ON THE USE OF FRACTALS FOR EFFICIENT MAP GENERATION

by
F. S. Hill, Jr. & S. E. Walker, Jr.
University of Massachusetts
Amherst, MA 01003

ABSTRACT

Techniques are described for compressing, storing, and regenerating outline map images in such "cramped" applications as cockpit instrument-panel displays with on-board microcomputers. A large map database is compacted into a vastly smaller one off-line, and stored on-board (or radioed as needed to the aircraft). During the regeneration process fractal curves are used to flesh out the compressed map outlines into visually compelling maps. Because the compression process retains key information that is later used in the fractalizing phase, an overall dramatic saving is made in dataset size and speed of regeneration.

KEYWORDS: Cartography, Data-Compression, Fractals, Chain-Encoding

I. INTRODUCTION

Although the state of the art in computer-assisted map generation and manipulation has advanced rapidly in the last 10 years (1,2), dealing with finely detailed maps on a computer is a very resource-hungry business. High quality maps require very large datasets (often 100,000 points or more) for storing the pertinent information, and several minutes are usually required to generate a new map display, even using large and fast mainframe computers.

On the other hand, there are several application areas where people want very rapid map generation and display. An aircraft pilot would benefit greatly from a built-in cockpit display of a map (See a simulated version in Figure 1). The display would show the plane's current position and heading on a backdrop of key geographical features and suggestive political boundaries. A ship's captain could navigate treacherous waters more safely and efficiently with an automatic map display which showed the ship's position relative to riverbank or shoal. One can also imagine overlaying a radar PPI display with additional data such as beacon and landmark positions as well as channel markers and even depth contours. Commercial fishing fleets would greatly benefit from map information showing, for instance, George's Bank overlaid with up-to-date LANDSAT data of most likely fish concentrations.

This borders on science fiction, if one must store and generate maps the way cartographers do, where great accuracy and fine displays are required. For the class of applications we are addressing, however the criteria are very different. The maps needed by the users mentioned above need only be highly INFORMATIVE and RECOGNIZABLE. One can reduce the storage requirements of a map many-fold, and yet still generate realistic-looking maps of great UTILITY.

The issue we address in this paper is: how can one reduce by about two orders of magnitude the number of data points that must be stored in a map database, and yet still rapidly generate maps which are both highly recognizable and useful to a human observer? The maps must not only capture IMPORTANT features accurately, they must also "look right". For instance, a coastline or riverbank must not be represented as just a few crude straight lines, it must undulate smoothly or be rugged as appropriate to match the original entity, or else the user will be distracted and irritated by the artifacts introduced.

II. EXAMPLE DESIGN GOAL: ON-BOARD MAP/NAVIGATION SYSTEM

As a way of motivating the approach treated here, consider the problem of rapid display of maps on board an aircraft, as suggested in

Figure 1 below. The plane has a conventional instrument panel, except there are fewer meters and dials. Mounted instead is a color video screen, and a keypad for easy interaction. Elsewhere in the plane are a small computer and mass storage devices (such as floppy or Winchester disks).

Prior to a flight the aircraft custodian loads in a map dataset relevant to the route. Map data for contingency routes are also loaded. During preflight checkout the pilot can browse through the maps at the touch of a few buttons. During flight the pilot activates the on-board OMEGA navigation receiver, and his current position flashes on each map. The displayed map scrolls automatically to keep the plane's position centered on the screen. The pilot can cause the display to zoom in or out, and to pan around at will.

If the pilot must make a course change into an unanticipated region, he need only insert the appropriate floppy disk to load in the new map data, or request new map information to be radioed from a nearby base, and this data is automatically stored in the on-board map database. As the plane finally nears its destination landing information is displayed. Current "approach plates" can be called up instantly: no longer must the pilot fumble through books of paper maps and approach plates to locate the proper one.

III. THE USUAL MAP GENERATION PROCESS

Consider standard map generation to see why the system envisioned above is not yet readily available.

Map data bases often consist of many blocks of data, each one an array (long (i), lat (i)), i=i,...NL of longitude/latitude real-number pairs. In most data sets the successive pairs are very closely spaced, since cartographers want a map to retain high accuracy even when only a very small region is expanded to full screen size.

To plot a new map the user specifies a rectangular "window" on the earth, and a particular viewing projection. The map generation scheme then simply treats the long/lat array as a "polyline", an unbroken sequence of straight line segments, clips it against the window, maps the window to the viewport in the usual fashion, and draws the straight lines between successive points.

By changing the window the user can "zoom" in or out on a particular area. For a large window the points in the polyline may appear very close together, while for a small window the straight lines are clearly visible, and can give the displayed map an artificial and unconvincing quality, with the insufficient "roughness" that the eye expects.

To enhance the naturalness of such coarse maps, "fractal curves" have been applied to map-making (3,4,5). Many shapes such as coastlines and river banks in nature are "self-similar": their apparent roughness is the same when one observes them at different scales. That is, the coastline of Maine has a certain "cragginess" when viewed from 100 miles up. When viewed from 10 miles up these crags become fleshed out and one now sees a more detailed level of cragginess superimposed, but the overall "sense of roughness" is nearly the same.

To "roughen" maps based on "sparse" line segments into a more realistic appearance, one "fractalizes" them into smaller line segments. Successive subdivisions of the segments are made, and the new midpoints are randomly perturbed as shown in Figure 2. The variance of each successive perturbation is controlled in such a fashion that the fractalized line appears self-similar. Below we describe how to enhance the usual fractalizing process to significantly assist the database compaction.

IV. NEW METHOD OF MAP COMPACTION AND REGENERATION

We propose to encode, store, and regenerate outline maps in two distinct phases:

a. "Mother" to "Daughter" database conversion: Each block of the original "mother" dataset is processed and re-encoded using chain-encoding (6) into an extremely compact array of bytes. This processing is done once at leisure for the entire mother dataset. A small amount of extra information is stored in the new chains to assist in both the compaction and regeneration.

b. Generation of Maps
For each desired map, the on-board daughter dataset is accessed to locate which chains overlap the window. As each chain is scanned, the links are fractalized to introduce the desired roughness for visual realism. Various hardware units and look-up tables are used in this phase to speed up the generation process.

IV a. Data Compaction Phase:

There are various ways to glean the important information from the parent data-base, and to generate the daughter data-base in a form which will aid in the map regeneration phase. One particularly effective way is to CHAIN ENCODE it (6), where one chooses a template such as that in Figure 3, with some "size" r (e.g. 0010 causes the chain to deviate by r in the northeasterly direction). For each array of long/lat pairs one finds the "closest" fitting chain of size r. The template shown

requires 4 bits per "link" to specify each new direction.

If the template size is large the chain will only approximate a sparse subset of the original data points. However, the daughter data set size has been vastly reduced: a large array of floating-point number pairs has been replaced by a much shorter array of 4-bit "nibbles". This is precisely the trade-off being studied here. Acceptable amounts of detail and accuracy are sacrificed in the service of compact datasets and rapid map generation. The smaller the template the more detail is retained, but the more links will be required for the chain. When r is smaller than the average distance between the original points the chain will match the original map rather well, although not exactly due to the discrete nature of the template.

In the encoding process richer templates might be used (e.g. one with 32 possible directions - requiring 5 bits per link) in order to reduce the "quantization noise" introduced by the template. Also one could encode direction CHANGES: each of 16 possible turns to the left being represented by a four-bit codeword. This is particularly useful if one wishes to ROTATE an entire map quickly (treated in a forthcoming paper (7)), as only the direction of the first link must be altered. For even greater coding efficiency, one could also use variable length codes to represent these relative direction changes, taking into account the relative probabilities of direction changes anticipated for a class of maps.

We add to the chain-encoding algorithm (and its possible variations) two new steps. During the search for the best chain, additional analysis is performed at two levels:

1) For the entire chain, the deviation of the original data points from the new chain is estimated statistically, to provide a measure of the "variance" of the map data and the "fractal parameter" H (3). This value is used in the fractal generation phase.

2) For each link of the chain, a "sense" bit is retained (Fig. 4): "0" indicates that the bulk of the original data points are situated clockwise (CW) from the link direction, while "1" denotes a CCW deviation. The sense bit ensures that the fractal reconstruction is on the correct side of the link, and will cause the fractal curve to match more closely the original map. One can therefore use larger template sizes r in the compaction phase for a given map accuracy.

Chain Formatting:

To access key data rapidly in the daughter database one must format each chain carefully. We use a linked list of chains. Each chain yields a curve that is to be drawn without break, but successive chains need not be continuous. Each chain has a header containing: i) the ID naming the region for easy user access when desired, ii) a pointer to the next chain header allows rapid threading through the entire dataset, and iii) "extent" values giving the minimum and maximum values each for the longitude and latitude, to facilitate searching for chains which overlap the window. Also in the header are the fields for the starting point of the chain, the number of links in the chain, the template size r, and a random-number "seed" to insure that the same fractal curves are produced each time the chain is used. This seed can simply be a start index into a random number table which is loaded once into memory at the start of a session. Hence subsequent random numbers are not calculated but are simply found in successive elements of this circular buffer.

Various choices exist for formatting each link of the chain. With a 4-bit template and 1 sense bit it is natural to store three links per word on a 16-bit microcomputer. For other word sizes one can envision schemes which use richer templates, and imbed more "sense" bits to aid in the regeneration phase.

IV b. The Map Generation Phase:

A method for fast generation of a map from the daughter dataset is briefly discussed, omitting many details. We are presently exploring several hardware/software blends, as well as the various cost/speed/space tradeoffs.

A window is selected, and the parameters for the window-to-screen mapping are found and stored. The daughter database is scanned, and each chain that overlaps the window is placed in an "active chain table" (to facilitate rapid scrolling). The template size is combined with these parameters to build simple tables to rapidly calculate the number of pixels each link spans, and the level of fractalization required to bring the final edges down near a pixel size is found (3,4). The chain is then scanned, and each link is fractalized using the sense bit and look-up tables, again striving for speed. Clipping is done very simply in hardware on a link-by-link basis since links are very short. There is no need for a Bresenham's algorithm to draw lines, since the fractal scheme generates individual points to "turn on" in the frame buffer.

Some Results

Figures 5-7 show various maps for purposes of comparison. An original map (Fig. 5) of 96,000 bytes based on CIA's World Data Bank

is compared with a chain-encoded version (Fig. 6) which uses only 2300 links. The offensive line segments are quite noticeable in this version. Figure 7 shows the chain-encoded and fractalized version, again using only 2300 links. The similarity of this map with the original is striking, even though it uses only 1/60 the storage space. As the algorithms are fine-tuned in our future studies and better use is made of the sense bits, we expect another factor of 2 or 3 in the compression. These preliminary studies were carried out on a VAX-11/780 with DI-3000. In this environment the fractalized chain-encoded map required 1/3 the generation time of the original. The reduction in image generation time is due to far fewer disk accesses (one initial access to load the 2300 bytes) and the high locality of the fractal algorithm that exploits the cache on the VAX. Figures 8 and 9 show a comparison of a zoom into the northeast portion of the map. Once again the reduced dataset (Fig. 9) compares very favorably with the original (Fig. 8).

CONCLUSION

Rapid generation of maps using a microcomputer requires significant reduction in the number of points that must be dealt with. The method presented uses chain encoding to achieve typical reductions of two orders of magnitude in the amount of data required. Chain encoding alone achieves data reduction but leaves an unsatisfying map which lacks the naturalness expected of outline maps. Incorporating fractalizing overcomes this, and restores the naturalness, at the expense of some accuracy. The investigation is demonstrating that the inclusion of a sense bit in the link data greatly assists in the map generation and improves the overall quality, permitting a larger template size and further reduction in the dataset size (7).

REFERENCES:

1. A.H. Schmidt & W.A Zaft "Programs of the Harvard Univ. Lab. for Computer Graphics and Spatial Analysis," Display and Analysis of Spatial Data, John Wiley & Son, New York, 1978.

2. J.A.B. Palmer "Computer Science Aspects of the Mapping Problem," Display and Analysis of Spatial Data, John Wiley & Son, New York, 1978.

3. A. Fournier & D. Fussell "Stochastic Modelling in Computer Graphics" SIGGRAPH '80 Conference Proceedings, July, 1980, Seattle, WA.

4. L.C. Carpenter "Computer Rendering of Fractal Curves and Surfaces" SIGGRAPH '80 Conference Proceedings, July, 1980, Seattle, WA.

5. G. Dutton "A Fractal Approach to the Control of Cartographic Detail" Proc. Computer Graphics '80, Brighton, UK, pp. 371-381.

6. H. Freeman & J.A. Saghri "Comparative Analysis of Line-Drawing Modelling Schemes" Comp. Graphics & Image Processing, Vol 12, 1980, pp. 203-223.

7. S.E. Walker, Jr. & F.S. Hill, Jr. "Microprocessor Based Real Time Map Generation" in preparation.

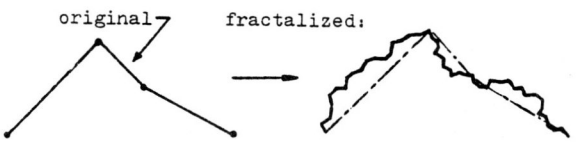


FIGURE 2. FRACTALIZING THE LINE SEGMENTS

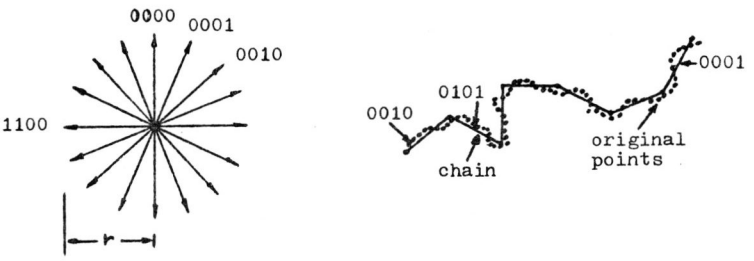


FIGURE 3. THE CHAIN TEMPLATE, & CHAIN ENCODING.

FIGURE 4. RETAINING THE SENSE BIT.

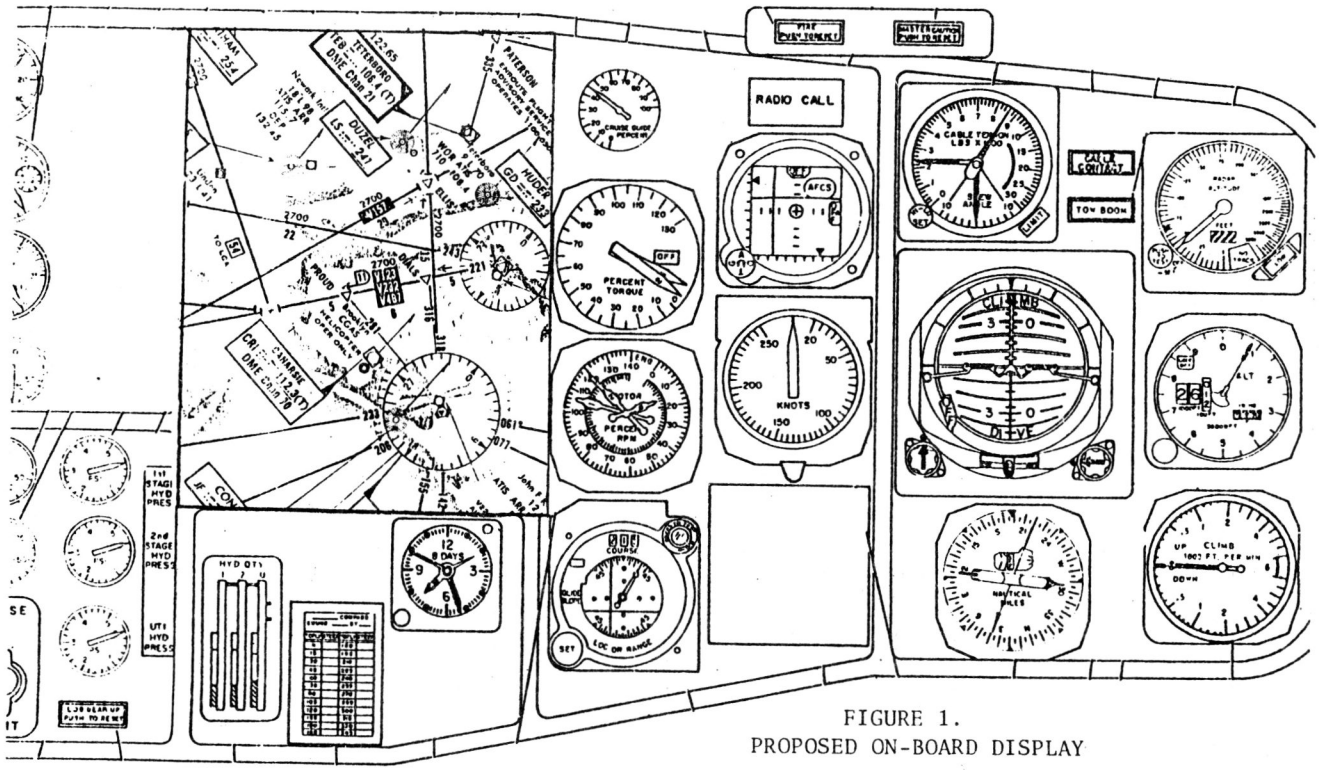


FIGURE 1.
PROPOSED ON-BOARD DISPLAY

FIGURE 5.
ORIGINAL MAP.

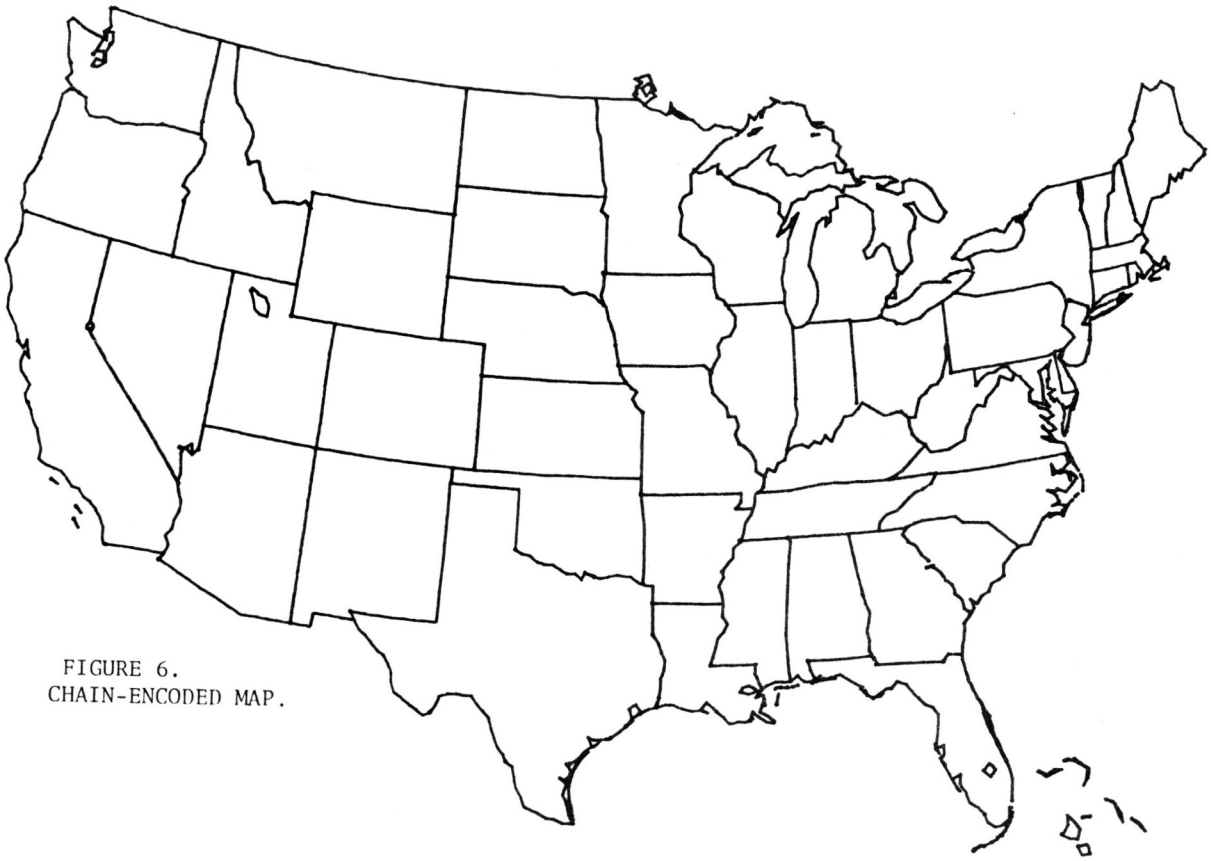
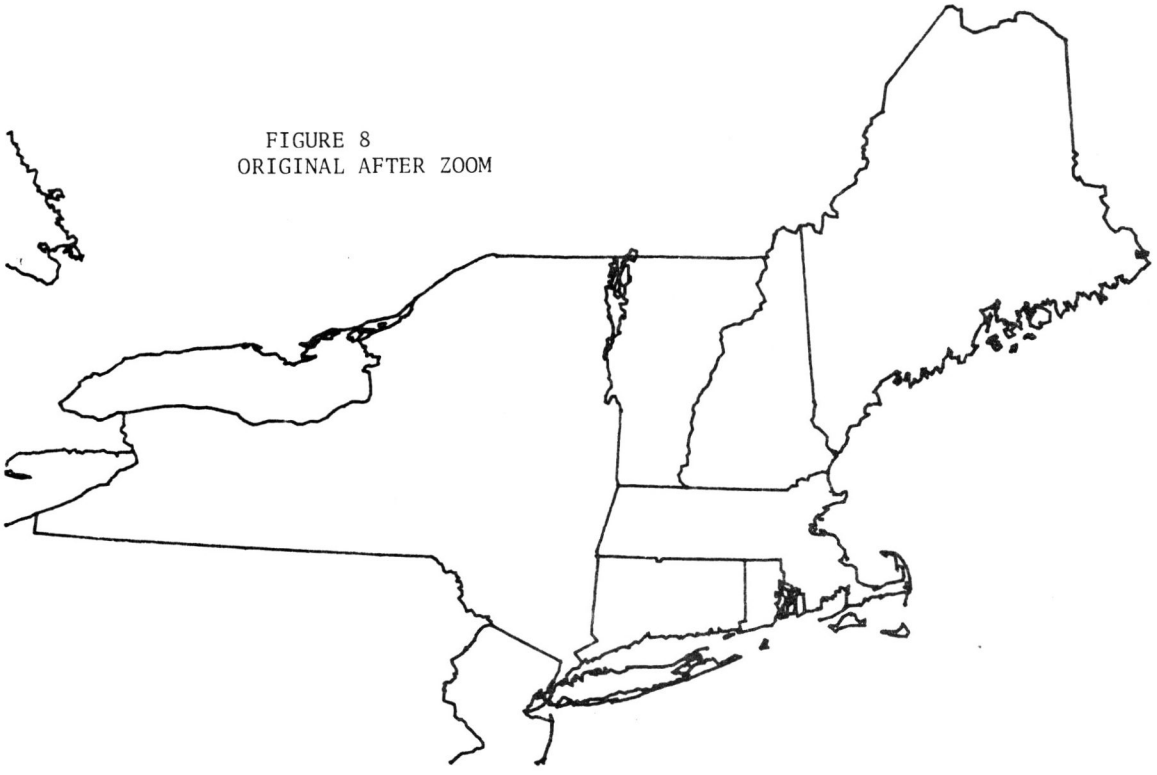


FIGURE 6.
CHAIN-ENCODED MAP.

FIGURE 8
ORIGINAL AFTER ZOOM



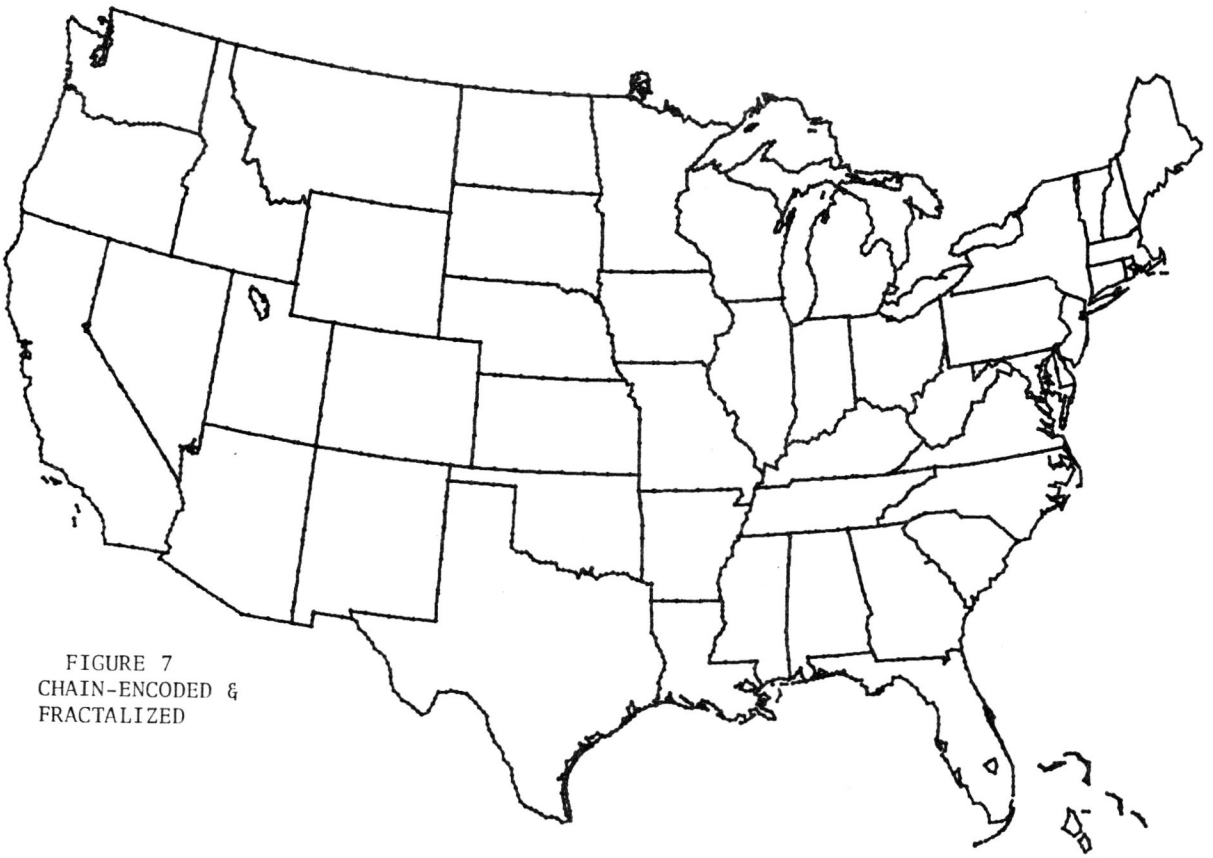


FIGURE 7
CHAIN-ENCODED &
FRACTALIZED

FIGURE 9
FRACTALIZED AFTER ZOOM.