Digital Halftones by Dot Diffusion

DONALD E. KNUTH
Stanford University

This paper describes a technique for approximating real-valued pixels by two-valued pixels. The new method, called dot diffusion, appears to avoid some deficiencies of other commonly used techniques. It requires approximately the same total number of arithmetic operations as the Floyd–Steinberg method of adaptive grayscale, and it is well suited to parallel computation; but it requires more buffers and more complex program logic than other methods when implemented sequentially. A "smooth" variant of the method may prove to be useful in high-resolution printing.

Categories and Subject Descriptors: I.3.3 [**Computer Graphics**]: Picture/Image Generation—*display algorithms*; I.4.1 [**Image Processing**]: Digitization—*quantization*; I.4.3 [**Image Processing**]: Enhancement—*grayscale manipulation*

General Terms: Algorithms

Additional Key Words and Phrases: Bilevel display, constrained average, edge enhancement, error diffusion, facsimiles, Floyd–Steinberg method, minimized average error, Mona Lisa, ordered dither, parallel computing, printing

1. INTRODUCTION

Given an $m \times n$ array A of real values between 0 and 1, we wish to construct an $m \times n$ array B of zeros and ones such that the average value of the entries $B[i,j]$ when (i,j) is near (i_0,j_0) is approximately equal to $A[i_0,j_0]$. In applications, A represents the light intensities in an image that has been scanned by some sort of camera; B represents a binary approximation to the image that might appear on the screen of a personal computer or on the pages produced by a laser printer.

2. ERROR DIFFUSION

An interesting solution to this problem was introduced by Floyd and Steinberg [7], who computed B from A by applying the following algorithm:

```
for $i := 1$ to $m$ do for $j := 1$ to $n$ do
  begin if $A[i,j] < \frac{1}{2}$ then $B[i,j] := 0$ else $B[i,j] := 1$;
  $err := A[i,j] - B[i,j]$;
  $A[i,j+1] := A[i,j+1] + err * alpha$;
  $A[i+1,j-1] := A[i+1,j-1] + err * beta$;
  $A[i+1,j] := A[i+1,j] + err * gamma$;
  $A[i+1,j+1] := A[i+1,j+1] + err * delta$;
  end.
```

The preparation of this paper was supported in part by National Science Foundation grant CCR-8610181.
Author's address: Computer Science Department, Stanford University, Stanford, CA 94305-3068.

Here α, β, γ, δ are constants chosen to diffuse the error, which is directed proportionately to nearby elements whose B values have not yet been computed. Floyd and Steinberg suggested taking $(\alpha, \beta, \gamma, \delta) = (\frac{7}{16}, \frac{3}{16}, \frac{5}{16}, \frac{1}{16})$. A similar but more complex method had previously been published by Schroeder [22].

The Floyd–Steinberg method often gives excellent results, but it has drawbacks. In the first place, it is an inherently serial method; the value of $B[m, n]$ depends on all mn of the entries of A. Furthermore, it sometimes puts "ghosts" into the picture; for example, when faces are treated by this approach, echoes of people's hairlines can occasionally be seen in the middle of their foreheads. Several other difficulties are discussed below.

The ghosting problem can be ameliorated by choosing $(\alpha, \beta, \gamma, \delta)$ so that their sum is less than 1; then the influence of $A[i, j]$ on remote elements decays exponentially. However, the ghosts cannot be exorcised completely in this manner. Suppose, for example, that $A[i, j]$ has the constant value a for all i and j, and let $\alpha + \beta + \gamma + \delta = \theta \le 1$. If a is very small, the entries of $B[i, j]$ for small i will all be zero, and the entries of $A[i, j]$ will build up to the limiting value

$$a(1 + \theta + \cdots + \theta^{i-1})(1 + \alpha + \alpha^2 + \cdots) = \frac{a(1 + \theta + \cdots + \theta^{i-1})}{1 - \alpha}$$

for large j. If we choose a so that this value is just slightly less than $\frac{1}{2}$, the $(i + 1)$st row will suddenly have many of its B values set to 1, after they had been 0 in all previous rows.

Floyd [personal communication, May, 1987] has found that ghosts disappear if the intensities $A[i, j]$ are rescaled. For example, we can replace each $A[i, j]$ by $0.1 + 0.8A[i, j]$. This works because the human eye is more sensitive to contrast than to absolute signal levels.

3. ORDERED DITHER

A second approach to the problem is the interesting technique of *ordered dither* [4, 16, 17]. Here we divide the set of all pairs (i, j) into, say, 64 classes numbered from 0 to 63, based on the values of i mod 8 and j mod 8 as follows:

0	32	8	40	2	34	10	42
48	16	56	24	50	18	58	26
12	44	4	36	14	46	6	38
60	28	52	20	62	30	54	22
3	35	11	43	1	33	9	41
51	19	59	27	49	17	57	25
15	47	7	39	13	45	5	37
63	31	55	23	61	29	53	21

If (i, j) belongs to class k, $B[i, j]$ is set to 1 if and only if $A[i, j] \ge (k + 0.5)/64$. In other words, each pixel is thresholded on the basis of the value in the corresponding position of the dither matrix. Notice that, if $A[i, j]$ has a constant value a, this method will turn on exactly t pixels in every 8×8 submatrix of B, where $|t/64 - a| \le 1/128$.

4. DOT DIFFUSION

The technique of ordered dither is completely parallel and ghost free, but it tends to blur the images. It would be nice to have a solution that retains both the sharpness of the Floyd–Steinberg method and the parallelism of ordered dither.

The following technique seems to have the desired properties. Let us divide the positions (i, j) into 64 classes according to (i mod 8, j mod 8) as above, but with the following matrix of class numbers:

34	48	40	32	29	15	23	31
42	58	56	53	21	5	7	10
50	62	61	45	13	1	2	18
38	46	54	37	25	17	9	26
28	14	22	30	35	49	41	33
20	4	6	11	43	59	57	52
12	0	3	19	51	63	60	44
24	16	8	27	39	47	55	36

(The intuition that suggested this matrix will be explained later; for now, let us simply consider this to be an arbitrary permutation of the numbers $\{0, 1, \ldots, 63\}$.) We now can perform the following diffusion algorithm:

```
for $k := 0$ to 63 do
  for all $(i, j)$ of class $k$ do
    begin if $A[i, j] < \frac{1}{2}$ then $B[i, j] := 0$ else $B[i, j] := 1$;
    $err := A[i, j] - B[i, j]$;
    ⟨Distribute $err$ to the neighbors of $(i, j)$ whose class numbers exceed $k$⟩;
    end.
```

Pixels of class 0 are computed first, then those of class 1, etc.; errors are passed to neighboring elements yet to be computed.

Each position (i, j) has four orthogonal neighbors (u, v) such that $(u - i)^2 + (v - j)^2 = 1$, and four diagonal neighbors (u, v) such that $(u - i)^2 + (v - j)^2 = 2$. One feasible way to do the error distribution in the diffusion algorithm is to proceed as follows:

```
⟨Distribute $err$ to the neighbors of $(i, j)$ whose class numbers exceed $k$⟩ =
  $w := 0$;
  for all neighbors $(u, v)$ of $(i, j)$ do
    if class$(u, v) > k$ then $w := w +$ weight$(u - i, v - j)$;
  if $w > 0$ then for all neighbors $(u, v)$ of $(i, j)$ do
    $A[u, v] := A[u, v] + err *$ weight$(u - i, v - j)/w$.
```

We can choose the weight function to be weight$(x, y) = 3 - x^2 - y^2$; this weighs orthogonal neighbors twice as heavily as diagonal neighbors. For efficiency, the weights and the lists of relevant neighbors should be precomputed once and for all, since the class numbers are independent of the A values.

A serial implementation of this method appears in [13]. The WEB program in that paper considers also a generalization in which white pixels orthogonally adjacent to black pixels are assumed to contribute some gray value to the total darkness; this tends to model the "dot gain" characteristics of certain output devices.

5. ERROR BOUNDS

Let us say that position (i, j) is a "baron" if it has only low-class neighbors. Barons are undesirable in the diffusion algorithm because they absorb all of the local error. In fact, "near-baron" positions, which have only one high-class neighbor, are also comparatively undesirable because they direct all the error to one place. The class structure of the matrix suggested above has only two barons (62 and 63), and only two near-barons (60 and 61). In contrast, the class structure of the matrix for ordered dither would be much less successful for diffusion, since it has 16 barons (48 to 63).

In most applications, the average error per pixel in the dot diffusion method will be at most the number of barons divided by twice the number of classes, if we average over a region that contains one pixel in each class. For example, we expect to absorb at most $\frac{2}{128}$ units of intensity per pixel in any 8×8 region if we use the matrix above, since the error committed at each pixel is compensated elsewhere except at the two baron positions, where we usually make an error of at most $\frac{1}{2}$.

However, it is possible to construct bad examples in which the entries of the matrix became negative or greater than 1; hence the maximum error does not simply depend on the number of barons. The worst case can be constructed as follows:

```
for $k := 0$ to 63 do bound[$k$] := 0.0;
for $k := 0$ to 63 do
  begin let $(i, j)$ be such that $k = \text{class}(i, j)$;
  for all neighbors $(u, v)$ of $(i, j)$ do if $\text{class}(u, v) > k$ then
    bound[class($u$, $v$)] := bound[class($u$, $v$)] + $\alpha_{uvij} * \max(0.5, \text{bound}[k])$;
  end.
```

Here α_{uvij} is the error diffusion constant from (i, j) to (u, v), as defined above. It follows that bound$[k]$ is the maximum error that can be passed to positions of class k from positions of lower classes. The maximum total error in a region containing one position of each class is the sum of bound$[k]$ over all baron classes k. Equivalently, it is the sum of $\max(0, 0.5 - \text{bound}[k])$ over all classes k. In the matrix above, we have bound[62] = bound[63] ≈ 4.3365; hence the average error per pixel is always less than $8.674/64 < 0.136$.

The original data must be chosen by a nasty adversary if the error is going to be this bad. On the other hand, an adversary who wants to defeat the ordered dither algorithm can make it commit errors of up to half a pixel on the average. (When (i, j) is of class k, let $A[i, j]$ be just less than $(k + 0.5)/64$; then the B's are entirely zero, but the A's have average density ≈ 0.5.)

The Floyd–Steinberg method has zero error by this criterion because all of its errors occur at the boundary, which has negligible area.

The following matrix has only one baron and one near-baron:

25	21	13	39	47	57	53	45
48	32	29	43	55	63	61	56
40	30	35	51	59	62	60	52
36	14	22	26	46	54	58	44
16	6	10	18	38	42	50	24
8	0	2	7	15	31	34	20
4	1	3	11	23	33	28	12
17	9	5	19	27	49	41	37

Therefore it might be better for dot diffusion than the matrix considered above. However, the barons in this case all line up rectilinearly, and this leads to a more noticeable visual texture. It is well known [12] that human eyes are less prone to notice the dots of a halftone grid if the dot pattern is rotated 45° than if the dots are rectilinear. If all entries of A are approximately $\frac{1}{64}$, the first matrix produces two nonzero values in every 8×8 submatrix, whereas the second matrix produces only one; yet the second matrix produces a less satisfactory texture.

The former class matrix was, in fact, suggested by dot patterns that are commercially used in halftone grids. If we imagine starting with a completely white matrix, and if we successively blacken all positions of classes $0, 1, \ldots,$ we obtain 45° grids of black dots that gradually grow larger and larger. When all classes <32 have been blackened, we have a checkerboard; from this point on, the blackening process essentially yields 45° grids of *white* dots that gradually grow smaller and smaller. Since the class number of (i, j) plus the class number of $(i, j + 4)$ is always equal to 63, the grid pattern of $63 - k$ white dots after k steps is exactly the same as the grid pattern of $63 - k$ black dots after $63 - k$ steps, shifted right 4. This connection of dot patterns to the diffusion pattern makes it reasonable to call the new method *dot diffusion*.

Dot diffusion can also be tried on a smaller scale, with the 4×4 class matrix

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

It can also be used with dots aligned at different angles, using patterns like those of Holladay [9], or with dots that are elliptical instead of circular.

6. ENHANCING THE EDGES

Jarvis and Roberts [11] and Jarvis et al. [10] discovered that ordered dither can be improved substantially if the edges of the original image are emphasized.

Their idea, in essence, is to replace $A[i,j]$ by

$$A'[i,j] = \frac{A[i,j] - \alpha\,\bar{A}[i,j]}{1-\alpha},$$

where $\bar{A}[i,j]$ is the average value of $A[i,j]$ and its eight neighbors:

$$\bar{A}[i,j] = \frac{1}{9}\sum_{u=i-1}^{i+1}\sum_{v=j-1}^{j+1} A[u,v].$$

The new values $A'[i,j]$ have the same average intensities as the old, but when $\alpha > 0$ they increase the difference of $A[i,j]$ from its neighbors. If $\alpha = 0.9$, these formulas simplify to a well-known equation (see [18, eq. 12.4-3]):

$$A'[i,j] = 9A[i,j] - \sum_{0<(u-i)^2+(v-j)^2<3} A[u,v].$$

The sum here is over all eight neighbors of (i,j).

Jarvis and Roberts formulated this "constrained average" method in another way, which made it seem inherently tied to the ordered dither technique. However, the equations above make it clear that edge enhancement can be used with any halftoning method.

7. EXAMPLES

Figures 1–12 on pages 252–263 demonstrate what happens when the three methods discussed so far are applied to two different images, with and without edge enhancement. Each image has 360 lines and 250 pixels per line.

Data for the first image were obtained from a high-quality television image of Leonardo da Vinci's most famous painting [15] in which each pixel was represented by a 1-byte value. Thus the data entries $A[i,j]$ all have the form $(r+0.5)/256$, where r is an integer, $0 \le r \le 255$.

The second image was computed artificially by formulas: Given i and j, let $x = (i-120)/111.5$ and $y = (j-120)/111.5$. If $x^2+y^2<1$, the value of $A[i,j]$ is $[(9+x-4y-8(1-x^2-y^2)^{1/2}]/18$; otherwise, $A[i,j]$ is simply $(1500i+j^2)/1000000$.

The edge-enhanced data $A'[i,j]$ were obtained from the original data by applying the simple formula above for the case $\alpha = 0.9$. Notice that this is amazingly effective in bringing out details of the Mona Lisa image; one can almost see the arches of the distant aqueduct in the background at the right of the picture. On the other hand, the transformation may have broadened Mona Lisa's mysterious smile.

When dot diffusion was applied to the Mona Lisa data, the magnitude of the error absorbed at baron positions was only 0.254 on the average. The values of $A[i,j]$ stayed between -0.63 and 1.67 throughout the processing. Only 47 of the 2835 baron positions absorbed an error whose magnitude was $\frac{1}{2}$ or more.

The average error committed by the ordered dither method was only slightly more than that of dot diffusion, in the case of Mona Lisa: The absolute difference between the A values and the B values, summed over all 8×8 blocks, turned out to be 0.596 on the average. However, the difference did exceed 1.0 in 260 of the 1440 blocks.

These illustrations have been generated with large square pixels so that the reader can plainly discern the on/off patterns. We would have to reduce them by a factor of 15 to get the equivalent of a medium-quality commercial screen for photographic halftones.

8. PROBLEMS

The ordered dither method produces a binary-recursive, "computery" texture that is unsuitable for most applications to publishing; such "cold" patterns are probably useful only when the underlying technology is intentionally being emphasized. The Floyd–Steinberg method usually gives much more pleasing results, but it too has occasional lapses where intrusive snakelike patterns call attention to themselves. The dot diffusion method, likewise, introduces a grainy texture of its own. Thus none of these approaches is wholly satisfactory, in the sense that a viewer presented with the illustrations at this size would instinctively find them attractive. We must hang them on a wall, stand back about 15 feet, and squint, before a continuous-tone effect can be perceived. (When such experiments are conducted, the Floyd–Steinberg examples tend to look better than the others.)

But the picture changes when we consider applications to printing. The author experimented with variations of these images on a conventional 300-pixel-per-inch laser printer (roughly a sixfold reduction from the size of the illustrations), and the results of Floyd–Steinberg and ordered dither proved to be quite unsatisfactory. Nonlinear effects of the xerographic process caused large dark blotches to appear in places where white pixels were fairly rare; there was a sharp jump between gray and black areas. In the author's experiments the best laser-printed Mona Lisa was produced by dot diffusion (see [13]); all other methods tried were clearly inferior.

Of course, laser printers are only a crude approximation to the high-resolution devices used in quality printing. Modern digital phototypesetters, with pixel sizes of about 25 micrometers (1000 pixels per inch), can produce excellent halftones by simulating the analog screening method that was used on older equipment. Indeed, the method of ordered dither—but with the 8×8 dot diffusion matrix in place of the 8×8 binary-recursive matrix shown earlier—is essentially a simulation of the traditional approach to halftones.

It is natural to suppose that we should be able to do an even better job than before, if only we could think of how to use the new machines in a cleverer way, because so many things are now possible that could not be done before. One might hope, for example, that the Floyd–Steinberg method (with sufficiently high resolution) might be able to reproduce Ansel Adam's photographs better than any previous method of printing has been able to achieve!

However, a moment's reflection makes it clear that the Floyd–Steinberg approach will be of no use at high resolution because of physical limitations. Tiny droplets of ink are simply unable to arrange themselves in patterns like those of Figure 1. The worst case probably occurs when we consider the case in which $A[i, j] = \frac{1}{2}$ for all i and j; then the Floyd–Steinberg algorithm produces a checkerboard of alternating black and white squares, and a printing machine will

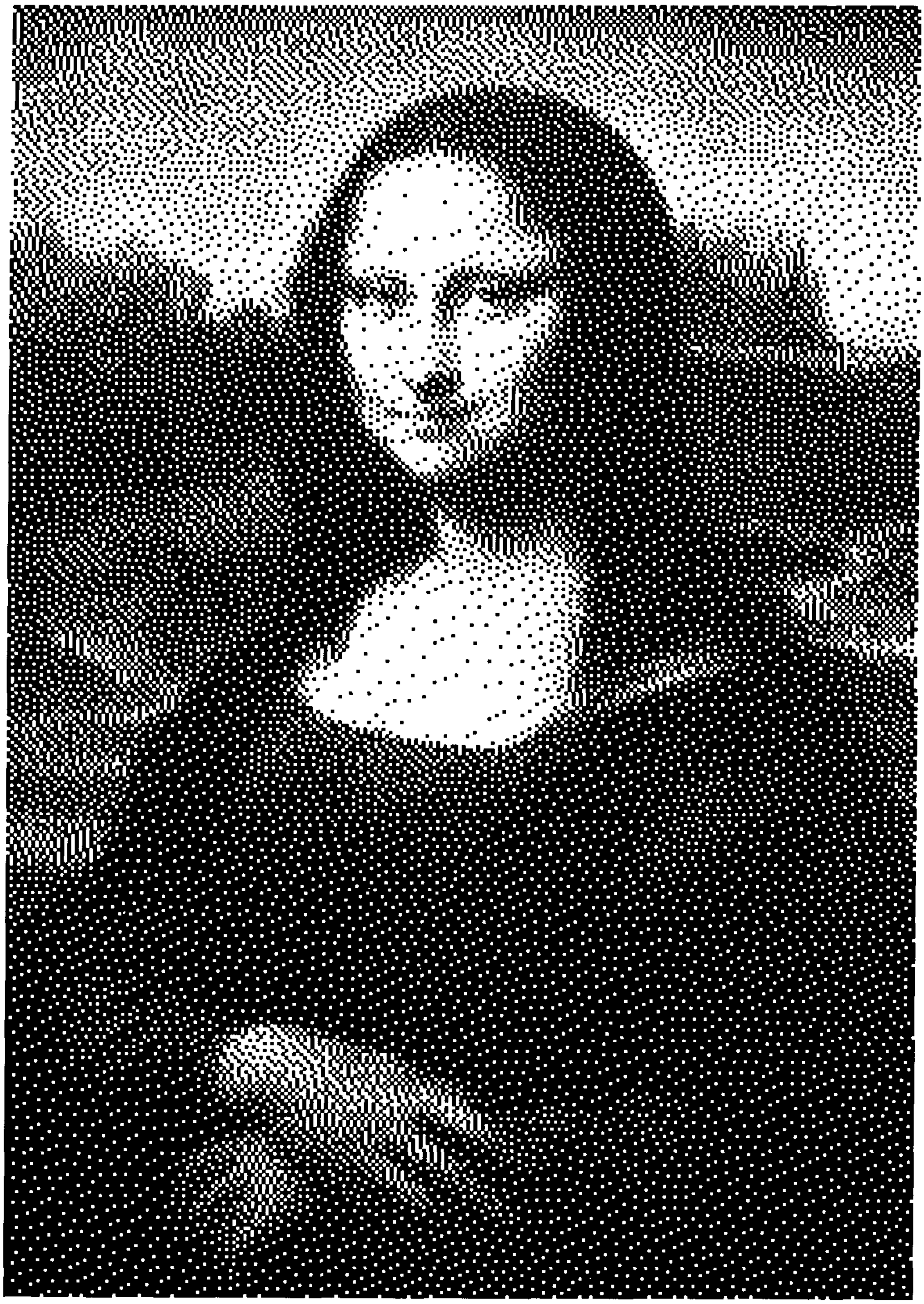


Fig. 1. Mona Lisa, digitized by the Floyd–Steinberg algorithm.

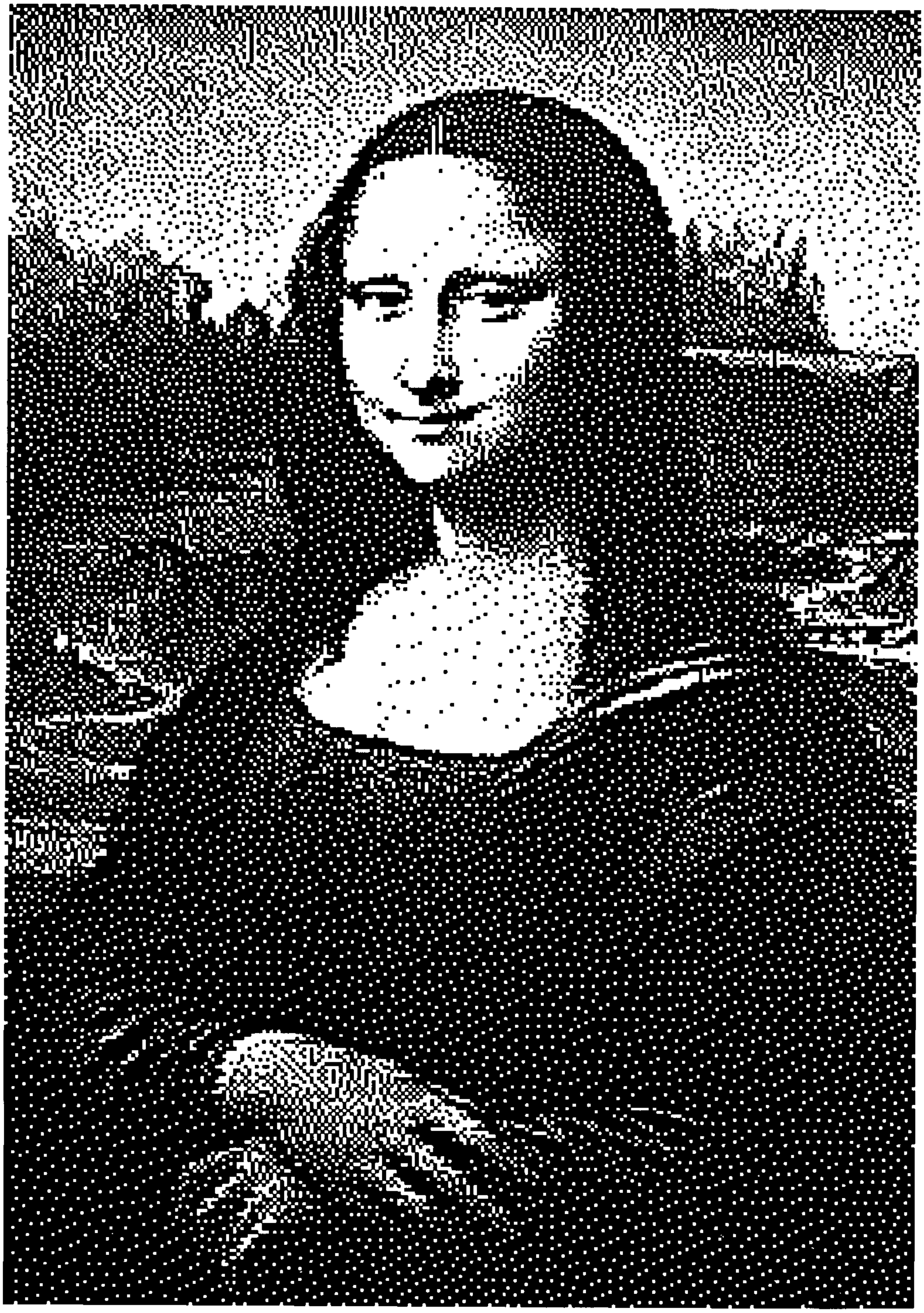


Fig. 2. Mona Lisa with enhanced edges, digitized by the Floyd–Steinberg algorithm.

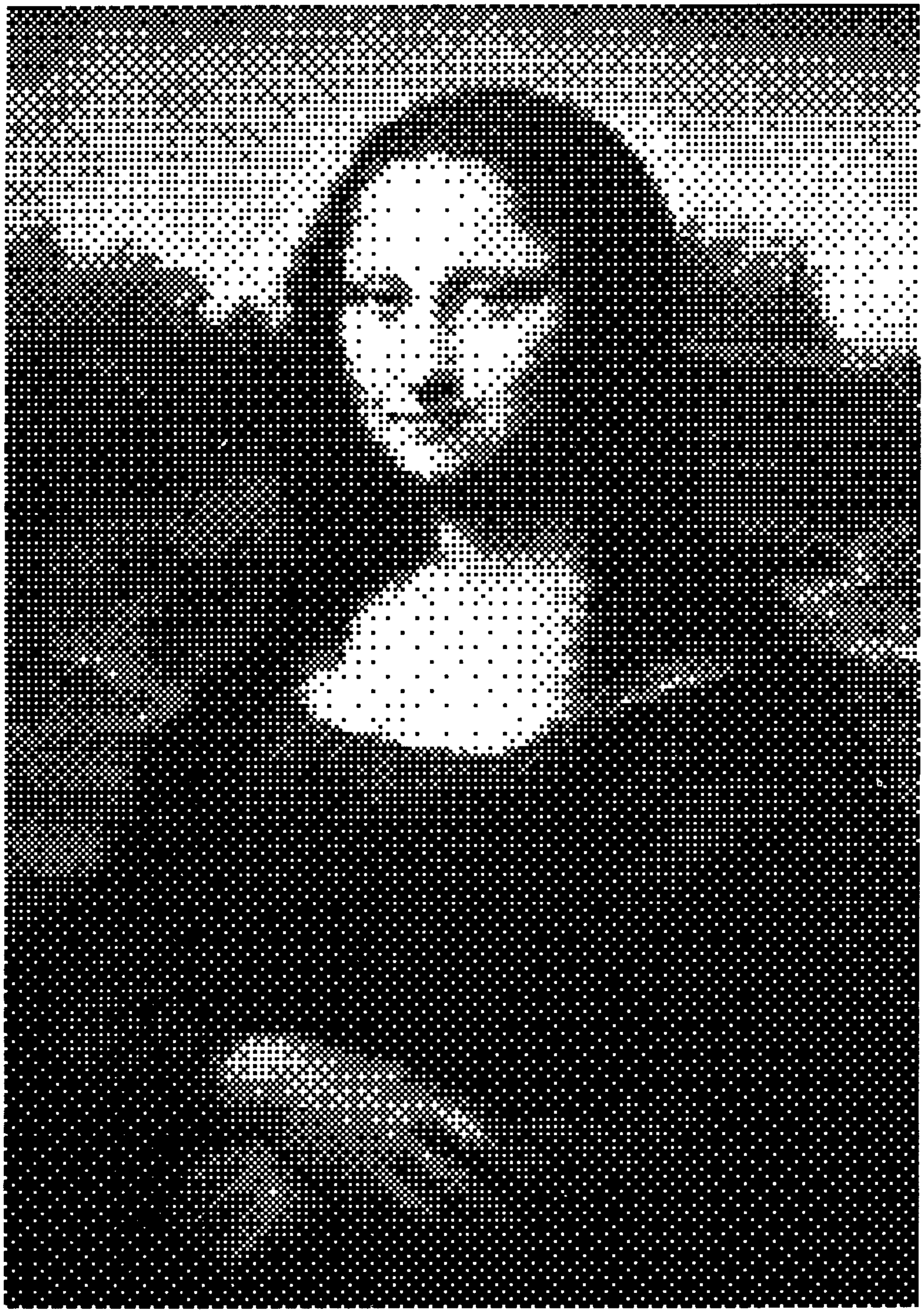


Fig. 3. Mona Lisa, digitized by the ordered dither algorithm.

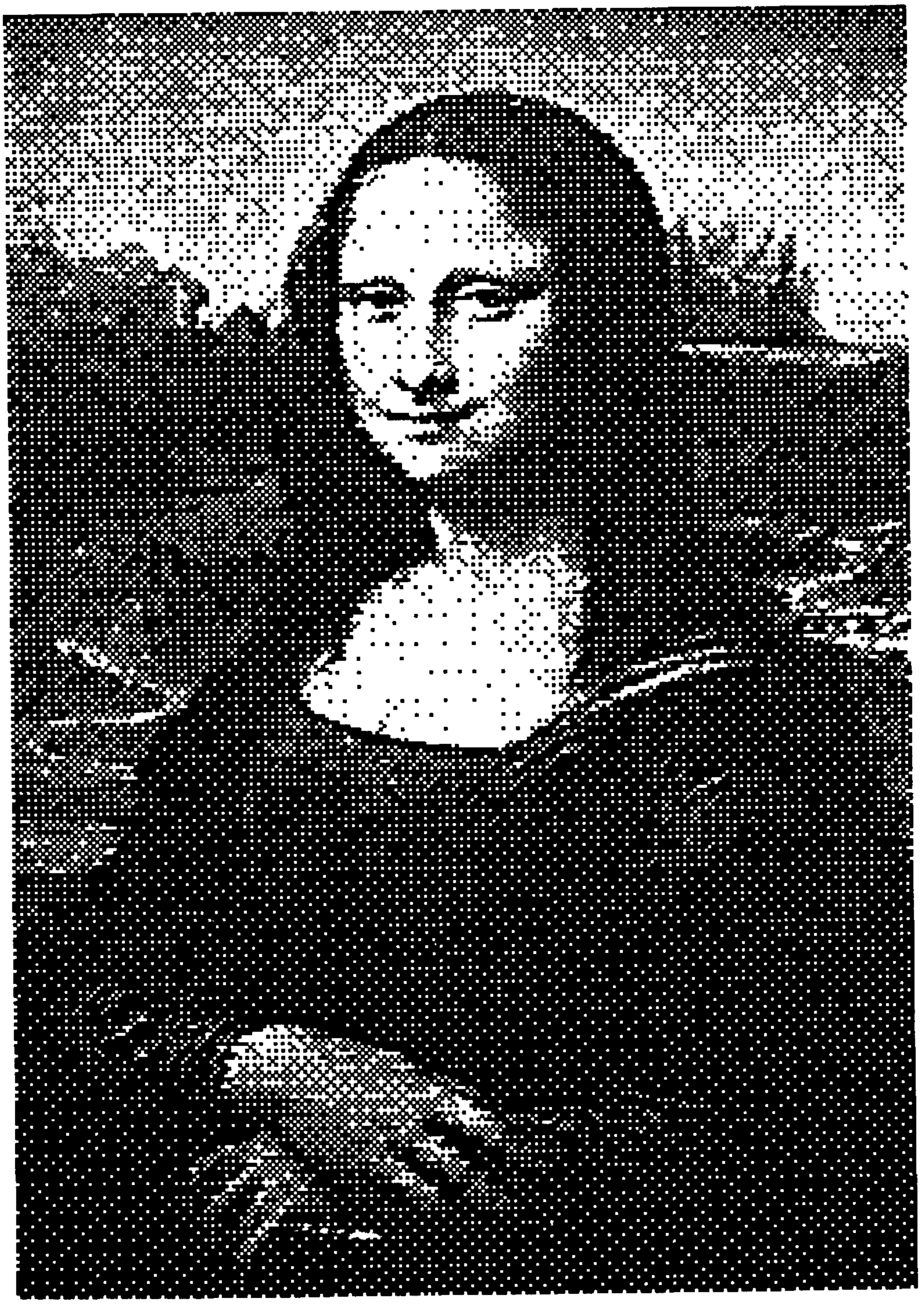


Fig. 4. Mona Lisa with enhanced edges, digitized by the ordered dither algorithm.

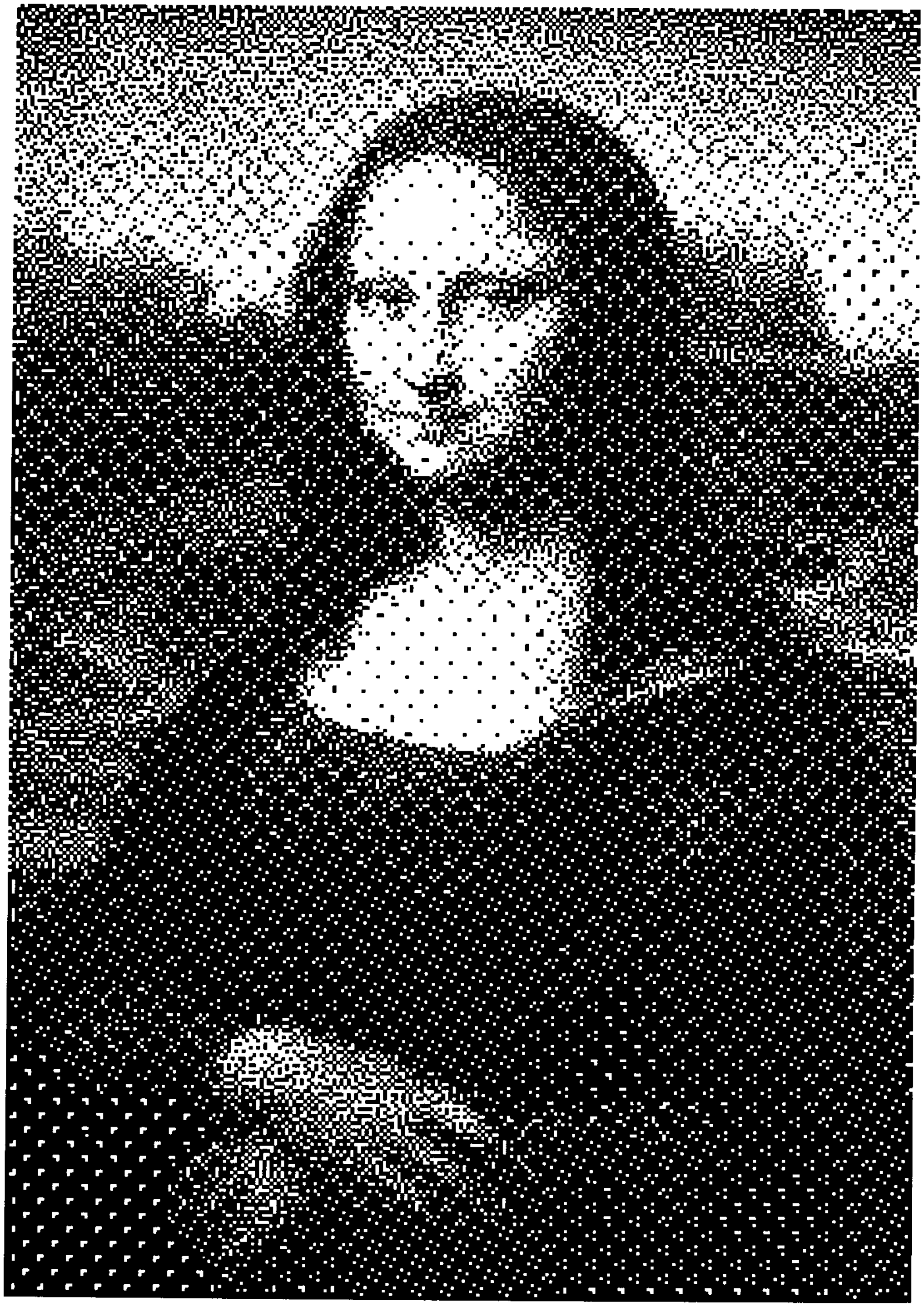


Fig. 5. Mona Lisa, digitized by the dot diffusion algorithm.

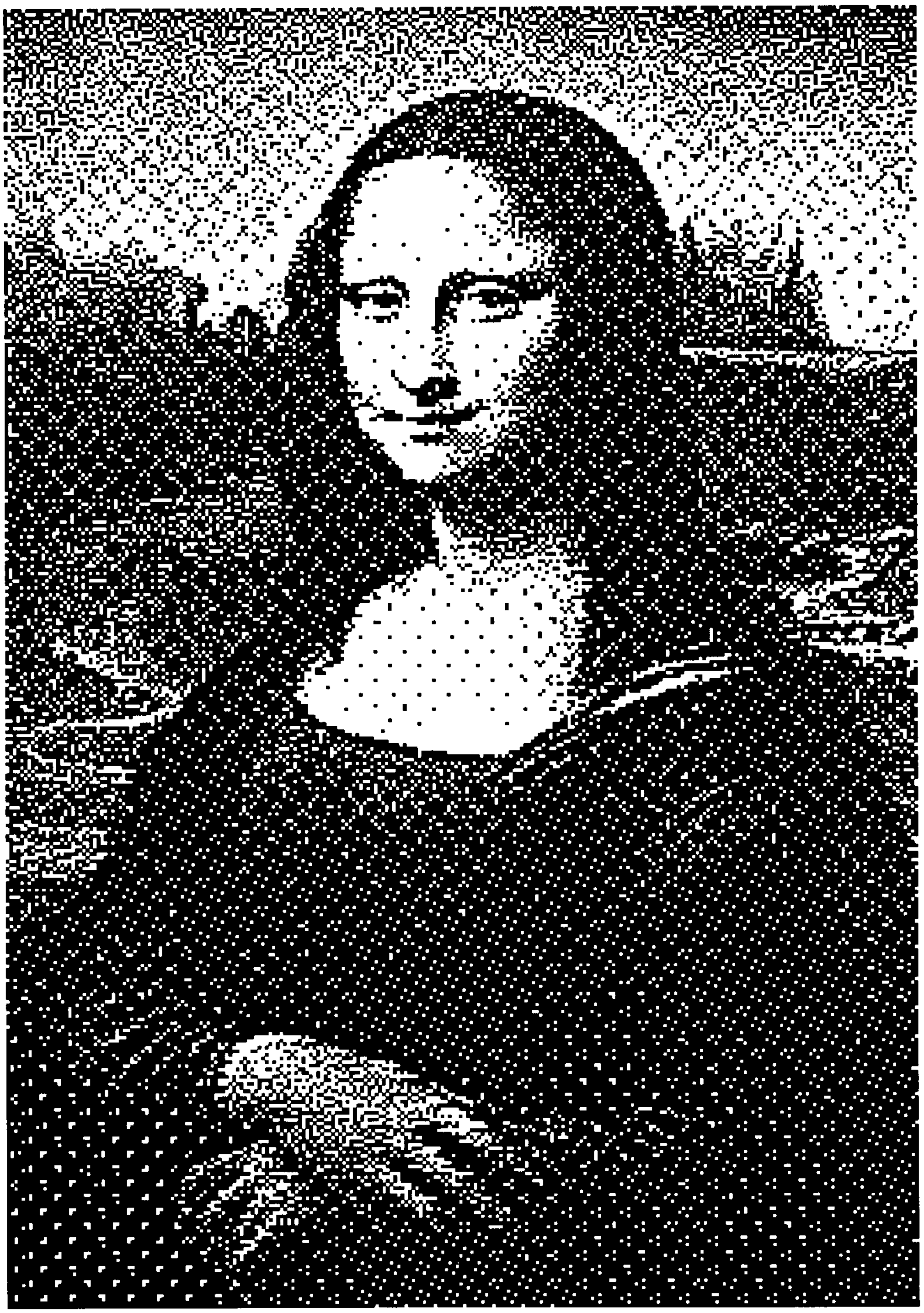


Fig. 6. Mona Lisa with enhanced edges, digitized by the dot diffusion algorithm.

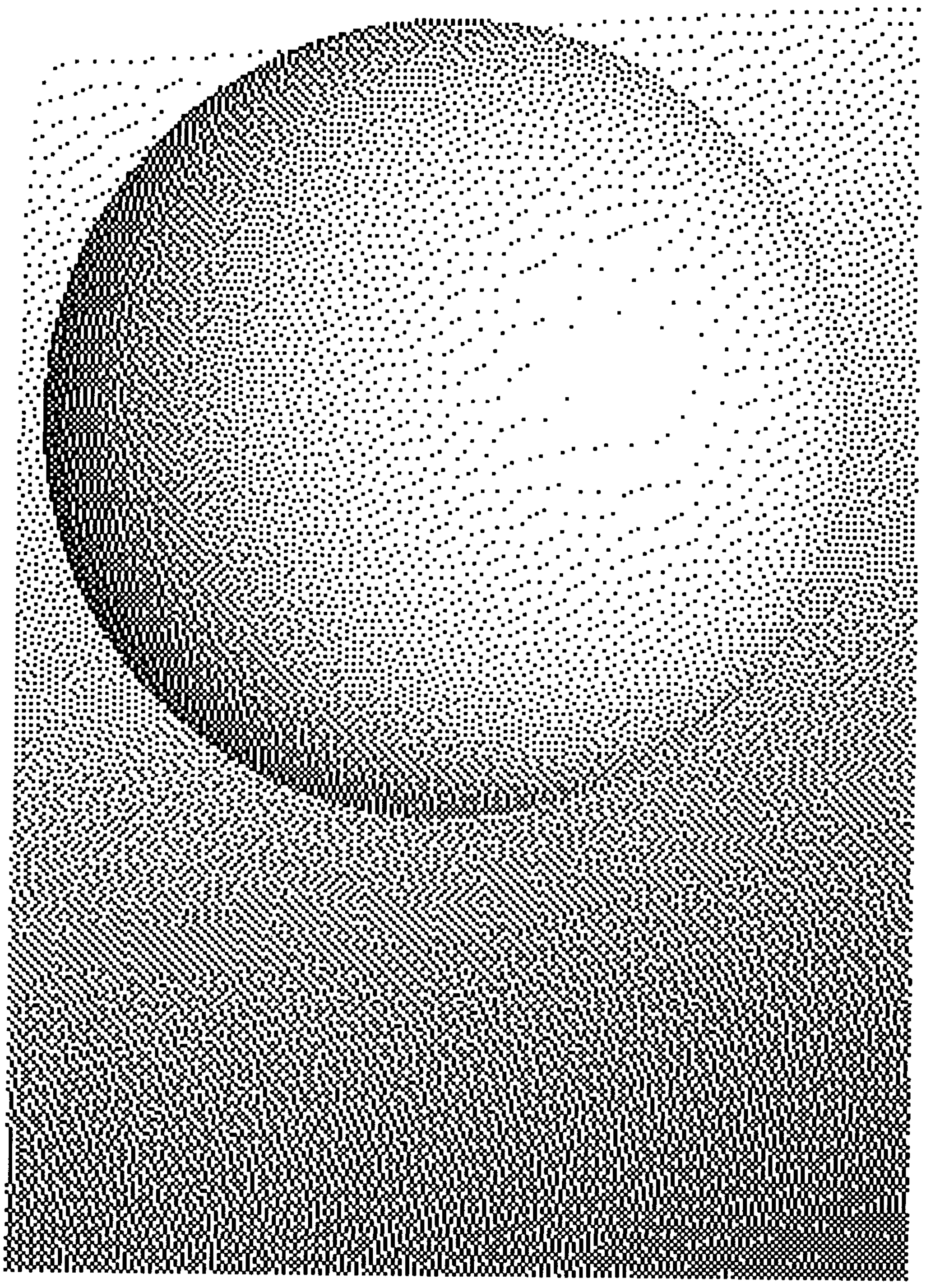


Fig. 7. Computed sphere, digitized by the Floyd–Steinberg algorithm.

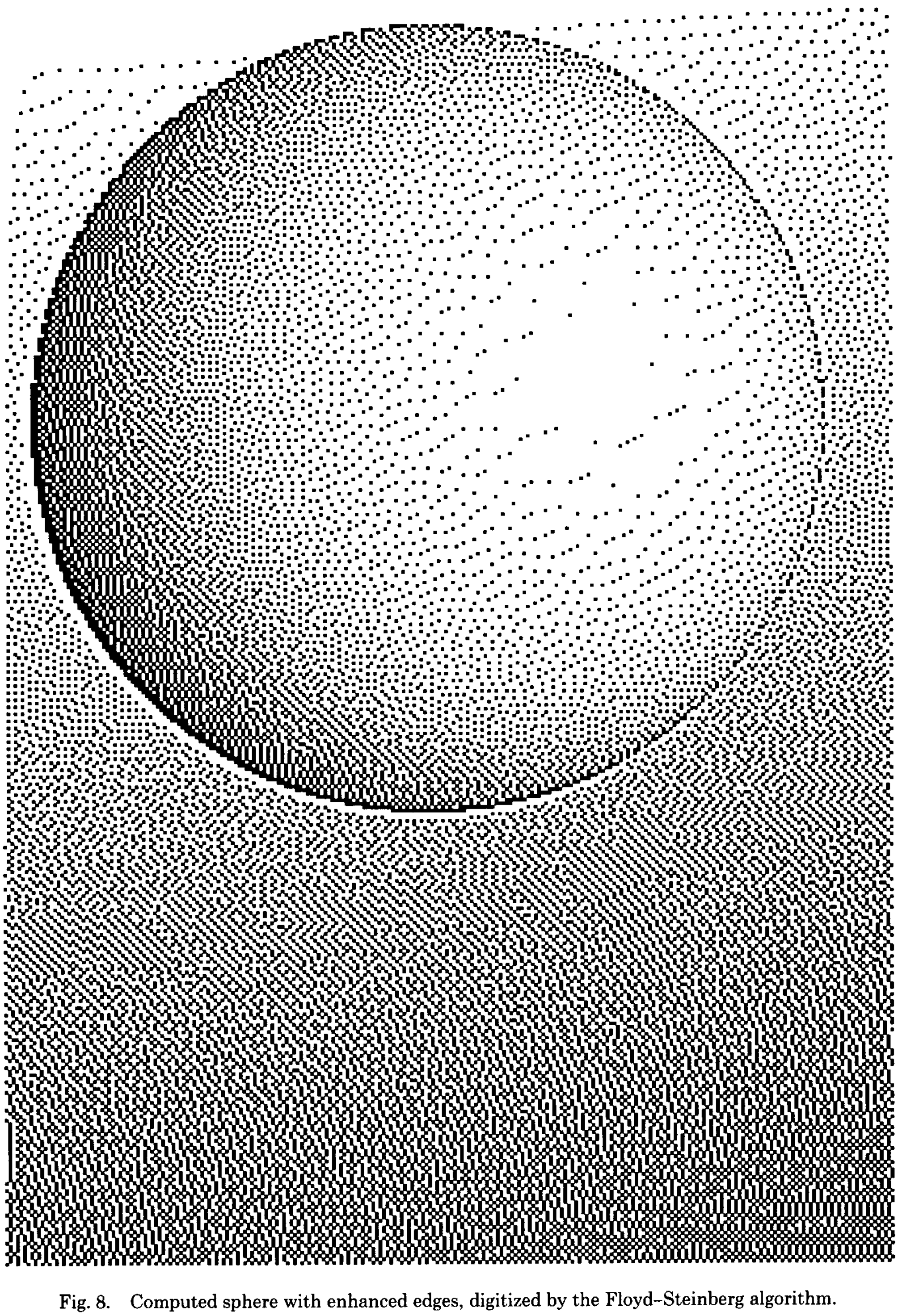


Fig. 8. Computed sphere with enhanced edges, digitized by the Floyd-Steinberg algorithm.

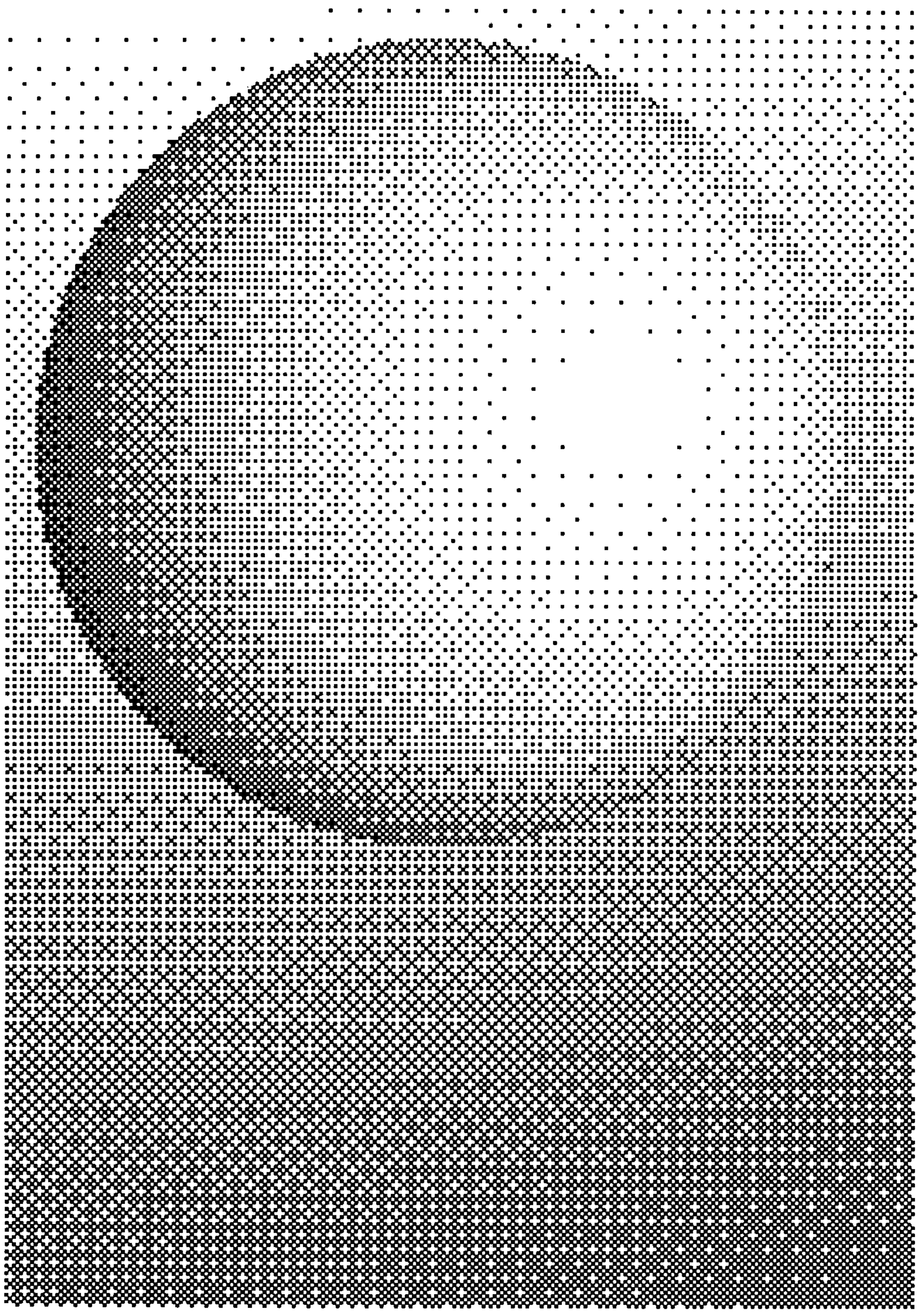


Fig. 9. Computed sphere, digitized by the ordered dither algorithm.

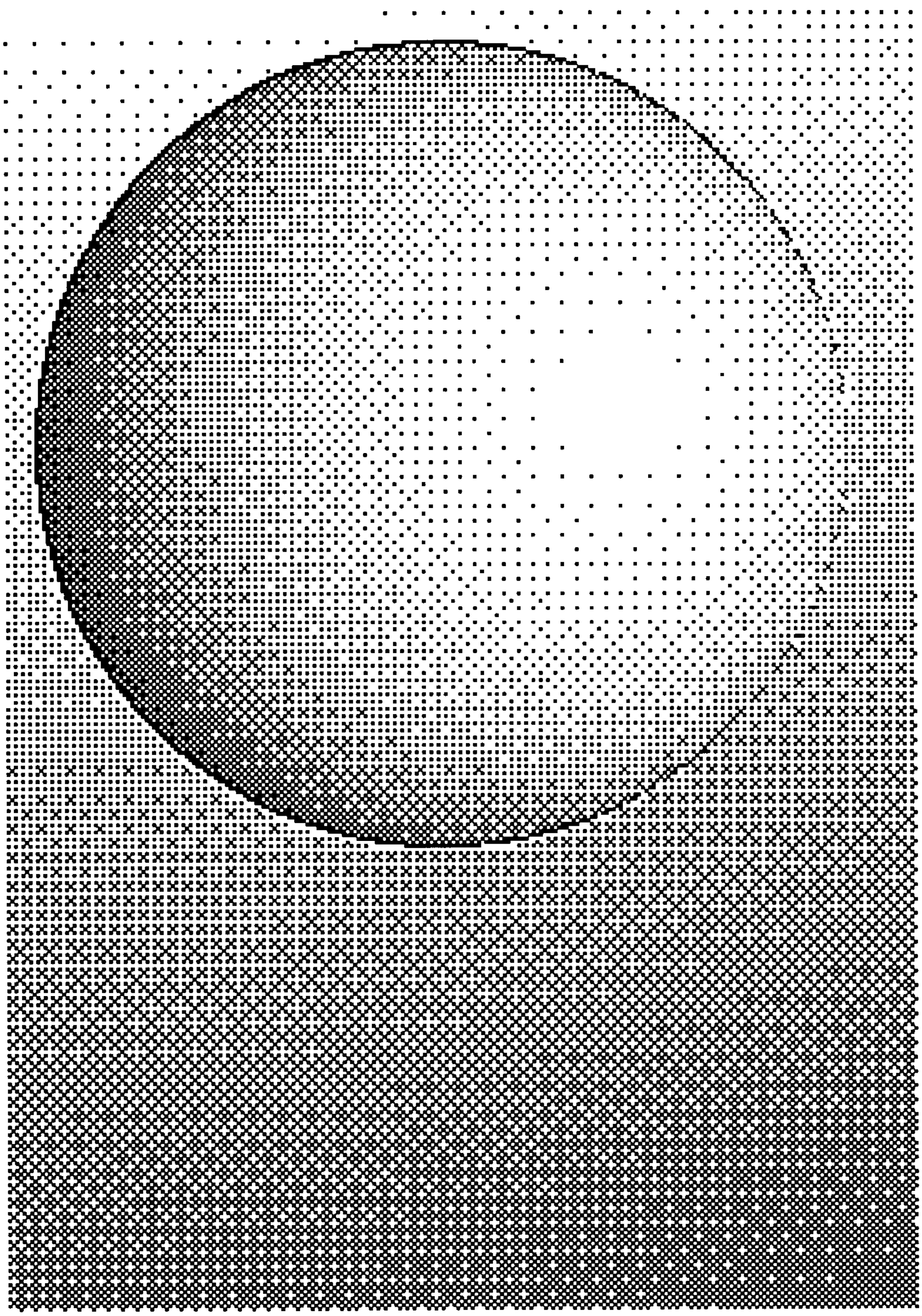


Fig. 10. Computed sphere with enhanced edges, digitized by the ordered dither algorithm.

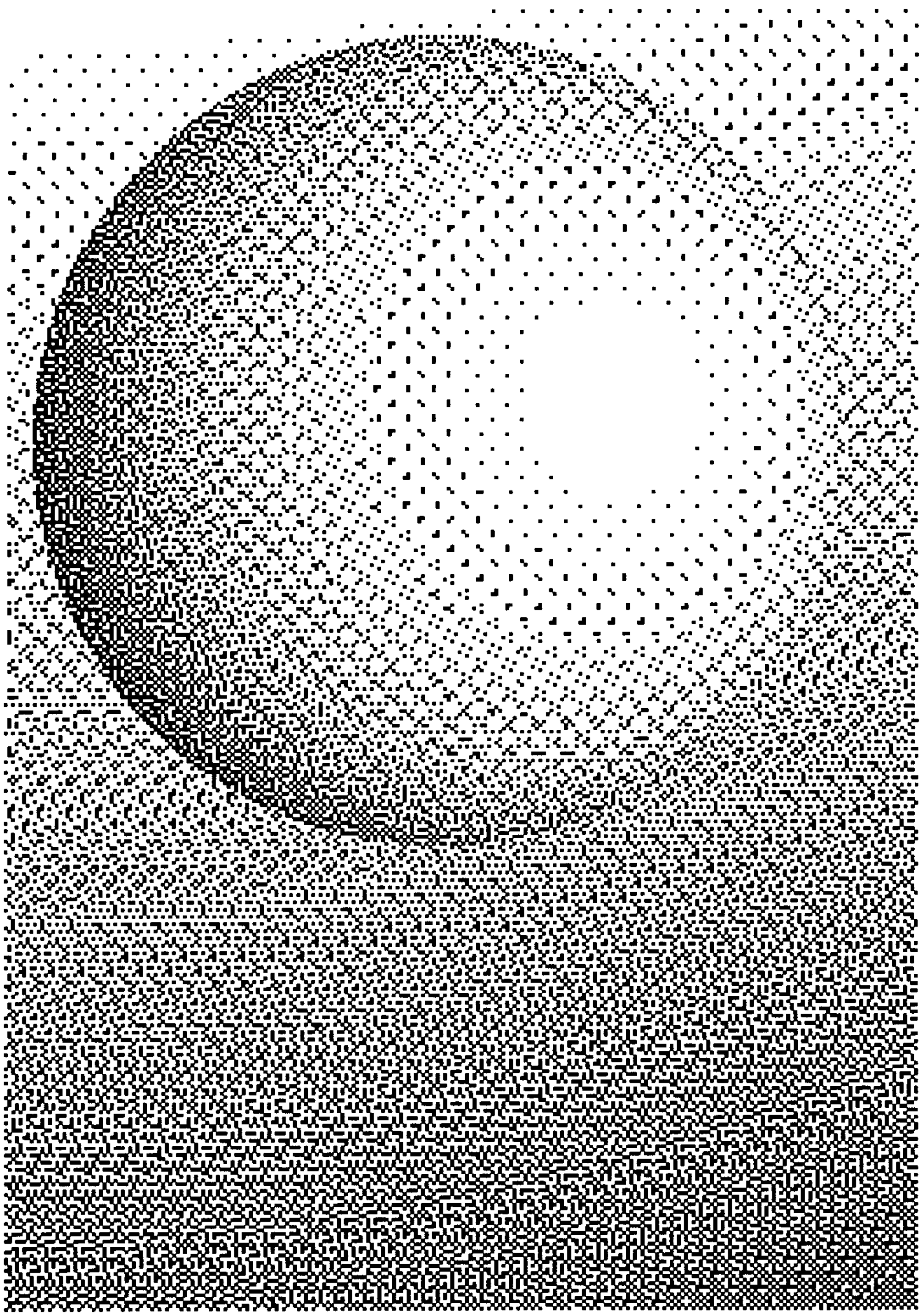


Fig. 11. Computed sphere, digitized by the dot diffusion algorithm.

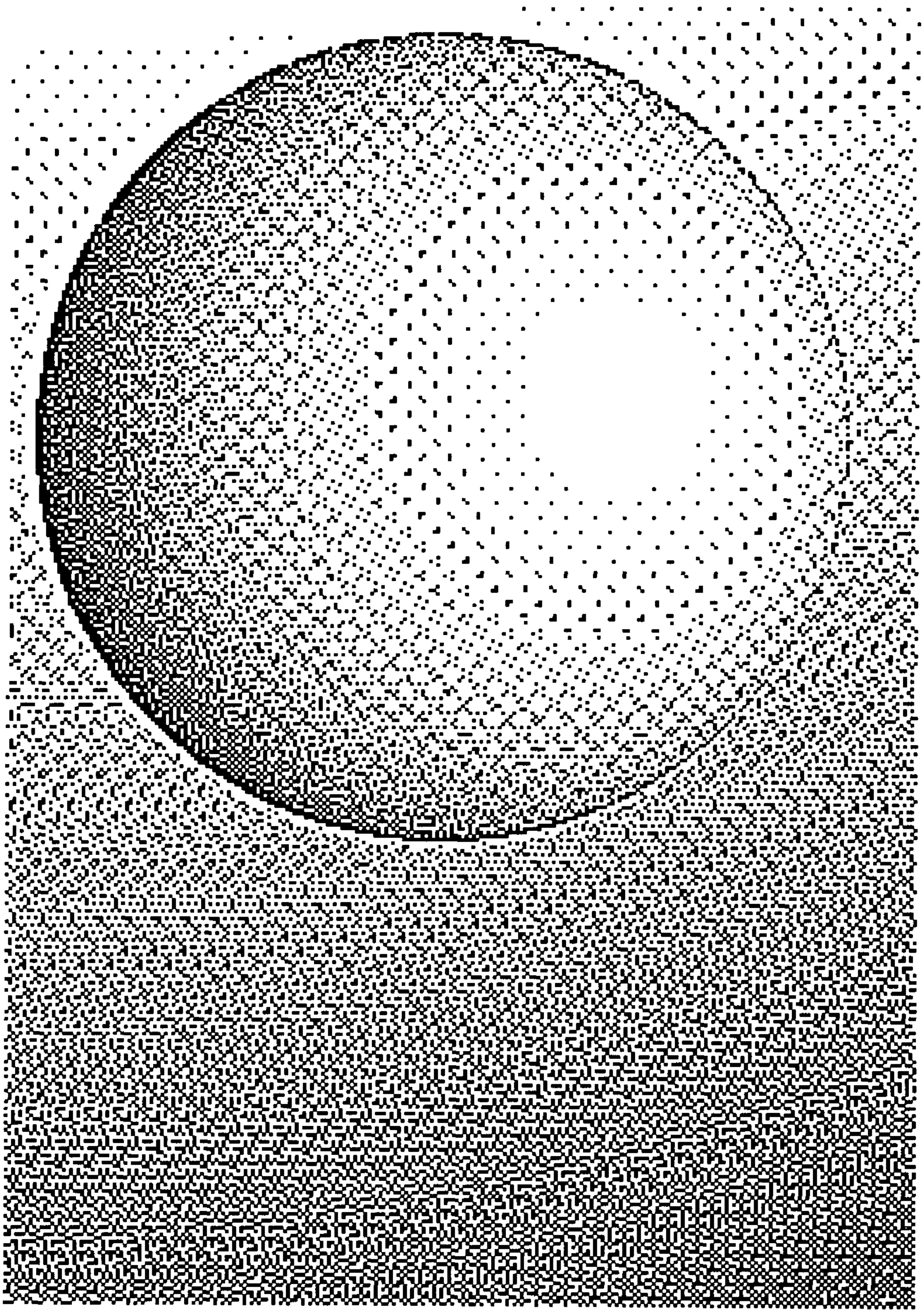
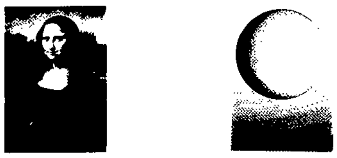


Fig. 12. Computed sphere with enhanced edges, digitized by the dot diffusion algorithm.

convert this into a horrible mess. Ordered dither, as in Figure 3, fails for the same reason. Both of these methods like to produce isolated black or white pixels.

Could dot diffusion, as described above, be useful at high resolution? Let us explore this by looking at a tenfold reduction of Figures 6 and 12:



In this reduced form there are 480 pixels per inch; if we rotate by 45° and divide by $4\sqrt{2}$ (which is the distance between "barons"), we see that the effect is analogous to an 85-line screen, which is roughly newspaper quality. These images are very small, and they would be even smaller on high-resolution devices; a medium-quality commercial screen has about 130 screen dots per inch, or about 720 unrotated bilevel pixels per inch. A real illustration of Mona Lisa would therefore have hundreds of times as much data; we must keep this "scale factor" in mind.

Mona Lisa does not look bad in this example, but there are serious deficiencies in the reproduction of the sphere. Our eyes will not notice a regular pattern of dots, if the dots are small enough, but we are quick to perceive changes in texture. The background tones of gray behind the sphere should be changing very gradually, but false contours show up because a slight change in intensity can make a large change in the pattern computed by dot diffusion. (Similar but less prominent false contours can be seen in the background of the Floyd–Steinberg output, Figures 7 and 8, especially where the intensities $A[i,j]$ are nearly $\frac{1}{2}$.)

Error diffusion methods are good at capturing the sharp details of a picture, but a successful method must also be "quiet" where the data show little activity. The preparer of a digital halftone must be willing to compose background music as well as the occasional fanfare. This example demonstrates that dot diffusion, as defined above, is unsuitable for general use at high resolution.

In fact, there is another good reason why dot diffusion breaks down. It can be shown that if $A[i, j]$ has the constant value $\frac{1}{2}$ for all (i, j), the dot diffusion algorithm defined above will produce a perfect checkerboard of alternating black and white. (Small checkerboard patches can be perceived in Figures 11 and 12.)

These observations lead us to conclude that our initial criterion, namely, that the average of $B[i, j]$ near (i_0, j_0) should be approximately $A[i_0, j_0]$, is not sufficient, in spite of its mathematical appeal. What we really want is a criterion that takes into account the distortions produced by a printing process, as well as the subsequent distortions and illusions produced by our optic nerves. In other words, human perception of $B[i,j]$ near (i_0,j_0), after printing, should be approximately the same as human perception of $A[i,j]$ near (i_0,j_0). Some steps in this direction have been taken by Allebach [2] and Dalton [6], who include a visual model in their experimental algorithms.

9. SMOOTH ERROR DIFFUSION

The discussion in the previous section seems to indicate that methods based on error diffusion are doomed, as far as applications to high-resolution printing are concerned. But we have not considered the full power of error diffusion. An important discovery was made by Billotet-Hoffmann and Bryngdahl [5], who realized that the Floyd–Steinberg method reproduces average gray levels even when the constant "$\frac{1}{2}$" is replaced by any other value! For example, if we set $B[i, j] := 1$ only when $A[i, j] > 0.6$, we will be setting $B[1, 1] := 0$ more often than before; but if we do, we will be distributing a larger error value; hence the neighboring pixels will be more likely to become 1. Billotet-Hoffman and Bryngdahl have found that if the thresholds vary slightly as a function of i and j, the resulting textures are improved.

Let us therefore consider a parallel algorithm of the following general form: All pixel positions (i, j) are divided into r classes, numbered 0 to $r - 1$, and we proceed as follows:

```
for $k := 0$ to $r - 1$ do
  for all $(i, j)$ of class $k$ do
    begin if $A[i, j] < \theta_k$ then $B[i, j] := 0$ else $B[i, j] := 1$;
    $err := A[i, j] - B[i, j]$;
    for $l := k + 1$ to $r - 1$ do
      begin let $(u, v)$ be nearest to $(i, j)$ such that class$(u, v) = l$;
      $A[u, v] := A[u, v] + err * \alpha_{kl}$;
      end;
    end.
```

A diffusion algorithm that will be useful at high resolution must have some sort of smoothness property. This means, intuitively, that small changes to the given pixel values $A[i, j]$ should produce small changes in the resulting binary values $B[i, j]$. For example, if all the A's increase, it would be nice if the B's all stay the same or increase. Let us therefore ask: Is there a sequence of parameter values θ_k and α_{kl}, for $0 \le k < l < r$, such that the general diffusion algorithm above has the following property?

If $A[i, j] = a$ initially, for all i and j, and if $(m - 0.5)/r < a < (m + 0.5)/r$ for some integer m, then the algorithm above should set

$$B[i, j] := \begin{cases} 1 & \text{if } \ 0 \le \text{class}(i, j) < m, \\ 0 & \text{if } \ m \le \text{class}(i, j) < r. \end{cases}$$

This condition states that the diffusion algorithm should act like a dither algorithm, when the data are constant.

Surprisingly, there is a simple solution to these nonlinear constraints. We may take

$$\theta_k = \frac{0.5}{r - k}, \qquad \alpha_{kl} = \frac{1}{r - k - 1}, \qquad \text{for all} \quad 0 \le k < l < r.$$

In particular, this threshold θ_k is strictly less than $\frac{1}{2}$, until we reach the baron class $k = r - 1$. For all smaller classes, the error is distributed equally to the higher class neighbors (i.e., it does not depend on l).

PROOF. Suppose that each $A[i, j]$ has the initial value $a = a_0$ and that $(m - 0.5)/r < a < (m + 0.5)/r$. If $m > 0$, the algorithm sets all $B[i, j]$ of class 0 to 1, and it sets all $A[i, j]$ of classes >0 to the value $a_1 = a_0 + (a_0 - 1)/(r - 1)$. Now we have $(m - 1.5)/(r - 1) < a_1 < (m - 0.5)/(r - 1)$. If $m > 1$, the algorithm sets all $B[i, j]$ of class 1 to 1, and it sets all $A[i, j]$ of classes >1 to $a_2 = a_1 + (a_1 - 1)/(r - 2)$. Hence $(m - 2.5)/(r - 2) < a_2 < (m - 1.5)/(r - 2)$; the process continues until we come to class m, with $A[i, j] = a_m$ and $-0.5/(r - m) < a_m < 0.5/(r - m)$. The algorithm now sets all $B[i, j]$ of class m to 1, and it sets all $A[i, j]$ of higher classes to $a_{m+1} = a_m + a_m/(r - m - 1)$. At this point we have $-0.5/(r - m - 1) < a_{m+1} < 0.5/(r - m - 1)$, hence the pattern persists. Q.E.D.

Although these threshold values θ_k appear to be very unsymmetrical with respect to 0 and 1, the stated smoothness property is symmetrical. Therefore the method is not so biased toward $B[i, j] = 1$ as it may seem. But there is a small bias. It can be shown, for example, that if $r = 2$ and if the continuous A values for classes 0 and 1 are chosen independently and uniformly at random, then the resulting binary B values will be (00, 01, 10, 11) with the respective probabilities $(\frac{3}{32}, \frac{5}{32}, \frac{20}{32}, \frac{4}{32})$; hence the total expected binary weight is $(5 + 20 + 8)/32 = 33/32$, slightly more than 1.

We can apply this method in the case $r = 32$, using the same class matrix as before, but with all class numbers divided by 2 (discarding the remainder). Each (i, j) now has 32 neighbors (u, v) that form a diamondlike pattern, defined by

$$-3 + |v - j| \le u - i \le 4 - |v - j|;$$

these 32 neighbors (including (i, j) itself) contain one element from each class.

Let us call the resulting algorithm *smooth dot diffusion*. The previously described dot diffusion method requires no more arithmetic operations than the ordinary Floyd–Steinberg algorithm; namely, 256 additions and 256 multiplications are needed to process each 8×8 block. In contrast, the smooth dot diffusion algorithm needs only 62 divisions per 8×8 block, since it distributes errors equally; but it performs 992 additions, so it is slightly more expensive.

The maximum value of $A[i, j]$ during the smooth diffusion algorithm with r classes will occur when the $A[i, j]$ are as large as possible subject to the condition that $B[i, j]$ be set to 0 for all classes $<(r - 1)$; this allows the baron to grow to the upper limit, $1.5 - 0.5/r$. The minimum possible value of $A[i, j]$ is more difficult to describe. When $r = 32$, it occurs when we choose A values as small as possible so that $B = 1$ for classes 0 to 19, and $B = 0$ for classes 20 to 30. This extreme case will make the baron's value $-6(\frac{1}{12} + \frac{2}{13} + \frac{2}{14} + \cdots + \frac{2}{30} + \frac{2}{31} + \frac{1}{32}) \approx -11.776$. For general r, let k be minimal such that $1/(r - k) + \cdots + 1/(r - 1) \ge 1 - 0.5/r$; then $k \approx r(1 - 1/e)$, and the least value that any $A[i, j]$ can assume is

$$\frac{k-2}{2}\left(\frac{1}{r-k} + \frac{2}{r-k-1} + \frac{2}{r-k-2} + \cdots + \frac{2}{r-2} + \frac{2}{r-1} + \frac{1}{r}\right) \approx -\frac{r}{e}.$$

Since smooth dot diffusion deals with rather large neighborhoods, an error can move to positions somewhat far from its source. For example, there is a propagation path from class 10 to 12 to 14 to 22 to 24 to 28 to 43 to 49 in the example matrix (before the class numbers have been divided by 2); this moves upward through 13 rows! However, the error is multiplied by small constants, so it is considerably dampened by the time it reaches the end of its journey. The only significant effect of such long paths is that a sequential implementation like that of [13] requires a buffer of some 23 rows; ordinary dot diffusion needs only 13.

10. COMPARISON WITH OTHER METHODS

Examples of methods intended for high-resolution printing are shown in Figures 13–16. Smooth dot diffusion appears in Figures 14 and 16. The maximum and minimum values of $A[i, j]$ actually occurring in the Mona Lisa example were 0.93 and -3.52; the average error dissipated per baron was 0.496.

The problematic false contours have disappeared from the sphere example; and if we compare Figure 14 subjectively to Figure 6, we might agree that the portrait of Mona Lisa has also improved a bit. These examples should produce excellent digital halftones at high resolution.

Paul Roetling [20, 21] has developed a somewhat similar parallel algorithm for high-resolution halftones. His method, called ARIES (alias-reducing image-enhancing screener), is essentially a modification of a dithering scheme in which the threshold levels are adjusted for each dot. ARIES first forms the set of all values $A[i, j] - k/r$ that contribute to a single dot, where position (i, j) corresponds to level k in the dither matrix and r is the total number of pixels per dot; then ARIES sets $B[i, j] := 1$ in the m positions (i, j) that score highest by this criterion, where m is chosen to equal the average intensity of the dot.

Figures 13 and 15 show the results of ARIES that correspond to the examples of smooth dot diffusion in Figures 14 and 16. In this case 32-pixel dots were used, again on the basis of the class matrix of dot diffusion with all class numbers divided by 2. (If (i_0, j_0) is a pixel of class 0, the 32 pixels of a dot are its 32 neighbors as defined above, namely, $(i_0 + \delta, j_0 + \epsilon)$ where $-3 + |\delta| \le \epsilon \le 4 - |\delta|$.) Again we obtain images that should make excellent halftones, but they do not seem to be as crisp as the results of smooth dot diffusion.

A generalization of ARIES has been suggested by Algie [1], who proposes a rank function of the form

$$A[i, j] - \alpha k,$$

where α is a tunable parameter. The ARIES scheme is the special case $\alpha = 1/r$; another scheme, called "structured pels" by Pryor et al. [19], is the special case $\alpha = 0$. If we let $\alpha \to \infty$, we get methods in which each dot is chosen from a fixed repertoire of shapes. Excellent results from such schemes have been obtained by Robert L. Gard, who uses alternating patterns of half-dots [8]. Another related algorithm has been described by Anastassiou and Pennington [3].

Printers traditionally distinguish between "halftones" and "line art"; each is treated differently. But it should not be necessary to make this distinction when high-resolution digital typesetting equipment is used. For example, a photograph might well contain textual information (such as a picture of a sign). Why should that text have to be screened, when it could be made more legible? Methods like ARIES and smooth dot diffusion are able to adapt to whatever an image requires.

It is, of course, possible to construct input data for which the output of smooth dot diffusion will be unsuitable for printing. For example, if each of the $A[i, j]$

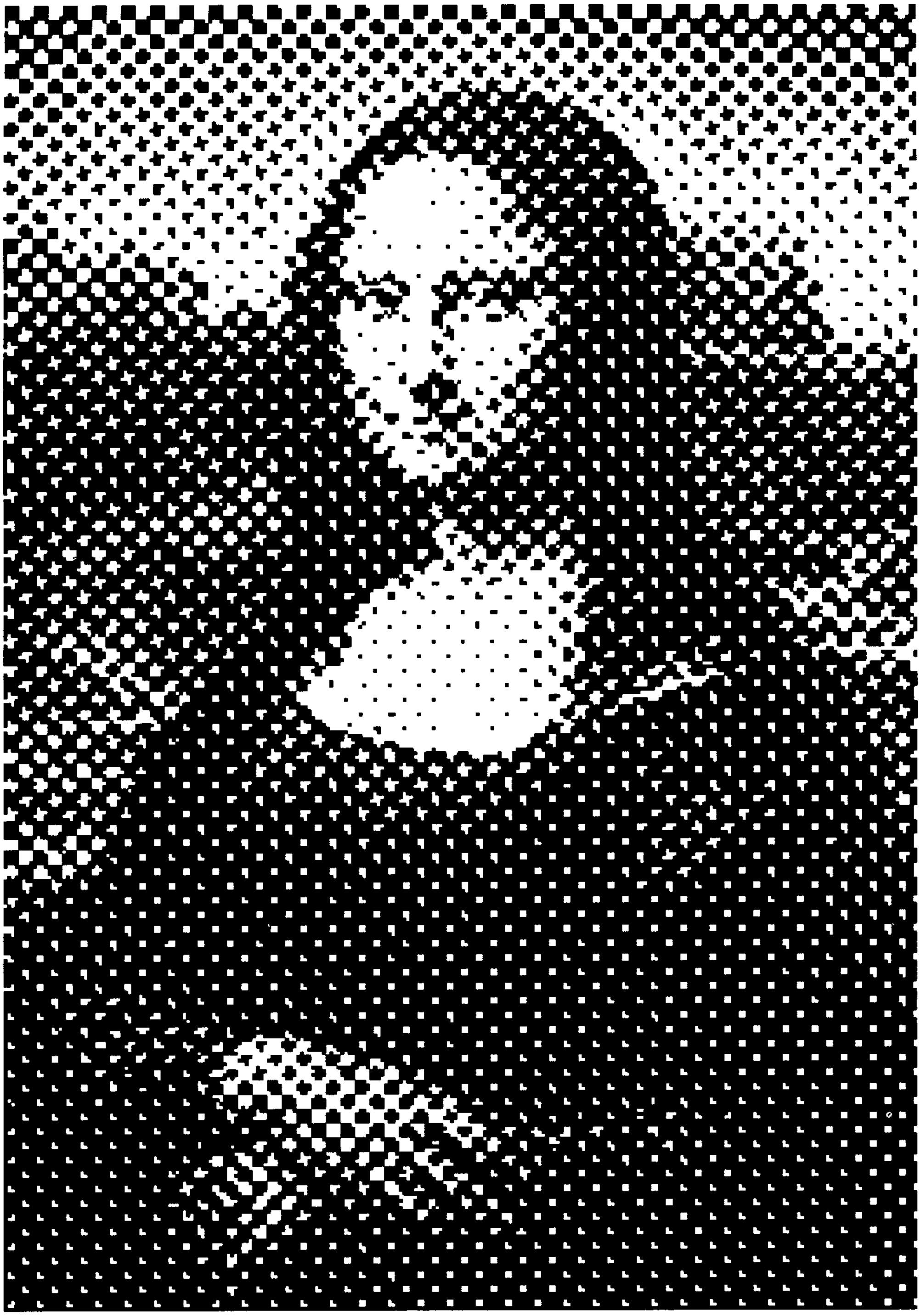


Fig. 13. Mona Lisa with enhanced edges, digitized by the ARIES algorithm.

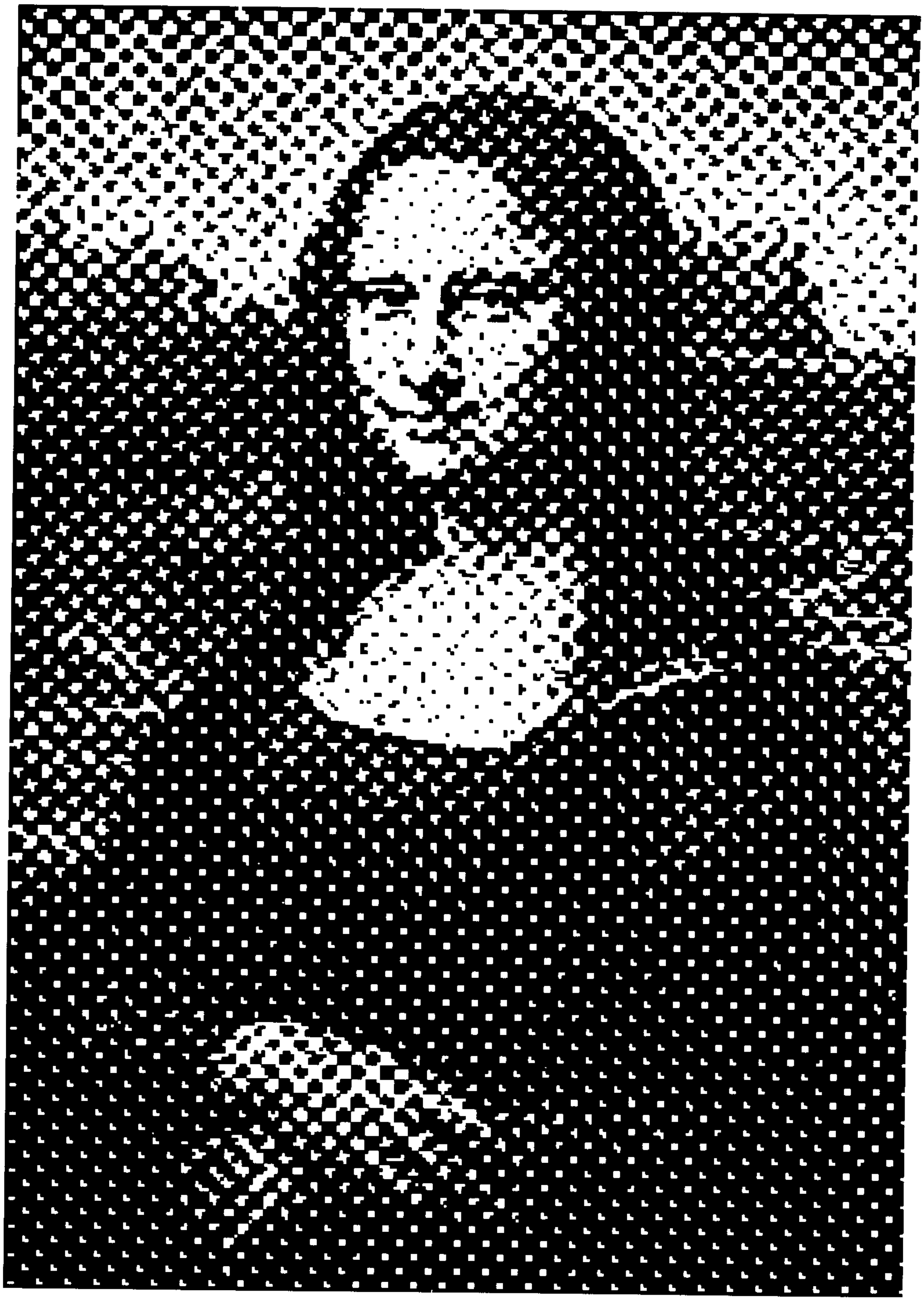


Fig. 14. Mona Lisa with enhanced edges, digitized by the smooth dot diffusion algorithm.

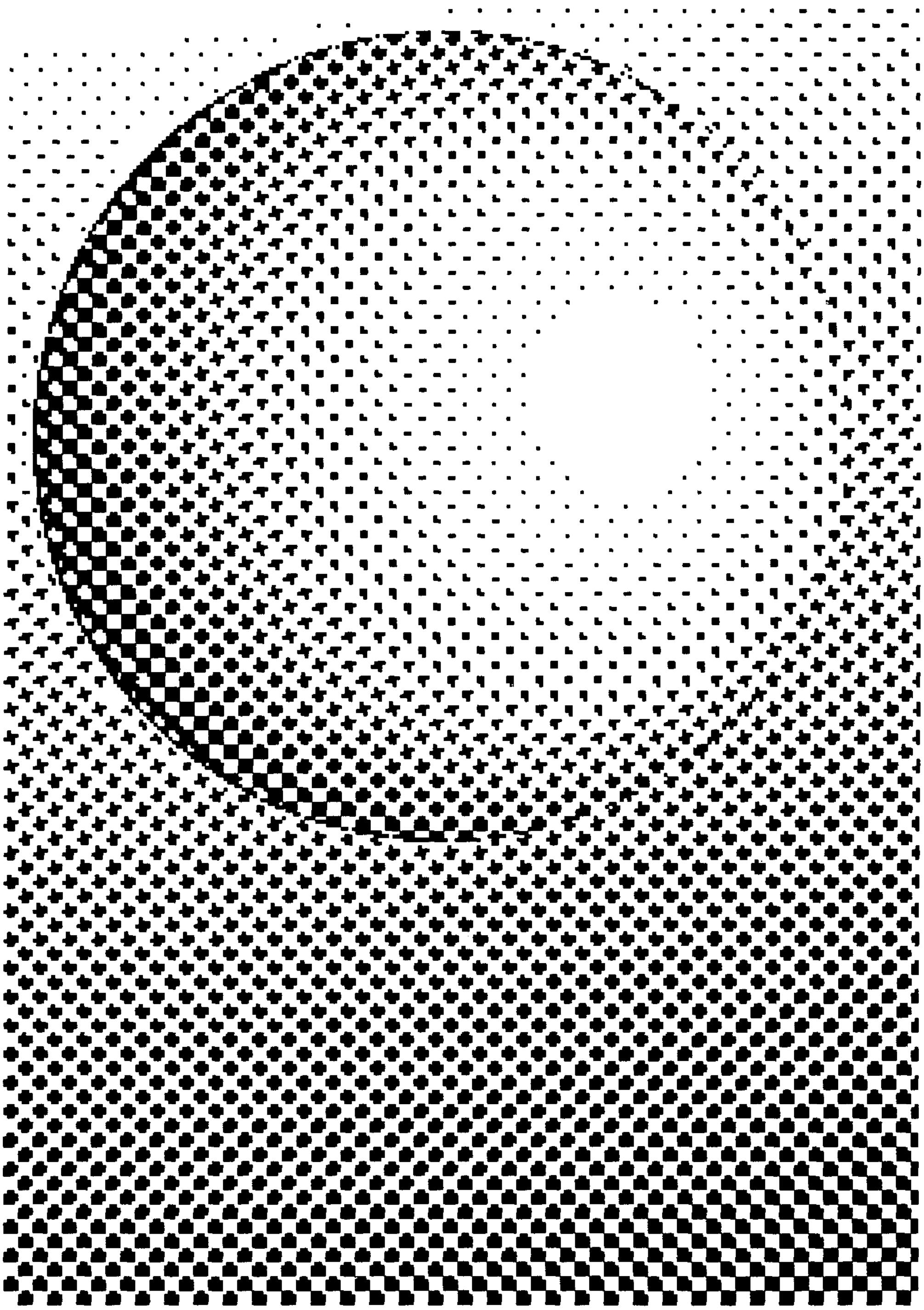


Fig. 15. Computed sphere with enhanced edges, digitized by the ARIES algorithm.

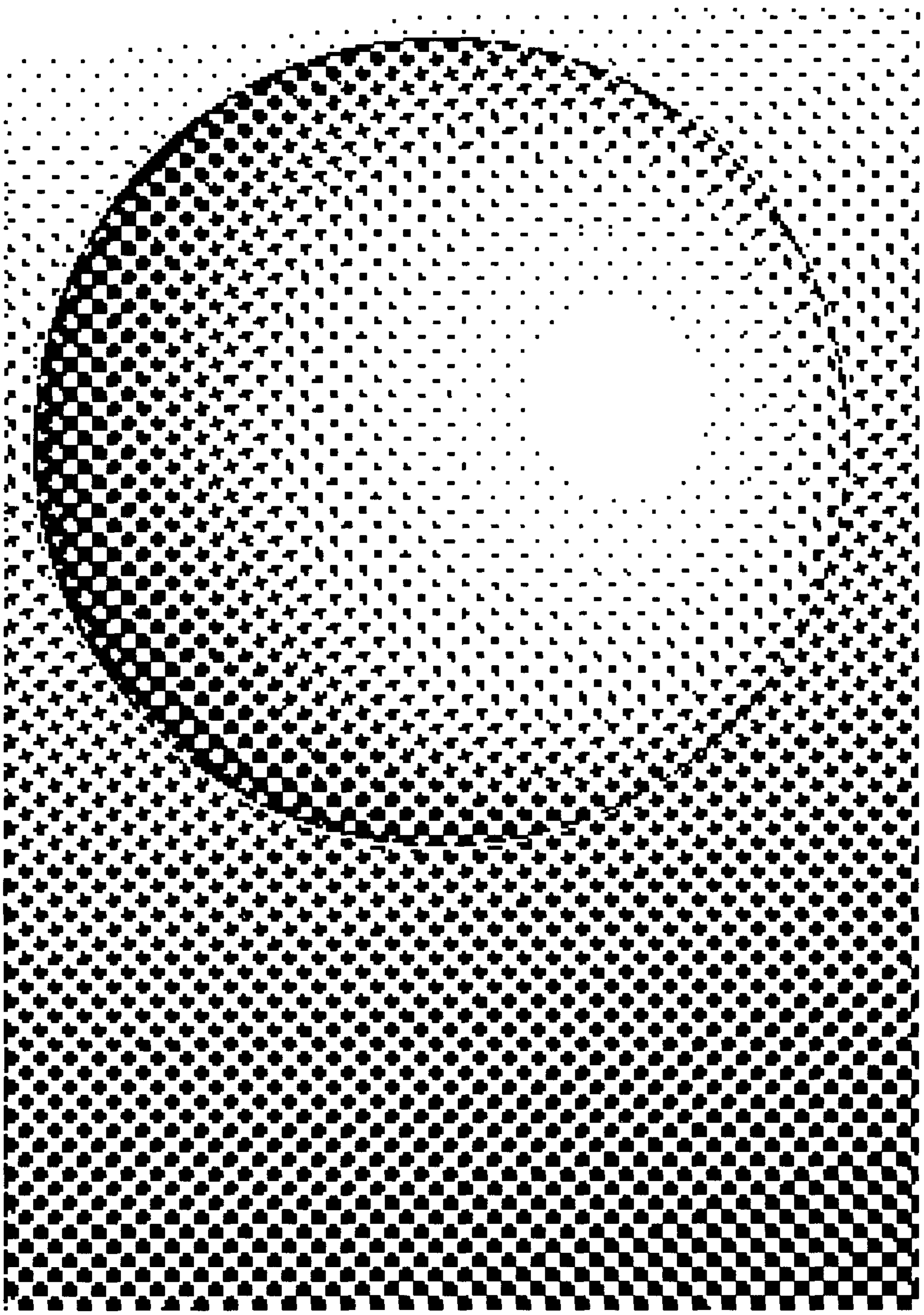


Fig. 16. Computed sphere with enhanced edges, digitized by the smooth dot diffusion algorithm.

values is already 0 or 1, smooth dot diffusion will simply set $B[i, j] := A[i, j]$ for all (i, j); we might be faced with values $A[i, j]$ that are unprintable, like a checkerboard. But we can assume that no such data will arise in practice; such noise can be filtered out before the digitization process begins.

At present, the author knows of no method that produces images of better quality for high-resolution digital phototypesetting than those produced by the smooth dot diffusion algorithm. However, it is obviously premature to make extravagant claims for this new method. Computational experience so far has been very limited.

ACKNOWLEDGMENTS

The referees were extremely helpful in pointing me to literature of which I was unaware; this led to substantial improvement of the methods and the exposition in this paper. I also wish to thank B. K. P. Horn for letting me study the results of his many experiments with computer-generated halftones, and Wally B. Mann for his help in capturing the Mona Lisa data.

REFERENCES

(Note: Reference [14] is not cited in text.)

1. ALGIE, S. H. Resolution and tonal continuity in bilevel printed picture quality. *Comput. Vision, Graph. Image Process. 24* (1983), 329–346.
2. ALLEBACH, J. P. Visual model-based algorithms for halftoning images. In *Proceedings of the SPIE (Society of Photo-Optical Instrumentation Engineering) 310, Image Quality* (1981), pp. 151–157.
3. ANASTASSIOU, D., AND PENNINGTON, K. S. Digital halftoning of images. *IBM J. Res. Dev. 26* (1982), 687–697.
4. BAYER, B. E. An optimum method for two-level rendition of continuous-tone pictures. *Conference Record, IEEE International Conference on Communications* (1973), vol. 1. IEEE, New York, pp. (26-11)–(26-15).
5. BILLOTET-HOFFMANN, C., AND BRYNGDAHL, O. On the error diffusion technique for electronic halftoning. *Proc. Soc. Inf. Display 24* (1983), 253–258.
6. DALTON, J. C. Visual model based image halftoning using Markov random field error diffusion. M.Sc. thesis, Dept. of Electrical Engineering, Univ. of Delaware, Newark, Del., Dec. 1983.
7. FLOYD, R. W., AND STEINBERG, L. An adaptive algorithm for spatial grey scale. *SID 75 Digest.* Society for Information Display, 1975, pp. 36–37.
8. GARD, R. L. Digital picture processing techniques for the publishing industry. *Comput. Graph. Image Process. 5* (1976), 151–171.
9. HOLLADAY, T. M. An optimum algorithm for halftone generation for displays and hard copies. *Proc. Soc. Inf. Display 21* (1980), 185–192.
10. JARVIS, J. F., JUDICE, C. N., AND NINKE, W. H. A survey of techniques for the display of continuous tone pictures on bilevel displays. *Comput. Graph. Image Process. 5* (1976), pp. 13–40.
11. JARVIS, J. F., AND ROBERTS, C. S. A new technique for displaying continuous tone images on a bilevel display. *IEEE Trans. Commun. COM-24* (1976), 891–898.
12. KLENSCH, R. J., MEYERHOFER, D., AND WALSH, J. J. Electronically generated halftone pictures. *RCA Rev. 31* (1970), 517–533.
13. KNUTH, D. E. Fonts for digital halftones. *TUGboat* 8 (1987), 135–160.
14. KNUTH, K. E., BERRY, J. M., AND OLLENDICK, G. B. An ink jet facsimile recorder. *IEEE Trans. Ind. Appl. IA-14* (1978), 156–161.
15. LEONARDO DA VINCI. La Gioconda. See, for example, *Leonardo da Vinci*, Reynal, New York, 1956, facing p. 144.
16. LIMB, J. O. Design of dither waveforms for quantized visual signals. *Bell Syst. Tech. J. 48*, 7 (1969), 2555–2582.

17. LIPPEL, B., AND KURLAND, M. The effect of dither on luminance quantization of pictures. *IEEE Trans. Commun. Tech. COM-19* (1971), 879-888.
18. PRATT, W. K. *Digital Image Processing.* Wiley, New York, 1978.
19. PRYOR, R. W., CINQUE, G. M., AND RUBENSTEIN, A. Bilevel image displays—A new approach. *Proc. Soc. Inf. Display 19* (1978), 127–131.
20. ROETLING, P. G. Halftone method with edge enhancement and Moiré suppression. *J. Opt. Soc. Am. 66* (1976), 985-989.
21. ROETLING, P. G. Binary approximation of continuous tone images. *Photograph. Sci. Eng. 21* (1977), 60-65.
22. SCHROEDER, M. R. Images from computers. *IEEE Spectrum 6*, 3 (Mar. 1969), 66–78. See also *Commun. ACM 12* (1969), 95-101.

Received January 1987; accepted May 1987