Fractal Images with TEX

ALAN HOENIG

Department of Mathematics
John Jay College of Criminal Justice
[use mailing address below]:
17 Bay Avenue
Huntington, NY 11743

ABSTRACT

This article is an additional demonstration that TEX can create non-textual images. Using halftone fonts created by Knuth, TEX can typeset fractal images. The images are somewhat coarse-grained; this is due to the limitations on the number of characters per page that can be typeset by TEX.

Sometimes one tires of the printed word, and at such black times, it's nice to know that TEX has its non-textual uses as well. Knuth [7] has previously demonstrated the possibility of typesetting halftone images using special halftone fonts. Shortly thereafter, Clark [2] was the first (apparently) to use these fonts and comment in writing on the experience. Their work is fascinating, but the drawback to the casual user of TEX who wants to participate in these experiments lies with specialized equipment one needs to generate the halftone data. I will discuss ways of using purely mathematical rules to generate images of beauty and realism.

1. Fractals

In the past 20 years or so, mathematicians have discovered new families of objects, shapes of great mystery, beauty, and intrigue. There is mystery and intrigue because, despite the simple mathematics used to generate them, they have lain undiscovered all this time. These shapes have become known as *fractals*. A precise definition of this term is out of place here; see [8]. It suffices to explain, I think, that they involve the disclosure of patterns of extraordinary complexity where one would naïvely expect to encounter chaos or monotony.

We set the stage with a recipe for generating a simple fractal pattern. Let z_i represent point i on the complex plane. We use the METAFONT mediation function to denote some other point lying some portion of the way between two given points. If $z = t[z_0, z_1]$, then z is the point lying on the line connecting z_0 to z_1 and located a fraction t of the way between z_0 and z_1.

Begin by locating the three corners of a non-degenerate triangle, and by selecting a random point z_0 lying somewhere within the triangle. Draw this point. Now:

1. Select one of the vertices of the triangle at random.[1] Let z_v be the chosen vertex.
2. Construct the midpoint z between z_0 and z_v; that is,

$$z = 1/2\,[z_0, z_v]$$

3. Draw this point. Re-label this point z_0, and return to step 1.
4. Continue this iteration until several tens of thousands of points have been generated and drawn.

What pattern will appear on the page? I am not ashamed that my own intuition "assures" me that a uniform spread of dots will appear. Nothing prepares me for what really appears.

[1] A roll of a single die makes a good random vertex generator. Choose vertex 1 if 1 or 2 show on the die; choose the second vertex if 3 or 4 show on the die; choose the third vertex otherwise.

But before displaying this pattern, let me make a connection between this procedure and TEX or METAFONT. The mathematical procedure that creates the midpoint is a special kind of object called an *affine transformation*. These are precisely the tools within METAFONT for rotating, shifting, stretching, and shrinking graphic objects. As a first attempt, we might try to create a single character using METAFONT to plot lots and lots of points. My success with METAFONT has been limited, largely because great quantities of dots cause METAFONT to exceed its capacity.

TEX provides an altogether more satisfactory approach. The idea is to use one of the halftone fonts suggested and demonstrated by Knuth [7]. Here's how to proceed.

We use an external computer program to generate the data. First, we translate the above algorithm into a high-level language (I used QuickBASIC or FORTRAN). The computer program should ask the computer to maintain a large 2×2 array, the four corners of which represent the corners of our diagram. Then, each time the computer generates a random point, it must determine which spot in the array best corresponds to the location of the dot on the page. Increment the value of this element in the array by 1. Do this several thousand, or tens of thousands, of times.

When done with this series of iterations, the program will spit this information out to a disk file. Instead of spewing rows of whole numbers, we ask the computer to spread the values in some way between the character **0** and **p**.[2] There are 64 characters lying between **0** and **p** in the ASCII convention, and Knuth's halftone characters occupy these positions. Next, print out *these* characters to the disk file. This creates an ASCII file which we feed into TEX. Of course, we need to make some adjustments so TEX processes the file properly — suppress interlineskipping and indentation, invoke the halftone font, and some other things. Here is what you see when you typeset this pattern (Fig. 1). Benoit Mandelbrot, the mathematician who is the father of things fractal, calls it the Sierpinski carpet.

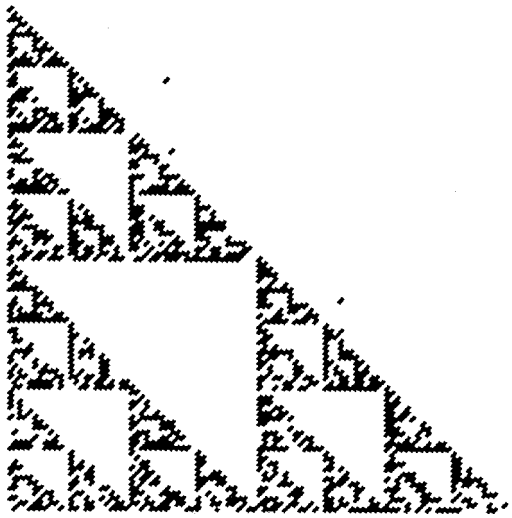

Fig. 1: Typesetting the Sierpinski carpet

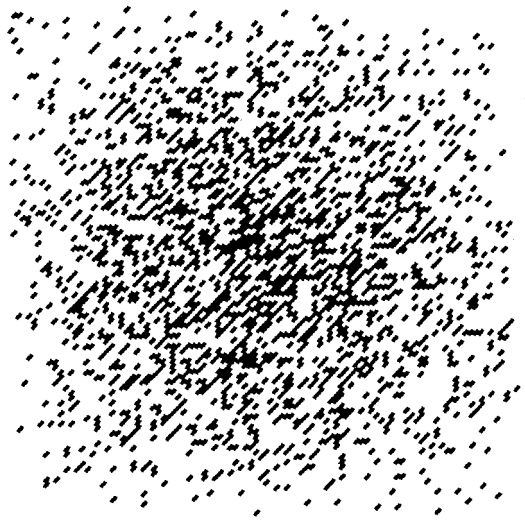

Fig. 2: A 4-cornered carpet: first attempt (unsuccessful)

The coarseness of this carpet's weave is due to at least two factors. There are various upper limits on the number of characters that TEX can typeset on a given page. Remember, each dot in a halftone is a separate character as far as TEX is concerned, and a single image involves far more characters than are usual on any single page of typeset matter. A greater dot density would be more pleasing. Furthermore, to enhance the contrast in these few figures, only two distinct characters were used in the typesetting, **0** if a pixel never contained any point, or **p** otherwise.

[2] Because of the peculiarities of human perception, it probably shouldn't be a linear spread; see the helpful figures on page 137 of reference [7] for insight into this problem.

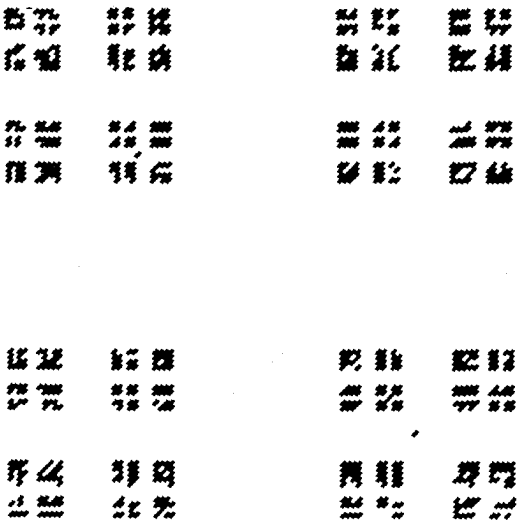

Fig. 3: A 4-cornered carpet: second attempt (successful)

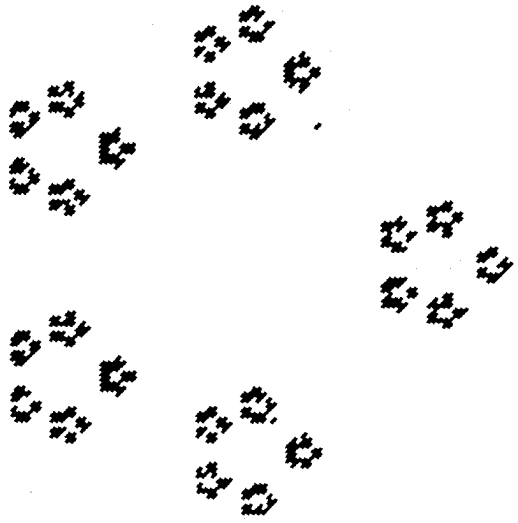

Fig. 4: A pentagonal Sierpinski carpet

2. Fractal Botany

Consider n transformations, and a set of probabilities p_1 through p_n, so that $\sum_{i=1}^{n} p_i = 1$. We will create a sequence of points, place them in an array, and then typeset this array as we did the Sierpinski carpets. The sequence begins with a single point, and successive points are found by transforming a predecessor point by one of the n transforms. Any of these transforms are chosen randomly, the only constraint on this random choice being that in the long run, the i^{th} transform should be chosen a fraction p_i of the time. A deep theorem guarantees that it is possible to choose the transforms and the probabilities so that the resulting pattern resembles real objects. These objects may appear uncannily real, perhaps because these images seem to repeat certain patterns at ever decreasing levels of scale, like Russian *matrushka* dolls. One example is the fern shown in Fig. 5. Some computer scientists find these images exciting; this technique holds promise that the large amounts of information one might think necessary to create images at a convincing level of detail can be easily summarized by some relatively small group of transformations.

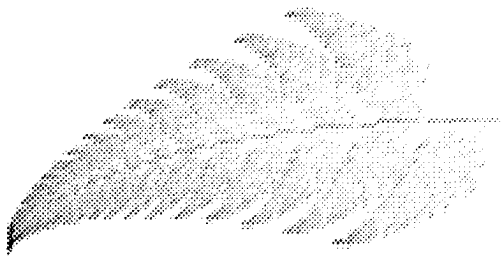

Fig. 5: A fractal fern

Fig. 6: A fractal, though stemless, fern

This fern is an outgrowth of the application of four transformations. Each transform w_i is determined by six constants. Let the six constants for transform w_i be a_i through f_i, and if $z = (x, y)$, then the transformed point $w_i(z)$ is

$$w_i(z) = \begin{pmatrix} a_i & b_i \\ c_i & d_i \end{pmatrix} \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e_i \\ f_i \end{pmatrix}$$

The following table shows the values for the fractal fern and the associated probabilities for each transform w_i:[3]

w	a	b	c	d	e	f	p
1	0	0	0	0.16	0	0	0.01
2	0.85	0.04	−0.04	0.85	0	1.60	0.85
3	0.20	−0.26	0.23	0.22	0	1.60	0.07
4	−0.15	0.28	0.26	0.24	0	0.44	0.07

I was especially intrigued by the first probability, $p_1 = .01$. It's hard to imagine what the contribution of w_1 can be to the final image since it is applied so seldom. It is easy to investigate by re-running the program with $p_1 = 0$ and $p_2 = .86$ (so the sum of the probabilities continues to equal 1). In the absence of w_1, we grow a peculiar stemless fern (Fig. 6).

Barnsley [1, p. 87] discusses this material in great detail, and displays some extraordinary examples of the fractalist's art. This book displays several striking plates — subjects include fields of sunflowers, Alaskan huskies, a young Bolivian woman, and scenes from the Black Forest. It would have been nice to use these transforms to typeset pictures, but the data is proprietary.

3. Mandelbrot and Julia Sets

There is a final group of fractal images we can generate. If we imagine that our page represents the complex plane, then complex points c within a so-called *Mandelbrot set* are those which lead to sequences of points which remain bounded. The sequence of points stem from a repeated iteration of a simple non-linear equation:

$$z_0 = 0$$
$$z_n = z_{n-1}^2 + c$$

that is, the sequence 0, c, $c^2 + c$, $(c^2 + c)^2 + c$, and so on. For some values of c, these sequences eventually generate elements which lie ever farther from the origin of the page. For some other values of c, no matter how many terms we examine, they remain within commuting distance of the origin. It is these latter points which comprise the Mandelbrot set.[4]

Using the apparatus available for all these projects — auxiliary program to generate data and prepare a TEX source file, typesetting using special halftone fonts, and final previewing or printing — leads to some interesting results. The shape of the Mandelbrot set surpasses rational expectation (Fig. 7).

The fuzzy outline of this set is not an artifact of the conversion to TEX halftones, and we can prove this by magnifying the image near the boundary of the set. The image fails to exhibit any further smoothness no matter how much the magnification is increased. Increasing levels of magnification reveal ever-increasing levels of complex patterns (Fig. 8). The patterns appear to me to be quite striking, but judge for yourself.

When I began playing with this material, I used METAFONT to generate the numerical data. Runs used to take eight hours! I have since learned how to speed up my calculations by a factor of about 24.

Pictures of related objects, *Julia sets*, deserve mention. Recall that in the preparation of the Mandelbrot set, we varied c. The sequence always began with $z_0 = 0$. To generate a Julia set, follow

[3] These values originally appear in [1].

[4] By the way, concepts such as "complex numbers," "sequences," and "distances from the origin" are not complex concepts at all. The reader is urged to seek out references [3]–[5] for an excellent introduction to these concepts and their connection to Mandelbrot and Julia sets.

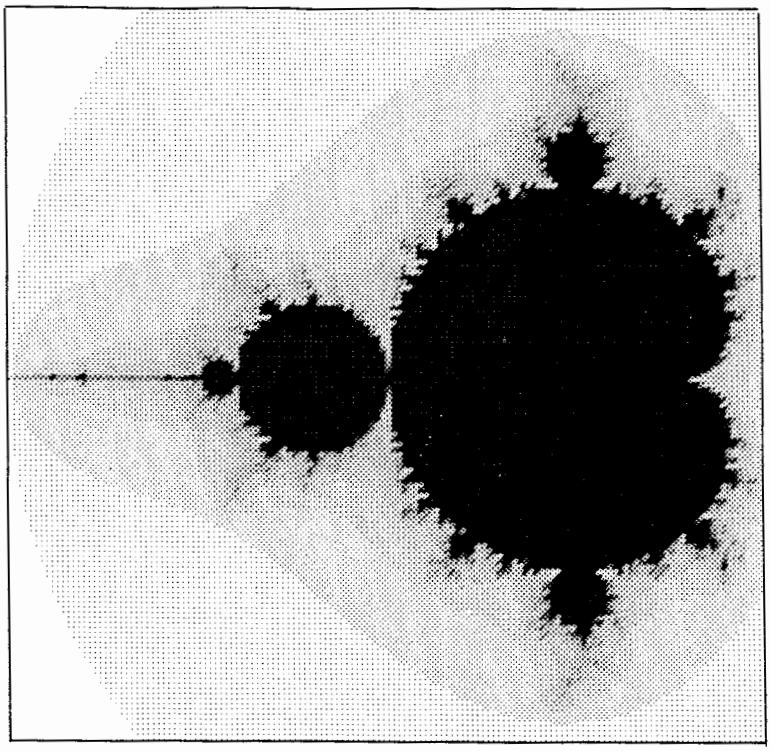

Fig. 7: The Mandelbrot set

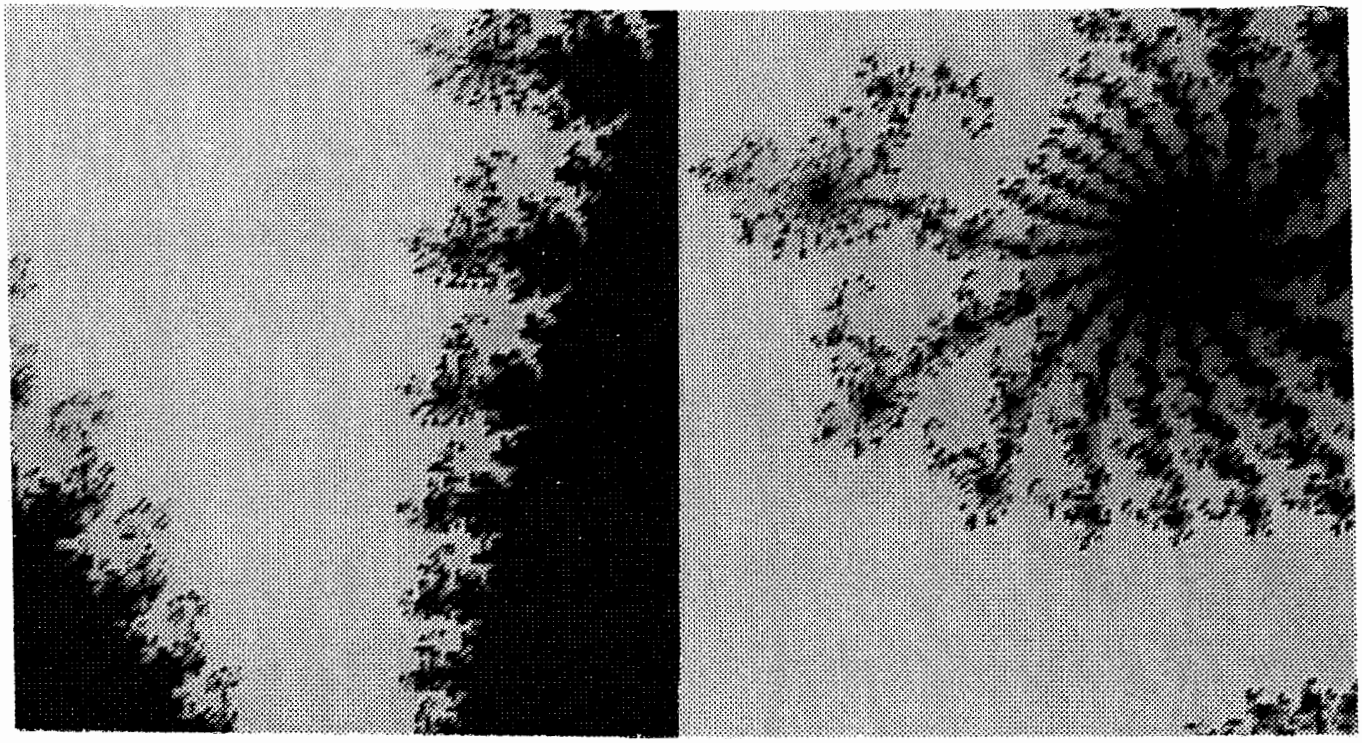

Fig. 8: The Mandelbrot set: several magnifications

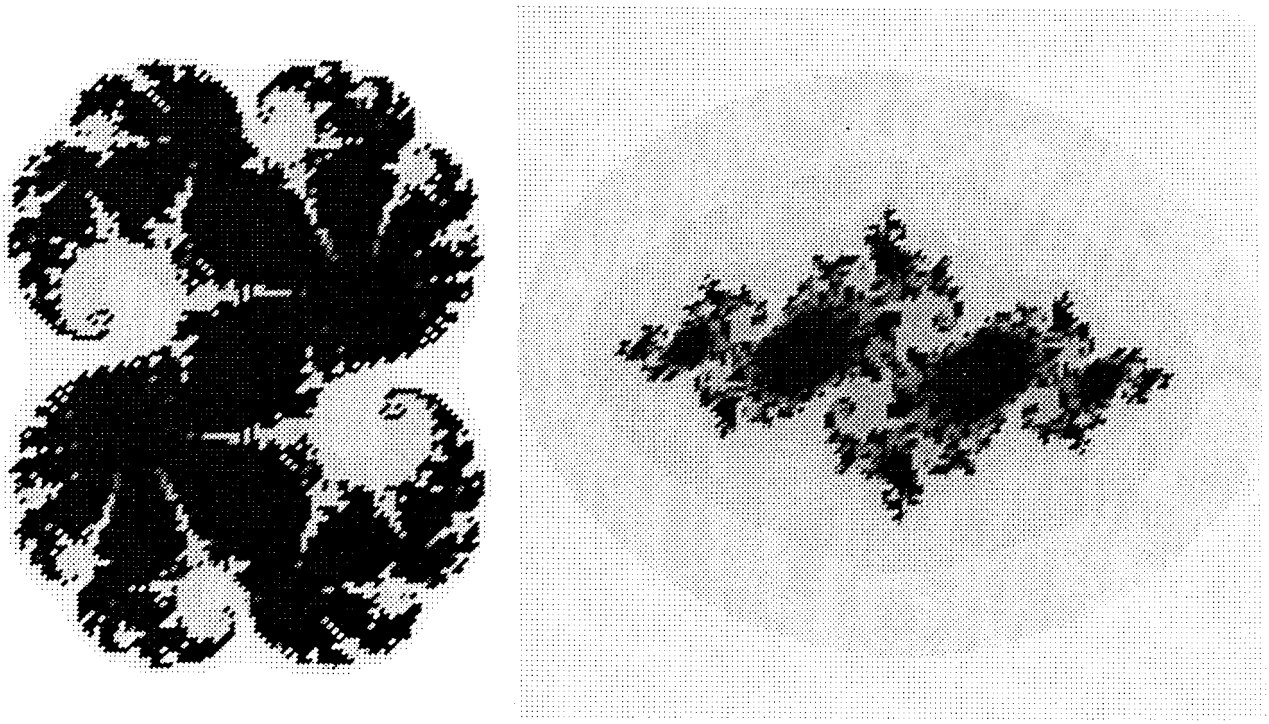

Fig. 9: Julia sets set with TEX

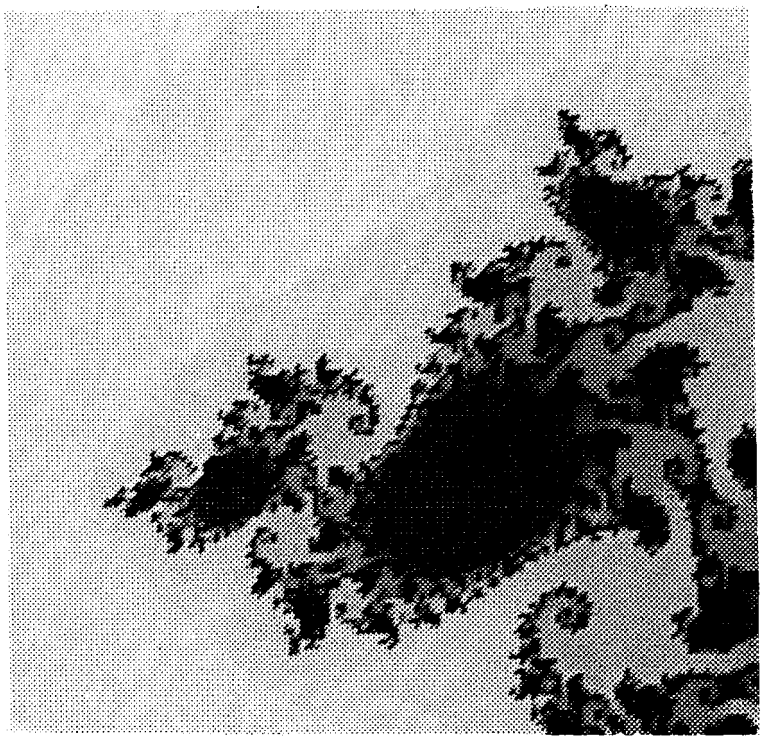

Fig. 10: Detail from a Julia set

the algorithm for the Mandelbrot set with a few modifications. In this case, fix a value of c and let the choice of z_0 range over the complex plain. The points of the Julia set consist of those z_0 for which the sequence z_0, $z_1 = z_0^2 + c$, $z_2 = z_1^2 + c$, ... remains bounded. Julia sets display their own good looks (Fig. 9).

As with the Mandelbrot set, magnifying the detail of a Julia set reveals new levels of intricate filigree (Fig. 10).

Of course, there are other kinds of halftone fonts that will suffice when typesetting these pictures. Knuth has suggested several of them [7], and their uses give interesting new looks to these pictures (Fig. 11).

Knuth has observed that if you enjoy "fooling around making pictures" and if you have suitable fonts, then "TEX will be a source of endless frustration/amusement for you, because almost anything is possible ..." [6, p. 389]. METAFONT is now as widely available as TEX, so it's easy to create special purpose fonts. I encourage readers to have as much fun making pictures as I have had.

Bibliography

[1] Barnsley, Michael. *Fractals Everywhere*. New York: Academic Press, 1988.

[2] Clark, Adrian F. "Halftone Output from TEX." *TUGboat* 8:270–274, 1987.

[3] Dewdney, A.K. "Computer Recreations." *Scientific American* 253, 2:16–24, August, 1985.

[4] Dewdney, A.K. "Computer Recreations." *Scientific American* 257, 5:140–145, November, 1987.

[5] Dewdney, A.K. "Computer Recreations." *Scientific American* 260, 2:108–111, February, 1989.

[6] Knuth, Donald E. *The TEXbook*. Reading, MA: Addison-Wesley, 1984.

[7] Knuth, Donald E. "Fonts for Digital Halftones." *TUGboat* 8:135–160, 1987.

[8] Mandelbrot, Benoit. *The Fractal Geometry of Nature*. New York: W.H. Freeman and Co., 1983.

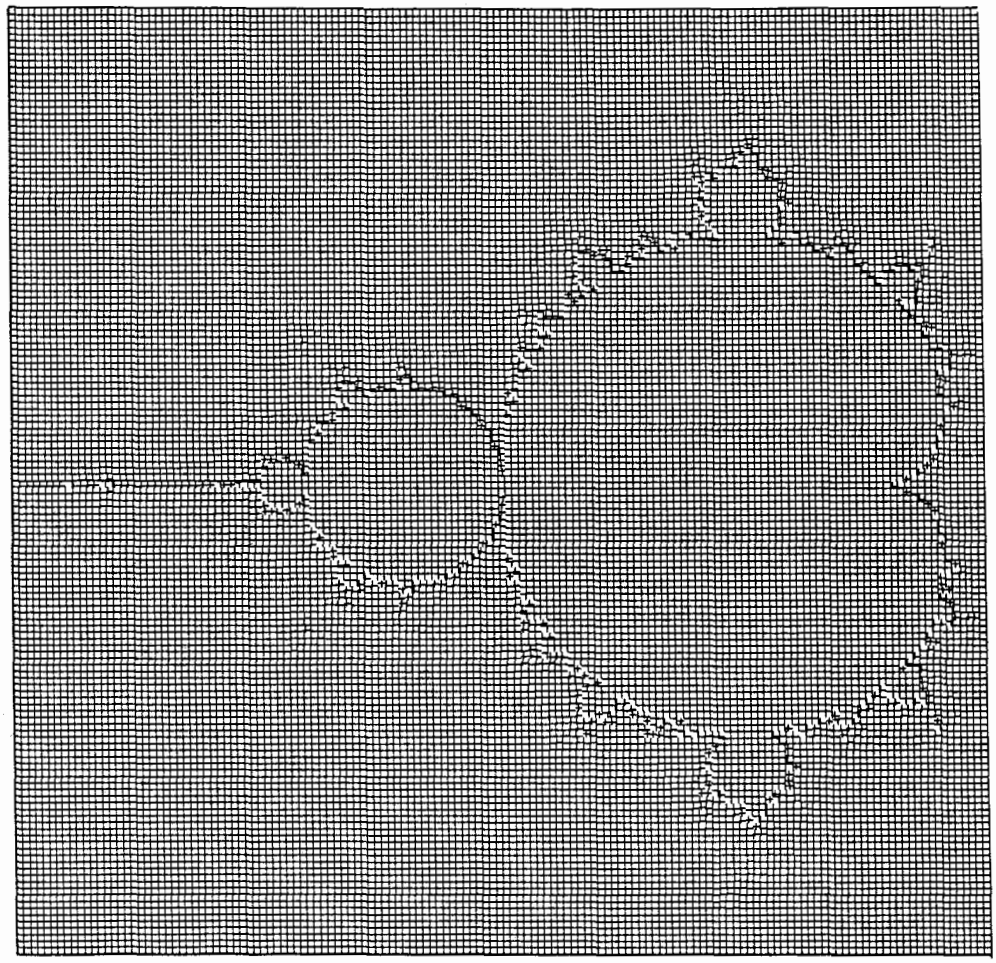

Fig. 11: New looks to the Mandelbrot set