IMAGE SEGMENTATION VIA FRACTAL DIMENSION

THESIS

Alan L. Jones

Captain, USAF

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY
AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

(1)

AFIT/GE/ENG/87D-29

IMAGE SEGMENTATION VIA FRACTAL DIMENSION

THESIS

Alan L. Jones

Captain, USAF

AFIT/GE/ENG/87D-29

AFIT/GE/ENG/87D-29

IMAGE SEGMENTATION VIA FRACTAL DIMENSION

THESIS

Presented to the Faculty of the School of Engineering

of the Air Force Institute of Technology

Air University

In Partial Fulfillment of the

Requirements for the Degree of

Master of Science in Electrical Engineering

Alan L. Jones, B.S.E.E., B.S.E.P.

Captain, USAF

December 1987

Preface

The purpose of this study was to investigate the application of algorithms derived from the study of fractal geometry to the general problem of pattern recognition. The original goal was to decompose an arbitrary input scene into a fractal basis set. It was hoped that this basis set would be orthogonal, such that expansions similar to Fourier series could be derived. The problem proved more difficult than anticipated, and the application of fractal geometry to image segmentation was then investigated as groundwork for further study on the basis set problem. Two widely used methods for calculating fractal dimension and an original hybrid technique were investigated. All three techniques appeared to be effective to some extent in segmenting images, but shared a common problem with the establishment of suitable thresholds for robust segmentation.

Through the course of this reserved time of study, I have had a great deal of encouragement and assistance from those around me. I would like to thank my faculty advisor, Dr. M. Kabrisky, for his receptiveness to this topic of study, Maj. P. Amburn for his enthusiasm and energy, Capt. Steve Rogers for his constructive comments as a committee member and reader, and Capts. Richard Roberts and Dennis Ruck for their assistance with the ITEX_TO_RMS conversion program. My family has been a constant source of relaxation and encouragement, and I wish to thank them for their understanding when studies beckoned, and their prayers for Godspeed and success. I am particularly indebted to my lovely wife Jennifer for her unending encouragement through those times when answers seemed distant; and my precious daughter Emily, who

often provided the refreshing distraction that helped clear my mind for the task at hand. I would most like to express thanks to my Lord and Saviour, Jesus Christ, who by his grace has faithfully seen me through this learning experience.

Alan L. Jones

Table of Contents

List of Figures

Abstract

The purpose of this study was to investigate the suitability of algorithms derived from the study of fractal geometry to the specific problem of image segmentation. The use of two widely used methods and an original hybrid technique for calculating fractal dimension is demonstrated. All three techniques appeared to be effective to some extent in segmenting images, but shared a common problem with the establishment of suitable thresholds for robust segmentation.

IMAGE SEGMENTATION VIA FRACTAL DIMENSION

I. Introduction

Overview

This chapter is intended to provide a brief discussion of the general pattern recognition problem, and discuss the fractal approach to a select area of that problem. Encompassed in the approach is a brief introduction to the fractal set, current fractal research as applied to pattern recognition, and the underlying assumptions used by the author. Next, a plan of attack is developed, the goals and scope of the present research are examined, and evaluation criteria considered. Finally, the support equipment used to carry out the research is discussed.

The Pattern Recognition Problem

Pattern recognition is of interest to the Air Force for automated data entry and reading machines, automated recognition of individuals for security systems, smart targeting systems for munitions, as well as other applications. It is widely accepted that the only effective general purpose pattern recognition machines at the present time are the visual systems of living organisms. One of the greatest difficulties to date has been to obtain an adequate mathematical

description of the targets of interest. To paraphrase Kabrisky's summarization of the inherent difficulty in pattern recognition:

> To find arbitrary targets in arbitrary backgrounds it would be useful to characterize the targets mathematically. In practice, much of the elegant mathematics developed for communication and radar systems (detection and estimation theory) falls apart in many realistic scene analysis problems. The "signal" implied by the shape of the target is unknown mathematically, as is that of the background noise. As a result, pattern recognition techniques are quite often composed of a combination of ad hoc steps that sometimes work depending on the particular data set and individual problem (10).

An examination of the difficulties facing any machine tasked with recognizing images brings the present lack of consistent success into sharper focus. Closely tied to this inability (in general) to adequately describe a target mathematically are the following five complications:

1. Translation - the target may not be in the center of the sensor's field of view, and in general is not.
2. Scale - the target may not be the same size as the model "image" that the pattern recognizer is looking for.
3. Rotation - the target may have an in-plane rotation that differs from the stored model.
4. Aspect - the target may have an out-of-plane rotation that differs from the stored model (for example the sensor is viewing the rear of the target, but the side view is the one stored).
5. Noise (10).

Various ad hoc approaches, some more successful than others, have been used to overcome these problems. One of the more successful approaches was an algorithm developed by Horev (8) and further expanded by Kobel and Martin (12). Their technique was shown in some cases to be effective in dealing with the translation, scale, and (to a limited extent) rotation of input images within cluttered scenes.

The Fractal Approach

The thrust of this research effort is to apply the branch of mathematics known as fractal geometry to the pattern recognition problem. Mandelbrot has defined a fractal set to be a set for which the Hausdorff Besicovitch dimension, D, strictly exceeds the topological dimension (14:15). That is, fractals have noninteger dimension. Consider the following statement by Pentland:

> The world that surrounds us, except for man-made environments, is typically formed of complex, rough, and jumbled surfaces .... If we are to develop machines competent to deal with the natural world, therefore, we need a representational framework that is able to describe such shapes succinctly. ...Fractal functions appear to provide such a model, in part, because many basic physical processes produce fractal surfaces (and thus fractals are quite common in nature), but perhaps even more importantly because fractals look like natural surfaces. ...This is important information for workers in computer vision because the natural appearance of fractals is strong evidence that they capture all of the perceptually relevant shape structure of natural surfaces (20:661,662).

The observation that fractals may be an appropriate measurement feature space for some classes of pattern recognition problems was precisely the motivation for this research effort.

Recent Fractal Research. Fractal analysis has been applied to a surprisingly diverse number of disciplines including landform analysis (14:25-33,267-276), cosmologic distributions (14:89-96), surface and volume analysis (14:109-115; 22), medical imaging (13), generation of computer graphics (5:424-435; 7; 16; 19; 23; 24:28-38), image compression (2), modelling of metalization growth on semiconductors (15; 24:12; 25), the analysis of fractures in metal and stone (14:461), and aspects of scene analysis (10; 20; 21), to specifically point out a few.

Pentland's research has demonstrated that fractal analysis can be extremely useful in segmenting an image (dividing it properly into areas of interest), and also exhibits an ability to determine whether or not the segments are appropriately modelled as fractals (i.e. natural objects) (20). Robust segmentation can be a useful first step in the pattern recognition process, depending on the pattern recognition technique used (10). Robust in this sense implies repeatability that is meaningful.

Assumptions. In virtually all disciplines to which fractal analysis has been applied, the fractal set used has been that known as fractional Brownian motion, or fBm. This research effort will make the assumption that the fractal set of interest is fBm. This fBm set possesses interesting properties that particularly suit it to pattern recognition in natural scenes. These properties include the following:

> 1. A three dimensional surface with a spatially isotropic fractal Brownian shape produces an image whose intensity surface is fractal Brownian and whose fractal dimension is identical to that of the components of the surface normal, given a Lambertian surface reflectance function and constant illumination and albedo.
> 2. A linear transformation of a fractal Brownian function is a fractal Brownian function with the same fractal dimension.
> 3. The fractal dimension of a fractal Brownian function is invariant over transformations of scale. (20:664-665)

The first of these properties assures that any fractal analysis performed on the two dimensional representation (using present day input sensors) of the scene stored in the computer hardware is valid, since none of the fractal dimension information is lost by acquiring an image rather than working with the object itself. It also indicates this form of scene analysis may solve the pattern recognition aspect problem. The second of these properties predicts that fractal analysis

may be useful in dealing with the problems of translation and rotation, since both are linear operations. The final property obviously indicates that fractal analysis will yield useful pattern recognition results independent of scale. Pentland has in fact demonstrated this final property over an eight to one scale range (20:668).

Plan of Attack

There was no previous AFIT research in this area, so all of the initial groundwork had to be laid. This included an investigation of fractals in general, algorithms to determine fractal dimension, segmentation techniques that used the fractal dimension, and the generation of software to accomplish these fundamental tasks. The next step was to apply these algorithms to natural images and investigate the performance of the segmenter. The final effort was to apply the refined algorithm to realistic Air Force images to investigate its performance.

Goals and Scope

The original goal of this research was to find a fractal kernel which would describe segmented regions of an image of interest. The end result would have been a "Fractal Transform", analogous to other familiar transforms (such as the Fourier transform), possibly rule-based in operation. This goal, however, proved to be unrealistic. The goals which were finally established can be divided into three broad areas. The first was to generate a appropriate fractal decomposition

of an initial set of images which would be used as standards. This yielded algorithms potentially capable of segmentation. The second goal was to actually segment natural images in a robust manner. The final goal was to examine typical images of Air Force interest to investigate the performance of the algorithm in realistic image environments. The scope of the research was limited to eight images consisting of three standard images, three images of natural scenes with trees, mountains, etc., and two images containing possible military targets. Each of these images was acquired using the video digitizer described below. Appendix A contains each of the images used.

Evaluation Criteria

The reference for evaluating success or failure in this effort was the human visual system, specifically the author's visual system. The achievement of each of the above goals, and each algorithm's relative merits, was determined strictly on the basis of the following questions: "Does this specific technique appear (visually) to work?", and, "How does it compare to the other techniques (visually)?"

Support Equipment

The primary equipment required for this effort was a MicroVAX II AI workstation manufactured by Digital Equipment Corporation (DEC). This workstation was running under MicroVMS 4.4 and was equipped with nine megabytes of main memory, two 71 megabyte hard disk drives, the

GPX color upgrade from DEC, a FORTRAN compiler (VAX FORTRAN 4.0), and a LISP interpreter/compiler (VAX LISP 2.0). There was also an occasional need for the use of the video digitizer that was supported by another DEC MicroVAX II, which was also running under MicroVMS 4.4. Both machines were linked for communication purposes via the DECNET at AFIT.

The digitizer board was a FG-100 series single board image processor manufactured by Imaging Technology Incorporated and was used to digitize images with eight bits of gray value resolution (256 gray levels) per picture element (pixel). The monochrome images were captured and stored as files by the ITEX 100 software library of image processing subroutines supplied by Imaging Technology Incorporated. A DAGE model 650 video camera equipped with a Canon 6x18, 18-108mm, f/3.5 TV zoom lens was used as the source of video input to the digitizer.

Conclusion

Pattern recognition is an inherently difficult task for a machine to perform at the present time, largely due to the inability to succinctly model real world images in a mathematically tractable form. Compounding this are the problems of translation, rotation, scale, aspect changes, and noise whose solution must be incorporated into the recognition algorithm. Fractals appear to be particularly well suited to modelling natural scenes. The applicability and potential of this form of analysis to pattern recognition has already been demonstrated. The thrust of the author's research was to first verify portions of previous fractal analysis, and second, to extend the application of this analysis to real world scenes of interest to the Air Force.

II. Fractals

Overview

This chapter is intended to provide the necessary background in fractal geometry to grasp the fundamentals of the reasoning behind the algorithms that were used in this research. A general introduction to fractals precedes a discussion of the two broadly classed types, deterministic fractals and random fractals. The various concepts embodied in random fractals are explored with an eye to those which are particularly relevant to the algorithms discussed in chapter three.

An Introduction to Fractals

It has only been recently that researchers have come to realize that fractal geometry is a useful tool for modelling the natural world around us. Yet, as Mandelbrot has pointed out, fractals do in fact succinctly model many natural phenomenon (14). These include, but are by no means limited to, coastlines (14:25-33), galactic clusters (14:84-96), craters (14:301-309), turbulence (14:97-105), flesh (14:147-150), linguistics (14:341-348), trees (14:151-165), meteorology (14:461), earthquakes (14:461), river discharges (14:247-255), and economics (14:335-340). The fractal dimension of a surface is also an excellent indicator of its perceived roughness (20:662; 21:254).

The standard connotations associated with dimension can be attributed to Euclid (circa 300 B.C.) who began his Book I (plane geometry) and Book XI (spatial geometry) with some of the following fundamental definitions:

1. A point is that which has no part.
2. A line is breadthless length.
3. The extremities of a line are points.
4. A surface is that which has length and breadth only.
5. The extremities of a surface are lines.
6. A solid is that which has length, breadth, and depth.
7. An extremity of a solid is a surface. (14:409)

A more general view of dimension was formulated by Hausdorff and Besicovitch. Hausdorff's contribution was the use of test functions (14:364) of the form $h(x) = g(d)x^d$ to place a mathematical measure on some object S. For instance, consider the area of a circle and its radial measure. In this case, the object S would be the circle, h(x) would be the area, g(d) a constant (pi), x the radius, and d equal to two. For Euclidean objects, the "d" in any Hausdorff test function will be an integer equal to the object's topological dimension. Besicovitch extended this concept (14:364) to the case where d is not an integer, and S is not a standard shape. Besicovitch went on to show that "for every set S there exists a real value D [corresponding to the d in Hausdorff's test function equation] such that the d-measure is infinite for d<D and vanishes for d>D" (14:364). It is this D that is the fractal dimension of the object in question.

As briefly discussed in chapter one, a fractal's dimension exceeds the topological dimension. But what does "fractal" mean?

> I coined fractal from the Latin adjective fractus. The corresponding Latin verb frangere means "to break:" to create irregular fragments. It is therefore sensible -- and how appropriate for our needs! -- that, in addition to "fragmented"

(as in fraction or refraction) fractus should also mean "irregular," both meanings being preserved in fragment.

The proper pronunciation is frac' tal, the stress being placed as in frac' tion. (14:4)

So fractals are irregular, fragmented, and usually of noninteger dimension. In fact, sets of integer dimension may or may not be fractal, but every set that has noninteger dimension is assuredly a fractal set (14:15).

A better understanding of this fractional dimension can be had by examining the concept of self-similiarity. "When each piece of a shape is geometrically similar to the whole, both the shape and the cascade that generate it are called self-similar" (14:34). Self-similarity is the foundation upon which all fractals are built, whether in a strict sense as is the case for deterministic fractals, or in a statistical sense in the case of random fractals. This property of self-similarity exists at any scale (theoretically), and for this reason fractals are often referred to as being scaling (24:4). This scaling property does not hold over all values in practical applications, but does hold over a wide enough range to be extremely useful. This range varies with the application; in computer generated graphics it can be as high as 10^6 to 10^8 (24), whereas in image data processing it may only be as high as four to ten (20:668). Typically the range is constrained by the field of view of the input sensor, the resolution of the input sensor, or the resolution of the output device, although other factors may contribute.

Another closely related concept is that of similarity dimension. As some mathematicians might interpret it, the similarity dimension and the Hausdorff Besicovitch dimension may differ, but in the instances

where they agree, it is also the fractal dimension (14:37). This similarity dimension can be calculated from the equation $Nr^D = 1$, or equivalently:

$$D = \log N / \log (1/r) \qquad (1)$$

where

N = the number of self-similar parts
r = the ratio of the part to the whole
D = the similarity dimension

Consider how this applies to Figure 1. In the case of the self-similar line segment, N is five, 1/r is five, and D is log(5) / log(5), or obviously one. In the case of the self-similar triadic Koch curve, attributed to von Koch 1904 (14:35), N is 4, 1/r is 3, and D is log(4) / log(3), or approximately 1.2618. This is in fact an instance where the similarity dimension is the fractal (Hausdorff Besicovitch) dimension. The following observation by Voss may provide additional insight:

> As D increases from one toward two the resulting "curves" progress from being "line-like" to "filling" much of the plane. Indeed, the limit [as] D [approaches] two gives a Peano or "space-filling" curve. The fractal dimension D, thus, provides a quantitative measure of wiggliness of the curves. Although these von Koch curves have fractal dimensions between one and two, they remain a "curve" with a topological dimension of one. The removal of a single point cuts the curve in two pieces. (24:6)

Deterministic Fractals

With minor exceptions, the bulk of the groundwork necessary for understanding deterministic fractals was established in the introduction. "The mathematical principle behind the generation of these [deterministic] fractal shapes involves the iteration of

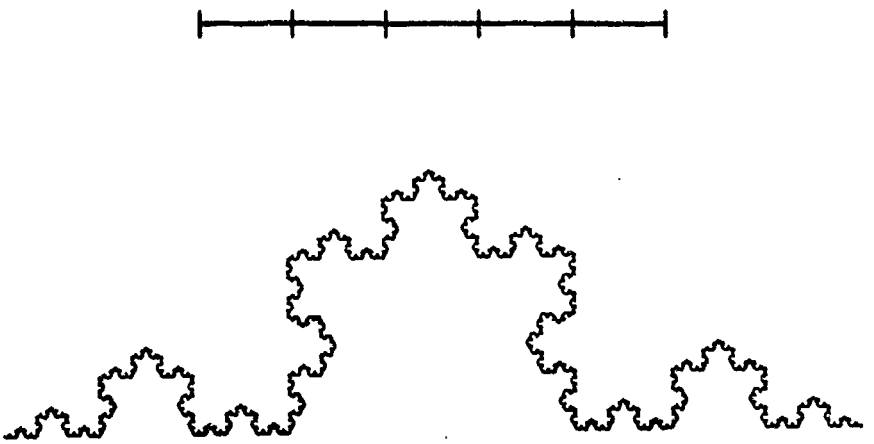


Figure 1. Standard vs Fractal Self-Similarity (14:44)

algebraic transformations" (16:62), specifically of mapping Euclidean space onto itself. This concept of mapping is the underlying principle used in the computer generation of deterministic fractals. As might be assumed, recursive programming is often the method used to implement these mappings (23). To this point, the only mapping considered was in the real plane, but other classes of deterministic fractals exist that are generated by mappings in the complex plane. These include what are known as Julia sets and the Mandelbrot set (14:180-192; 19; 23), with transformations typically of the form $f(z) = z^2 - u$ where u is a complex number and z may or may not be. Figure 2 is an illustration of an example from the Mandelbrot set, which uses a mapping in the complex plane. Note that the fractal boundary of this figure is quite different from the triadic Koch curve of Figure 1. These transformations, whether in the real or complex plane, are applied as

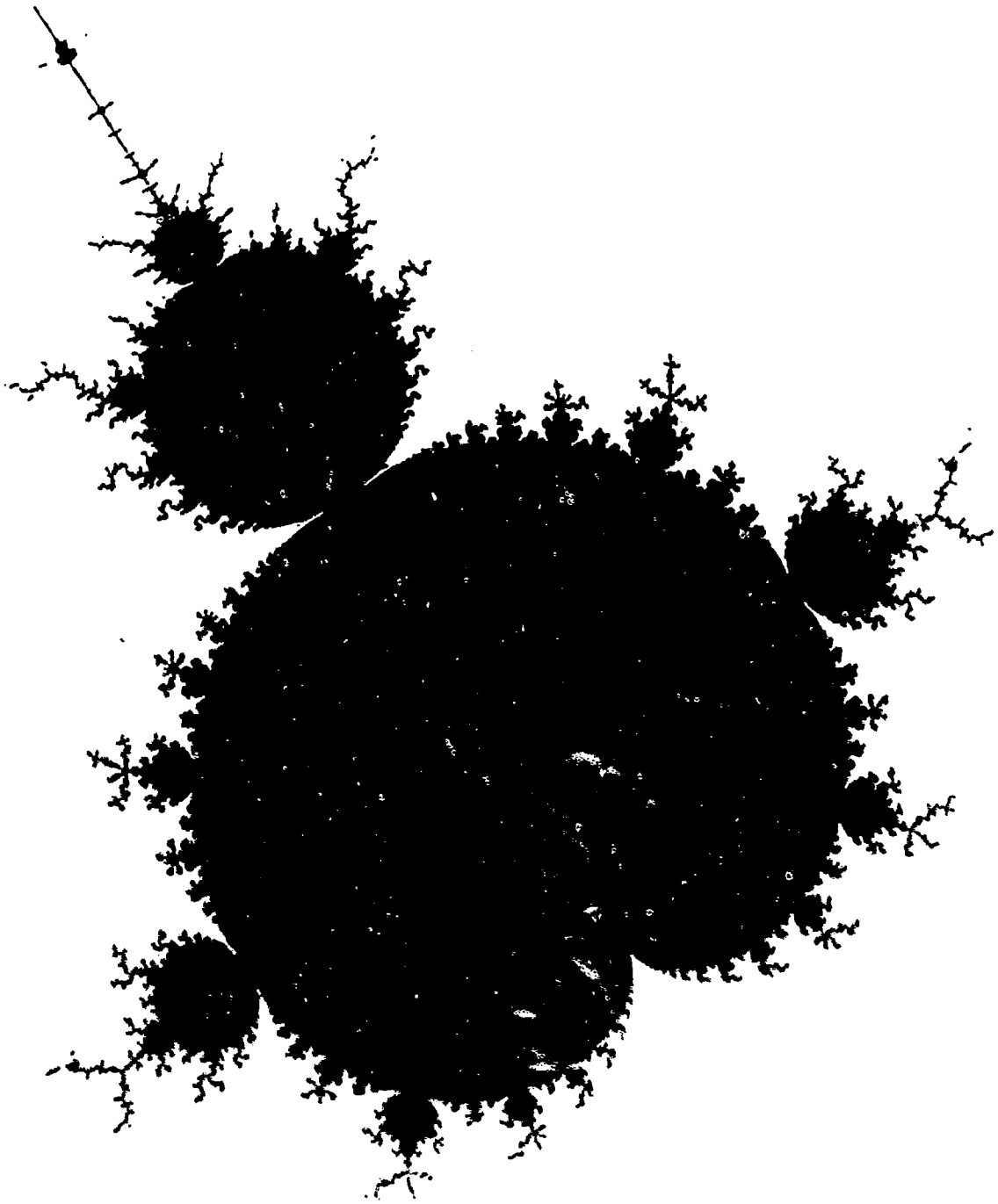


Figure 2. The Mandelbrot Set (3)

shown in Eq (2) (16:62). Note how this equation clearly illustrates the inherently recursive nature of deterministic fractal generation.

$$x_0 = x,\ x_1 = T(x),\ x_2 = T[T(x)],\ x_3 = T\{T[T(x)]\},\ \ldots \qquad (2)$$

where

x_0 = the initial point in the plane (real or complex)
T = the functional form of the transformation

Random Fractals

The primary difference in deterministic and random fractals is in the interpretation of self-similarity. Deterministic fractals look exactly the same at all scales ; random fractals do not. Each smaller portion will look like, but not exactly like, the larger object from which it came (24:7). The fractal dimension of random fractals can still be calculated from Eq (1) and formulations similar to it, but now the D calculated is an average (24:7). To add another element of complication, random fractals may also exhibit statistical self-affinity, rather than the simpler case of statistical self-similarity. Under an affine transformation, the mapping described by Eq (2) can now have different scaling factors for each of the Euclidean coordinates (2:17). Thus, the X coordinates may be scaled differently than the Y coordinates, which in turn may be scaled differently than the Z coordinates, for example. Compare this then to the simpler case of statistical self-similarity, where each of the coordinates would be scaled equally. The fractals formed by such an affine transformation are properly known as self-affine, rather than self-similar, but the distinction is not always clearly made in the literature. The term

self-similar is often used in a generic sense that incorporates self-affinity (14:349).

As pointed out in chapter one, the fractal set of interest in this research is that known as fractional (or fractal) Brownian motion (fBm), and it is this subject which will occupy the remaining material in this chapter. Strictly speaking, fBm are statistically self-affine, which can cause algorithms for calculating fractal dimension to produce ambiguous results depending on the measurement technique used (24:19,20). The reason is simple: the scaling may vary with each Euclidean dimension, and it is inevitably these scalings which are used in some fashion to calculate the fractal dimension.

Fractional Brownian motion has its basis in ordinary Brownian motion, first described by R. Brown around 1827 while observing particles in a liquid suspension under a microscope (23). This erratic movement can be modelled by a random process whose increments are Gaussian distributed (14:351; 23; 24:16;). Additionally, the mean square of these increments has a variance proportional to the difference in the independant variable (23; 23416). In the case where the independent variable is time (t), this can be more succinctly stated by Eq (3):

$$\langle |X(t_2) - X(t_1)|^2 \rangle = K\,|t_2 - t_1| \qquad (3)$$

where

$\langle\ \rangle$ denotes statistical expectation
K denotes a proportionality constant

Figure 3 is a typical sample of one dimensional Brownian motion.

Eq (3) can be generalized to (24:16)

$$\langle |X(t_2) - X(t_1)|^2 \rangle = K\,|t_2 - t_1|^{2H} \qquad (4)$$

where

< > denotes statistical expectation
K = a proportionality constant
H = the Hurst exponent, in interval [0,1] (14:249)

Eq (4) is a mathematical generalization of Brownian motion in one dimension that will model fBm. When H is equal to 1/2, the formulation is simply that of ordinary Brownian motion. The cases of greatest interest are for H in the intervals [0,1/2) and (1/2,1]. In the former case, the increments of X(t) are positively correlated; and in the latter case, negatively correlated (24:16). The Hurst exponent is directly related to the fractal dimension of the process being modelled by the relation (24:24)

$$D = E + 1 - H \qquad (5)$$

where

D = the fractal dimension
E = the Euclidean dimension
H = the Hurst exponent

The effect of H on a typical trace can be seen in Figure 4. By examining Figure 4 and knowing that fractal dimension increases with perceived roughness, it should be obvious to the reader that a larger H corresponds to a smaller fractal dimension, which is precisely what Eq (5) indicates.

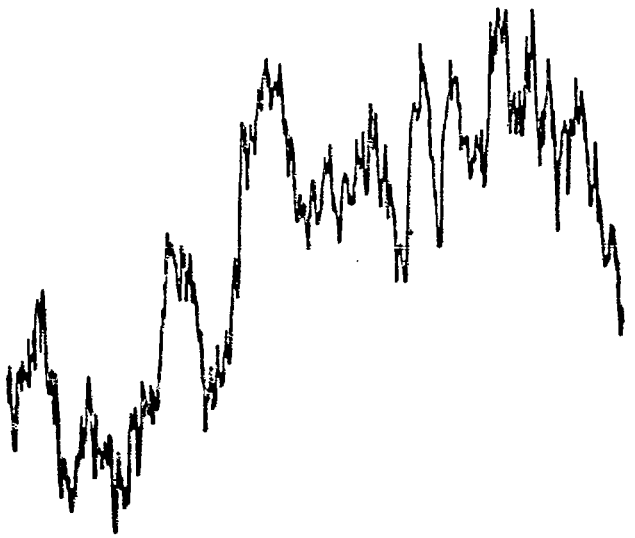


Figure 3. One Dimensional Brownian Motion (23)

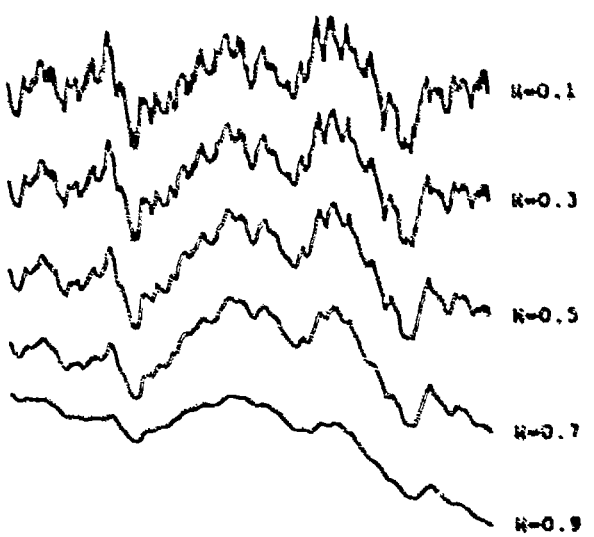


Figure 4. The Effect of H on fBm (23)

Conclusion

When it comes to fractals, there really is "no new thing under the sun. Is there any thing whereof it may be said, See, this is new? it hath been already of old time, which was before us" (4). As has been pointed out, objects which can be modelled well by fractal geometry pervade the world around us. An understanding of this geometry, which is new, moves it from a past of captivating mathematical oddities to the realm of a truly useful analytical tool in the present.

III. Experimental Method

Overview

In the discussion that follows, two techniques commonly seen in the literature for determining the fractal dimension, D, will be reviewed, as well as a original hybrid technique investigated by the author. The two types of threshold processing employed and the reasoning behind them will also be discussed. The actual application of these algorithms is discussed within each sub-section. Appendix B contains the actual programs used.

Dimension Measurement

A wide variety of techniques for measuring D are discussed in the literature. Two of these are used by the author, and are apparently the most commonly used; they are the methods employing the $1/f^x$ nature of the power spectral density (PSD) of the Fourier transform of the signal of interest, and what is known as box dimension. Both of these techniques are particularly well suited to implementation in FORTRAN (25), largely due to its array handling capabilities and the highly optimizing nature of the compiler. The final original technique used does not calculate D specifically, but does borrow from concepts used in the computer generation of fBm. It was investigated simply because of its ease of implementation and relatively fast computation time.

In each case, the general approach used was to divide the 512 X 480 pixel image into 8 X 8 pixel regions, which will be referred to as sub-images throughout the rest of this discussion, for subsequent processing. This size of sub-image yielded a segmentation resolution of 64 X 60 regions. The number of regions could have been increased, of course, by simply choosing a smaller size sub-image, but the other consideration was the Gaussian nature of the mean square increments of fBm. It was the author's conviction that 64 pixel samples should be sufficient to insure (within a reasonable margin of error) that the increments between intensity values were in fact Gaussian in nature, but this was never investigated. The three fractal dimension measurement techniques that follow were applied to each of the 8 X 8 pixel sub-images.

PSD Rolloff. The PSD of fBm is proportional to $1/f^x$ (24:24), where x can be related to the Hurst exponent H by (24:24; 23):

$$x = 2H + 1 \tag{6}$$

(For a development of this relation, see Appendix C.) Substituting for H in Eq (5), yields a relationship between the spectral rolloff x, the Euclidean dimension E, and the fractal dimension D:

$$D = E + (3 - x) / 2 \tag{7}$$

This technique was implemented by making use of the Wiener-Khintchine relation (6:209), which simply states that the magnitude squared of the Fourier transform of a signal (i.e. - the PSD) is a Fourier transform pair with the autocorrelation of the original signal. It was felt that a simple X and Y autocorrelation would be advantageous from the viewpoint of computation time versus implementing

an 8 X 8 two dimensional fast Fourier transform from which the PSD could then be calculated. Thus, the PSD was determined using autocorrelations in the X and Y directions (with no offset in Y or X, respectively) for each of the 8 X 8 sub-images, which were then transformed with a discrete cosine transform to obtain the spectral coefficients. The spectral rolloffs in X and Y were then calculated by simply arithmetically averaging the slope in the seven harmonics present (the DC term was ignored), and these two were then arithmetically averaged to yield a rolloff that was characteristic of the entire 8 X 8 pixel sub-image. This was then used to calculate the fractal dimension, D, via Eq (7). The Euclidean dimension E was assumed to be three.

Box Dimension. Consider an image comprised of a set of m points, S, in E-dimensional Euclidean space, which a priori can be considered to be equiprobable, and with associated probablility measure P(m,L). "P(m,L) is the probability that there are m points within an E-cube of size L centered about an arbitrary point [m] in S" (Voss:24). This probability measure P(m,L) is then normalized such that the summation over all the points m in S is equal to one for each value of L (24:25). The mass dimension (or box dimension) is then simply the first moment with respect to m of P(m,L) (24:25). This is used to calculate the fractal dimension D from the relation (24:25)

$$M(L) = KL^{D} \tag{8}$$

where

M(L) = the mass dimension for a particular value of box size L
K = a constant of proportionality
D = the fractal dimension

This technique was implemented by first normalizing all pixel gray values in the 8 X 8 pixel sub-image to the interval [1,8]. This can be thought of as yielding a cube 7 units on a side in the image intensity space. This also functioned as a form of energy normalization. P(m,L) was initialized by considering it as a discrete distribution of 64 bins, each of which represented the number of points within the cube, for each value of L. There were 64 bins, because for large enough values of L, the cube would contain all of the image intensity points. The bins were then filled by considering each point in the sub-image to be the center of a cube ranging from an L of two units on a side up to an L of seven units on a side, and counting the number of points within that cube. This yielded the six distributions P(m,2), P(m,3), ..., P(m,7); each of which was then normalized to one, so that it could be treated as a discrete probability density function. The first moment with respect to m was then calculated for each P(m,L) yielding M(2), M(3), ..., M(7). A value proportional to the fractal dimension D of the sub-image was then calculated using Eq (8) by calculating the average slope of a log-log mapping of L vs. M(L).

Hybrid Brown. This technique has no precedent in the literature, although it borrows from concepts used in the computer generation of fBm for terrain simulations via one dimensional Brownian functions (see reference (23)). In this method, the 8 X 8 pixel sub-image intensity surface is normalized in X, Y, and Z to [0,1]. Note that this mapping is substantially different from the past techniques in which the X and Y coordinates retained their mapping on the interval [1,8], and their values only changed in integer intervals. The sub-image array is then

sorted based on the intensity value (the Z coordinate) from the lowest value to the highest while simultaneously storing the path taken through the array. This path is then treated as three-space Brownian motion, and the increments from point to point in X, Y, and Z as the path is traced are found. This yields 63 increments in each of the coordinates X, Y, and Z. For each of the coordinates, the mean square of these increments is then calculated. Since the assumption is that the path is indeed Brownian, each coordinate's mean square value can be assumed to be independent from the other two. The mean square that characterizes the 8 X 8 pixel sub-image simply becomes the addition of the three previously calculated values. The Hurst exponent is then calculated via Eq (4), where the independent variable interval is assumed to be 1/63. The interval chosen might appear to be somewhat arbitrary, but there are 63 increments that are an artifact of the 64 element sub-image, and it seemed fitting that an interval of 1/63 was appropriate. It is important to note that what is being calculated in this technique is not the fractal dimension of the sub-image; it is merely a statistical calculator based on fractal Brownian motion. It was investigated because of the relatively quick computation time involved, and to compare its performance in its ability to segment to the known fractal dimension calculators.

Threshold Techniques

The natural problem that arises once the calculators finish is how to organize the data into some graphically meaningful output. Two approaches were tested. In each case, the fractal-based measurement

was used to determine which color (or equivalently, intensity) bin a sub-image would be mapped to. The sub-images were mapped to four different values.

Linear. This method was implemented by scaling the working array, which was the 64 X 60 array containing the output of the calculators, to range in value from zero to four. Thresholds were then set at one, two, and three, and each of the corresponding elements in the array was then assigned a value based on which of the thresholds the region was between.

Adaptive. This method used histogram manipulation to determine where the minima were in a 256 bin histogram of sub-image values, which in turn were used to set the array thresholds. It was implemented by first scaling the working array to values in the interval [0,256]. The number of values in the working array in the interval (255,256] was then placed into bin 255, the number of values in the interval (254,255] into bin 254, etc. The histogram was then numerically integrated by simply replacing each bin value by the running total sum, and an average slope for the entire integrated histogram was then calculated. The slope from bin to bin was then compared to the average. Any slope that was greater than the average was considered to be due to either a peak or the rising edge of a peak in the original histogram. This approach allowed the computer to search for maxima, and indirectly, the minima as well. The search proceeded from the largest intensity values downward until three minima were found. These bin values between peaks then corresponded to the three thresholds that were set for assigning working array elements to their segmented

value. These threshold values were simply an artifact of the natural clusters of fractal dimension. The hope was, of course, that the clusters corresponded to what the human eye would have segmented in the original image.

Modified Linear. This method can be thought of as scaling the working array into eight levels. The lower five levels were all mapped to black, the next higher level to a dark gray, the next higher level to a light gray, and the upper level to white. This is still a linear method for establishing thresholds, it simply ignores the lower four values. This technique emphasises variations in the upper half of the values in the working array histogram, and essentially ignores those variations in the lower half.

Conclusion

The implemenation of a fractal dimension calculator is not intuitively obvious. Two standard techniques were presented along with a discussion of the specific implementation used by the author. Additionally, an original technique and the reasoning behind it was discussed. Finally, the three methods used for establishing thresholds and the implementation of each was discussed.

IV. Results

Overview

This chapter examines the success of the three techniques for calculating fractal dimension as applied to the segmentation of the eight images mentioned in chapter one. Each technique is discussed separately with accompanying examples. Additional results can be found in Appendix D.

Output Comments

All three methods of establishing thresholds were found to be inadequate. The adaptive threshold never successfully set the thresholds as it was hoped. This was perhaps due to the jaggedness of the histogram profiles. The linear method was in almost every case an absolute failure. The most successful of the thresholding methods was the modified linear method. After examining histograms of the distribution of fractal dimension within various scenes, the reason became obvious. It appears that the fractal dimension calculators do in fact differentiate between different types of natural objects, as evidenced by the clusters observed in the histograms. The problem with this is that the clusters vary in number, location, and with the fractal dimension calculator used. This was precisely the reason that the linear method failed completely. Additionally, these clusters tended to be within the upper half of the histogram which explains why

the modified linear threshold technique met with some measure of success. All graphic results were obtained using the modified linear threshold technique.

PSD Rolloff

This technique appears to work from histogram distributions and is relatively fast, typically processing an image in about two minutes. This technique does not separate the clusters within the histogram enough to yield visually pleasing results with the present threshold approaches. This can be readily seen by considering Figure 6, which is the result of the simple input image of Figure 5, as opposed to the result of Figure 7, which is the result of the more complex image of Figure 8. Note that intuitively Figure 5 would contain one prominent cluster in the histogram, but Figure 7 would contain several.

Box Dimension

With the present threshold methods, this technique seems to be the most effective in segmenting an image, but is very slow, typically processing an image in 45 minutes or more. Figure 9 is an example of the effectiveness of this approach on the image shown in Figure 8. This technique tends to spread out the locations of the clusters within the histogram, and thus is more likely to separate them with the fixed thresholds established by the modified linear method of thresholding. This technique was also used without first scaling the intensity values, yielding virtually identical results.

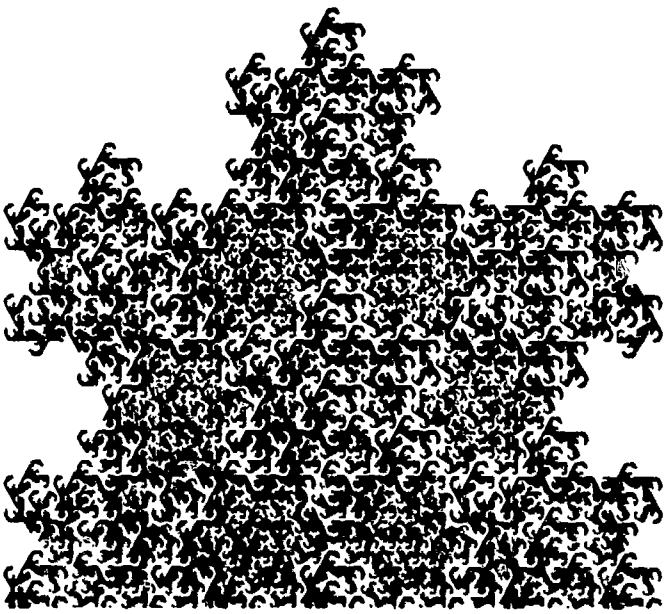


Figure 5. Input Image (14:68)

Figure 6. Segmentation via PSD Rolloff

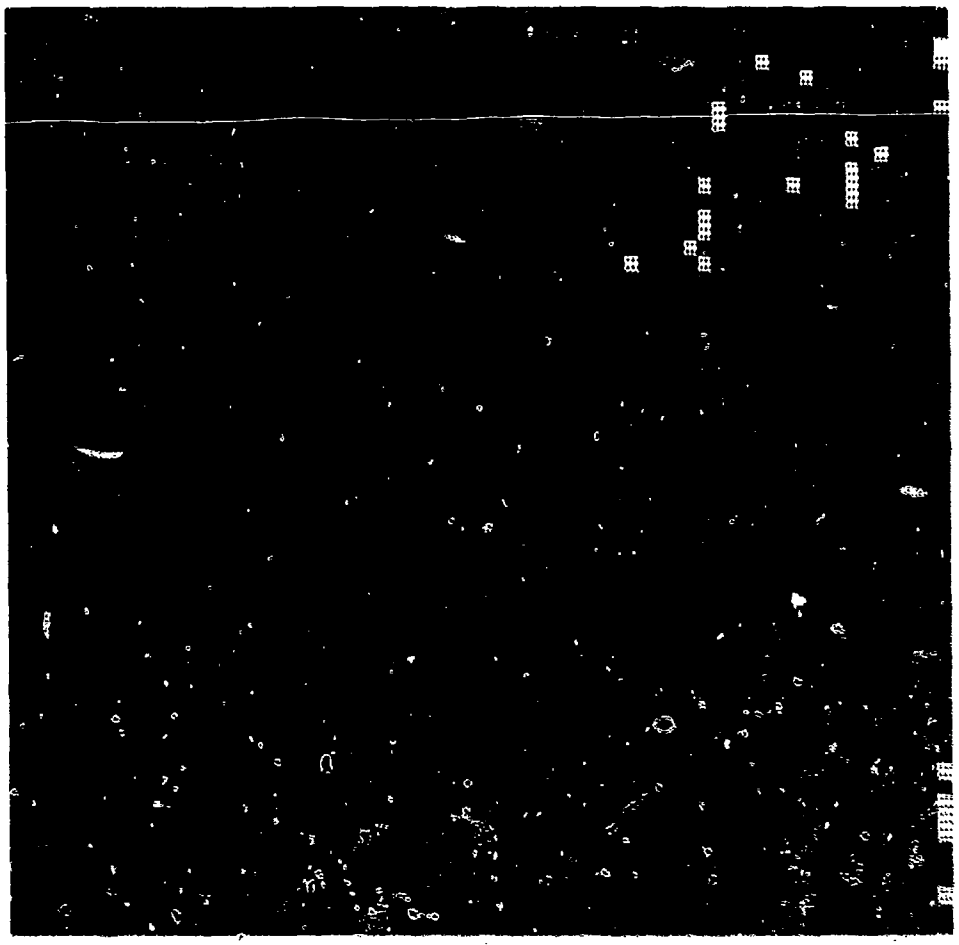


Figure 7. Segmentation via PSD Rolloff

Figure 8. Input Image (18)

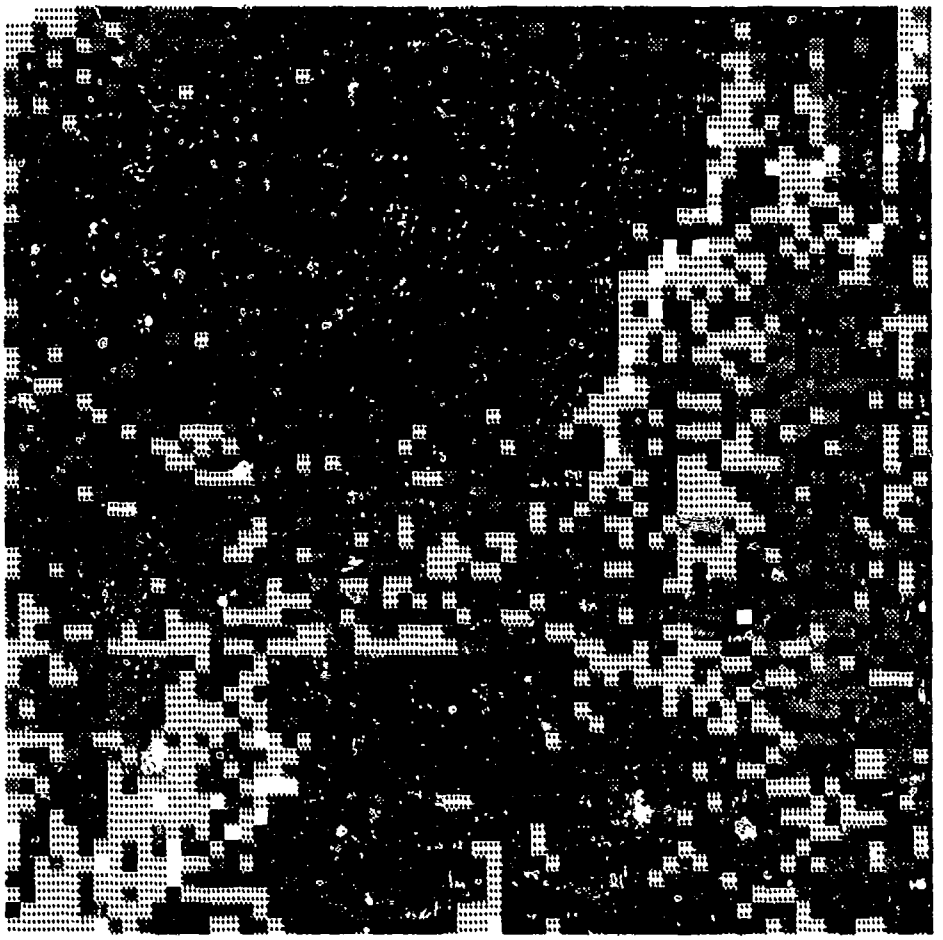


Figure 9. Segmentation via Box Dimension

Hybrid Brown

This technique also appears to work, although like the PSD technique, it does not separate the clusters within the fractal dimension histogram as well as the box dimension technique. It is also relatively fast, typically processing an image in about two minutes. Figures 10 and 11 are examples of the output of this technique with Figures 5 and 8 as input images, respectively. Note the similarity of these results to that of the PSD rolloff technique.

Figure 10. Segmentation via Hybrid Brown

Figure 11. Segmentation via Hybrid Brown

V. Conclusions and Recommendations

Conclusions

Some general conclusions can be drawn from the results presented in the previous chapter. One of the most important is that the use of modelling natural objects using fractal geometry shows great promise in image processing. The use of fractal geometry to mathematically model regions of interest in an image is a fundamentally different approach than the Euclidean models and Euclidean-extracted measurement spaces used in pattern recognition to date. Of the three techniques investigated, all three appear to work to some extent. The PSD rolloff technique and hybrid Brown technique both fail in cluttered images, apparently due to an inability at the present to adaptively set the necessary thresholds to provide robust segmentation. All of the techniques appear capable of extracting useful image information from noisy (real world) images, the box dimension technique demonstrably so.

Recommendations

The following general areas of investigation appear to be worth pursuing. The foremost area of investigation is that of adaptively setting the thresholds, such that each of the techniques examined in this research would be useful in terms of robust segmentation. Two possible approaches to this might be searching for minima in a curve that has been spline-fit to the histogram profile; or secondly,

correlating a generic peak with the histogram and thresholding the result to determine the location of the maxima (and thus the minima). Secondly, investigate the distribution of intensity increments within each of the processed sub-images. If in fact regions have distributions that are not Gaussian, theory indicates the region is a likely candidate to be part of a man-made object, simply because the model for fBm does not fit the data. More simply stated, natural objects fit the fBm model, man-made objects do not.

Areas of investigation concerning the specific implementation of the software as it presently exists are those of pre-processing and region resolution. Preprocessing the image may aid the dimension calculators in achieving the desired end. No edge enhancement, or any other form of pre-processing, was employed before passing the image to the fractal dimension calculators. Also, is it in fact necessary to use 8 X 8 pixel sub-images? There is probably some optimum size of sub-image that is a good tradeoff between processing speed and processed image resolution.

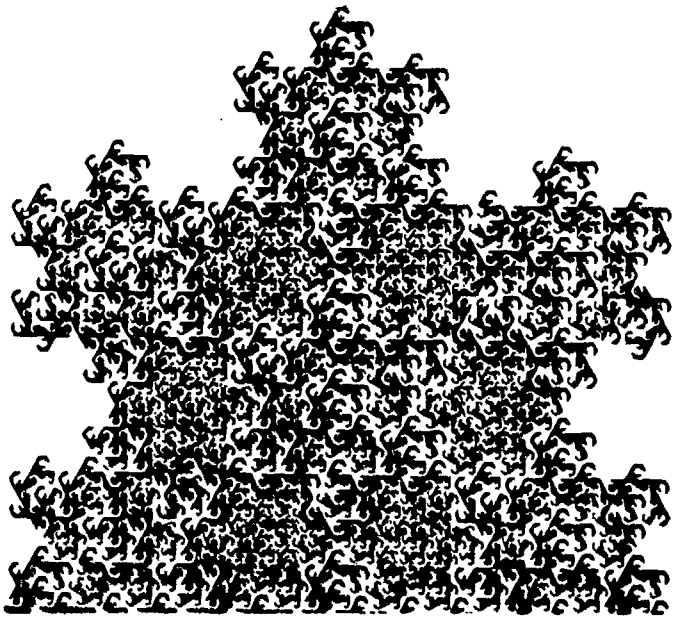
Turning to the original goal of this research, which was to find a fractal transform, one area of current research has been encountered which may further progress on this problem. The general approach would be based on an extension of the work by Barnsley discussed in reference (2). Barnsley's efforts focus on finding a set of graphic kernels (that can be modelled by fractal geometry) which can be used to reconstruct an entire object. The general approach is to use a fixed set of probabilistic affine transformations known as iterated function set codes to generate this image. The image can be thought of as

closely representing the object in a mean square error sense. If this approach could be automated (it is not presently) and applied to an input image that has been partitioned on a grid, it may be possible to find a fractal kernel that represents each partition. The fractal dimension of this kernel can then be compared to the fractal dimension of the entire partition using the calculators developed in this research in an iterative fashion until some predetermined measure of fit is obtained. This approach may or may not work, contingent on whether it is a true assumption that the fractal dimension of the kernel used to generate a partition is equal (or proportional) to the fractal dimension of the partition itself. An issue not discussed by Barnsley is the orthogonality of such fractal kernels, or of any effects that might be caused by discontinuities in the fractal kernel from partition to partition. It is not intuitively clear that any such decomposition would yield orthogonal kernels similar to the harmonics used in a Fourier transform. It is also quite possible that there are in fact discontinuity phenomenon similar to the Gibb's phenomenon exhibited by discontinuities in a signal which is Fourier transformed.

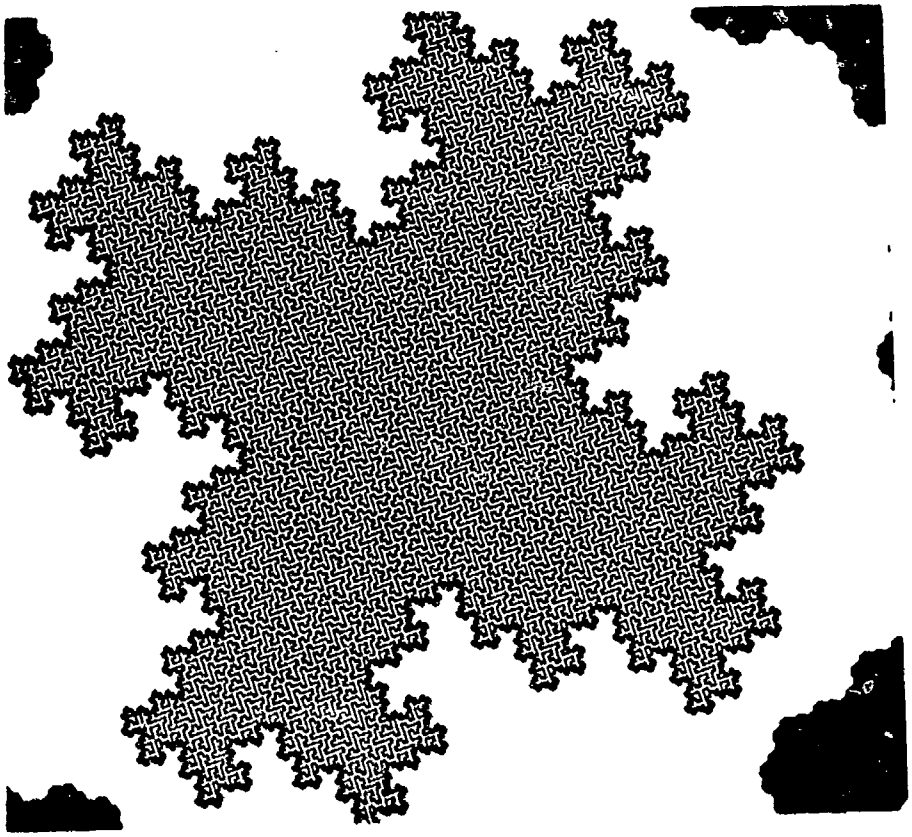
The final recommendation is specifically targeted to anyone pursuing future work with fractal geometry. The author considers it a necessity that at a minimum references (5), (20), (23), and (24) be studied and thoroughly understood. These references will provide a solid foundation for further study in the application of fractal geometry to image processing.

Appendix A: Input Images

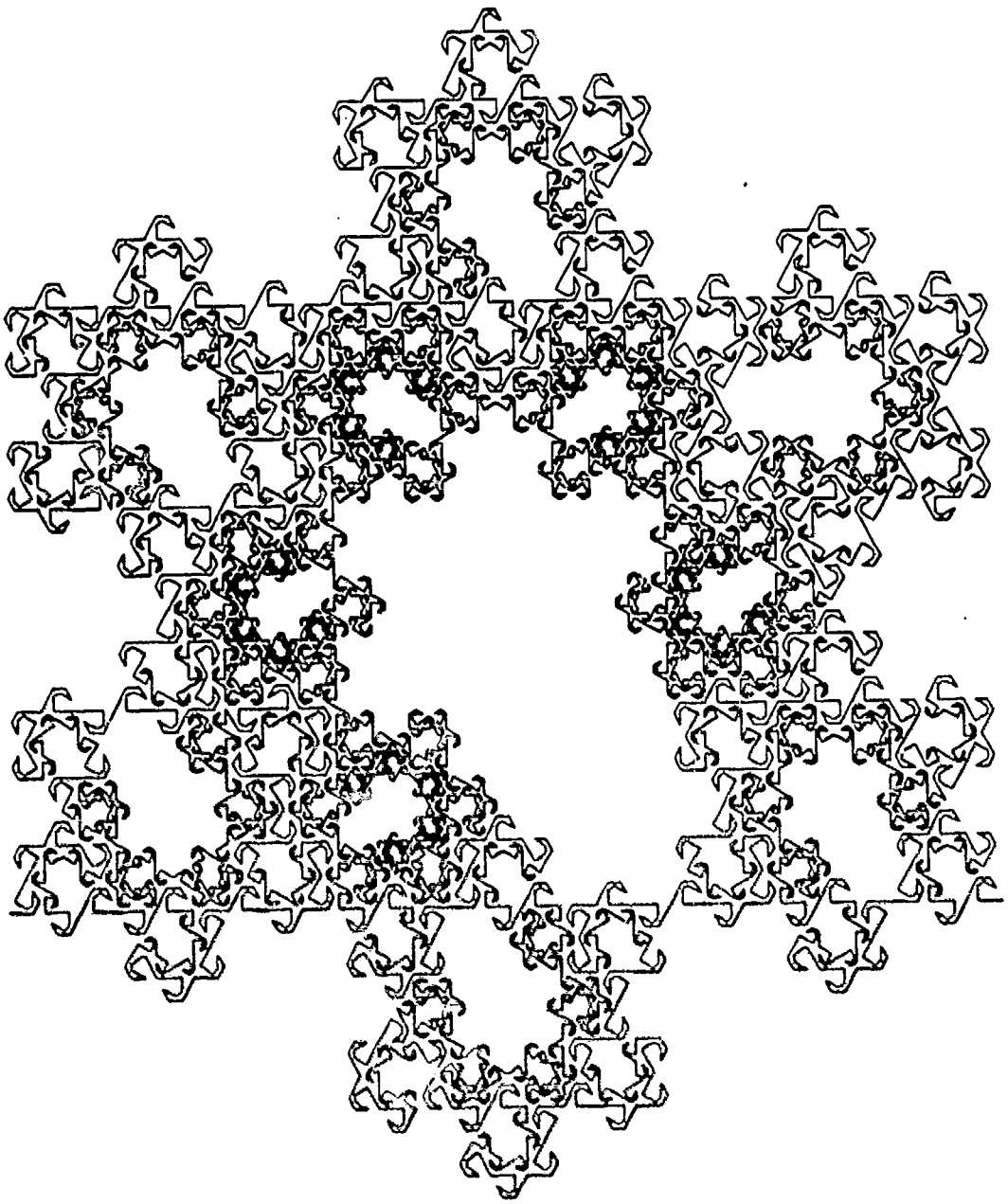
This appendix simply displays in numerical order the images that were used in this investigation. The file in which the image was stored is given for each. These images were acquired with the digitizer and video camera described in chapter one. The files were converted into a format readable by the FORTRAN compiler using the ITEX_TO_RMS program written by Capt. Richard Roberts.



1.SSI (14:68)

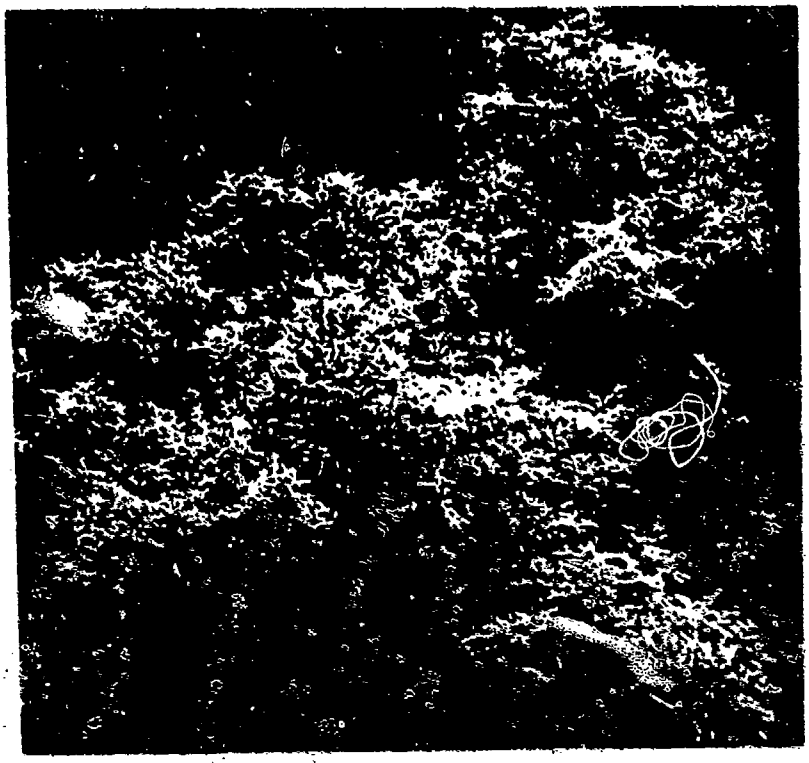


2.SSI (14:49)



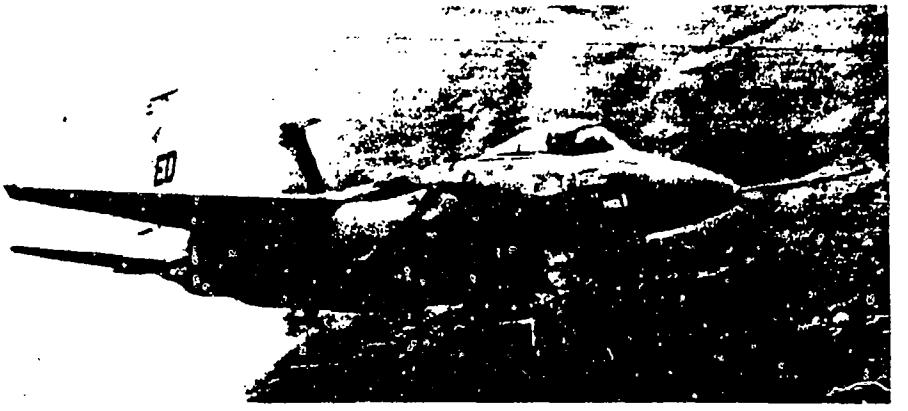
3.SSI (14:146)

4.SSI (18)



5.SSI (17)

6.SSI (2)



oto:ype McDonnell Douglas F-15C fitted with conformal fuel tanks and with a LANTIRN navigational pod under its port engine air intake

l external stores stations remain available with the CFTs use, and McDonnell Douglas has developed for the F-15 iew weapon attachment system which can extend the erating radius with large external loads by up to 40 per it. Known as tangential carriage, it involves the instal- ion of rows of stub pylons on the lower corner and

MSIP improvements include a tactical electronic system consisting of a Northrop Enhanced ALQ-135 internal countermeasures system, Loral ALR-56C radar warning receiver, Tracor ALE-45 chaff disperser, and Magnavox electronic warfare warning system. A further $274·4 million contract was received in December 1983. Flight testing of

F15.SSI (9:450)

F16XL.SSI (9:412)

Appendix B: Software

This appendix is simply a listing of the various routines that were used to investigate image processing using fractal-based algorithms. Included are the programs that were used to convert the original output of the ITEX digitizer board (the only C program), load the image files into the VAX FORTRAN environment, display images on the GPX hardware, and all of the processing routines. Many thanks to Capt. Richard Roberts and Capt. Dennis Ruck for their help in adapting the ITEX conversion program to suit the author's needs.

```
*
*
*       This subroutine calculates the rolloff in the PSD of the FFT
*       spectrum by calculating the autocorrelation in X and Y.
*
*
        SUBROUTINE AUTO(ARRAY,SUB_ARRAY,X_INDEX,Y_INDEX)
*
*       Define variables ...
*
        INTEGER*4 X_INDEX, Y_INDEX
        REAL*8 SUB_ARRAY, CORR_X, CORR_Y
        DIMENSION CORR_X(0:7), CORR_Y(0:7)
        DIMENSION ARRAY(64,60), SUB_ARRAY(8,8)
*
*       Loop over each shift ...
*
        DO 1010 K=0,7
*
*       Zero the correlation vectors ...
*
           CORR_X(K) = 0.0
           CORR_Y(K) = 0.0
*
*       Loop over every element of the sub-image ...
*
           DO 1000 J=1,8
              DO 1000 I=K+1,8
*
*       Autocorrelation across X ...
*
                 CORR_X(K) = CORR_X(K) + SUB_ARRAY(I,J)
     1           * SUB_ARRAY(I-K,J)
1000       CONTINUE
           DO 1010 J=K+1,8
              DO 1010 I=1,8
*
*       Autocorrelation across Y ...
*
                 CORR_Y(K) = CORR_Y(K) + SUB_ARRAY(I,J)
     1           * SUB_ARRAY(I,J-K)
1010       CONTINUE
*
*       Pass the correlation values to the Fourier routine ...
*
        CALL DCT(CORR_X)
        CALL DCT(CORR_Y)
        DO 1020 K=0,7
*
*       Add small value to keep log function from crashing ...
*
           CORR_X(K) = CORR_X(K) + 0.000001
           CORR_Y(K) = CORR_Y(K) + 0.000001
*
*       Take the log of each harmonic ...
*
           CORR_X(K) = DLOG10(CORR_X(K))
           CORR_Y(K) = DLOG10(CORR_Y(K))
1020    CONTINUE
*
*       Calculate the rolloff in the PSD in log-log form ...
*
*       Initialize slope values to zero ...
*
        SLOPE_X = 0.0
        SLOPE_Y = 0.0
*
*       Calculate slope in each increment and add together ...
*
        DO 1030 I=1,6
           SLOPE_X = SLOPE_X + (CORR_X(I) - CORR_X(I+1)) /
     1      (ALOG10(FLOAT(I)) - ALOG10(FLOAT(I+1)))
```

```
         SLOPE_Y = SLOPE_Y + (CORR_Y(I) - CORR_Y(I+1)) /
     1      (ALOG10(FLOAT(I)) - ALOG10(FLOAT(I+1)))
1030  CONTINUE
*
*     Crudely calculate the slope of the line by averaging ...
*
      SLOPE_X = SLOPE_X / 6.0
      SLOPE_Y = SLOPE_Y / 6.0
*
*     Calculate the fractal dimension in X and Y ...
*
      DIM_X = 3.0 + 0.5 * (3.0 + SLOPE_X)
      DIM_Y = 3.0 + 0.5 * (3.0 + SLOPE_Y)
*
*     Average and put result in the work array ...
*
      ARRAY(X_INDEX,Y_INDEX) = (DIM_X + DIM_Y) / 2.0
      RETURN
      END
*
*
*     This subroutine simply swaps two values (used for sorting) ...
*
*
      SUBROUTINE SWAP(X,Y)
*
*     Define variables ...
*
      REAL*8 X, Y, TEMP
*
*     Perform the swap ...
*
      TEMP = X
      X = Y
      Y = TEMP
      RETURN
      END
*
*
*     This subroutine calculates dimensionality by treating the
*     image pixel values as Brownian motion as the "particle"
*     traces out a three space path over ([0,1],[0,1],[0,1]) ...
*
*
      SUBROUTINE BROWN(ARRAY,SUB_ARRAY,X_INDEX,Y_INDEX)
*
*     Define variables ...
*
      LOGICAL*2 ERROR
      INTEGER*4 X_INDEX,Y_INDEX
      REAL*4 ARRAY
      REAL*8 SUB_ARRAY, SORT, DELTA, SUM, SQUARE
      DIMENSION ARRAY(64,60), SUB_ARRAY(8,8), DELTA(63)
      DIMENSION SQUARE(3), SORT(64,3), SUM(3)
*
*     Normalise the array SUB_ARRAY to min = 0.0, max = 1.0 ...
*
      CALL NORM(SUB_ARRAY,ERROR)
*
*     Check error trap on return from NORM ...
*
      IF (ERROR .EQ. .TRUE.) THEN
         ARRAY(X_INDEX,Y_INDEX) = 3.0
         RETURN
      END IF
*
*     Now associate normalised X, Y, and Z  coordinates so
*     that the actual path will be known ...
*
      DO 1000 J=1,8
         DO 1000 I=1,8
            SORT((J-1)*8+I,1) = (I-1) / 7.0
            SORT((J-1)*8+I,2) = (J-1) / 7.0
```

```
            SORT((J-1)*8+I,3) = SUB_ARRAY(I,J)
1000    CONTINUE
*
*       Sort ascending Z values ...
*
        DO 1010 I=1,63
*
*       Initialize Z ...
*
           VALUE = SORT(I,3)
*
*       Test each of the Z values ...
*
           DO 1010 J=I+1,64
              IF(SORT(J,3) .LT. VALUE) THEN
                    VALUE = SORT(J,3)
                    CALL SWAP(SORT(I,1),SORT(J,1))
                    CALL SWAP(SORT(I,2),SORT(J,2))
                    CALL SWAP(SORT(I,3),SORT(J,3))
              END IF
1010    CONTINUE
*
*       Loop through X, Y, Z ...
*
        DO 1020 J=1,3
           SUM(J) = 0.0
           SQUARE(J) = 0.0
*
*       Get the increments ...
*
           DO 1020 I=1,63
              DELTA(I) = SORT(I+1,J) - SORT(I,J)
*
*       Calculate sum and square values for the variance ...
*
              SUM(J) = SUM(J) + DELTA(I)
              SQUARE(J) = SQUARE(J) + DELTA(I) ** 2.0
1020    CONTINUE
*
*       Calculate the variance in X, Y, and Z ...
*
        VAR_X = (63.0 * SQUARE(1) - (SUM(1) ** 2.0)) / 3906.0
        VAR_Y = (63.0 * SQUARE(2) - (SUM(2) ** 2.0)) / 3906.0
        VAR_Z = (63.0 * SQUARE(3) - (SUM(3) ** 2.0)) / 3906.0
*
*       Assuming Brownian motion implies that the variances in
*       X, Y, and Z can be treated as independent Gaussian R.V.'s
*
        VAR = VAR_X + VAR_Y + VAR_Z
*
*       Now calculate H (arbitrarily assumes delta T is 1/63) ...
*
        H = ALOG10(VAR) / (2.0 * ALOG10(63.0))
*
*       And place in array ...
*
        ARRAY(X_INDEX, Y_INDEX) = 3.0 - ABS(H)
        RETURN
        END
*
*
*       This subroutine converts the work array into an image array.
*
*
        SUBROUTINE CONVERT(IN,OUT)
        BYTE OUT
        REAL*4 IN
        DIMENSION IN(64,60), OUT(512,480)
*
*       Place into 8 X 8 blocks in the array OUT ...
*
        DO 1000 J=1,60
           DO 1000 I=1,64
```

```
            DO 1000 K=1,8
                  DO 1000 L=1,8
                     OUT((I-1)*8+L,(J-1)*8+K) = (IN(I,J) - 128)
1000    CONTINUE
        RETURN
        END
*
*
*       This subroutine performs a discrete cosine transform on the
*       autocorrelation to be used in calculating the rolloff of the
*       power spectral density.
*
*
        SUBROUTINE DCT(IN)
        REAL*8 IN, PSD
        DIMENSION IN(0:7), PSD(7)
*
*       Initialize PSD ...
*
        DO 1000 I=1,7
           PSD(I) = 0.0
1000    CONTINUE
*
*       Calculate the Fourier transform ...
*
        DO 1010 K=1,7
           DO 1010 N=0,7
              PSD(K) = PSD(K) + IN(N) * DCOSD(DBLE(N*K)*45.0)
1010    CONTINUE
*
*       Load result back into the array to be passed ...
*
        DO 1020 I=1,7
           IN(I) = PSD(I)
1020    CONTINUE
        RETURN
        END
*
*
*       This subroutine images an array of bytes in a writing
*       mode that uses the byte value to determine the color
*       (or shade of gray) of  the pixel.
*
*
        SUBROUTINE DRAW(IM_ARRAY,COLOR,HEADER)
        BYTE IM_ARRAY
        LOGICAL*2 COLOR
        CHARACTER*32 HEADER
        REAL*4 RED, GREEN, BLUE, INTENSITY
        DIMENSION IM_ARRAY(512,480)
        DIMENSION RED(256), GREEN(256), BLUE(256), INTENSITY(256)
*
*       Include the UIS graphics routines ...
*
        INCLUDE 'SYS$LIBRARY:UISUSRDEF'
*
*       Create a virtual color map  ...
*
        VCM_ID = UIS$CREATE_COLOR_MAP(256)
*
*       Create a virtual display and associate the colormap ...
*
        VD_ID = UIS$CREATE_DISPLAY(0.0,0.0,1024.0,864.0,16.3,15.3,VCM_ID)
        CALL UIS$ENABLE_DISPLAY_LIST(VD_ID)
*
*       Check to see if color or shades of gray ...
*
        IF (COLOR .EQ. .FALSE.) THEN
           DO 1000 I=0,255
*
*       Black to white in 256 steps ...
*
              INTENSITY(I) = FLOAT(I) / 255.0
```

```
1000      CONTINUE
*
*       Fill the color map ...
*
          CALL UIS$SET_INTENSITIES(VD_ID,0,256,INTENSITY)
        ELSE
*
*       Initialize all color vectors to contain 0.0 ...
*
          DO 1010 I = 1,256
             RED(I) = 0.0
             GREEN(I) = 0.0
             BLUE(I) = 0.0
1010      CONTINUE
*
*       Fill them with color ...
*
          DO 1020 I=1,85
             VAL = 0.30 + (FLOAT(I) * (0.69 / 85.0))
             BLUE(I) = VAL
             GREEN(85 + I) = VAL
             RED(170 + I) = VAL
1020      CONTINUE
*
*       Don't forget white ...
*
          RED(256) = 1.0
          GREEN(256) = 1.0
          BLUE(256) = 1.0
*
*       Fill the color map ...
*
          CALL UIS$SET_COLORS(VD_ID,0,255,RED,GREEN,BLUE)
        END IF
        CALL UIS$SET_WRITING_MODE(VD_ID,0,1,UIS$C_MODE_COPY)
*
*       Display the input image array
*
         CALL UIS$IMAGE(VD_ID,1,0.0,0.0,512.0,480.0,512,480,0,IM_ARRAY)
*
*       Window the image ...
*
        WD_ID = UIS$CREATE_WINDOW(VD_ID,'SYS$WORKSTATION',HEADER)
*
*       This subroutine works with a small bug ...
*       ... to get the header and buried windows move the
*       pointer to the side of the image and push the left
*       mouse button ..
*       select EXIT from the menu ...
*
        RETURN
        END
*
*
*       This subroutine is used to read in the ITEX digitizer files.
*       The files must have been previously converted by the
*       ITEX_TO_RMS executable file to an RMS format (identified
*       by the .SSI extension in the directory [IMAGES]) to be
*       used by LOAD.
*
*
        SUBROUTINE LOAD(ARRAY,INFILE)
        BYTE ARRAY
        CHARACTER*32 INFILE
        DIMENSION ARRAY(512,480)
        TYPE 1000
1000    FORMAT(//' Image fileNAME.EXT? '/)
*
*       Read in the desired image file name ...
*
        ACCEPT 1010, INFILE
1010    FORMAT(A32)
        INFILE = 'IM:'//INFILE
```

```
*
*       IM is a logical to access the image files on the SMV2A ...
*
*       Connect image file to unit 1 ...
*
        OPEN(1, FILE=INFILE, READONLY,
        1    RECORDTYPE='VARIABLE',
        1    STATUS='OLD', ACCESS='SEQUENTIAL', FORM='UNFORMATTED')
*
*       Read in the image ...
*
        DO 1020 J=1,480
           READ (1) (ARRAY(I,J),I=1,512)
1020    CONTINUE
        CLOSE(1)
        RETURN
        END
*
*
*       This subroutine calculates dimensionality using what is
*       known as box (or mass) dimension.
*
*
        SUBROUTINE MASS(ARRAY,SUB_ARRAY,X_INDEX,Y_INDEX)
*
*       Define variables ...
*
        LOGICAL*2 ERROR
        INTEGER*4 X_INDEX, Y_INDEX, LO_X, LO_Y, HI_X, HI_Y
        REAL*8 SUB_ARRAY, MASSBIN, M
        DIMENSION MASSBIN(7,64), M(7), ARRAY(64,60), SUB_ARRAY(8,8)
*
*       Zero all of the elements (bins) in MASSBIN and M  ...
*
        DO 1000 I=1,7
           M(I) = 0.0
           DO 1000 J=1,64
              MASSBIN(I,J) = 0.0
1000    CONTINUE
*
*       Count points within L X L X L box throughout the region ...
*
*       First loop increments box size ...
*
        DO 1010 L=2,8
*
*       Next two loops increment over each point in region ...
*
           DO 1010 J=1,8
              DO 1010 I = 1,8
*
*       Box limits in I ...
*
                 MIN_I = SUB_ARRAY(I,J) - L
                 MAX_I = SUB_ARRAY(I,J) + L
*
*       Box limits in Y ...
*
                 LO_Y = J - L
                 IF (LO_Y .LT. 1) THEN
                    LO_Y = 1
                 END IF
                 HI_Y = J + L
                 IF (HI_Y .GT. 8) THEN
                    HI_Y = 8
                 END IF
*
*       Box limits in X ...
*
                 LO_X = I - L
                 IF (LO_X .LT. 1) THEN
                    LO_X = 1
                 END IF
```

```
                HI_X = I + L
                IF (HI_X .GT. 8) THEN
                   HI_X = 8
                END IF
*
*       Count the points in the box ...
*
                DO 1010 K=LO_Y,HI_Y
                   DO 1010 N=LO_X,HI_X
                      IF ((SUB_ARRAY(N,K) .LT. MAX_Z)
     1                      .AND. (SUB_ARRAY(N,K) .GT. MIN_Z)) THEN
                            MASSBIN(L-I,(J-1)*8+I) =
     1                      MASSBIN(L-1,(J-1)*8+I) + 1.0
                      END IF
*
*       Increment L, I, J, K, N ...
*
1010    CONTINUE
        DO 1040 L=1,7
*
*       Normalize the mass distribution to 1.0 for each value of L ...
*
           DO 1020 I=1,64
              M(L) = M(L) + MASSBIN(L,I)
1020       CONTINUE
           DO 1030 I=1,64
              MASSBIN(L,I) = MASSBIN(L,I) / M(L)
1030       CONTINUE
*
*       Calculate the mass dimension for each L ...
*
           DO 1040 I=1,64
              M(L) = M(L) + (I * MASSBIN(L,I))
1040    CONTINUE
*
*       Now estimate the expected value of D for each L ...
*
        DO 1050 L=1,6
           M(L) = (DLOG10(M(L+1)) - DLOG10(M(L))) /
     1       (DLOG10(DBLE(L+1)) - DLOG10(DBLE(L)))
1050    CONTINUE
        D = 0.0
        DO 1060 L=1,6
           D = D + M(L)
1060    CONTINUE
*
*       Place calculated D in working array ...
*
        ARRAY(X_INDEX,Y_INDEX) = D / 6.0
        RETURN
        END
*
*
*       This subroutine simply normalizes the array SUB to
*       contain    Z values between 0 and 1, sets an error flag if all
*       Z values are equal.
*
*
        SUBROUTINE NORM(SUB,ERR)
        REAL*8 SUB, MN, MX
        DIMENSION SUB(8,8)
        LOGICAL*2 ERR
*
*       Initialize min, max, and ERR ...
*
        MN = 255.0
        MX = 0.0
        ERR = .FALSE.
*
*       Find the min and max Z values (to normalize and as
*       an error trap)  ...
*
        DO 1000 J=1,8
```

```
            DO 1000 I=1,8
               IF (SUB(I,J) .LT. MN) THEN
                     MN = SUB(I,J)
               END IF
               IF (SUB(I,J) .GT. MX) THEN
                     MX = SUB(I,J)
               END IF
1000     CONTINUE
*
*        This is the error trap ...
*
         IF(JIDINT(MN) .EQ. JIDINT(MX)) THEN
*
*        Simply set the error flag ERR and exit ...
*
            ERR = .TRUE.
            RETURN
         END IF
*
*        Normalise the array SUB to min = 0.0, max = 1.0
*
         RANGE = MX - MN
         DO 1010 J=1,8
            DO 1010 I=1,8
*
*        Remove the bias, scale values ...
*
               SUB(I,J) = (SUB(I,J) - MN) / RANGE
1010     CONTINUE
         RETURN
         END
*
*
*        This subroutine processes the segmented output array into 4
*        bit patterns (to emulate shades of gray) which are then sent
*        to the DEC LN03 Plus laser printer.
*
*
         SUBROUTINE PRINT(ARRAY,THRESHOLD)
         BYTE DCS, ST, FF, LF, CR
         CHARACTER*2 CRLF
         CHARACTER*5 INIT
         CHARACTER*6 WHITE, GRAY4, GRAY18, BLACK, SPACE
         INTEGER*4 THRESHOLD
         REAL*4 ARRAY
         DIMENSION ARRAY(64,60), THRESHOLD(3)
*
*        Initialize printer control bytes and graphics strings ...
*
         DCS    = '90'X                    !Graphics mode
         ST     = '9C'X                    !Graphics data terminator
         LF     = 'A'X                     !For 1 inch top margin
         CR     = 'D'X                     !Standard carriage return
         FF     = 'C'X                     !Form feed to eject page
         CRLF   = '$-'                     !Graphics carriage return and line feed
         INIT   = '9;;0q'          !Initialize graphics characteristics
         WHITE  = '??????'         !Sixel graphics string
         GRAY4  = '?@?@?@'         !          "
         GRAY18 = 'TITITI'         !          "
         BLACK  = '~~~~~~'         !          "
         SPACE  = '      '         !Move image to middle of page
*
*        Prompt to ensure printer is ready ...
*
         PAUSE 'Printer switched? CONTINUE (CR), or EXIT (CR) ... '
*
*        Open the printer ...
*
         OPEN (ACCESS='SEQUENTIAL', FORM='UNFORMATTED', FILE='CSA0:', UNIT=3)
*
*        Center image to be printed ...
*
         WRITE(3) LF,LF,LF,LF,LF,LF,CR,SPACE,SPACE
```

```
*
*       Place printer in graphics mode ...
*
        WRITE(3) DCS, INIT
*
*       Begin to send actual graphics data ...
*
        DO 1000 J=1,60
           WRITE(3) CRLF
           DO 1000 I=1,64
              IF (ARRAY(I,J) .LE. THRESHOLD(1)) THEN
                    WRITE(3) BLACK
              ELSE IF (ARRAY(I,J) .LE. THRESHOLD(2)) THEN
                    WRITE(3) GRAY16
              ELSE IF (ARRAY(I,J) .LE. THRESHOLD(3)) THEN
                    WRITE(3) GRAY4
              ELSE
                    WRITE(3) WHITE
              END IF
1000    CONTINUE
*
*       Terminate graphics and eject page ...
*
        WRITE(3) ST,FF
*
*       Close printer and exit ...
*
        CLOSE(3)
        RETURN
        END
*
*
*       This subroutine is used to manipulat the work array in
*       a visually useful form.
*
*
        SUBROUTINE PROCESS(IN,VALLEYS)
        LOGICAL*2 ERROR, CHANGE, TEST
        CHARACTER*1 ANSWER
        INTEGER*4 BIN, VALLEYS
        REAL*4 IN, MN, MX, RANGE, SLOPE
        DIMENSION IN(64,60), BIN(0:255), VALLEYS(3), TEST(64,60)
*
*       Initialise min and max ...
*
        MN = 255.0
        MX = 0.0
*
*       Find the min and max values ...
*
        DO 1000 J=1,60
           DO 1000 I=1,64
              IF ((IN(I,J) .LT. MN) .AND. (IN(I,J) .NE. 0.0)) THEN
                    MN = IN(I,J)
              END IF
              IF (IN(I,J) .GT. MX) THEN
                    MX = IN(I,J)
              END IF
1000    CONTINUE
*
*       Normalise the array IN to min = 0.0, max = 255.0 corresponding
*       to the gray values on the GPX ...
*
        RANGE = MX - MN
        DO 1010 J=1,60
           DO 1010 I=1,64
*
*       Remove the bias, scale values ...
*
              IF (IN(I,J) .NE. 0.0) THEN
                    IN(I,J) = 255.0 * (IN(I,J) - MN) / RANGE
              END IF
```

```
1010    CONTINUE
        VALLEYS(1) = 127
        VALLEYS(2) = 159
        VALLEYS(3) = 223
        CALL PRINT(IN,VALLEYS)
        RETURN
        END
*
*
*       This subroutine writes the work array to disk.
*
*
        SUBROUTINE WRITEFILE(ARRAY,OUTFILE,TYPE)
*
*       Define variables ...
*
        LOGICAL*2 TYPE
        CHARACTER*1 CHARRAY
        CHARACTER*32 OUTFILE
        REAL*4 ARRAY
        DIMENSION ARRAY(64,60), CHARRAY(64,60)

*
*       Associate OUTFILE with a VMS filespec ...
*
        OUTFILE = 'DUA1:[AJONES.DATA]'//OUTFILE//'.DAT'
*
*       Create an outfile and associate it with a logical unit ...
*
        OPEN (FILE=OUTFILE, FORM='FORMATTED', STATUS='NEW', UNIT = 2)
*
*       Write the work array to disk ...
*
        IF (TYPE .EQ. .TRUE.) THEN
           DO 1010 J=1,60
              DO 1010 I=1,57,8
*
*       8 rows of 8 reals for each row of the work array ...
*
                    WRITE(2,1000) (ARRAY(I+K,J),K=0,7)
1000                FORMAT(8V10.4)
1010       CONTINUE
        ELSE
*
*       Write out character graphics of 64 characters for
*       each row of the work array ...
*
           DO 1020 J=1,60
              DO 1020 I=1,64
                    IF (IINT(ARRAY(I,J)) .EQ. 0) THEN
                       CHARRAY(I,J) = ' '
                    ELSE IF (IINT(ARRAY(I,J)) .EQ. 1) THEN
                       CHARRAY(I,J) = 'E'
                    ELSE IF (IINT(ARRAY(I,J)) .EQ. 2) THEN
                       CHARRAY(I,J) = 'V'
                    ELSE
                       CHARRAY(I,J) = 'X'
                    END IF
1020       CONTINUE
           DO 1040 J=1,60
              WRITE(2,1030) (CHARRAY(I,J),I=1,64)
1030          FORMAT(64A1)
1040       CONTINUE
        END IF
*
*       Done with file -- so close it ...
*
        CLOSE(2)
        RETURN
        END
```

```
      PROGRAM MAIN
      INCLUDE 'GLOBAL.FOR'
      CALL LOAD(IMAGE,STRING)
      TYPE 3
3     FORMAT(/' Processing image ... work array line # '/)
      DO 10 J=1,60
         TYPE 5,J
5        FORMAT(I4)
         DO 10 I=1,64
*
*     Segment image into 8 X 8 sub-images ...
*
            CALL SUB(IMAGE,SUB_IMAGE,I*8-7,J*8-7)
*
*     Calculate fractal dimension of each via hybrid Brown ...
*
            CALL BROWN(WORK,SUB_IMAGE,I,J)
10    CONTINUE
      FLAG = .FALSE.
*
*     Display original image ...
*
      CALL DRAW(IMAGE,FLAG,STRING)
      PAUSE
*
*     Process fractal dimension results and dump to printer ...
*
      CALL PROCESS(WORK,THRESHOLDS)
*
*     Convert to an image file (ie - to bytes) ...
*
      CALL CONVERT(WORK,IMAGE)
*
*     Take a look at final image ...
*
*     CALL DRAW(IMAGE,FLAG,STRING)
      PAUSE
      END
```

```
/*****************************************************************************
*                                                                            *
*      Program name        ITEX_to_RMS.c                                     *
*       Author                  Richard Roberts                              *
*                                                                            *
*       Special thanks to Stevve Johnson and Rick Raines for assistannce     *
*                                                                            *
   *
*       This program inputs a file from disk into an array. The image        *
*      stored by the ITEX100 software is read in and converted into          *
*      decimal format.  Each pixel is stored into the two dimensional        *
*      array with a color value between 0 and 255.                           *
*                                                                            *
*       The program then writes the image to disk in the .SSI format.        *
*       The program assumes images of size 512 by 480.                       *
*                                                                            *
*****************************************************************************/

#include "sys$library:stdio.h"
#include "sys$library:rms.h"

int    i,j,k;                          /* loop control variables */
char   name[256];                    /* name of image file */
char   test[2];                    /* left or right image select var */
char   char_header[63];              /* array containing header char info */
int    num_header[63];               /* array containing header num info */
char   comment[256];                 /* array containing user image comment */

/* change these array values to match ITEX image */

char   im_char_array[512][480]; /* array containing character pixel values */
int    im_num_array[512][480];  /* array containing decimal pixel values */

FILE   *afile, *fopen ();

#define RECORD_SIZE (sizeof record)

int     rms_status;
struct  FAB  fab;
struct  RAB  rab;

/* change row_data array if not using 512 pixel rows */

struct
  {
        char    row_data [512];
  }  record;
char    *filename;
int     index;
int     last;
char    response;

main()
{
printf("\n\nCONVERT ITEX100 Format to .SSI\n\n");
response = 'Y';
while ( (response=='Y') | (response=='y') )
{

get_info();
read_disk();

/* Initialize RMS file */

/* Change file extension to SSI */
last = strlen(&name) - 1;
name[last-2] = 'S';
name[last-1] = 'S';
name[last]   = 'I';

fab = cc$rms_fab;
```

```
fab.fab$l_fna = &name;
fab.fab$b_fns = strlen(&name);
fab.fab$b_org = FAB$C_SEQ;
fab.fab$b_rat = FAB$M_CR;
fab.fab$w_mrs = 0;

rab = cc$rms_rab;
rab.rab$l_fab = &fab;

rms_status = sys$create (&fab);

rms_status = sys$connect (&rab);

/* change loop counters in not 512 X 480 image -- note "<" in loop */

       for(j=0; j<480; j++)                  /* write image pixel colors */
       {
           for(i=0; i<512; i++)
           {
                   record.row_data[i] = im_num_array[i][j] - 128;
           }
            rab.rab$b_rac = RAB$C_SEQ;
            rab.rab$l_rbf = &record;
            rab.rab$w_rsz = RECORD_SIZE;
            rms_status = sys$put (&rab);
       }

/* Close RMS file */
rms_status = sys$close (&fab);

printf("Convert Another File? (Y/N) ");
scanf("%s",&response);
};  /* end while */

}


/*************************************************************************/
get_info()
{
       cls();
       printf("\n Enter the name of the file (include .EXT)\n>");
       scanf("%s",&name);

       return;
}



/*************************************************************************/
read_disk()
{
       afile = fopen(name, "r");        /* open file for read only */
       printf("\n\n\n\n\n\n\n Please wait while file is processed.......\n ");

       for(k=0; k<=63; k++)                          /* read header info */
       {
           char_header[k] = getc(afile);
           num_header[k] = char_header[k];
           if(num_header[k]<0)
             num_header[k] = num_header[k] + 256; /* correct for wrap around */
       }


       if(num_header[12]==1 || num_header[12]==2) /* check file format type */
       {
          cls();
          printf("\n File can not be processed.");
          printf("\n File format must be eight_bit");
          printf("\n\n\n\n\n\n Press RETURN to quit.\n:");
          getchar();
          getchar();
          return;
```

```
        }


        for(j=0; j<num_header[2]; j++)          /* read comment area */
            comment[j] = getc(afile);

/* change loop counters if not 512 X 480 image -- note "<" in loop */

        for(j=0; j<480; j++)                    /* read image pixel colors */
        {
            for(i=0; i<512; i++)
            {
                    im_char_array[i][j] = getc(afile);
                    im_num_array[i][j] = im_char_array[i][j];
                    if(im_num_array[i][j]<0)
                       im_num_array[i][j] = im_num_array[i][j] + 256;
                     record.row_data[i] = im_num_array[i][j] - 128;
            }
        }

        fclose(afile);
        return;
}

/*************************************************************************/
cls()
{
printf("\n\n");
return;
}
```

Appendix C: Fractal Dimension via PSD Rolloff

This appendix is simply a derivation of the relationship described by Eq (6). It is taken in its entirety from reference (23).

Consider the following fundamental property of fBm: If X(t) denotes fBm with Hurst exponent $0<H<1$, then the properly scaled function

$$Y(t) = r^{-H}X(rT)$$

for a given $r>0$ has the same statistical properties, and thus the same spectral density, as X(t). From this basic property and the use of some elementary calculus, the results of Eqs (6) and (7) can be deduced as follows.

Let us fix some $r>0$, and let

$$Y(t,T) = \begin{cases} Y(t) = r^{-H}X(rT), & 0<t<T \\ 0 & , \text{otherwise} \end{cases}$$

and adopt the notation

$F_X(f,T)$, $F_Y(f,T)$	Fourier transforms of X(t,T), Y(t,T)
$S_X(f,T)$, $S_Y(f,T)$	Spectral densities of X(t,T), Y(t,T)
$S_X(f)$, $S_Y(f)$	Spectral densities of X(t), Y(t)

We then compute the Fourier transform of Y(t,T)

$$F_Y(f,T) = \int_0^T Y(t)e^{-2\pi ift}dt = r^{-H}\int_0^{rT} X(s)e^{-2\pi i(f/r)s}ds/r$$

where we have substituted s/r for t and ds/r for dt in the second integral. It follows then, that

$$F_Y(f,T) = r^{-(H+1)}F_X(f/r,rT)$$

Thus, for the spectral density of Y(t,T)

$$S_Y(f,T) = r^{-(2H+1)}(1/rT)|F_X(f/r,rT)|^2$$

and in the limit as T approaches infinity, or equivalently as rT approaches infinity, we conclude

$$S_Y(f) = r^{-(2H+1)} S_X(f/r)$$

But, since Y is just a properly scaled version of X, their spectral densities coincide such that

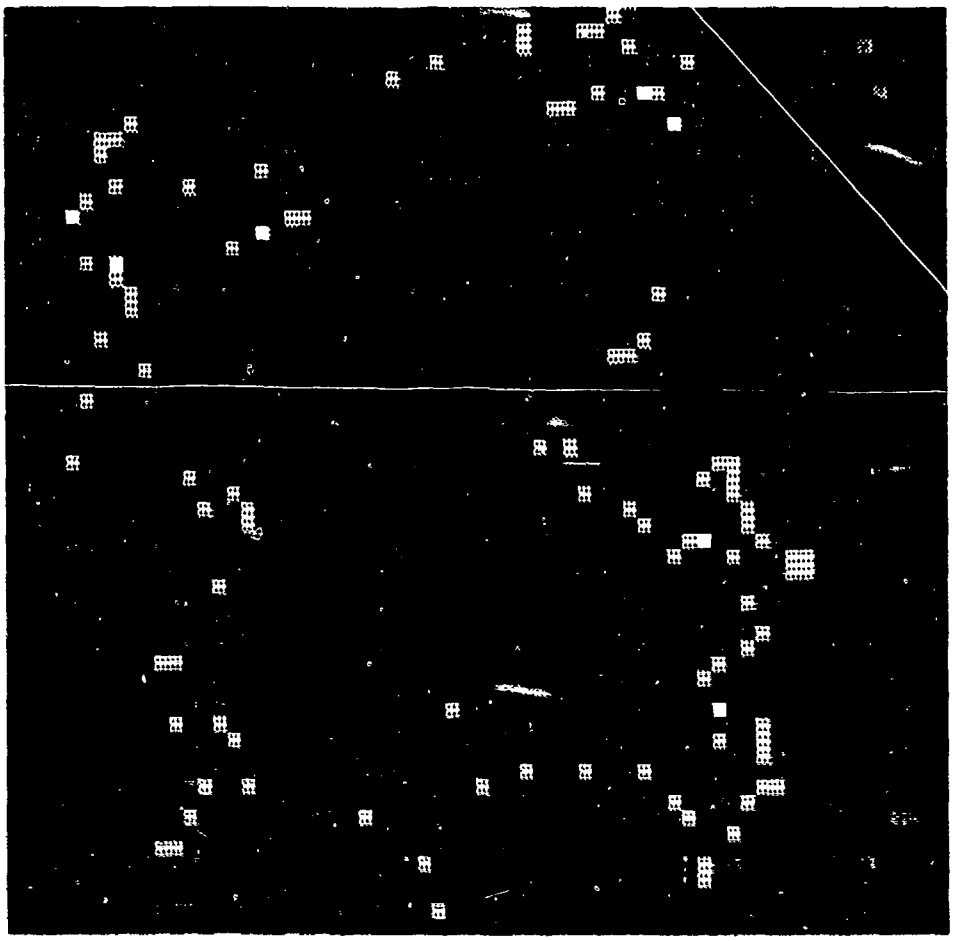
$$S_X(f) = r^{-(2H+1)} S_X(f/r)$$

Now we formally set $f=1$ and replace $1/r$ again by f to obtain the desired result

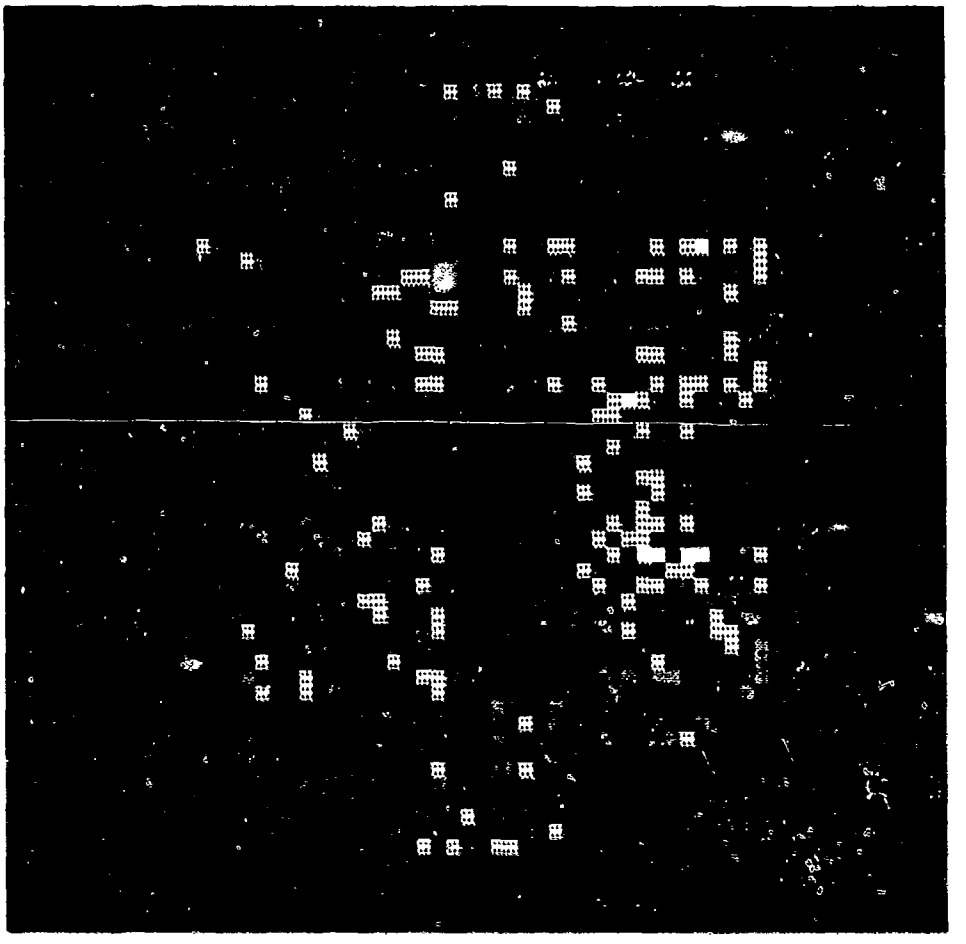
$$S_X(f) \propto f^{-(2H+1)}$$

Appendix D: Output

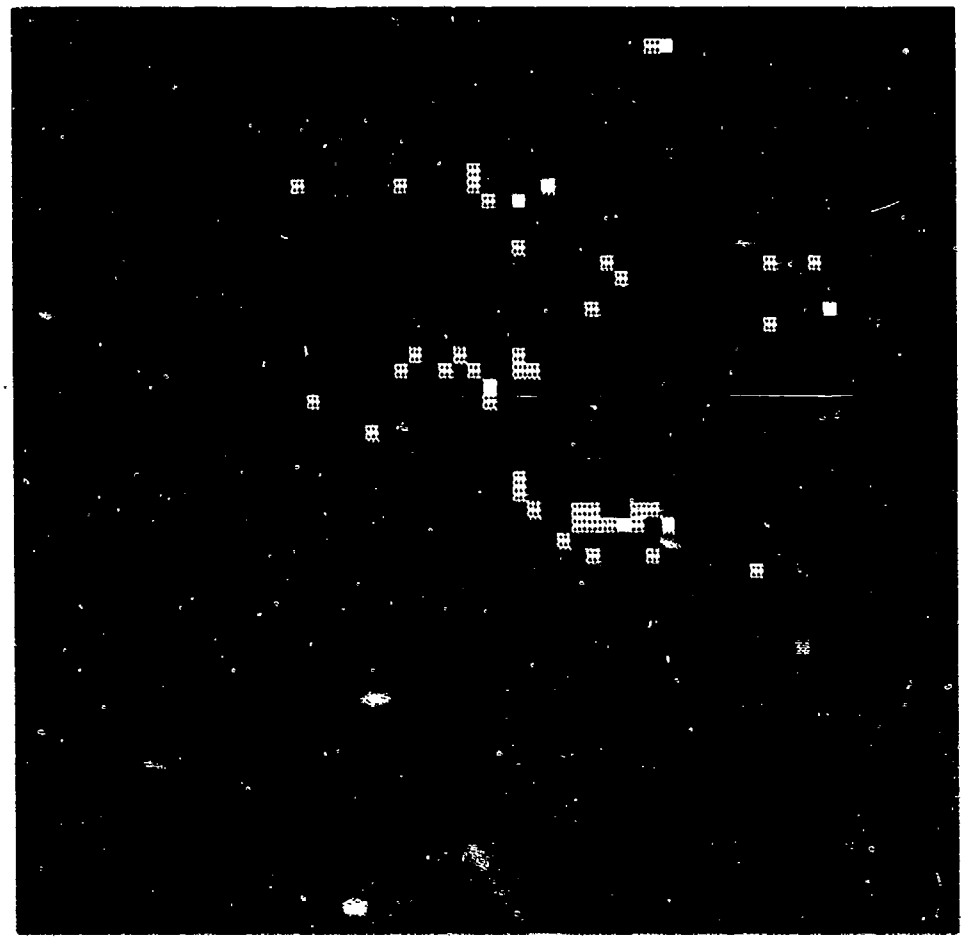
This appendix contains the graphic output using the modified linear method of thresholding. Each figure is labelled to indicate the fractal dimension calculator and input image (refer to Appendix A) used.



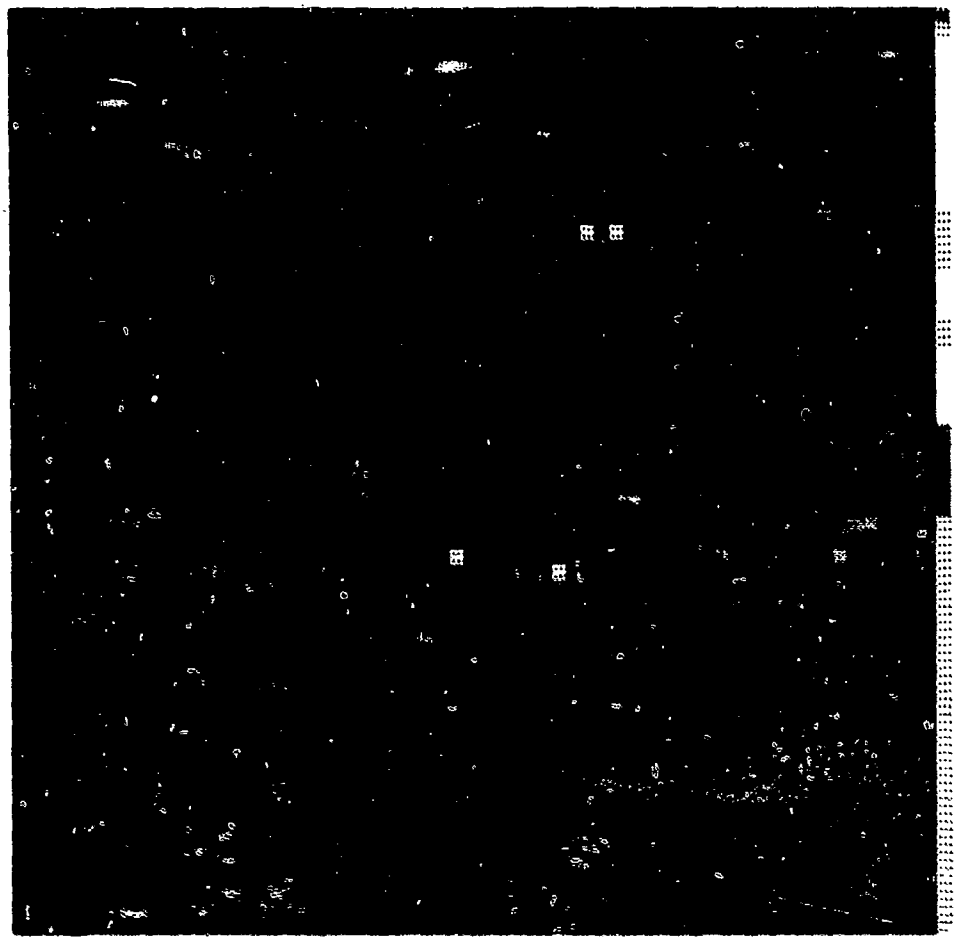
PSD Rolloff / 2.SSI



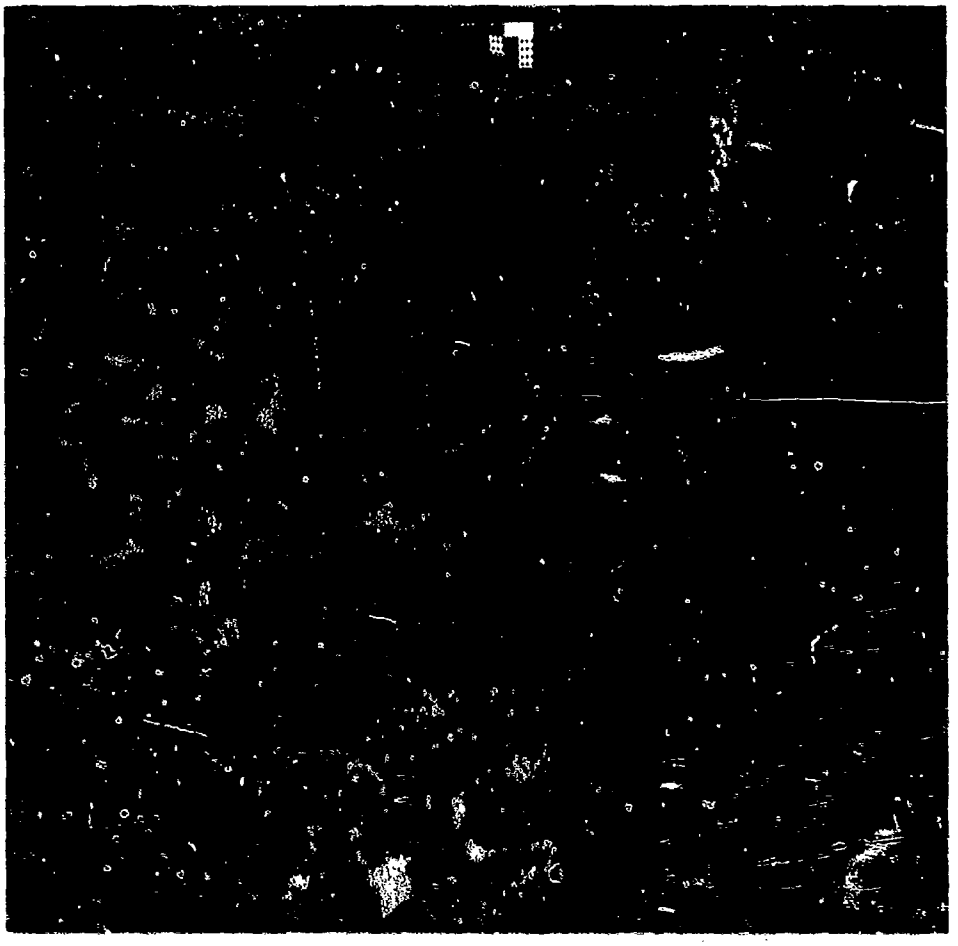
PSD Rolloff / 3.SSI



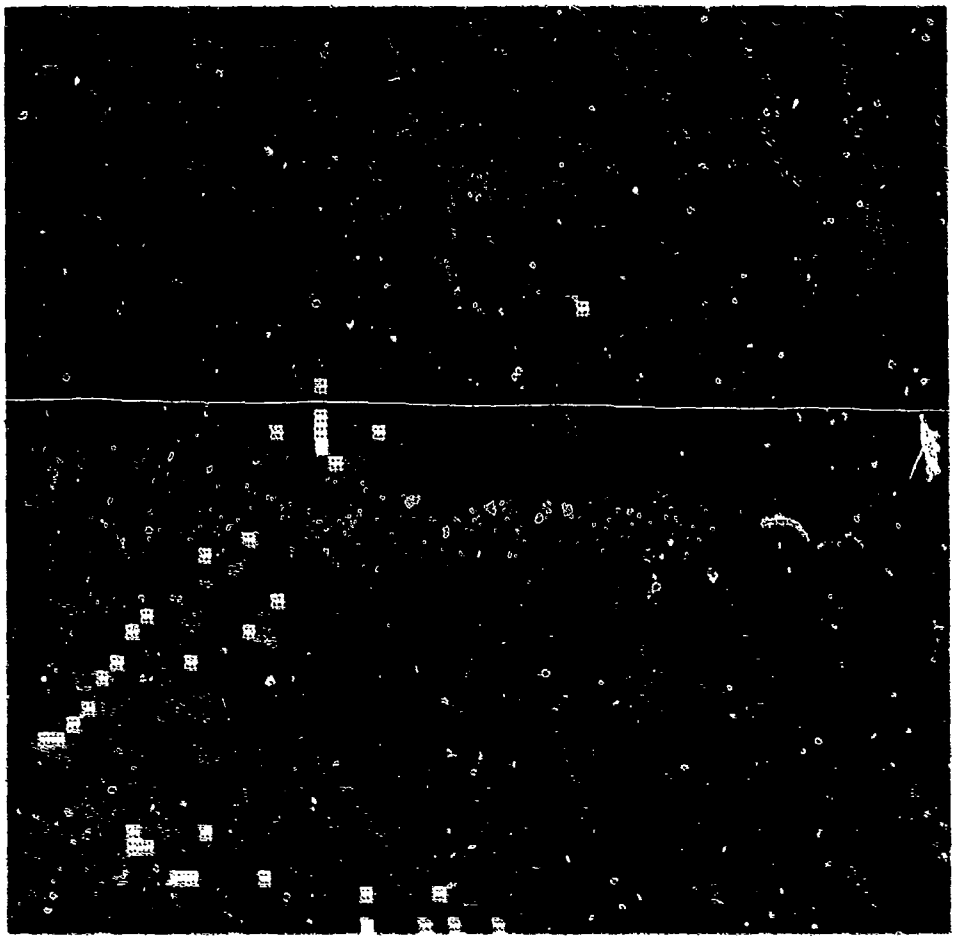
PSD Rolloff / 5.SSI



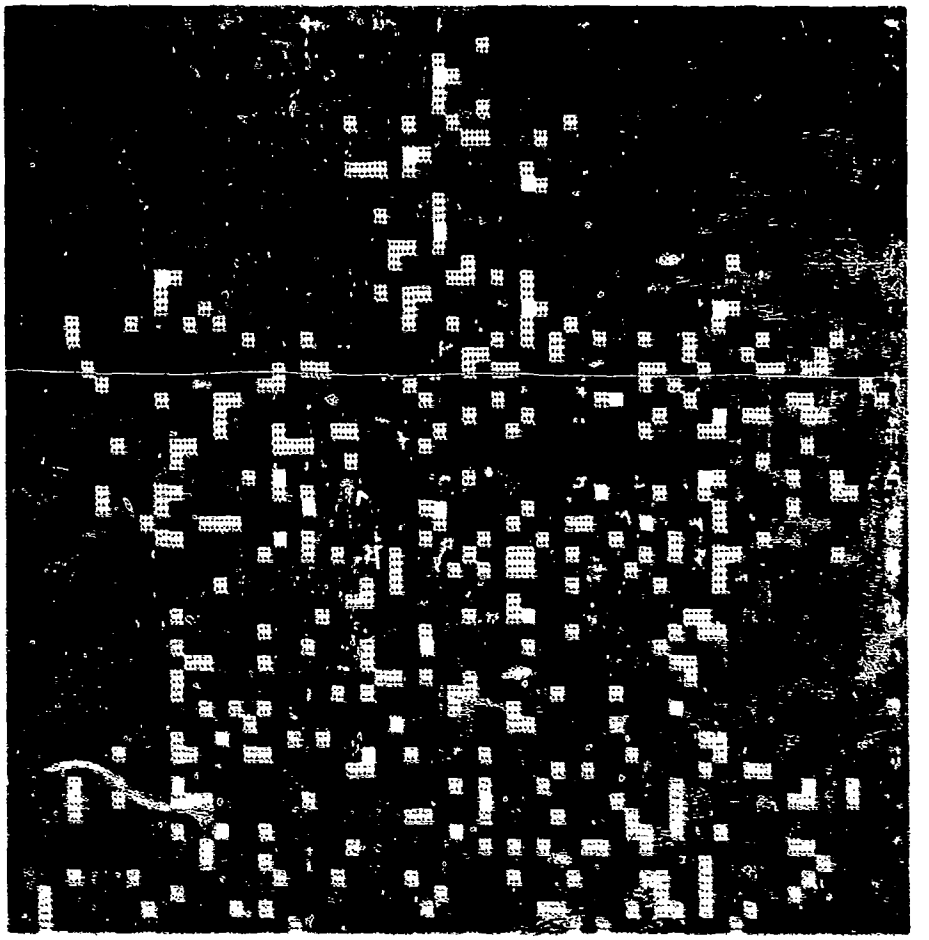
PSD Rolloff / 6.SSI



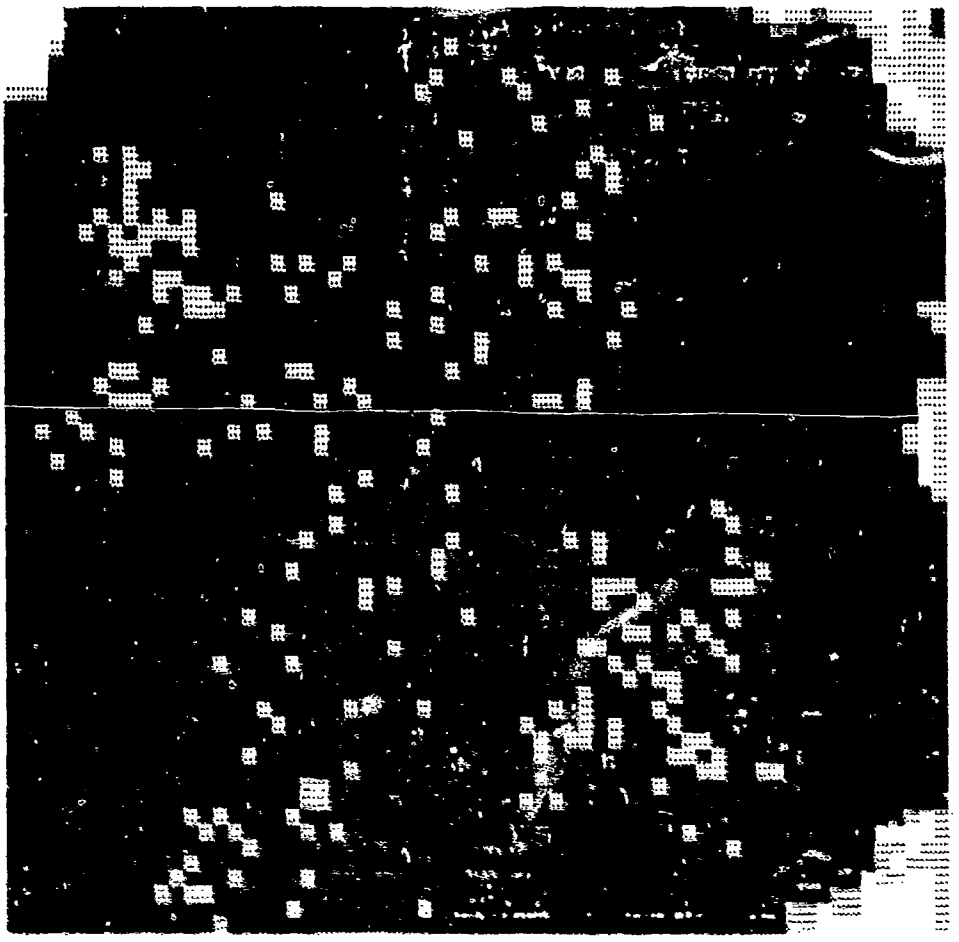
PSD Rolloff / F15.SSI



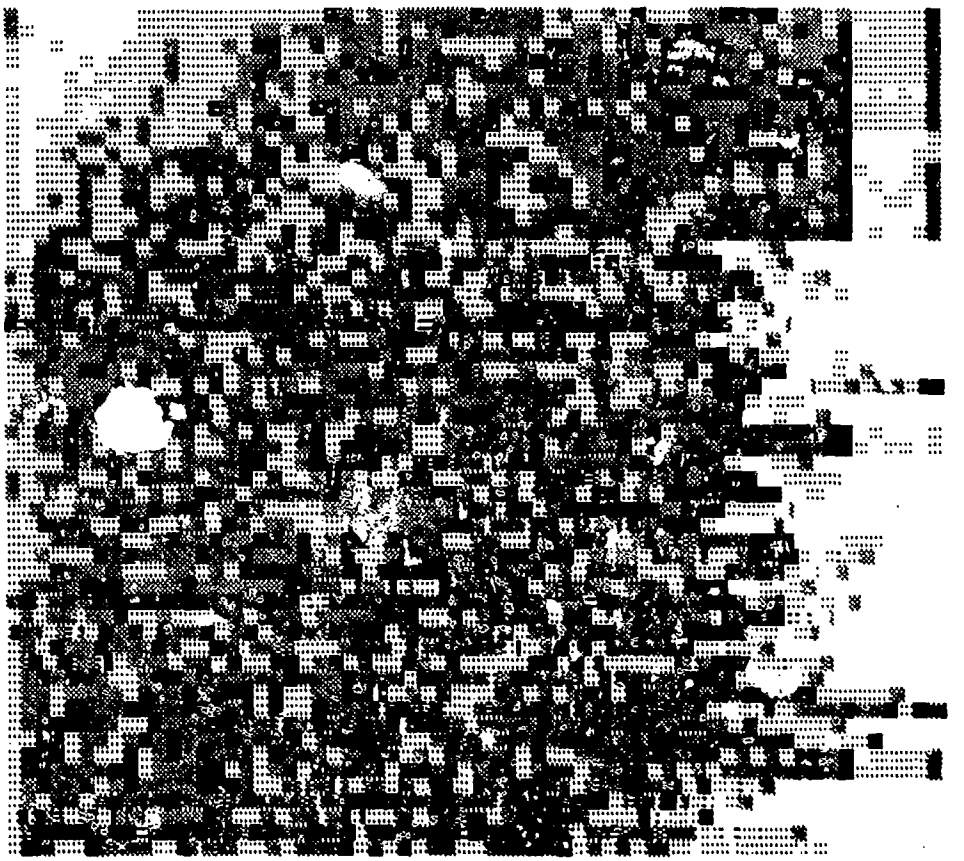
PSD Rolloff / F16XL.SSI



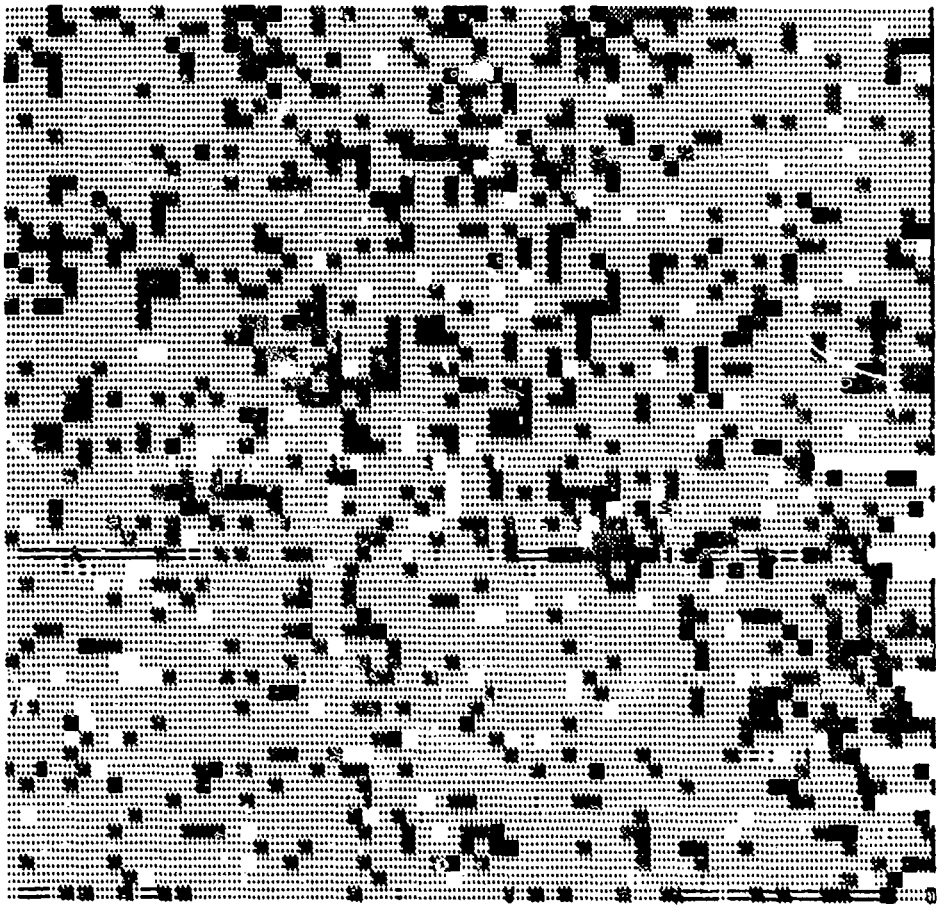
Box Dimension / 1.SSI



Box Dimension / 2.SSI

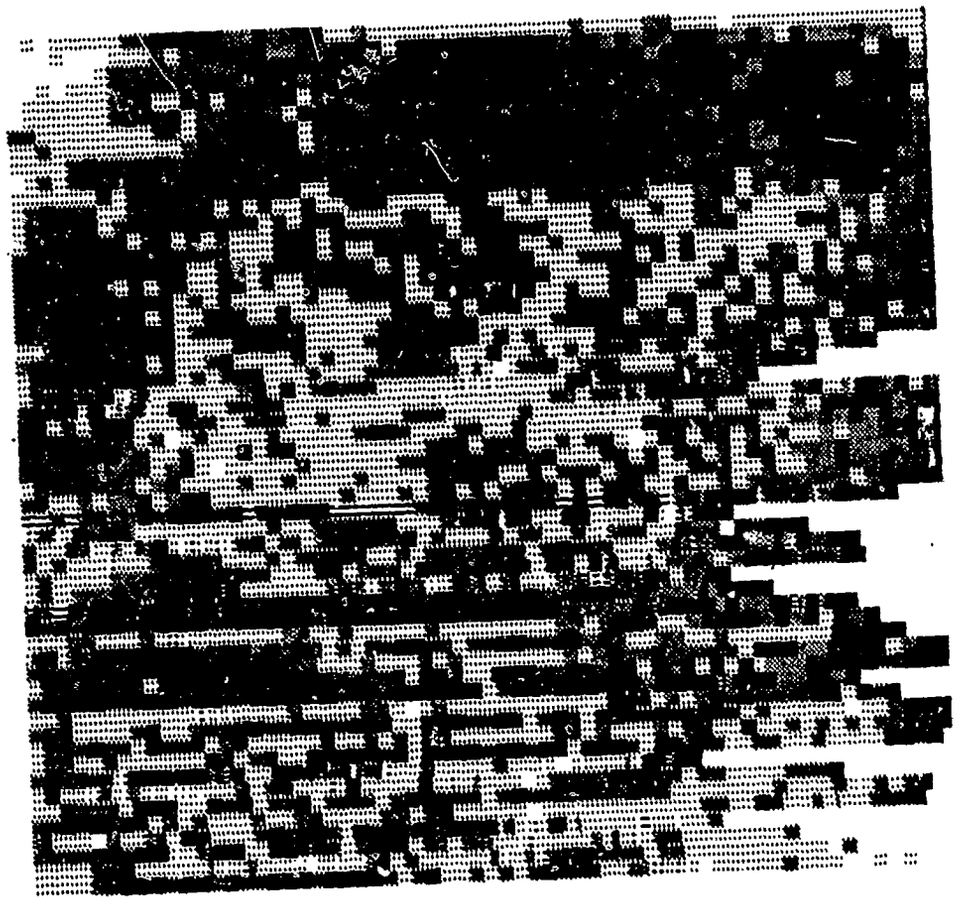


Box Dimension / 3.SSI

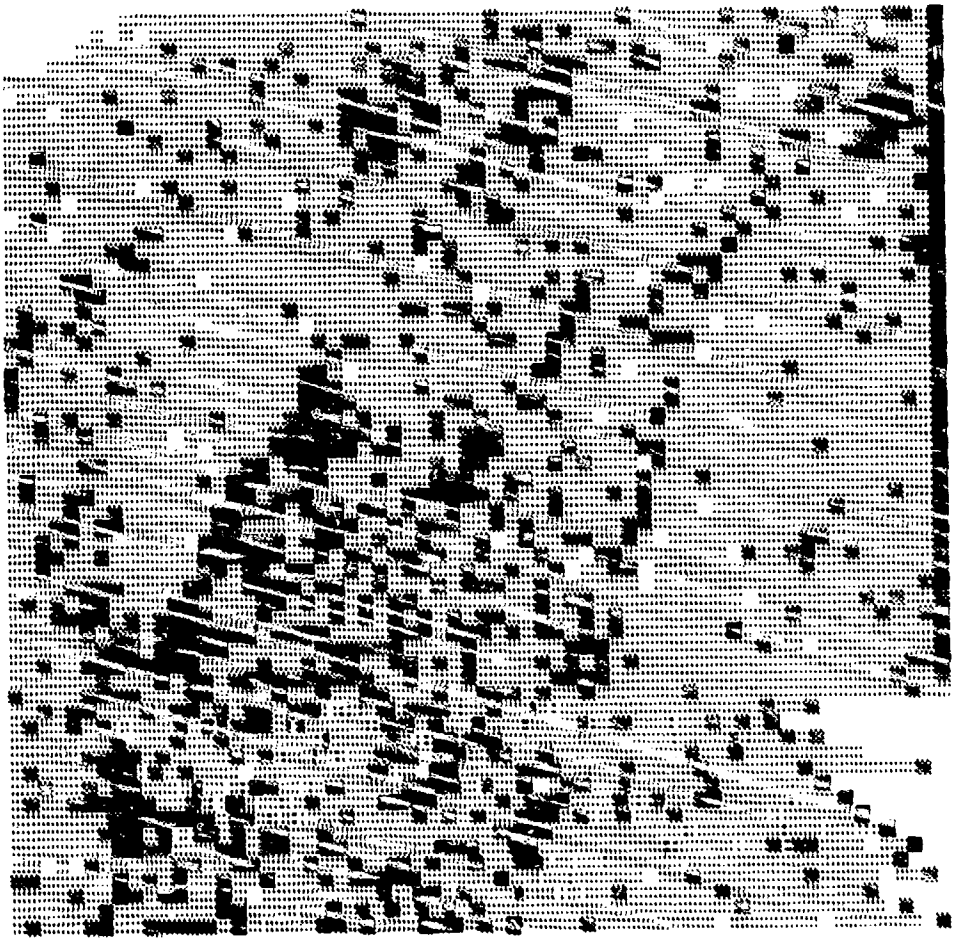


Box Dimension / 5.SSI

Box Dimension / 6.SSI

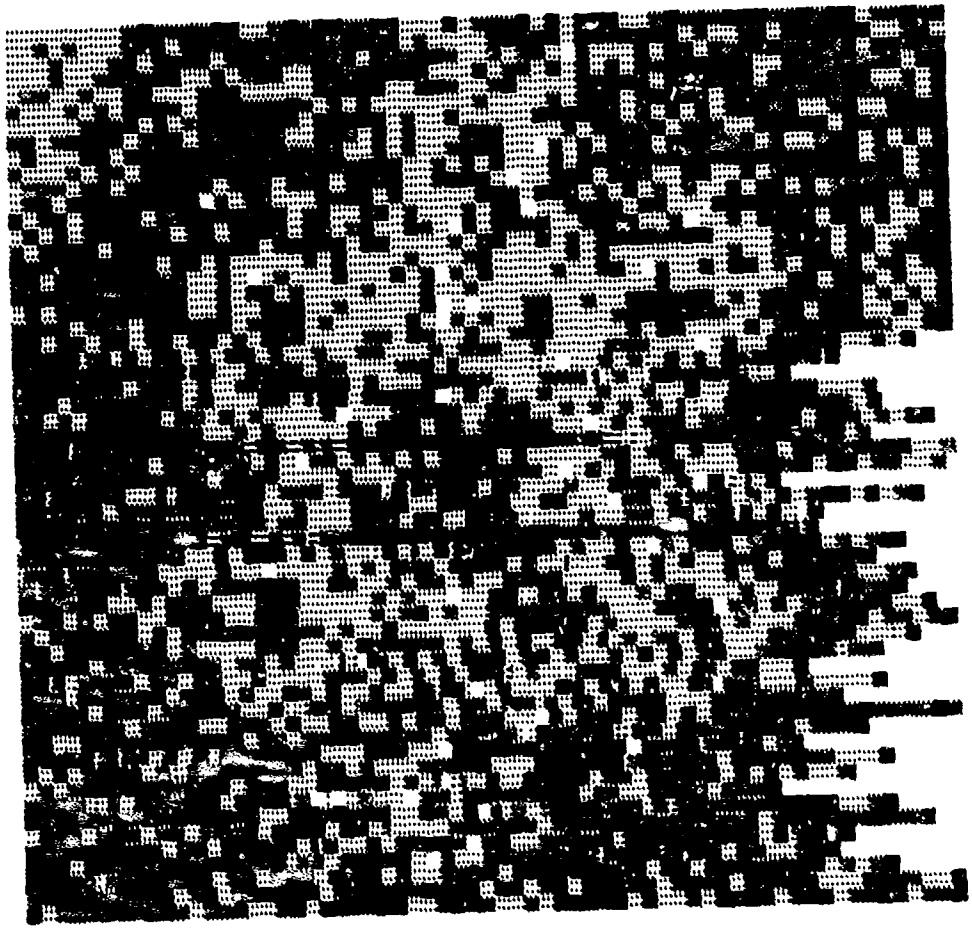


Box Dimension / F15.SSI

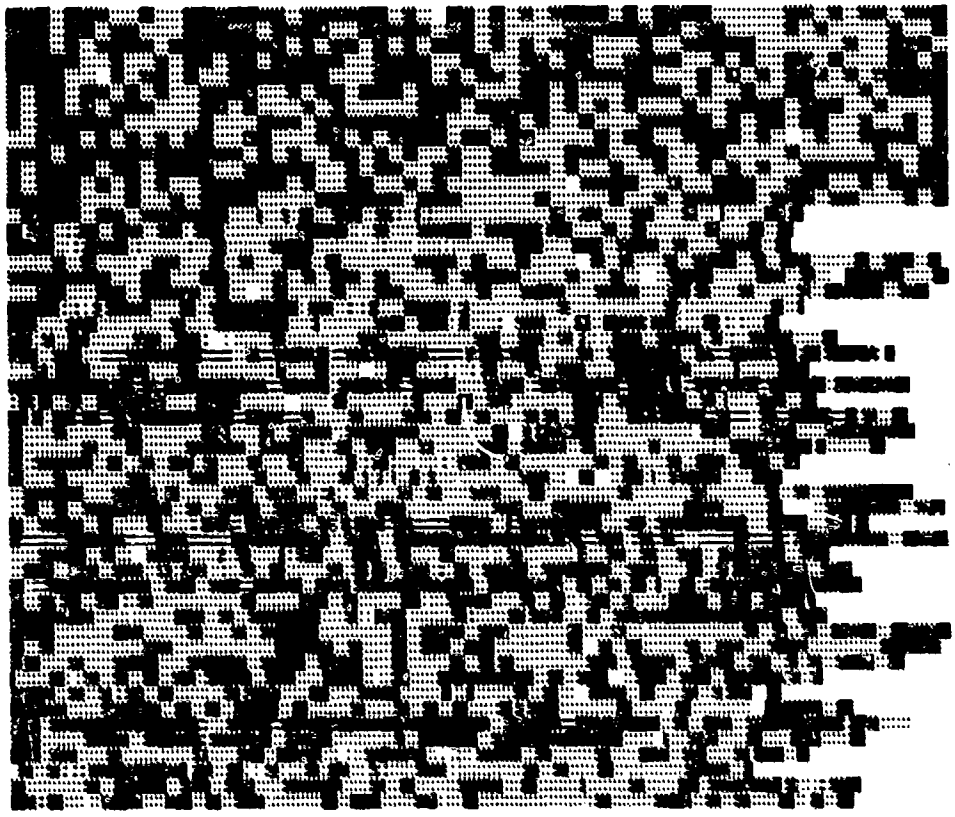


Box Dimension / F16XL.SSI

Hybrid Brown / 2.SST

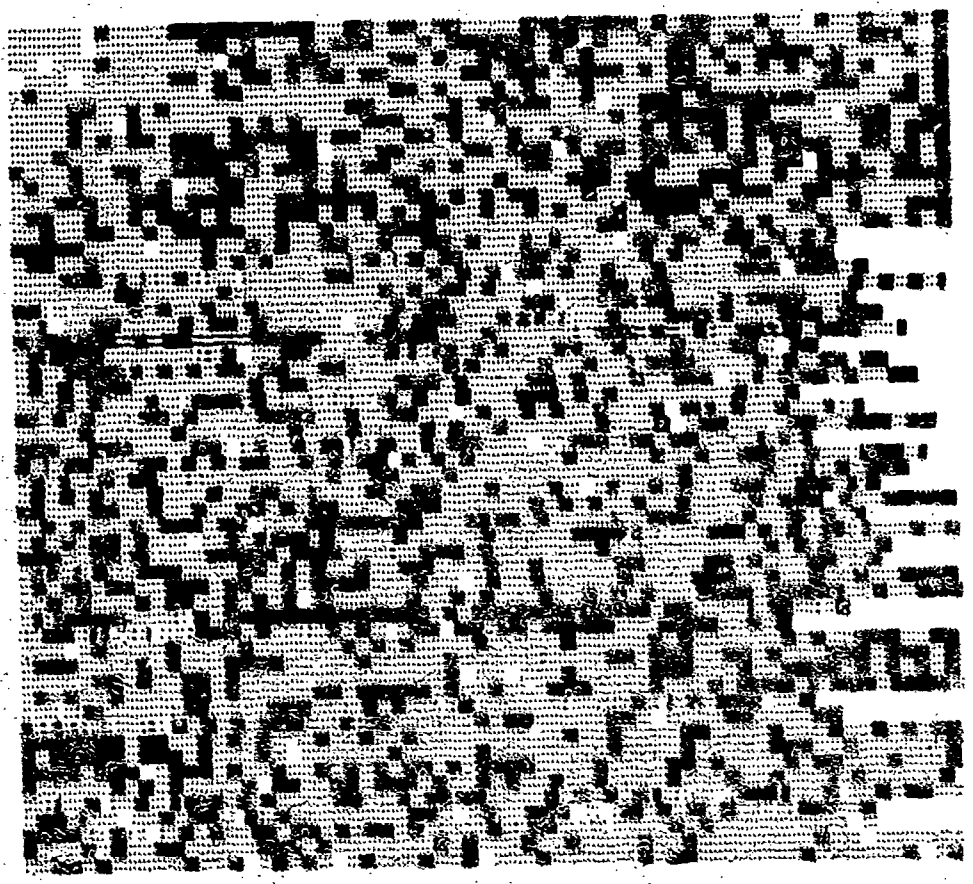


Hybrid Brown / 3.SSI

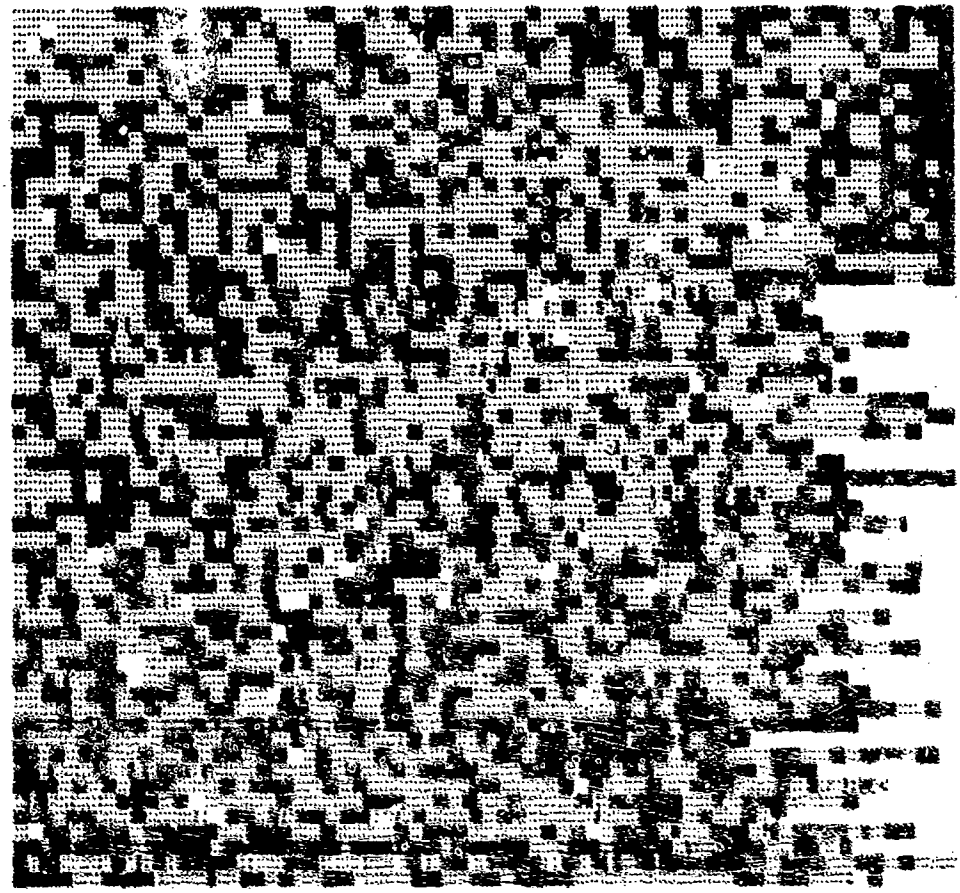


Hybrid Brown / 5.SSI

Hybrid Brown / 6.SSI



Hybrid Brown / F15.SSI



Hybrid Brown / P16XL.SSI

Bibliography

1. Backpacker, January 1984, 46.

3. Barnsley, Michael F. "Fractal Modelling of Real World Images," SIGGRAPH '87 Fractals: Introduction, Basics, and Perspectives, Published by the ACM as material for Course 15 of SIGGRAPH '87.

3. Communications of the ACM, 28: Cover (November 1986)

4. Ecclesiastes 1:9-10, Holy Bible, King James Version. Camden, New Jersey: Thomas Nelson Inc.

5. Fournier, Alain, et. al. "Computer Rendering of Stochastic Models," Communications of the ACM, 25: 371-384 (June 1982).

6. Gardner, William A. Introduction to Random Processes With Applications to Signals and Systems. New York: Macmillan Publishing Company, 1986.

7. Harrington, Steven. Computer Graphics A Programming Approach. New York: McGraw-Hill Book Company, 1987.

8. Horev, Moshe. Picture Correlation Model for Automatic Machine Recognition. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, 1980.

9. Jane's All the World's Aircraft 1985-86. London, England: Jane's Publishing Co., Ltd.

10. Kabrisky, Matthew. Class notes EENG 620 Pattern Recognition I, Air Force Institute of Technology (AU), Wright-Patterson AFB OH, Winter Qtr. 1987.

11. Keller, James M., et. al. "Characteristics of Natural Scenes Related to the Fractal Dimension," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-9: 621-627 (September 1987).

12. Kobel, Capt William G. and Martin, Capt Timothy. Distortion-Invariant Pattern Recognition In Non-Random Noise. MS thesis, AFIT/GE/ENG/86D-20. School of Engineering, Air Force Institute of Technology (AU), Wright-Patterson AFB OH.

13. Lundahl, T., et. al. "Texture Analysis of Diagnostic XRay Images by Use of Fractals," SPIE Visual Communications and Image Processing, 707: 23-30 (1986).

14. Mandelbrot, Benoit B. The Fractal Geometry of Nature. New York: W. H. Freeman and Company, 1983.

15. Mandelbrot, Benoit B., Public lecture. University of Cincinnati, Cincinnati OH, 16 September 1987.

16. Norton, Alan. "Generation and Display of Geometric Fractals in 3-D," Computer Graphics, 16: 61-67 (July 1982).

17. Outside, April 1984, 59. Photo by Dennis Brooks.

18. Outside, December 1984, 100. Photo by Carr Clifton.

19. Peitgen, Heinz-Otto "Fantastic Deterministic Fractals," SIGGRAPH '87 Fractals: Introduction, Basics, and Perspectives, Published by the ACM as material for Course 15 of SIGGRAPH '87.

20. Pentland, Alex P. "Fractal-Based Description of Natural Scenes," IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-6: 661-674 (November 1984).

21. Pentland, Alex P. "Shading into Texture," From Pixels To Predicates: Recent Advances In Computational Vision. Edited by Alex P. Pentland. Norwood, New Jersey: Ablex Publishing Company, 1986.

22. Pfeifer, P., et. al. "Ideally Irregular Surfaces of Dimension Greater Than Two in Theory and Practice," Surface Science, 126: 569-572. (1983).

23. Saupe, Dietmar "Algorithms for Random Fractals," SIGGRAPH '87 Fractals: Introduction, Basics, and Perspectives, Published by the ACM as material for Course 15 of SIGGRAPH '87.

24. Voss, Richard D. "Fractals in Nature: Characterization, Measurement, and Simulation," SIGGRAPH '87 Fractals: Introduction, Basics, and Perspectives, Published by the ACM as material for Course 15 of SIGGRAPH '87.

25. Voss, Richard D., Public lecture. University of Cincinnati, Cincinnati OH, 16 September 1987.

VITA

Captain Alan L. Jones was born on 16 January 1961 in Yankton, South Dakota. He graduated from high school in Flandreau, South Dakota, in 1979 and attended South Dakota State University, from which he received the degrees of Bachelor of Science in Electrical Engineering and Bachelor of Science in Engineering Physics. He was commissioned in the USAF upon graduation through the ROTC program. His initial tour of duty was with the Foreign Technology Division, Wright-Patterson AFB, Ohio were he served as an avionic systems engineer until entering the School of Engineering, Air Force Institute of Technology, in June 1986.

Permanent Address: 701 W. Third Ave.

Flandreau, South Dakota 57028

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	1b. RESTRICTIVE MARKINGS
2a. SECURITY CLASSIFICATION AUTHORITY	3. DISTRIBUTION/AVAILABILITY OF REPORT Approved for public release; distribution unlimited
2b. DECLASSIFICATION/DOWNGRADING SCHEDULE	
4. PERFORMING ORGANIZATION REPORT NUMBER(S) AFIT/GE/ENG/87D-29	5. MONITORING ORGANIZATION REPORT NUMBER(S)
6a. NAME OF PERFORMING ORGANIZATION School of Engineering 6b. OFFICE SYMBOL (If applicable) AFIT/ENG	7a. NAME OF MONITORING ORGANIZATION
6c. ADDRESS (City, State, and ZIP Code) Air Force Institute of Technology Wright-Patterson AFB, OH 45433-6583	7b. ADDRESS (City, State, and ZIP Code)
8a. NAME OF FUNDING/SPONSORING ORGANIZATION 8b. OFFICE SYMBOL (If applicable)	9. PROCUREMENT INSTRUMENT IDENTIFICATION NUMBER
8c. ADDRESS (City, State, and ZIP Code)	10 SOURCE OF FUNDING NUMBERS PROGRAM ELEMENT NO. / PROJECT NO. / TASK NO / WORK UNIT ACCESSION NO

11. TITLE (Include Security Classification)

IMAGE SEGMENTATION VIA FRACTAL DIMENSION (UNCLASSIFIED)

12. PERSONAL AUTHOR(S)

Alan L. Jones, Captain, USAF

13a. TYPE OF REPORT	13b TIME COVERED	14. DATE OF REPORT (Year, Month, Day)	15 PAGE COUNT
MS Thesis	FROM _____ TO _____	1987, December	97

16. SUPPLEMENTARY NOTATION

17. COSATI CODES			18 SUBJECT TERMS (Continue on reverse if necessary and identify by block number)
FIELD	GROUP	SUB-GROUP	
12	9		Image Processing, Mathematical Models, Fractals, Fractal Geometry, Segmentation, Algorithms

19 ABSTRACT (Continue on reverse if necessary and identify by block number)

Thesis Advisor: Dr. Matthew Kabrisky

Approved for public release IAW AFR 190-1.
LYNN E. WOLAVER 31 Dec 87
Dean for Research and Professional Development
Air Force Institute of Technology (AU)
Wright-Patterson AFB OH 45433

20 DISTRIBUTION/AVAILABILITY OF ABSTRACT	21 ABSTRACT SECURITY CLASSIFICATION
☒ UNCLASSIFIED/UNLIMITED ☐ SAME AS RPT ☐ DTIC USERS	UNCLASSIFIED
22a NAME OF RESPONSIBLE INDIVIDUAL Dr. Matthew Kabrisky, Prof.	22b TELEPHONE (Include Area Code) (513)255-2576
	22c OFFICE SYMBOL AFIT/ENG

Continued from block 19: Abstract

The purpose of this study was to investigate the suitability of algorithms derived from the study of fractal geometry to the specific problem of image segmentation. The use of two widely used methods and an original hybrid technique for calculating fractal dimension is demonstrated. All three techniques appeared to be effective to some extent in segmenting images, but shared a common problem with the establishment of suitable thresholds for robust segmentation.

Keywords:

44

FED 18