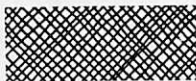



THEORY & PRACTICE



On the bathtub algorithm for dot-matrix holograms

By Brian Hayes

 It all began with a quick-and-dirty program to draw a contour map. I had an equation defining a surface—giving the elevation, z , for any combination of x and y coordinates—and I needed to see what the surface looked like. The best format for my immediate needs was that of a topographic map, showing contours of equal elevation.

The end of the story is that I got the map I needed. It was reasonably clear and accurate; it served its purpose; in short, it was an utter bore and I have nothing more to say about it. What I want to speak of here is an unexpected discovery (otherwise known as a mistake) made along the way. The illustrations that accompany this column, which I have taken to calling dot-matrix holograms, are the fruits of that discovery.

Robot surveyor

The usual method of drawing a topographic map calls for following individual contour lines. In effect, you build a robot equipped with a sensitive altimeter and instruct it to wander over the landscape while maintaining constant elevation. If the robot is plotting the contour at elevation $z = 10$, it first searches the terrain until it finds a point where the altimeter reads 10. Then it moves in whatever direction keeps the altimeter reading steady. Meanwhile, you are tracing the projection of the robot's motion onto a plane.

A program based on this procedure is likely to be messy. To follow a contour line, the robot must measure the elevation of various points near its current position until it finds a point at the right altitude. This seems straightforward until you begin to think about some of the pathological cases that could confuse the poor automaton. What happens if there is no point at the right elevation? (The robot might be at a peak, or it might be in the pit of a depression.) What happens if there are two points at the correct height? (The robot might be near a saddle point, where two contours at the same elevation approach closely but do not touch.) What happens if all the points are at the same

height? (The robot might be on a level plane.)

With care and ingenuity these problems can be solved. The proof is that contour-following programs do exist, and apparently they work well. Chris Johnston of the NASA Lewis Research Center described a particularly clever one in "Contour Plots of Large Data Sets" (*COMPUTER LANGUAGE*, May 1986, pp. 63-67). Nevertheless, writing such a program is not a job to be undertaken in haste.

I had another reason for avoiding the contour-tracing algorithm. The method is ideally suited to driving an x - y pen plotter—the points along a contour are generated in the order they will be drawn—but I did not have a pen plotter on hand. My only graphic output device was a dot-matrix printer, which expects to receive data row by row; the printer is a raster device, whereas a plotter is a vector device. Of course one could convert from vector to raster form, but this would add yet another level of complexity to the program. It seemed there must be an easier way.

Eureka

As Archimedes learned, when you come up against a problem that seems harder than it ought to be, the best course is to take a long soak in the tub. That's where I found a simpler approach to topographic mapping.

Suppose you had a solid model of the terrain to be mapped. You could put it in the bathtub, taking care to level it, and add half an inch of water every day or two. Minerals deposited at the water line would create a series of evenly spaced bathtub rings on the surface of the model. Photographing the rings from directly overhead would then yield a contour plot. All the tricky analysis needed to follow a path of constant elevation is taken care of by the infallible tendency of a liquid to seek its own level. No matter how many pits, peaks, saddle points, or other peculiarities the surface has, the water is never fooled.

Can this algorithm be adapted for use with a computer rather than a bathtub? It turns out the conversion is easy, at least for a raster device such as a dot-matrix printer. One can think of the printer as

having a two-dimensional array of pins—a "bed of nails"—covering the entire page. Any pin in the array can either be fired, thereby printing a dot on the page, or prevented from firing, so that a white space is left. In the computer version of the bathtub algorithm, a pin is fired only if the corresponding position on the solid model would lie at the water line during one of the stages in the filling of the tub.

Consider a map of the function $z = f(x, y)$. Assume a contour line is to be drawn at $z = 10$, $z = 20$, $z = 30$, and so on. Each pin in the imaginary bed of nails has a position defined by a particular pair of x and y values. To determine whether or not a pin should be fired, substitute its x and y coordinates into the function and solve for z . If the result is an even multiple of 10 (or, in other words, if z modulo 10 is equal to 0), the point lies on a contour line and the pin is fired. If z has any other value, the point is left blank. When this calculation is repeated for every pin position, a complete topographic map emerges, with a contour at every 10th unit of elevation.

The bathtub algorithm has an appealing simplicity. There is no need to trace individual contour lines. Indeed, lines and curves do not exist in the program; they appear in the finished map only because the human eye and brain tend to connect nearby points to form a continuum. There is no need for a vector-to-raster conversion; the points can be calculated in the same sequence they are sent to the printer. Most important, the plotting routine requires no intelligence to deduce where a contour is going to go next; the program merely evaluates an equation at each point in the map and makes a binary decision based on the result. It is a dumb program.

One small refinement is needed to make the algorithm work reliably. The modulus operation yields a value of 0 only if z is exactly equal to a multiple of the contour interval. Thus a dot is printed only when it lies precisely on an infinitely thin bathtub ring. For a printer with a finite number of pins this criterion for selecting points is too restrictive. Sections of the contour

lines will pass between the pin positions, and the map will show only a sprinkling of isolated dots rather than the illusion of a continuous curve.

The ideal solution to this problem is to select from the array of possible dots all those that lie closest to a contour line. This is feasible, but it is almost as complicated as contour-following. A much simpler approach is to define an approximate modulus function, which prints a dot whenever z is within some specified range of the contour elevation. For example, a contour at $z = 10$ might be drawn by printing dots at all pin positions whose elevation is between 9.5 and 10.5.

This change does have one untoward effect. In a standard topographic map all the contour lines have a uniform thickness. Allowing a contour to span a range of elevations means it will be thicker where the surface is nearly flat and thinner where it is steep. Whether this feature of the map is regarded as a defect or as an additional visual clue to the slope of the surface is a matter of taste. In the illustrations reproduced here the contours are adjusted to have the same thickness as the spaces between them.

When I stumbled onto the bathtub algorithm, the idea seemed so simple and obvious that I was sure someone else had thought of it before. I still suspect that the method has been known for years, but I have found no evidence of it. Sedgwick L. Simons Jr. of the University of Houston in Texas described a related technique in *BYTE* in 1983, but his method is more complex and is meant for use with a pen plotter. (As this column was going to press, related work was reported by John E. Connert of the University of Minnesota. See A.K. Dewdney's "Computer Recreations," *Scientific American*, Sept. 1986, pp. 14-23.)

Mapping an egg carton

I wrote the contour-plotting program for a printer with a graphics resolution of 60 dots per inch. Hence the imaginary bed of nails for a square map six inches on a side has 360^2 (or 129,000) pins. The equation defining the surface must be evaluated at each of these points.

Of course when it comes to writing a program for a real machine one can no longer pretend that the printer has a bed of nails with 130,000 pins. It has just eight pins, which are swept across the page to

print a line. Each byte received by the printer specifies a pattern of eight dots. A sequence of 360 bytes prints a horizontal strip six inches wide and eight dots deep; 45 such strips create a square map.

It took a few hours to cobble together a functioning program. (A version of this program, called *Hologram.pas*, is available from the *COMPUTER LANGUAGE Bulletin Board Service*.) The equation I chose for the initial test was $z = x^2 + y^2$, which defines a paraboloid, a surface that curves upward like the inside of a mixing bowl, growing steeper with distance from the origin. I asked the program to draw a contour every 100 units over a range of x and y values extending from -100 to $+100$. What I expected to see was a series of concentric circles getting closer together toward the outer edges of the map. The contour lines would be circular because on a paraboloid all points at the same distance from the center are at the same elevation. They would crowd together toward the edges because the slope of the surface increases with radius.

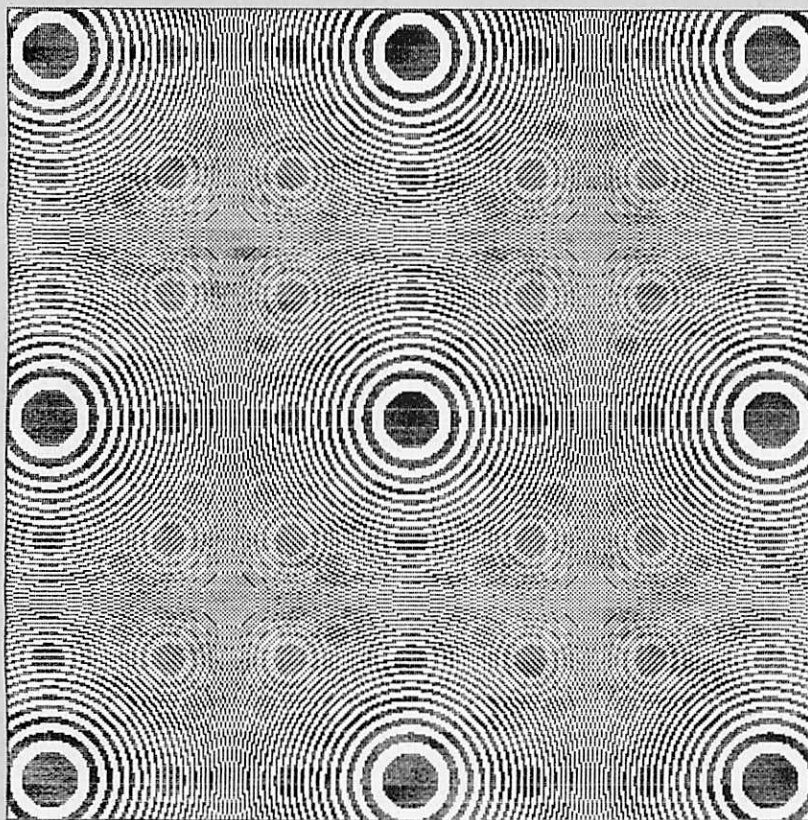
What I did see, rolling out of the printer one line at a time, was a pattern much like the one in Figure 1. There were circles, all right, and lots of them, but they did not all share a common center. Instead the pattern looked like a contour map of the inside of an egg carton, with multiple peaks and cavities arranged on a square grid. On looking at the map more closely I was able to discern shadowy evidence of smaller circles, arranged on a grid one-third as large as the main grid. And there were faint and fuzzy hints of still smaller circles, at one-sixth the main grid spacing. The egg carton appeared to have dimples.

It looked like an interesting bug. Most programs that run amok produce less than what you want, not more; this one seemed to be inventing elaborate ornamentation for a simple geometric map. The complexity of the pattern was particularly puzzling. A quadratic equation such as $z = x^2 + y^2$ simply cannot convey enough information to create a surface with multiple hills and valleys. An equation for the surface of an egg carton—even without dimples—would have to be much more complex, with dozens of terms and with higher powers of x and y .

Making waves

At this point I was ready for another soak in the tub, but a shower would have been more appropriate. Looking at the pattern again, I began to see it as ripples formed by raindrops falling on the surface of a pond. There were even suggestions of interference where waves from adjacent drops overlapped. The map also looked vaguely like the interference pattern cre-

Contours on a paraboloid



Periodic array of bull's-eyes is created when a topographic map is digitized at too low a resolution. The surface being mapped is defined by the equation $z = x^2 + y^2$, or (in polar coordinates) $z = r^2$. Only the rings of the central bull's-eye are real contour lines; the rest are artifacts of the sampling process.

Figure 1.

ated when light passes through an array of pinholes. Some X-ray diffraction patterns, which are used in the study of large molecules, have a similar texture. So do holograms: interference patterns (recorded with laser light) that yield a three-dimensional image of an object.

What do waves and interference patterns have to do with contour lines and bathtub rings? In point of fact, the contours *are* waves: the alternating dark and light bands are analogous to the peaks and troughs of a water wave or the varying electromagnetic potentials of a light wave. A dark band followed by a light band constitutes one cycle of the waveform. The wavelength is the distance from the leading edge of one dark band to the leading edge of the next. The spatial frequency of the wave is the number of cycles per inch.

Thinking of the contours as waves soon led to an understanding of what went wrong with the paraboloid map. There was nothing the matter with the algorithm or the program; the problem lay with the test case I had chosen.

Along the x axis of the map the elevation of the surface ranged from 0 at the center to 10,000 at each edge (where x has values of +100 and -100). I had asked for a contour line every 100 units of elevation, and so the map would have 100 dark and light bands from center to edge. Near the corners of the square the contour density would be more than 100 per in. The minimum distance in which the printer can represent a dark-and-light cycle is two dot positions: one dot must be on and the next must be off. At 60 dots per inch there are 30 such pairs per inch. I was trying to draw a 100-cycles-per-inch pattern with a blunt, 30-cycles-per-inch crayon.

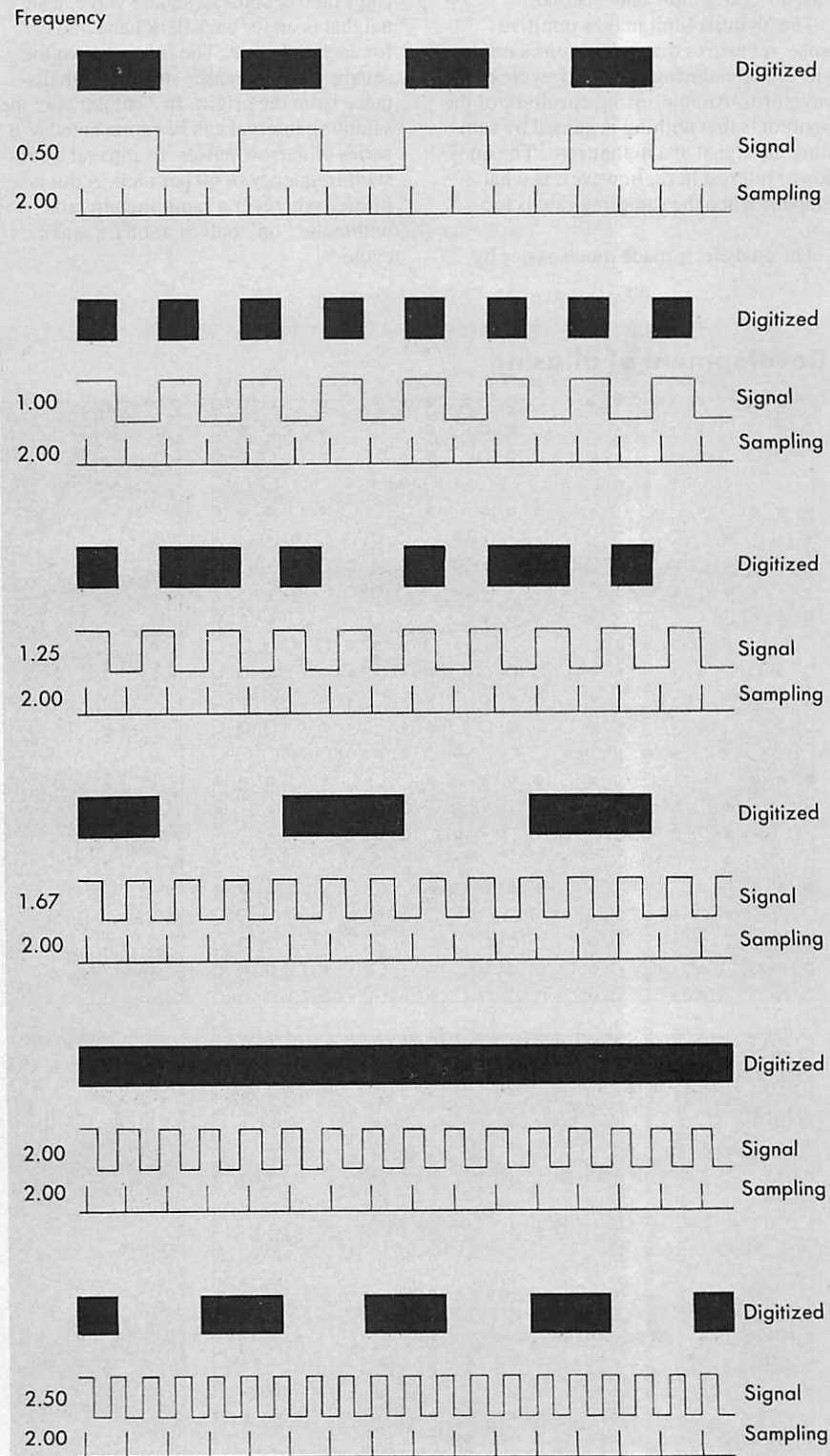
The fix was obvious. When I increased the contour interval from 100 units to 500 units, the expected pattern of concentric circles emerged from the printer. Decreasing the range of x and y values from ± 100 to ± 20 had the same effect. In a few minutes I had my map.

Beyond the Nyquist limit

It is always gratifying to get a program working correctly, but in this case the finished map could not compete in visual interest with the bizarre array of raindrop ripples I had glimpsed along the way. Moreover, knowing why the original test case failed did not explain how the failure created such an elaborately embellished pattern.

The key to understanding where all the ripples come from is the Nyquist sampling theorem, formulated in 1928 by Harry Nyquist of the AT&T Dept. of Development and Research (a predecessor of Bell Laboratories). The theorem states that for a waveform to be accurately digitized, the sampling frequency must be at least twice

The Nyquist limit



Sampling a waveform yields an accurate digital representation only if the sampling frequency is at least twice the signal frequency (the Nyquist limit). The alternating dark and light bands of a contour map can be viewed as a square-wave signal; a dot-matrix printer digitizes the bands at a fixed sampling rate determined by the printer's pin spacing. As the signal frequency increases beyond the Nyquist limit, the frequency of the digitized output declines, reaching 0 at twice the limit; the output frequency then climbs again. The spurious low-frequency signals are aliases.

Figure 2.

the highest frequency in the signal. For example, a musical passage with frequencies of up to 20,000 Hz requires at least 40,000 samples per second.

The Nyquist limit makes intuitive sense: it ensures that at least one sample will be taken during each half-cycle of the waveform. An interesting corollary of the theorem is that nothing is gained by sampling the signal at a higher rate. The question of interest here, however, is what happens when the sampling rate is too low.

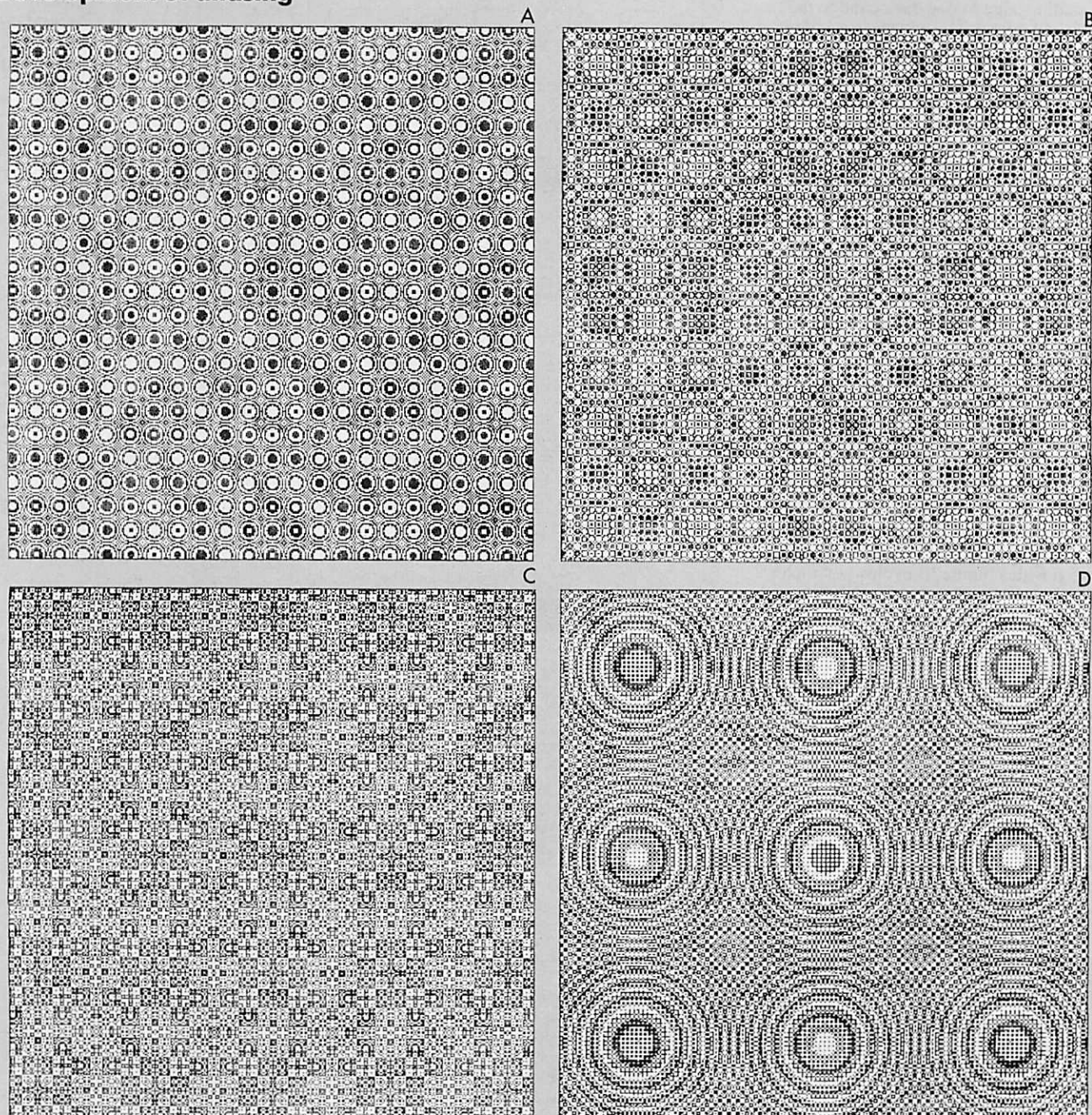
The analysis is made much easier by

reducing the problem to one dimension and considering only the points along the positive x axis. The sequence of bathtub rings then becomes a square wave, a signal that is on for each dark band and off for each light one. The frequency of the square wave increases steadily with distance from the origin. In a similar way the sampling interval can be represented as a series of narrow pulses, or pips, at a constant frequency of 60 per inch. A dot is printed wherever a sampling pip falls within the "on" half of a square-wave cycle.

Near the center of the map the sampling works as it should. Each "on" period spans several pips and accordingly is printed as a block of consecutive dots. As x increases, however, the square waves become narrower, and eventually the width of an on-off cycle falls to $1/30$ th of an inch. This is the Nyquist limit, where there is exactly one sampling pip per "on" period and one per "off" period. Each dark band is printed as a single black dot and each light band appears as a one-dot space.

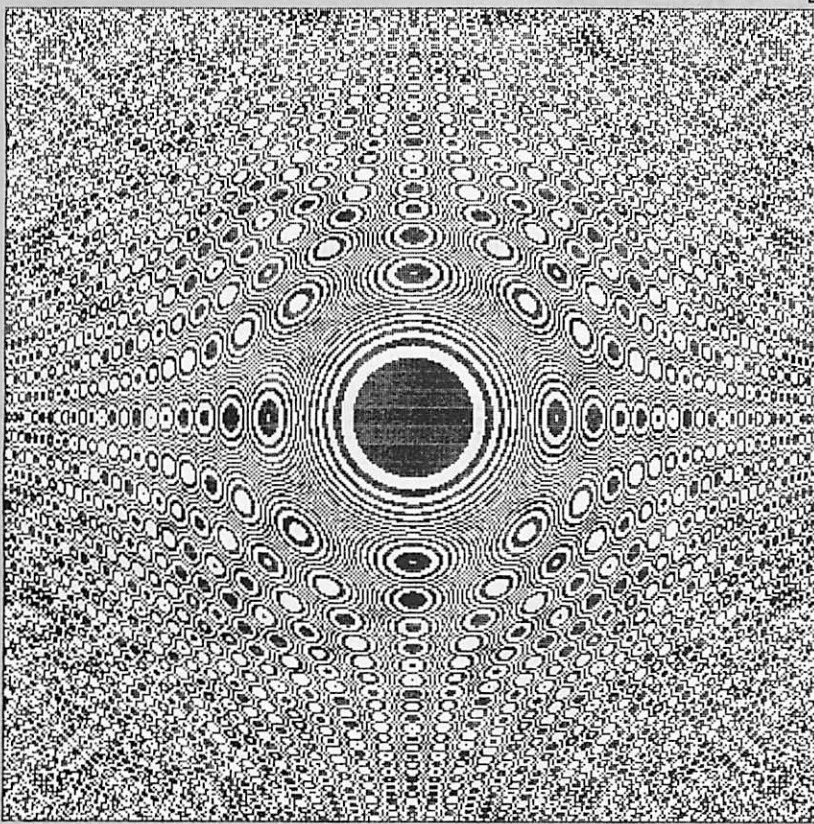
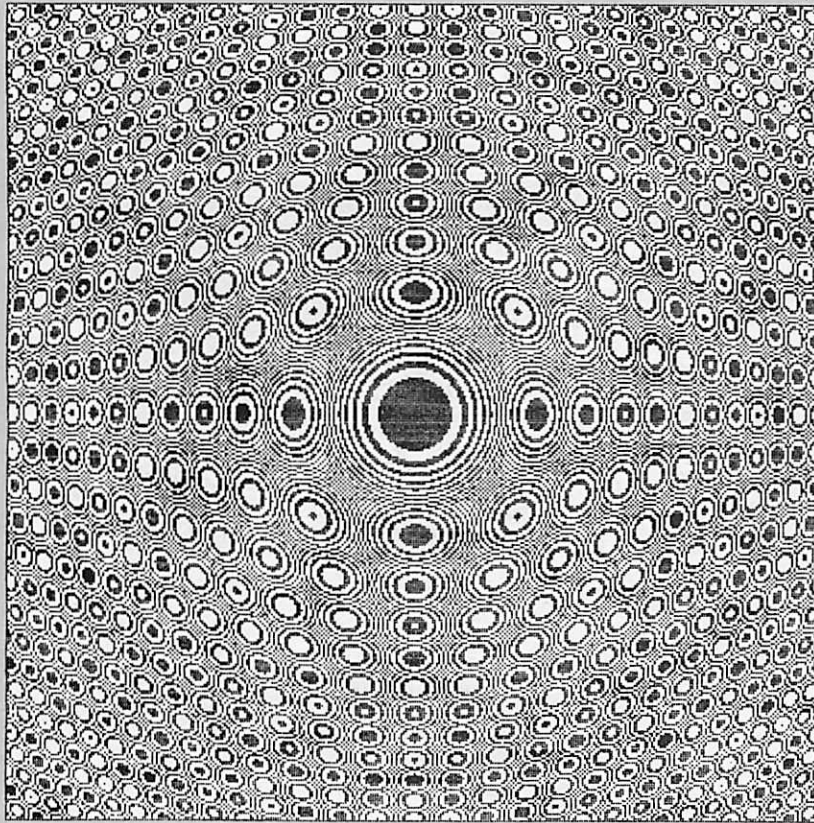
Beyond the Nyquist limit the sampling

Development of aliasing



Contour maps of a paraboloid exhibit progressively more severe aliasing. The surface is the same one mapped in Figure 1, where the maximum spatial frequency is about three times the Nyquist limit. In Figure 3A the maximum frequency exceeds the limit by about 30 times, in Figure 3B by 100 times, and in Figure 3C by 1,000 times. At this point the shape of the surface is completely obscured, and yet circular forms appear again in Figure 3D, where the frequency is roughly 10^{12} times the Nyquist limit.

Cubic and quartic surfaces



Surfaces with the same general form as a paraboloid but a steeper slope give rise to distorted arrays of circles and ellipses. The surfaces are defined by the equations $z = r^3$ (Figure 4A) and $z = r^4$ (Figure 4B).

cannot keep up with the undulations of the waveform. Figure 2 shows what happens. At first, when the signal frequency is just slightly beyond the Nyquist limit, a few "on" periods are missed because they happen to fall between two sampling pips. As a result an extra space is inserted. "Off" periods are overlooked in the same way, causing two adjacent pins to be fired. As the signal frequency increases further, such missing samples become more common and dots are printed in larger and more widely spaced blocks. In other words, as the input frequency goes up the frequency of the digitized output goes down.

At twice the Nyquist limit, where the signal frequency is equal to the sampling rate, the output frequency falls to 0. Each sampling pip comes at the same relative position in the signal waveform, so that the signal appears to be either constantly on or constantly off. The dots are either all black or all white. The sampling mechanism cannot know that the signal goes through a complete cycle between samples.

At three times the Nyquist limit, alternating dots and spaces are again printed. The signal goes through one-and-a-half cycles in each sampling interval, so that every other pip falls in an "on" period. At four times the limit another solid band of either black or white appears. The periodic variation in the frequency of the digitized pattern continues indefinitely. Whenever the signal frequency is an odd multiple of the Nyquist limit, every second dot is turned on. Where the frequency is an even multiple of the limit, all the dots are either on or off, depending on the relative phase of the sampling interval and the signal waveform.

An analysis of this kind can explain the pattern of dots along any radius of the paraboloid map. Examining the pattern along the x axis, for example, shows that the spatial frequency repeatedly goes from 0 to the maximum resolution of 30 cycles per inch and then back to 0.

What is harder to fathom is how a violation of the Nyquist limit gives rise to the specific two-dimensional pattern seen in Figure 1. Only the rings of the central bull's-eye are real contours, marking curves of equal elevation; all the other circles are artifacts of the sampling process. If you tried to follow one of the phony contours on foot, you would find that the path is far from level. Nevertheless, all the bull's-eyes are perfect replicas of the central one.

The regularity of the pattern can be attributed to two factors. First is the symmetry of the paraboloid itself. Everywhere on a paraboloidal surface the slope increases in direct proportion to the radius; this is just the rate of change

C & PASCAL PROGRAMMERS

Blaise Computing provides a broad range of programming tools for Pascal and C programmers, with libraries designed for serious software development. You get carefully crafted code that can be easily modified to grow with your changing needs. Our packages are shipped complete with comprehensive manuals, sample programs and source code.

C TOOLS PLUS

\$175.00

NEW! Full spectrum of general-purpose utility functions; windows that can be stacked, removed, and accept user input; interrupt service routines for resident applications; screen handling including EGA 43-line text mode support and direct screen access; string functions; and DOS file handling.

PASCAL TOOLS/TOOLS 2

\$175.00

Expanded string and screen handling; graphics routines; easy creation of program interfaces; memory management; general program control; and DOS file support.

VIEW MANAGER

\$275.00

Complete screen management; paint data entry screens; screens can be managed by your application program; block mode data entry or field-by-field control. Specify C or IBM/MS-Pascal.

ASYNCH MANAGER

\$175.00

Full featured asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; baud rates up to 9600; modem control and XMODEM file transfer. Specify C or IBM/MS-Pascal.

Turbo POWER TOOLS PLUS

\$99.95

NEW! Expanded string support; extended screen and window management including EGA support; pop-up menus; memory management; execute any program from within Turbo Pascal; interrupt service routine support allowing you to write memory resident programs; schedulable intervention code.

Turbo ASYNCH PLUS

\$99.95

Complete asynchronous communications library providing interrupt driven support for the COM ports; I/O buffers up to 64K; XON/XOFF protocol; and baud rates up to 9600.

RUNOFF

\$49.95

NEW! Text formatter written especially for programmers; flexible printer control; user-defined variables; index generation; and general macro facility. Crafted in Turbo Pascal.

EXEC

\$95.00

Program chaining executive. Chain one program from another even if the programs are in different languages. Shared data areas can be specified.

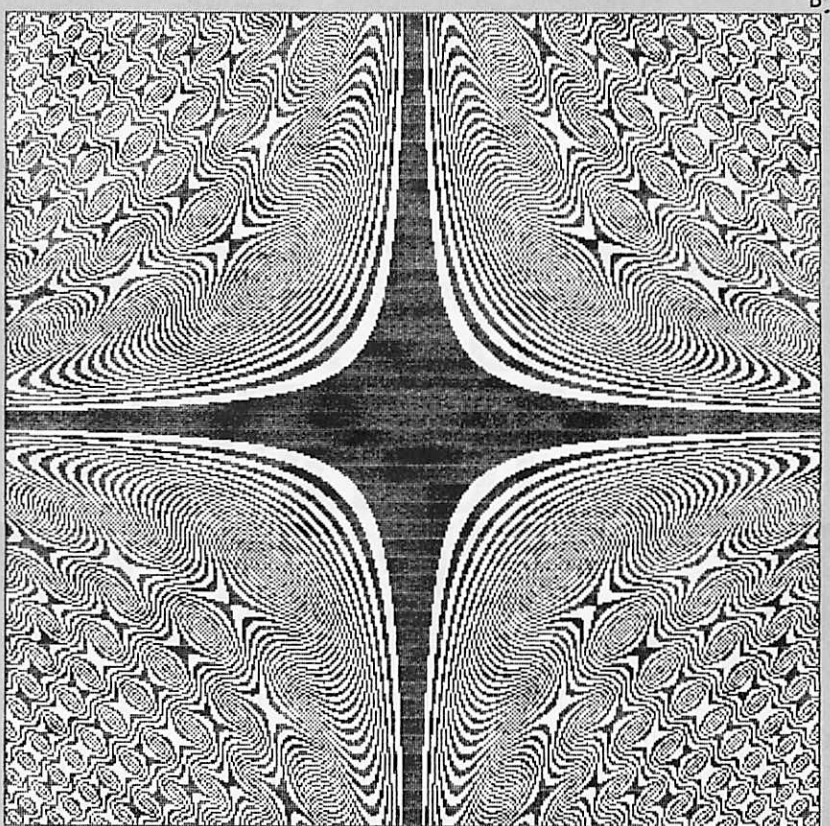
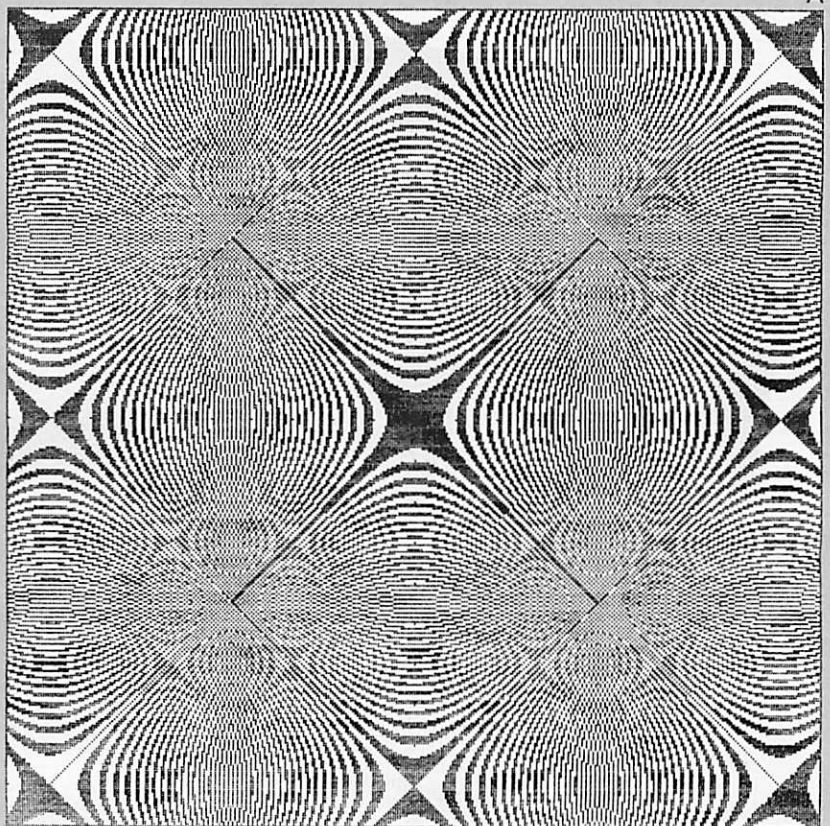
ORDER TOLL-FREE 800-227-8087!



BLAISE COMPUTING INC.

2560 Ninth Street, Suite 316 Berkeley, CA 94710 (415) 540-5441

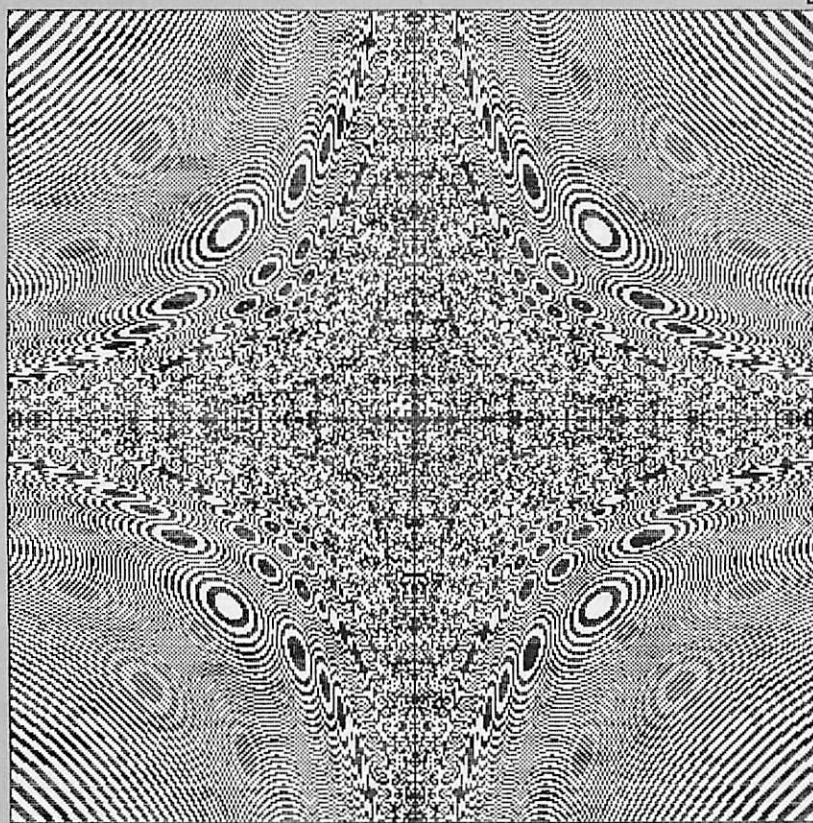
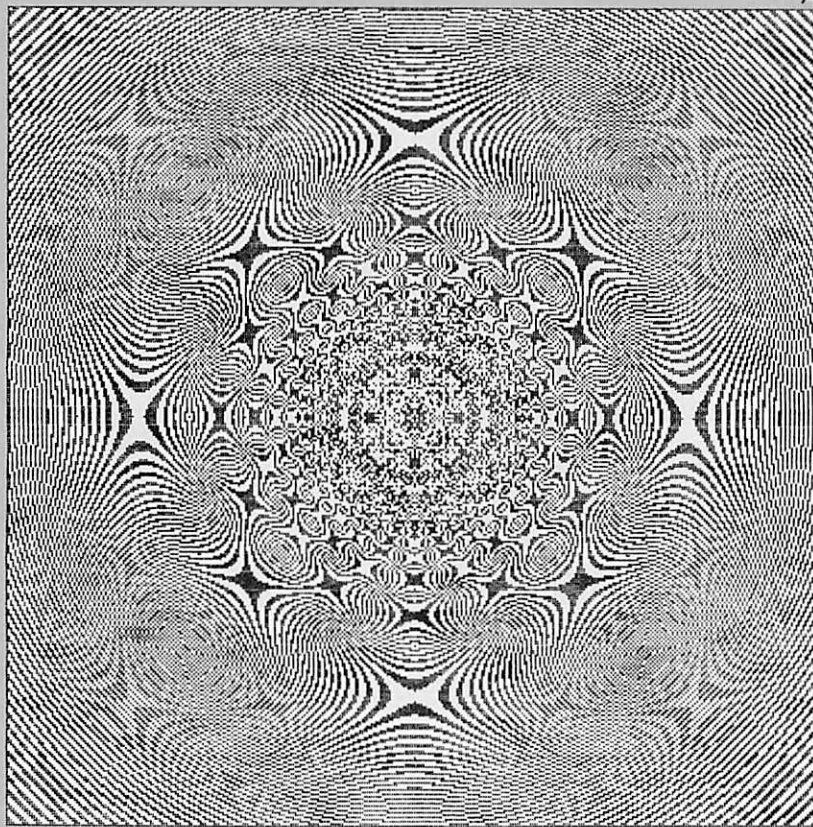
Surfaces with hyperbolic contours



Four-pointed stars are a common feature of aliased topographic maps whose genuine contour lines are hyperbolas. In Figure 5A the defining equation is $z = x^2 - y^2$; the surface has two lobes that bend upward and two that bend downward, meeting in a saddle point at the origin. In Figure 5B the equation is $z = x^2 y^2$, and there are four upward-sweeping lobes.

Figure 5.

Poles and cusps



Surfaces that approach infinity at the origin compress the entire spectrum of aliased textures into a single map. The equation in Figure 6A is $z = 1/r$, defining a surface that rises steeply near the center and is undefined at the origin itself. In Figure 6B the equation is $z = 1/xy$, and the surface forms cusps along both axes. Note that although Figure 6A has circular contours, the figures in the alias pattern are hyperbolic; likewise, Figure 6B has hyperbolic contours but an alias pattern made up of ellipses.

needed to create a uniform lattice of rain-drop ripples. The second factor is the geometry of the printer pins, which are laid out on a square grid. If someone made a printer with pins in a honeycomb arrangement, the bull's-eyes would form a hexagonal lattice rather than a rectangular one.

The faint gray circles interspersed among the larger and more conspicuous ones are also sampling artifacts. Note that as the signal frequency increases beyond the Nyquist limit, the sampled output does not change smoothly from on/off to on-on/off-off, and so forth. Instead there are unbalanced intermediate states such as on-on/off and on/off-off. It is these unbalanced dot sequences that create the shadow patterns.

Aliasing

The spurious low-frequency signals that appear when a waveform is undersampled are called aliases. In computer graphics the most familiar manifestation of aliasing is the jagged, stair-step edge that appears when a diagonal line is displayed on a low-resolution raster device. The stair-step effect has a number of similarities to the patterns seen in aliased contour maps. For example, the spatial frequency of the stair steps, like that of the false contours, varies periodically as a function of slope. The stair-step frequency is 0 when the line is horizontal; it rises to a maximum at 45 degrees and returns to 0 when the line is vertical.

When aliasing turns a line into a staircase it is a nuisance. But as the illustrations reproduced here suggest, aliasing can also create some intriguing and rather decorative patterns. Figure 3 shows the development of progressively more severe aliasing in a contour plot of a paraboloid. As the contour interval is reduced, the lattice of intersecting ripples becomes a mosaic of dark and light tiles. When the spatial frequency of the contour lines is hundreds of times greater than the Nyquist limit, the map degenerates into a texture that seems to bear no relation to the form of the surface. The underlying pattern is still present, however. Indeed, the characteristic overlapping rings suddenly reappear at a much higher multiple of the Nyquist limit, where the waveform goes through some 10^{12} cycles between sample points.

Circles and ellipses are a recurring motif of aliased maps whose basic contour lines are circles. Figure 4 shows maps of the cubic and quartic analogues of a paraboloid (surfaces on which elevation is proportional to the third or fourth power of the radius). They can be interpreted as distorted versions of the paraboloidal map.

Surfaces that have a saddle point or a

cusps tend to yield a different repeating figure: a four-pointed star, marking the convergence of four sets of hyperbolic contour lines. The equation $z = x^2 - y^2$ is obviously similar to the paraboloid equation, and yet the surface it defines is very different. There are upward-sweeping lobes along the x axis and downward-sweeping ones along the y axis, which meet in a saddle point at the origin. The contour plot (Figure 5A) has the same symmetries as the paraboloid map, but all the circular forms are replaced by stars. The same stars, somewhat distorted, turn up again in the map of $z = x^2y^2$, a surface with four lobes oriented along the diagonals (Figure 5B).

Although the false contours generally mimic the form of the real ones, there are exceptions. Figure 6A maps a surface that is essentially the inverse of a deep bowl; far from the origin it is nearly flat, but near the center it rises to form a steep pole. The contours are circular, but the alias patterns are hyperbolic stars. Figure 6B, on the other hand, has hyperbolic contours that converge on cusps, or asymp-

totes, along both axes; nevertheless, aliasing covers the surface with ellipses.

One sign of an interesting program is that you cannot readily predict its output. The contour-mapping program would appear to qualify under this criterion. Even for the very simple and symmetrical equations discussed so far, it is difficult to guess what a highly aliased map will look like; when the equations become more complicated, prediction is all but impossible. The surface defined by $z = 100xy - x^2 - y^2$ slopes smoothly toward negative infinity along the x and y axes but has four humps a little ways from the origin on the diagonals. What does the aliased map look like? If you guessed a bakery tray of cinnamon buns (Figure 7), you have better geometric intuition than I do.

Dot-matrix laser

The resemblance of aliased contour maps to interference patterns is not mere coincidence. An interference pattern forms when two waves are superimposed or added. Where the two component waves are in phase, the sum has a large amplitude; where the waves are out of phase, they cancel and the sum is 0. The digitizing process effectively adds waves at

the sampling frequency and at the frequency representing the slope of the surface. The resulting contour map can be regarded as a product of interference.

A hologram is a special kind of interference pattern made with the coherent light emitted by a laser. (Light is said to be coherent when all of its waves are in phase.) In making a hologram, laser light is split into two parts. One beam goes directly to a piece of photographic film, while the other beam first strikes the object being recorded and is then reflected onto the film. At the plane of the film the two beams interfere, exposing the photographic emulsion wherever the waves are in phase. The phase at the emulsion depends on the length of the path followed by the reflected beam.

Under a microscope a hologram is seen to be a maze of fine interference fringes. It does not look like the object recorded, but if laser light is shone through the film, the stored interference pattern acts on the waves to reconstruct the original reflected waveform. The result is a three-dimensional image of the object.

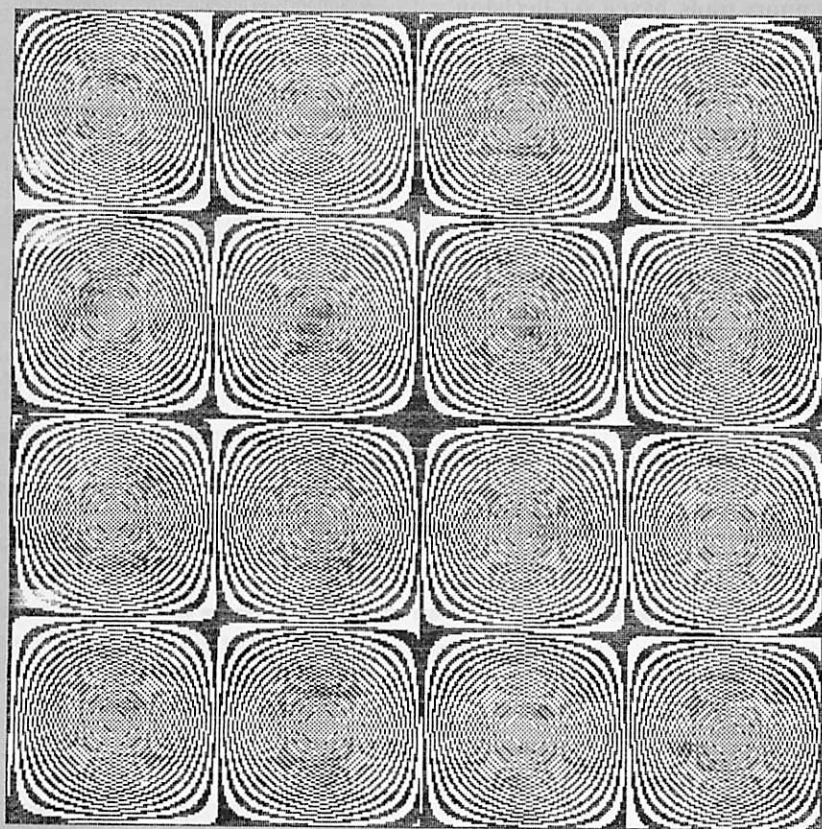
Aliased contour maps are crude holograms made with a long-wavelength laser and coarse-grained photographic film. Suppose a source of coherent light has been set up at a great height above a paraboloid. Parallel rays of light fall on the surface, and those reflected straight back along their original path are intercepted by a piece of film. Meanwhile, another beam from the same source goes directly to the film. A grain of the photographic emulsion is exposed if the two waves striking it interfere constructively. Whether the interference is constructive depends on the phase of the reflected beam and hence on the elevation of the surface. To be more precise, the phase depends on the elevation modulo the wavelength of the light.

This procedure for making a hologram is exactly equivalent to the bathtub algorithm for plotting a topographic map. The wavelength of the laser light corresponds to the contour interval; the grain size of the film is the pin spacing. Although the dot-matrix hologram does not look like a map of the paraboloidal surface, it encodes all the information needed to reconstruct that surface. If only we had a laser of the appropriate wavelength, we could shine it through the hologram and create an image of the paraboloid. ■

References

- Arnold, Steven M. "Computer-Generated Holograms." *Scientific Honeyweller* (Dec. 1984): 44-54.
- Johnston, Chris. "Contour Plots of Large Data Sets." *COMPUTER LANGUAGE* (May 1986): 63-67.
- Simons, Sedgwick L., Jr. "Make Fast and Simple Contour Plots on a Microcomputer." *BYTE* (Nov. 1983): 487-492.

Cinnamon buns



An unlimited variety of two-dimensional alias patterns remains to be explored. This unusual tiling of the plane with square swirls is based on the equation $z = 100xy - x^2 - y^2$.

Figure 7.