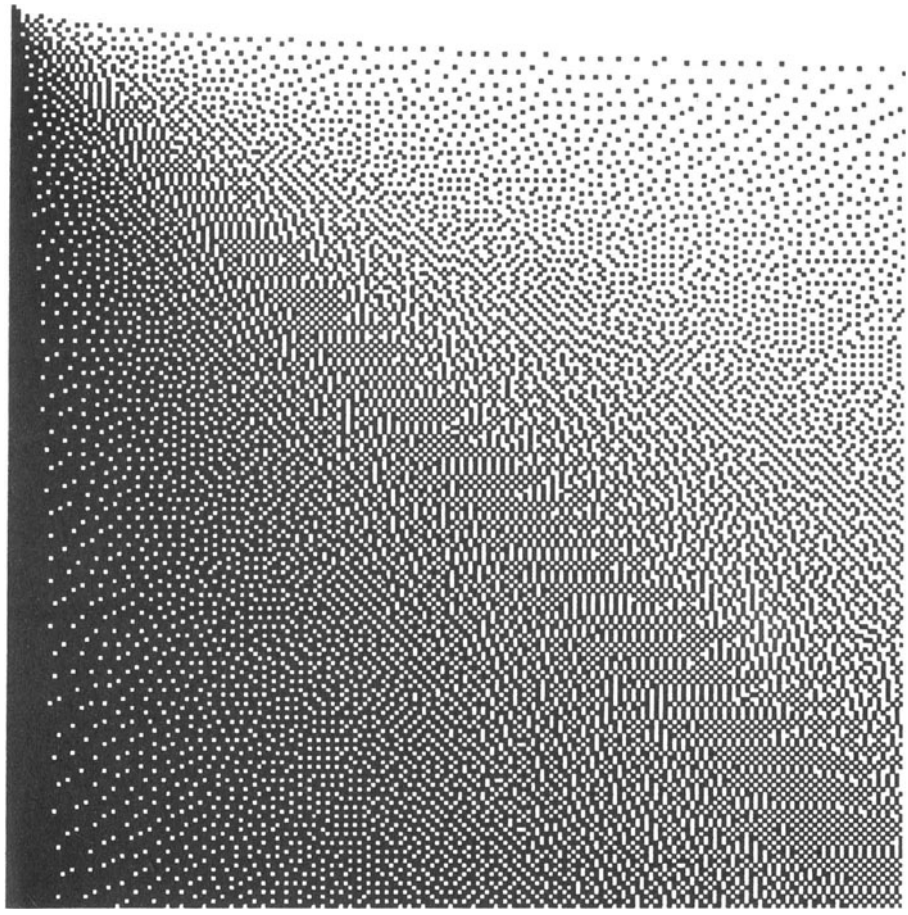


Figure 4: Floyd-Steinberg error diffusion of arctangent. The jarring texture on the diagonal of this image is "checkerboarding." In higher resolution images, this appears as "tearing." It is especially apparent in images produced on less expensive laser and ink jet printers which produce round black pixel; these black pixels by necessity encroach on the areas of neighboring white pixels, resulting in nonomonotonic (let alone nonlinear) printer responses.

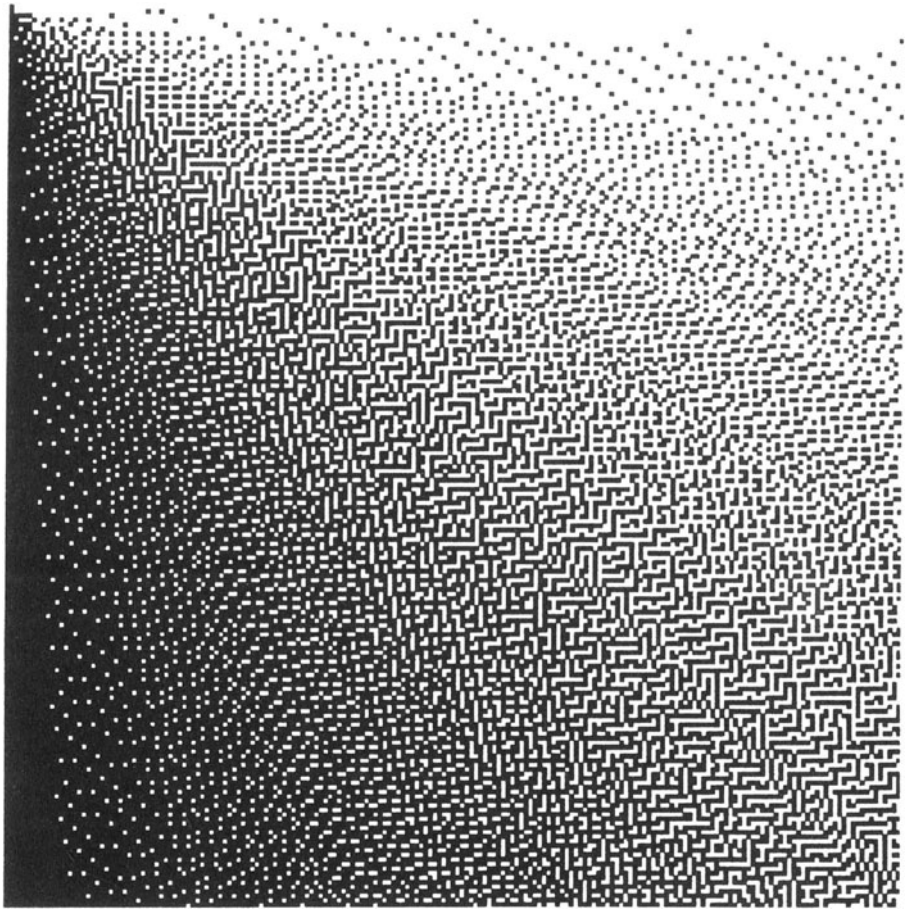


Figure 5: LPS error diffusion of arctangent.

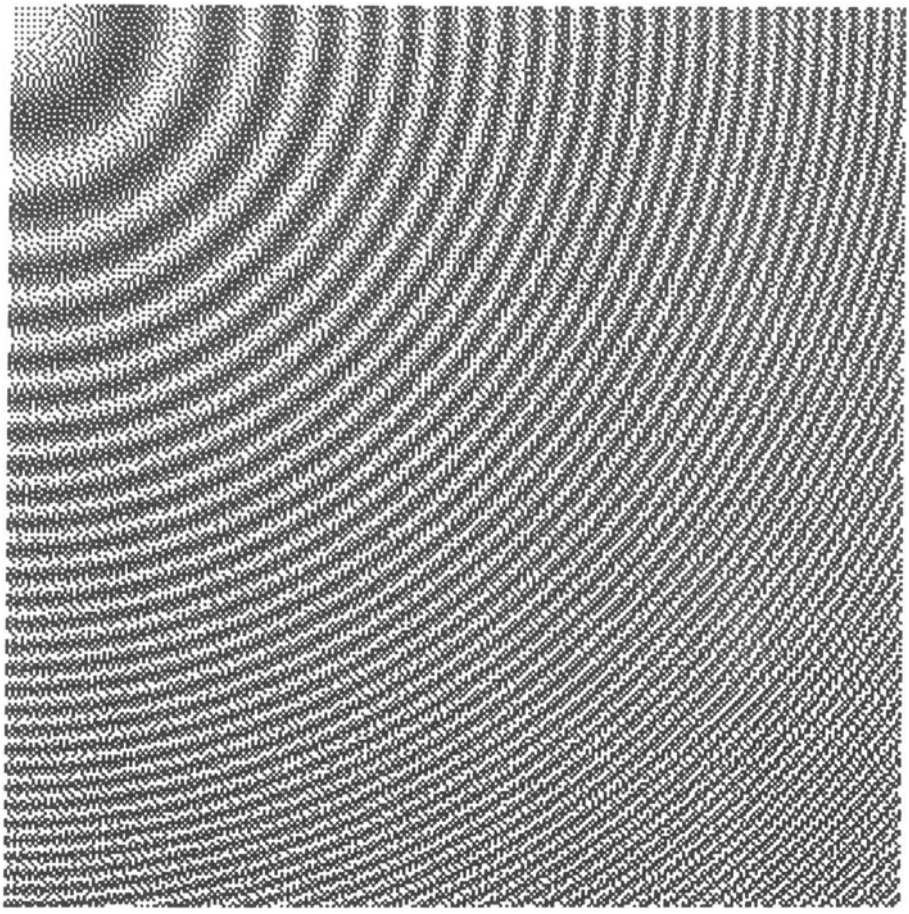


Figure 6: Floyd-Steinberg error diffusion of ripples.

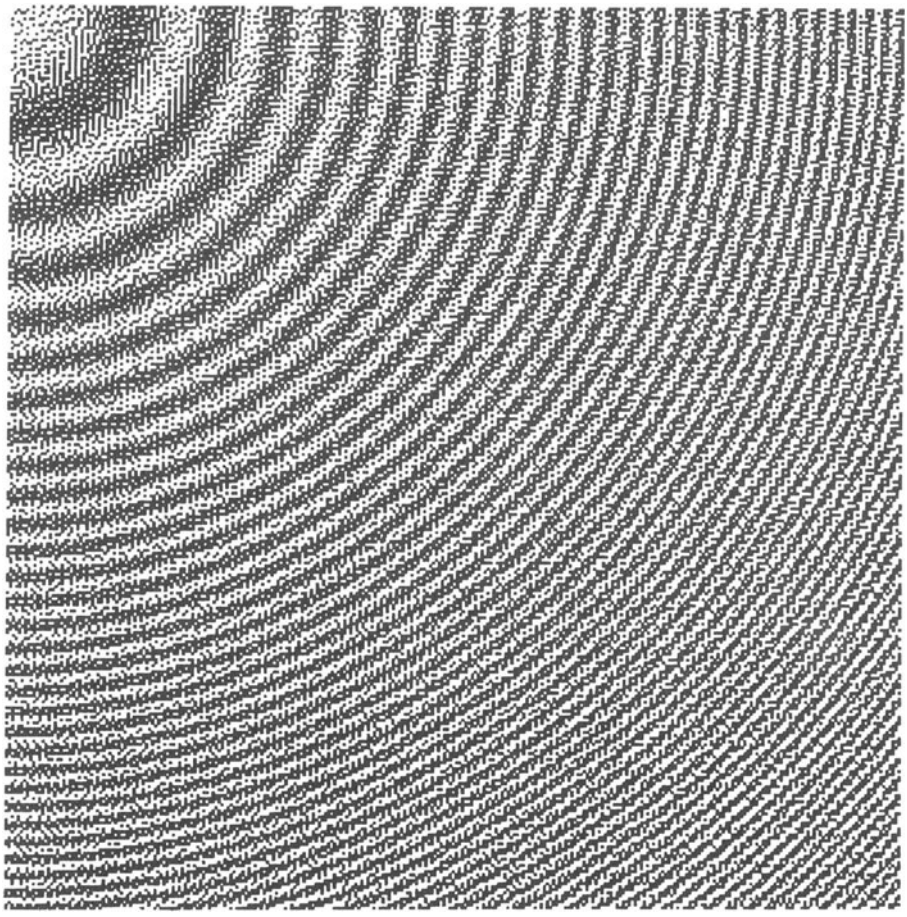


Figure 7: LPS error diffusion of ripples.