EVALUATING THE ACCURACY OF LINE THINNING ALGORITHMS AFTER PROCESSING SCANNED LINE DATA

by

Loretta J. Bush

Thesis submitted to the Faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

MASTER OF SCIENCE

in

GEOGRAPHY

APPROVED:

L. W. Carstensen, Chairman

J. B. Campbell

R. W. Ehrich

April, 1993

Blacksburg, Virginia

EVALUATING THE ACCURACY OF LINE THINNING ALGORITHMS AFTER PROCESSING SCANNED LINE DATA

by
Loretta J. Bush

Committee Chairman: Laurence W. Carstensen
Geography

(ABSTRACT)

The development and rapid growth of computer mapping has led to many discussions concerning the accuracy of techniques used to generate these computer representations. The purpose of this study is to analyze the accuracy of thinning methods applied to scanned map data, which is only one in a series of processes used to accomplish digital conversion of conventional maps.

In preliminary tests, nine thinning methods based on the successive layer removal process are evaluated. Seven raster images are thinned using these methods. The raster results are compared based on the number of pixels deleted and on the number of retained pixels that fall either on or off the medial axis of the original matrix. The four algorithms that produce the best results are then used for final testing.

For the final tests, 25 digital lines are plotted and scanned. The raster images are thinned using the four successive layer removal methods and a line following method developed for this study. The raster output is evaluated using the preliminary testing method. The final vector output is compared to the original input based on line length, anchor line length, and fractal dimension.

ACKNOWLEDGEMENTS

I must first thank the two men who started me down this long path, which leads me back to Western Kentucky University. Ron Dilamarter (Dr. D) was my first geography professor. Through the knowledge he shared and the puns that kept me laughing, I was convinced that geography was the minor that I was looking for tc complement my math major. Conrad Moore taught my second geography class. His enthusiasm in the classroom convinced me that a minor was not enough, and so two majors became my goal. I took as many classes as my schedule would allow from these two men. I look forward to sharing this victory with Conrad at a future Geo Pig Roast in Kentucky. Unfortunately, Dr. D died of cancer before I graduated from Western, but he is still in my heart and in my work.

My thanks also extends back to Carol White, my friend and cooperative education advisor. With her help, I was able to get hands-on experience in computer cartography before graduation. This experience increased my desire to learn more about computer cartography. The support and encouragement that I received from Carol and Ben Keller gave me the confidence to continue my education in grad school.

Here at Virginia Tech, Bill Carstensen and Jim Campbell shaped my course of graduate study. Except for Jim's stats class, I enjoyed studying under these two men. The infinitely long, late nights in the

computer cart lab will always be etched in my mind (and in the minds of all the people who had to deal with me the next morning). I appreciate their support as my committee members. I know that they would have preferred me to have completed this thesis before accepting a job. It has been a tremendous personal struggle to find the motivation and the time to complete this paper.

I gratefully acknowledge the financial assistance provided by my graduate assistantship, General Dynamics, and CIT. I am also indebted to Ben Keller, Joe Uknalis, and Scott Rapier who out of friendship kept me fed during the third semester when the financial support was not quite enough. Finally, thanks to my parents for allowing me to "borrow" the money that made the last semester at Tech possible.

TABLE OF CONTENTS

PAGE

PAGE

LIST OF FIGURES

FIGURE PAGE

FIGURE PAGE

LIST OF TABLES

CHAPTER 1

INTRODUCTION

I. BACKGROUND

The development and rapid growth of geographic information systems (GIS) have created a requirement for inexpensive and rapid acquisition of digital data. Much of the information needed to operate an effective GIS is recorded in analog form as conventional maps (Berry,1987; Burrough,1986). Data recorded on paper maps can be converted to digital form using either a manual digitizer or an automatic raster scanner.

Manual digitizing can introduce a considerable amount of human error in the data. Three types of human errors are discussed by Jenks (1981): physiological, psychological, and logical. Physiological errors are created by muscle spasms that cause the operator's hand to involuntarily move the cursor from the desired position. Psychological errors occur when the operator can not observe the true center of the line or is unable to move the cursor exactly along its center. Logical errors exist if the operator does not properly select points essential to preservation of the character of the line. The extent of these human errors can be limited by shortening the number of consecutive hours that each operator must spend digitizing and by providing training in point selection. A portion of these errors can be eliminated in the editing phase, but there is no way to completely

purge the file of erroneous points. As long as manual digitizing is used to gather digital information, computer cartography will continue to be faced with these human errors.

Use of automatic raster scanners eliminates the tedious and exacting process of manual digitizing that leads to human error (Leberl and Olson,1982). A scanner consists of a high resolution sensor and a low powered light source. The scanner moves systematically across the source material. Levels of light reflected from the map determine the presence or absence of line features. Output is a binary matrix (Burrough,1986) in which 0 represents background and 1 represents linear features. The matrix may be processed in its basic form, or in a converted distance matrix format, as described by Rosenfeld and Pfaltz (Rosenfeld and Pfaltz,1966; Peuquet,1981; Arcelli and di Baja,1989). The distance value increases toward the center of the raster line from both horizontal and vertical directions (Figure 1a,b). Diagonal distance is not considered. This transformation is completed after two passes through the matrix. On the first pass, the matrix is read from top-to-bottom, left-to-right, and only neighbors directly above and left of the given pixel are evaluated. When a non-zero pixel is encountered, it is given the minimum value between these two neighbors, plus one (Figure 1c). On the second pass, the matrix is read from bottom-to-top, right-to-left, and the neighbors directly below and right are evaluated. When a non-zero pixel is encountered, it is given the minimum value among these two neighbors

```
  11111
 11111
11111
11111
11111
111111
 111111
  111111
  1111111
   1111111
    1111111
     111111
```

a. Binary Matrix

```
  11111
 12221
12321
12321
12321
123321
 123321
  123321
  1234321
   1233321
    1222221
     111111
```

b. Distance Matrix

```
-- P1 --
P2 P0 --
-- -- --
```

P0 = min (P1 + 1, P2 + 1)

c. First Pass

```
-- -- --
-- P0 P3
-- P4 --
```

P0 = min (P3 + 1, P4 + 1, P0)

d. Second Pass

Figure 1. Transformation of a Binary Matrix into a Distance Matrix

plus one and itself (Figure 1d). Thus, the medial axis of the raster line falls along the maximum distance values. With this method, a line of any width can be transformed such that the medial axis can be identified.

Software packages developed to process scanned data are not without problems. The perfect program that in one iteration can handle all the different colors, tones, dashed lines, and symbols that compose most maps without requiring operator interaction has not been -- and probably never will be -- developed. Printing defects, dirt, fingerprints, and creases can complicate the process. Scanning the color separates that form composite maps can help overcome most of these problems (Burrough,1986), assuming the separates are in good physical condition. A pre-scanning process used by the Canada Geographic Information System is to manually scribe lines so that they have distinct edges (Peuquet,1981), but this process is time-consuming and can lead to some of the same problems as manual digitizing. Since words and symbols create special problems for character recognition, map separates that lack these details produce better raster images. These features can be added after the map is in digital form with more precision and clarity. Some advanced post-scanning software that can recognize words and symbols cannot efficiently process characters and line features in the same operation. Separate passes through the data set must be conducted when using these software packages (Antenucci and Fain,1991).

Due to imperfections in conventional maps and the limitations of scanners, it is impossible to scan an image such that the raster lines are only one pixel wide. Subtle variations in line width easily generate raster lines thicker than one pixel or, more seriously, create breaks not present in the original. Therefore, the map must be scanned at a resolution coarse enough to ensure that no undesired breaks occur, which in turn generates thick lines. Because thick lines are not desirable for either analysis or display, scanned lines are thinned so that they become one pixel wide. Numerous algorithms used to conduct this thinning process are mentioned in literature and shall be discussed in greater detail in the next two chapters.

There is also the danger of scanning the lines too thickly. Excessively thick raster lines cause loss of line character during the thinning process. Scanning at course resolutions can cause lines that run close together, such as contours, to bleed into each other, creating extremely thick line segments. Lines that consist of tight curves can collapse on themselves, creating extensive line spurs when thinned. Intersections can be completely altered in form and position. Therefore, a balance must be achieved between scanning the image too thinly and creating breaks and scanning the image too thickly and losing line character. This balance depends on the nature of the image being scanned and the thinning algorithm being applied.

There are at least three approaches to thinning raster lines: successive layer removal, line following, and edge detection. The

layer removal method processes the image by starting with the outer layer of the line pixels then moving inward with each successive iteration. In this way, edge pixels are gradually deleted until the remaining skeleton is unit width (Peuquet,1981; Baruch,1988; van Vliet and Verwer,1988). Line following methods select points along the center of the raster line (Baruch,1988; Fulford,1981; Peuquet, 1981). Raster-to-edge methods detect transitions between raster regions and map background. These edges are converted to vector line casings. The midpoints or centerline data between the edge casings are computed, and then the edges are deleted (Holmes,1991).

A. Quality of Data

Another source of potential error when converting conventional maps to digital form is error associated with the map itself (Burrough,1986). Since paper maps are human representations of geographical phenomena, they contain human errors. These errors are compounded by errors introduced during the conversion process, be it digitizing or scanning. Therefore, it is extremely important that high quality source material be collected for use in any geographic information system. Input data determine the potential level of accuracy and precision that the computer image can achieve (Blakemore, 1984; Jenks,1981).

Chrisman (1982) discusses the difficulties involved in collecting quality information for the base data in a GIS. Ideally source

material should contain a reliability diagram along with a description on how the information has been collected. The reliability diagram should be converted to a reliability overlay within the GIS, which corresponds to the computer map file. Data collection descriptions should be readily accessible if any questions should arise pertaining to methods used or comparability of map layers. The reliability diagram is useful in ensuring that a quality base map is converted to digital form; plus the reliability overlay gives the GIS operator guidelines as to the validity of the data. These information sources can be beneficial in establishing and maintaining routine procedures designed to update base map data. Unfortunately efforts to implement such quality control and maintenance practices have been greatly neglected in the development of most present geographic information systems (Chrisman,1982; Sugawara,1989). Analyzing the converted data, correcting any errors, and adjusting the documentation can require up to 25 percent of the time allotted for establishing the initial map data base (Antenucci and Fain,1991).

When using scanned map data, information should be available concerning the map source, the scanning resolution, and the possible use of any filters. Once the scanned data have been skeletonized, a description of the thinning and/or vectorization method(s) should be provided. The advantages and disadvantages of implementing the methods and the effects they have on the reliability of the data should be specified. Most research mentioned in literature concerning line

thinning addresses the problem from the perspective of pattern recognition and is not primarily concerned with line accuracy. If scanned maps are to be used for analysis in geographic information systems, the accuracy of the thinned raster line needs to be addressed directly. The resulting skeleton should meet specific standards.

B. Standards For Thinning Raster Data

Hilditch (1969) outlines four basic requirements that thinning algorithms need to satisfy. These requirements are thinness, position, connectivity, and stability. First, the skeleton should consist of one unit wide raster lines. Second, the position of the line should not be biased towards one side of the line or the other; it should lie in the center of the scanned line. In cases in which the scanned line has an odd raster width, there is a definite raster center which the line should follow. When the scanned line has an even raster width, there is no true raster center and, out of necessity, the line must be shifted in one direction or the other by choosing one of the two pixels that make up the center of the raster line. The third requirement is that the connectivity of the image should be preserved so that no gaps appear in the final image that are not present in the original. The final requirement, stability, is in regard to two situations. First, the algorithm should not erode away the endpoints of the lines once they have been defined to a width of one pixel. And second, the last point within a circular image should be preserved.

In terms of cartography, circular image stability mainly refers to scanned point data. Point features are some of the least important features to be collected when scanning a hardcopy source. Unless the source is in pristine condition, attempting to distinguish point data from noise is a futile and time-consuming endeavor. It is more expedient and accurate to manually digitize point features from the hardcopy source after processing the scanned information. Therefore, in practice, circular image stability is the least significant requirement outlined by Hilditch. All other requirements specified are equally important when processing scanned maps as they are when processing scanned chromosomes, which is Hilditch's intent.

There is an additional requirement for the skeleton that is not addressed by Hilditch or most of the other authors reviewed. This omission is probably due to the fact that most of them did not examine map data. If a person is working with two map sheets of adjacent areas, he would expect to be able to follow a line from one map to the other. Therefore, in terms of connectivity, the algorithm should not only preserve the connectivity within the interior of the line, but it also should preserve the connectivity between maps where lines cross map edges. Consequently, computer map files can be linked together for both analysis and display purposes without substantial editing. Unfortunately, creating a seamless data base is a difficult task even with a perfect thinning method. The maps scanned do not always provide

adequate information to successfully align the map edges (Antenucci and Fain,1991).

C. Processing Methods

Sequential processing can greatly alter the position of thinned lines. With this method, when it is determined that a pixel can be deleted, it is immediately removed, directly effecting the neighboring pixels and resulting in lines completely shifted to the edge of the raster line. The alternative, parallel processing, eliminates this problem by evaluating each pixel independent of the status of its neighbors. One thinning method presented uses sequential processing (Chapter 2, Section E). All other methods mimic parallel processing. When it is determined that a pixel can be deleted, the pixel is not removed immediately but is flagged for deletion by recoding it to a value not used in the original matrix. Once one complete pass is made through the matrix, all pixels flagged for deletion are removed, thereby promoting more centrally located skeletons.

D. Noise Effects

Another major concern is the effect of scanning noise on the thinning process (Chen and Hsu,1988; Arcelli and di Baja,1989; Hilditch,1969; Greenlee,1987; Holt et. al.,1987; Deutsch,1972). Image noise appears as slight changes along edges of raster lines. It often consists of protrusions or intrusions of only one pixel. If the

algorithm is sensitive to these subtle changes, the position of the final line may be shifted or small line spurs generated.

Spurious pixels can be eliminated at different processing phases. Filters can be used during the scanning process. Convolution filters can compress the intensity of the histogram or minimize the effects of ink diffusion. Histogram filters average the values of neighboring pixels in 2x2 to 8x8 blocks. Cleanup filters assign the intensity of the center pixel to neighboring pixels in 3x3 or 5x5 blocks (Baumann,1991). Once scanned, a smoothing routine can be applied before the thinning process to remove slight protrusions (Hilditch, 1969; Xia,1989). Arcelli and di Baja (1989) locate insignificant protrusions during processing by calculating the distance of the protrusion from the core of the raster line. In postprocessing, raster lines identified as significantly short can be deleted. Similarly, after vector conversion, the length of the lines can be calculated, and spurs discarded (Greenlee,1987).

From working with images of chromosome spreads, Hilditch (1969) confronts the problem of disjoint segments touching in places after scanning. To overcome this difficulty, the holes created are expanded so that they connect with the background pixels, thereby disconnecting the scanned segments. In other cases, holes and border intrusions caused by imperfections in the lines being scanned need to be eliminated (Greenlee,1987; Arcelli and di Baja,1989). Greenlee provides a lookup table designed to fill in these small voids that can

alter the position of thinned raster lines. A value is assigned to each 0-pixel based on position and status of the eight neighboring pixels (Figure 2). If the value equals an entry in the table, the pixel is changed from 0-background to 1-line.

E. Scale and Resolution

The existence of numerous professional studies on line thinning methods illustrates the difficulty of developing an algorithm perfect for all situations. There will always be some final pixels that do not fall on the exact center of the raster line. The goal is to minimize these pixels. The amount of mapping error that is introduced by these pixels is determined as much by the scale of the map and the resolution of the scanning process as it is by the thinning method being used.

The larger the map scale, the smaller the area covered by each pixel. A one pixel shift on a 1:24,000 map constitutes less error than a one pixel shift on a 1:1,000,000 map, assuming they are scanned at the same dots per inch (dpi). The greater the scanning resolution, the smaller the area covered by each pixel. A one pixel shift on a 1:24,000 map scanned at 150 dpi introduces half the error of a one pixel shift on the same map at 75 dpi. Therefore, the measured error is both scale and resolution dependent. Whether or not any of these errors are significant depends on how the thinned maps are used. Simply increasing the scanning resolution to decrease the amount of error caused by an individual pixel is not necessarily the best

order of neighbors:

```
1 2 3
4 P 5
6 7 8
```

value of neighbors:

```
64 128 1
32  P  2
16  8  4
```

example neighborhood:

```
0 1 1
1 P 1
1 1 0
```

value of P: P = 128+1+32+2+16+8 = 187

result: fill void

Figure 2. Greenlee's Value Assignment

solution. An increase in scanning resolution also increases the thickness of the raster lines generated. As mentioned previously, excessively thick raster lines cause loss of line character during the thinning process, potentially increasing the quantity of erroneous pixels.

The width of the lines before scanning should influence the scanning resolution. One resolution may create some thick raster lines while generating gaps in others. Therefore, the scanning resolution should be selected based on the width of the thinnest line that needs to be extracted from the hardcopy source. The resolution should ensure that, after scanning, these lines are at least the minimum raster width desired to convert them into continuous line data. The scale of the map should dictate a minimum scanning threshold to limit the amount of potential error produced by a single pixel. In this manner, the final raster file will contain all the required information at the lowest resolution possible to provide quality output.

II. PURPOSE AND OUTLINE

Since much of the information needed to operate an effective geographic information system is recorded in analog form as conventional maps, the established practice of utilizing automatic scanners to convert these maps to digital form will continue to prosper. Most research mentioned in literature concerning line

thinning addresses the problem from the perspective of pattern recognition and is not primarily concerned with the cartographic perspective of line accuracy. This study is an attempt to bridge the research and literary gap between pattern recognition and line accuracy. Since scanned maps are being used for analysis in geographic information systems, the accuracy of the thinned raster line must be addressed directly.

The purpose of this study is to analyze the accuracy of thinning methods applied to scanned map data, which is only one in a series of processes used to accomplish digital conversion of conventional maps. Even though filling, expanding, and smoothing techniques can improve the results obtained by thinning, none are employed in this study. The influence of these editing techniques add other factors to the analysis process, making it difficult to identify the origin of any errors introduced during processing. Errors that may be present in the source data are also disregarded. The means for gaging the accuracy of the source material is not generally available. To simplify the analysis process, digital lines used as input are assumed to be accurate. Hence, the purpose of this study is to only evaluate the amount of error introduced by the scanning and thinning processes. The line thinning standards outlined by Hilditch are used as the foundation for evaluating the effectiveness of the algorithms implemented in this study.

Lines are scanned to create binary raster matrices in which 0's represent background and 1's represent line components. Several thinning algorithms reported in literature and one of my own are applied to thin raster lines to unit width. Depending on the algorithm, either a binary or distance matrix is used (Chapters 2 & 3). Error measures are computed to determine the effect each thinning process has on the position and character of the lines (Chapter 2 & 4).

In the preliminary testing phase, nine thinning methods based on successive layer removal are evaluated. Seven raster images are thinned using these methods. The results are compared based on the number of pixels deleted and on the number of retained pixels that fall either on or off the medial axis of the original matrix (Chapter 2). The four algorithms that produce the best results are then used for final testing. The line following method which I developed is evaluated during final tests. The edge detection method is not tested, since an algorithm has not been located in current literature to form a base for the program.

For the final testing phase, 25 digital lines are plotted, scanned, and then thinned by the five methods. The raster output is evaluated using the preliminary testing method. The final vector output is then compared to the original input based on line length, anchor line length, and fractal dimension (Chapter 4). The raster and vector results are combined to clarify the overall accuracy of each method tested (Chapter 5).

CHAPTER 2

SUCCESSIVE LAYER REMOVAL

I. BINARY MATRIX

Algorithms that utilize the successive layer removal principle have two things in common. They operate on a 3x3 pixel window, and they are used on binary matrices. Therefore, this section deals primarily with methods that fall in this category. Variations among results from the different algorithms arise from the manner in which variables are calculated and definitions of delete conditions. Variables evaluated usually consist of the number of line neighbors, the crossing number, and whether or not a gap would be formed if the pixel is removed.

The crossing number is the number of black and white transitions around a given pixel. One straightforward method of determining this value is to compare each pair of adjacent neighbor pixels (P1 to P2, P2 to P3,..., P8 to P1) while circumventing the pixel in question (P0) in a clockwise direction (Figure 3). If the pair does not have the same values (either a 0-1 or 1-0 transition), then the crossing number is incremented by one. Not all algorithms use this exact method to calculate the crossing number, but the basic principle remains the same. The number of line neighbors is the number of non-zero pixels among the eight neighbor pixels.

order of neighbors:

P1	P2	P3
P8	P0	P4
P7	P6	P5

example neighborhood:

1	1	0
1	1	0
1	0	0

crossing number = 2

line neighbors = 4

example neighborhood:

1	0	1
0	1	1
0	1	0

crossing number = 6

line neighbors = 4

Figure 3. Crossing Numbers and Number of Line Neighbors

Six algorithms have been programmed from information provided in professional journals. The programs are written in Turbo C for execution on an AT&T PC 6300. The articles selected include Hilditch (1969), Stefanelli and Rosenfeld (1971), Deutsch (1972), Chen and Hsu (1988), Greenlee (1987), and Landy and Cohen (1985). An application of each method is provided after its description by thinning the raster lines illustrated in Figure 4.

The thinned matrix is visually evaluated on eight aspects. The first four correspond to the requirements outlined by Hilditch (1969). All of the algorithms preserve connectivity.

A. Are lines thicker than one pixel wide?

B. Are lines centered?

C. Are endpoints extensively eroded?

D. Are circular features visible?

Four questions, developed after reviewing lines thinned during the developmental stage and observing shortcomings within the images, also are included.

E. Are intersections intact?

F. Are line positions influenced by noise?

G. Is the 90 degree corner distinguishable?

H. Are additional line segments present?

```
111
 111
 111
  111
   111
   111
    111
    1111            1111
     111            1111
      111            111
      1111          1111
      1111          1111
       1111         111
       1111          111
        111         1111
        111         1111
        111         1111
        111         1111
        111         1111
        111         1111
        1111        1111
1111111111111111111111111111111
1111111111111111111111111111111
1111111111111111111111111111111
1111111111111111111111111111111
        1111                111
        111                 111
        111                 111
        111                 111
        111                 111
       1111                 111
        111                 111
        111                1111
        11                1111
                         1111
                        111
     11                111
    1111              111
    1111             111
     11             111
                   111
```

Figure 4. Test Matrix 1

Only algorithms that produce the best results will be used in the final testing phase.

A. The Hilditch Algorithm

The well-known and frequently cited thinning algorithm developed by Hilditch (1969) is used in this study. Within the algorithm four initial variables are computed. First, she reasons that a pixel should lie on the edge of a line before deleting it; therefore, at least one axially adjacent neighbor must equal zero. To ensure that endpoints of thinned lines are not eroded away and that circular features do not completely disappear, two similar variables are calculated. To prevent endpoint erosion, there must be at least two line neighbors before deletion. To prevent the disappearance of circular features, only line neighbors not flagged for deletion are included in a second count, and one or more neighbors are required for deletion. Finally, the crossing number is computed.

In addition to the standard crossing number, another crossing number needs to be computed. If the pixel under consideration for deletion has any axially adjacent neighbors that are flagged for deletion, a second crossing number is determined by temporarily removing these neighbors. This second crossing number ensures that connectivity is not altered by removing a combination of flagged pixels. All flagged pixels are permanently removed after one complete pass through the matrix. (Appendix A)

As illustrated by Figure 5, the algorithm completely satisfies all requirements considered significant by Hilditch (A thru D). This does not necessarily mean that the image is perfectly thinned. There is a distortion of the T-intersection (E). There is also a minor shift in line position due to noise (F).

B. The Modified Stefanelli-Rosenfeld Algorithm

The method which I refer to as the modified Stefanelli-Rosenfeld algorithm has gone through several modifications. Originally, in 1971, Stefanelli and Rosenfeld introduced the method as a simplified version of the Hilditch algorithm described in the previous section. As in the Hilditch algorithm, the number of non-zero neighbors and the standard crossing number are determined. The number of neighbors has to be greater than one and less than seven and the crossing number has to be one before considering the pixel for deletion. The simplification consists of two products and two corresponding crossing numbers. If the first product (p2 * p0 * p6) is greater than zero, a crossing number is computed after shifting the 3x3 processing window up one pixel (Figure 6). If the second product (p0 * p6 * p4) is greater than zero, a crossing number is computed after shifting the window left one pixel. Either the product equals zero or the corresponding crossing number does not equal one before flagging the pixel for deletion. The products and crossing numbers are intended to preserve connectivity.

```
---
 ---
 -1-
  -1-
   -1-
   -1-
    -1-
    --1-            ----
     -1-            ----
      -1-            -1-
      --1-          --1-
      --1-          --F-
       --1-         -F-
       --1-          -F-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        --1-        --1-
----------1----------E--------
----------1--------1E-E-------
--11111111311111111----11111--
----------1-----------------1-
        --1-                -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
       --1-                 -1-
        -1-                 -1-
        -1-                --1-
        --                -11-
                         -1--
     --                 -1-
    ----               -1-
    --1-              -1-
     --              -1-
                    ---
```

-	Deleted Pixel		
1	Line Pixel	E	Altered Intersection
3	Intersection	F	Effect Of Noise

Figure 5. Matrix Thinned by the Hilditch Algorithm

```
order of neighbors:          P7 P0 P1
                             P6    P2
                             P5 P4 P3


shift processing window:     p7 p0 p1
                             p6 P0 p2
                             p5 p4 p3
                             -- -- --
```

Figure 6. Stefanelli and Rosenfeld's Processing Window

After programming this version and discovering that connectivity is not preserved, Carstensen has added two more sets of products and crossing numbers to rectify the problem. If the third product (p2 * p4 * p6) does not equal zero, the processing window is shifted down one pixel and the crossing number is computed. If the fourth product (p0 * p2 * p4) does not equal zero, the window is shifted right one pixel. The delete conditions remain the same as for the first two pairs. This modification corrects the connectivity problem. Consequently, the final method has very few similarities to the original Hilditch method. The Turbo C version of Carstensen's FORTRAN program is used in this study. (Appendix B)

The modified Stefanelli-Rosenfeld algorithm has one drastic problem, namely erosion of endpoints. As shown in Figure 7, a complete line and intersection are missing from the final image, which is not unusual for this method (C,E). Other problems with the example includes diagonal lines of two pixel width (A) and the disappearance of the circular feature (D). A minor problem includes a slightly altered 90 degree corner (G).

C. The Deutsch Algorithm

In 1972, Deutsch developed three thinning algorithms to process rectangular, hexagonal, and triangular arrays. Mainly, Deutsch was interested in greatly reducing the number of picture elements needed to facilitate accurate character recognition. The algorithm designed

```
---
 -1-
 -1-
  -1-
   AA-
   -1-
    AA-
    -AA-            ----
     -1-            ----
      AA-            ---
      -1--          ----
     • --1-         ----
        -1--        ---
        --1-         ---
         -1-        ----
         -1-        ----
         -1-        ----
         -1-        ----
         -1-        ----
         -1-        ----
         -1--       ----
---------1--------------------
----C11113111111111CE1111G---
---------1-----------------GG-
---------1------------------1-
         -1--                -1-
         -1-                 -1-
         -1-                 -1-
         -1-                 -1-
         -1-                 -1-
        --1-                 -1-
         -1-                 -1-
         -1-                -AA-
         --                -AA-
                          -1--
     --                  -1-
    ---- D              -1-
    ----               -1-
     --               -1-
                     -1-
```

-	Deleted Pixel	C	Eroded Endpoint
1	Line Pixel	D	No Circular Feature
3	Intersection	E	Altered Intersection
A	Line Too Wide	G	No Corner

Figure 7. Matrix Thinned by the Modified Stefanelli-Rosenfeld Algorithm

to work on hexagonal arrays was found tc produce the best overall results. The algorithm for triangular arrays was considered the least effective. For both comparison and convenience, however, the algorithm for rectangular arrays has been used in this study.

Deutsch's algorithm uses two subiterations to preserve the connectivity of the lines. The crossing number and the number of line neighbors are determined, accompanied by several additional computations. The variables considered constitute eight different deletion patterns.

During the first pass through the matrix, four categories of deletion patterns are considered. For a crossing number equal to one, the algorithm only deletes pixels on the south and east sides of the line. At least one of the three axially adjacent neighbors, designated by X in Figure 8a, has to equal zero in both patterns before the pixel is marked for deletion. If the crossing number equals two, at least one of the corners, designated by Y in Figure 8b, has to equal one. Once a complete pass has been made through the matrix, all pixels marked for deletion are removed.

For the second pass, the mirror images of the above deletion patterns are used. If the crossing number equals one, pixels on the north and west sides of the lines are considered. At least one of the pixels, designated by X in Figure 8c, has to equal zero. For a crossing number of two, at least one Y in Figure 8d has to equal one

First Subiteration:

```
- X -          - - -        At least one
- * X   and    X * X        X equals 0
- X -          - X -
```

a. Crossing Number Is One

```
Y 1 0          0 0 Y        At least one
0 * 1    or    0 * 1        Y equals 1
0 0 Y          Y 1 0
```

b. Crossing Number Is Two

Second Subiteration:

```
- X -          - X -        At least one
X * -   and    X * X        X equals 0
- X -          - - -
```

c. Crossing Number Is One

```
Y 0 0          0 1 Y        At least one
1 * 0    or    1 * 0        Y equals 1
0 1 Y          Y 0 0
```

d. Crossing Number Is Two

Figure 8. Deutsch Delete Conditions

before flagging the pixel. The algorithm switches between these two subiterations until no more pixels can be deleted.

This process retains 90 degree angles. If these corners are not considered to be vital to the character of the lines, Deutsch provides a final option to delete these pixels. In this way the algorithm can be slightly adapted to meet the needs of the user. (Appendix C)

Thinning the example lines with the Deutsch algorithm raises the question of identification of correct intersections. Neither of the intersections in Figure 9 consists of aligned 90 degree angles, but are they completely wrong? In the context of raster lines, concessions must be made for restrictions imposed by the pixel unit. Even though the intersections are not exactly joined, it is difficult to judge their correctness. The cross intersection has one pixel that is not consistent with the general pattern (B), but the intersection as a whole is not wrong. The T-intersection is wrong, not because it is shifted one pixel but because the shift creates an obtuse angle instead of the desired right angle (E).

Other problems exist with the Deutsch algorithm. Diagonal lines are not always thinned adequately (A). This can be overcome by adding the option to delete 90 degree angles (Figure 10). With or without this option, the thinning process adds an extra hook to the end of some lines (H). Finally, the circular feature is present, but it is a short line (D) instead of a single point.

```
---
 -1-
 -1-
  -1-
   -1-
   -1-
    -1-
    -1--            ----
     -1-            HH1-
      -1-            -1-
      -1--          --1-
      -1--          --1-
       -1--         --1
       -1--          -1-
        -1-         --1-
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1--        -1--
---------1----------1---------
-------13311--------E11-------
-111111-B---1111111E---111111-
---------1------------------1-
        -1--                -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
       --1-                 -1-
        -1-                 -1-
        -1-                --1-
        -1                -11-
                         -1--
     --                 AA-
    ----               -1-
    -DD-              AAA
     --              -1-
                    HAA
```

-	Deleted Pixel		
1	Line Pixel		
3	Intersection	D	No Circular Feature
A	Line Too Wide	E	Altered Intersection
B	Off-Centered LIne	H	Addition To Line

Figure 9. Matrix Thinned by the Deutsch Algorithm with 90 Degree Angles Retained

```
---
 -1-
 -1-
  -1-
   -1-
   -1-
    -1-
    --1-            ----
     -1-            H1--
      -1-            -1-
      -1--          --1-
      --1-          --1-
       -1--         --1
       --1-          -1-
        -1-         --1-
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1--        -1--
---------1----------1---------
-------13311--------E11-------
-111111-B---1111111E---11111--
---------1-----------------1-
        -1--                -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
       --1-                 -1-
        -1-                 -1-
        -1-                --1-
        -1                -11-
                         -1--
     --                 -1-
    ----               -1-
    -DD-              -1-
     --              -1-
                    H1-
```

-	Deleted Pixel		
1	Line Pixel	D	No Circular Feature
3	Intersection	E	Altered Intersection
B	Off-Centered Line	H	Addition To Line

Figure 10. Matrix Thinned by the Deutsch Algorithm with 90 Degree Angles Deleted

Zhang and Suen (1984) propose a simple thinning method that is basically a truncation of the Deutsch method. Their method is based on 4-neighbor connectivity, whereas all methods discussed thus far address 8-neighbor connectivity. Only conditions listed above for a crossing number of one are used. In the first subiteration pixels on the south and east edges of the lines are deleted (Figure 8a), followed by the pixels on the north and west edges in the second iteration (Figure 8c). Therefore, diagonal lines and curves of two pixel width, along with 90 degree angles, remain. Due to these problems, this algorithm is not programmed for this study.

D. The Chen-Hsu Algorithm

Chen and Hsu (1988) present an expanded version of the Zhang-Suen algorithm. The only notable variation between this method and the original Deutsch algorithm is that the deletion of 90 degree angles is part of the main process, instead of forming a final cleanup option. Therefore, when the crossing number of two is encountered, the pixels designated by Y in Figures 11a and 11b, can be any combination of ones and zeros. This one difference does produce noticeably different results. (Appendix D)

The Chen-Hsu algorithm appears to form a visible improvement over the Deutsch algorithm. Intersections in Figure 12 are aligned at 90 degree angles, and extra line extensions disappear. There is still a

```
Y 1 0          0 0 Y        Y can equal
0 * 1    or    0 * 1        either 1 or 0
0 0 Y          Y 1 0
```

a. First Subiteration

```
Y 0 0          0 1 Y        Y can equal
1 * 0    or    1 * 0        either 1 or 0
0 1 Y          Y 0 0
```

b. Second Subiteration

Figure 11. Chen and Hsu Delete Conditions

```
---
 B--
 -1-
  -1-
   B--
   -1-
    B--
    -1--            ----
     -1-            --1-
      B--            -1-
      -1--          --F-
      --1-          -F--
       --1-         -1-
       --1-          F--
        -1-         --F-
        -1-         --1-
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1--        -1--
---------1----------1---------
--1111111311111111113111111--
---------1------------------1-
---------1------------------1-
        -1--                -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
       --1-                 -1-
        -1-                 -1-
        ---                -1--
        --                -1--
                         -1--
     --                 -1-
    ---- D             -1-
    ----              -1-
     --              -1-
                    -1-
```

-	Deleted Pixel	B	Off-Centered Line
1	Line Pixel	D	No Circular Feature
3	Intersection	F	Effect Of Noise

Figure 12. Matrix Thinned by the Chen-Hsu Algorithm

problem with the circular feature (D). In addition, single pixels move off-center more frequently (B) and noise has a negative influence (F).

E. The Greenlee Algorithm

Delete conditions outlined by Greenlee (1987) are based on the lookup-table principle previously described in reference to Greenlee's method of filling small voids in raster lines (Chapter 1). Greenlee lists very specific thinning rules. Once programmed, the method deletes <u>every</u> pixel in the matrix. In observing the behavior of the algorithm, conditions listed allow endpoints to be deleted after the lines are thinned to one pixel width.

This misrepresentation is a disappointing development. The Greenlee algorithm is the only method found during my literature review that is specifically designed to work on map data and provides enough information to program. None of the other successive layer removal methods investigated in this study are designed with cartographic purposes in mind. Most of the research on cartographic line thinning methods is being conducted by private industry and is protected by proprietary rights.

F. The Landy-Cohen Algorithm

Landy and Cohen (1985) worked on a project addressing the feasibility of developing a visual communication system for the deaf and hearing impaired. The purpose of their study was to produce

"intelligible" images of the hand gestures included in the American Sign Language (ASL). Landy and Cohen stated that the quality of the thinned images was not the primary importance as long as speakers of ASL could understand the transmitted images.

The Landy-Cohen algorithm is divided into two stages. The first phase accomplishes the majority of thinning. Only two variables are computed in this stage: the crossing number and the number of line neighbors. A pixel is deleted when the crossing number equals two and the number of neighbors is greater than two and less than six. These conditions are applied until no more pixels can be deleted.

The second phase is the cleanup process. For this stage, a third variable is computed to determine whether a gap will be created if the given pixel is removed. Diagonal lines and curves that remained two pixels wide after the first stage are thinned to one pixel width. Additional points are also deleted from intersections without altering the connectivity. (Appendix E)

As evident by Figure 13, results from the Landy-Cohen algorithm display numerous problems. Lines are not always centered (B). Intersections are generated at the ends of lines (H) sometimes due to noise (FH). Desired intersections are altered (E). Circular features become lines (D). Basically, this algorithm is not designed to work on raster lines that average more than two pixels in width.

```
B--
 B--
 -1-
  -1-
   -1-
   -1-
    -1-
    --1-            ---H
     -1-            HH-H
      -1-            -3-
      --1-          --1-
      --1-          --1-
       --1-         --B
       --1-          -1-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        -1--        --1-
----------1----------1-------H
----------1----------E-------H
----------1-----------E1-----H
BBBBBBBBE-EBBBBBBBBBBE--BBBB-H
        -E--                -3-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
       FH3-                 -1-
        -1-                 -1-
        -1-                --1-
        1-                ---B
                         --BB
     --                 --B
    ---D               --B
    DDD-              --B
     --              --B
                    HHB
```

-	Deleted Pixel	D	No Circular Feature
1	Line Pixel	E	Altered Intersection
3	Intersection	F	Effect Of Noise
B	Off-Centered Line	H	Addition To Line

Figure 13. Matrix Thinned by the Landy-Cohen Algorithm

II. DISTANCE MATRIX

Three algorithms programmed are designed to thin matrices consisting of distance values (Chapter 1). Two of these methods are modifications of algorithms mentioned above, Landy and Cohen (1985) and Hilditch (1969). The third method comes from an article by Suetens, Dierckx, Piessens, and Oosterlinck (1981). The same test matrix (Figure 4) is used to illustrate the thinning results.

A. The Modified Landy-Cohen Algorithm

Since it appears that most of the problems associated with the original Landy-Cohen algorithm result from sequential processing, a matrix of distance values is used in an attempt to overcome this difficulty. The conditions for deletion remain the same as listed above, but the distance values control the deletion process. Now instead of searching for just any line pixels, the algorithm searches first for pixels with the value of 1. After all possible 1-pixels are removed, it searches for pixels with the value of 2, and so on. With this adjustment, the thinned lines move to a more central position, instead of shifting to the raster edge. The main drawback of this alteration is the increase in processing time, because more passes through the matrix are required.

As seen in Figure 14, the modification produces only slight improvements over the original Landy-Cohen algorithm. The modified process centralizes most of the lines (B). Problems with the

```
B--
 AA-
 -1-
  B--
   AA-
   -1-
    AA-
    -AA-            ---H
     -1-            H--H
      AA-            H3-
      -1--          --1-
      -AA-          -AA-
       -1--         -1-
       -AA-          -B-
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1--        -1--
---------1----------1--------H
---------1---------A31A------H
--11111113111111111A--A111113-
BB-------1------------------1-
        -1--                -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
       FH3-                 -1-
        -1-                 -1-
        -1-                -AA-
        H1                -AA-
                         --1-
     --                 --B
    --D-               --B
    --D-              --B
     -D              --B
                     HHB
```

-	Deleted Pixel	B	Off-Centered Line
1	Line Pixel	D	No Circular Feature
3	Intersection	F	Effect Of Noise
A	Line Too Wide	H	Addition To Line

Figure 14. Matrix Thinned by the Modified Landy-Cohen Algorithm

intersections are solved. Line spurs are still generated to the same extent (H,FH), and the circular feature is still shown as a line (D). In addition, double wide lines are introduced (A). Hence, the modification made in the Landy-Cohen algorithm does not bring about the desired improvements. Problems concerning delete conditions remain.

B. The Modified Hilditch Algorithm

When developing her line thinning algorithm as part of her analysis of images of chromosome spreads, Hilditch (1969) found that results are improved when the nature of the chromosome is taken into account. Chromosomes have density. The density is highest towards the middle, but not necessarily the true center, of the image, then reduces gradually outward. A distinct boundary is not always present. Therefore, Hilditch adjusts the algorithm to peel off pixels with a lower density first, forcing the thinned image to lie on the highest density ridge. Therefore, the Hilditch algorithm is modified to be used with a matrix of distance values to represent different densities. My modification is not expected to cause any major alterations, since the original method produces impressive results. Subtle refinements in the thinned line are expected.

In comparing Figure 15 to Figure 5 (original Hilditch results) there is only a one-pixel difference between the two examples; Figure 15 has an additional pixel at the end of the top left-hand line.

```
---
 -1-
 -1-
  -1-
   -1-
   -1-
    -1-
    --1-            ----
     -1-            ----
      -1-            -1-
      --1-          --1-
      --1-          --F-
       --1-         -F-
       --1-          -F-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        -1-         --1-
        --1-        --1-
----------1----------E--------
----------1--------1E-E-------
--11111111311111111----11111--
----------1-----------------1-
        --1-                -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
       --1-                 -1-
        -1-                 -1-
        -1-                --1-
        --                -11-
                         -1--
     --                 -1-
    ----               -1-
    --1-              -1-
     --              -1-
                    ---
```

-	Deleted Pixel		
1	Line Pixel	E	Altered Intersection
3	Intersection	F	Effect Of Noise

Figure 15. Matrix Thinned by the Modified Hilditch Algorithm

Hilditch found improvements because density matrices often have several layers of pixels with the same value. With distance matrices, layers uniformly increment towards the center. Therefore, processing time is increased, but results remain about the same.

C. The Suetens-Dierckx-Piessens-Oosterlinck Algorithm

Influenced by the work of Deutsch (1972), Suetens, Dierckx, Piessens, and Oosterlinck (1981) made two adjustments to the Deutsch algorithm. In processing photonegatives of line drawings, continuous photonegatives are converted to discrete arrays with pixel values ranging from 1 to 128. Pixels with lower density values are tested for deletion first. Therefore, a distance matrix is used again to represent density values.

An additional deletion condition is introduced. Suetens, Dierckx, Piessens, and Oosterlinck contend that if the crossing number is two then the pixel must have more than two line neighbors before it can be marked for deletion. This condition addresses only one pattern group. In Figure 16, the central pixel will be retained by this modified Deutsch algorithm. The Suetens et. al. algorithm does not provide the option of deleting 90 degree angles, but the occurrence of such corners is reduced by the modifications. (Appendix F)

With respect to intersections, the Suetens et. al. algorithm is an improvement over the Deutsch algorithm, as illustrated by Figure 17. There are still slight problems with line spurs (H), off-centered lines

```
0 0 0        Includes all rotations of
0 1 0        this pattern
0 1 1
```

Figure 16. Suetens-Dierckx-Piessens-Oosterlinck Save Condition

```
B--
 AA-
 -1-
  -1-
   -1-
   -1-
    -1-
    --1-            ----
     -1-            -H1-
      -1-            -1-
      -1--          --F-
      --1-          -F--
       -1--         -1-
       --1-          -F-
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1-         -1--
        -1--        -1--
---------1----------1---------
---------1---------1311-------
--11111113111111111----11111--
---------1------------------1-
        -1--                -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
        -1-                 -1-
       --1-                 -1-
        -1-                 -1-
        -1-                --1-
        --                -11-
                         -1--
     --                 -1-
    ---- D             -1-
    ----              -1-
     --              -1-
                    H1-
```

-	Deleted Pixel	B	Off-Centered Line
1	Line Pixel	D	No Circular Feature
3	Intersection	F	Effect Of Noise
A	Line Too Wide	H	Addition To Line

Figure 17. Matrix Thinned by the Suetens-Dierckx-Piessens-Oosterlinck Algorithm

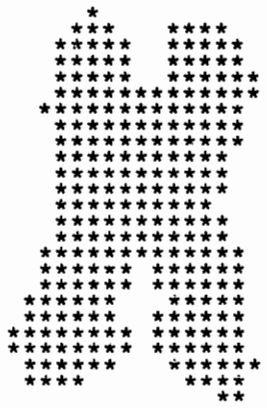
(B), two-pixel-wide lines (A), and loss of the circular feature (D). By correcting the intersections, the Suetens et. al. algorithm has introduced more problems with straight line segments than is apparent in the Deutsch algorithm.

III. TEST RESULTS

In addition to the test matrix shown in previous sections, six other matrices are thinned in the preliminary tests. The second matrix contains the same line pattern as the first one, except that raster lines are 4 to 5 pixels wide (Figure 18a). The remaining test matrices have been adopted from journal articles. By using the examples given in articles, I am able to determine if I am receiving the same results as the author from the algorithm being presented. Therefore, two test matrices come from Hilditch (1969; Figures 18b,c) and one matrix comes from each of the following: Deutsch (1972; Figure 18d), Chen and Hsu (1988; Figure 19a), and Landy and Cohen (1985; Figure 19b). Since original vector coordinates for the test lines are not available, the accuracy of the final vector output can not be tested.

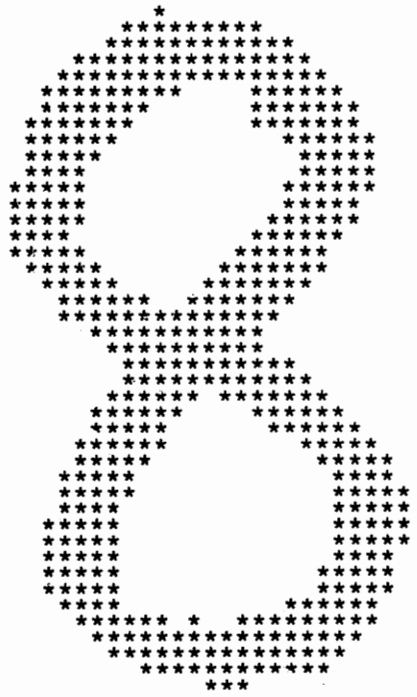
To determine the number of pixels that the final raster matrix should contain, pixels are manually deleted from each input matrix. Manual deletion introduces some subjectivity. Since a totally objective evaluation method is not available, the distance matrix

a. Matrix 2 - Bush



b. Matrix 3 - Hilditch

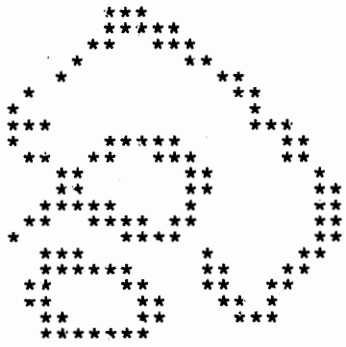
c. Matrix 4 - Hilditch



d. Matrix 5 - Deutsch

Figure 18. Test Matrices

a. Matrix 6 - Chen-Hsu



b. Matrix 7 - Landy-Cohen

Figure 19. Test Matrices

(Figure 20a) is used as a guideline for placement of the final raster line to help minimize human error within the manual deletion process. The central ridge of maximum values defined by the distance matrix represents the medial axis. The medial axis defines the path that the final raster line should follow, since it is the mathematical raster center of the line. Pixels that are not part of the medial axis are manually deleted (Figure 20b). The desired final line consists only of pixels on the medial axis. The medial axis can contain two pixel wide lines where the original raster line has even width. Either of these pixels are a correct position for the final line, but not both. Therefore, when determining the desired number of pixels in the final raster line, only one of these pixels is counted. The value of the medial axis is determined by adding together the values of the desired pixels.

The medial axis shown in Figure 20b appears to shorten the ends of the raster line. When a line is scanned the width increases in all directions from all points. Therefore, ends of lines are extended in the same manner that sides are expanded. Thinning algorithms must remove pixels from the ends of thick raster lines in order to return the thinned lines to their correct position.

The final raster lines are compared using three measures related to the medial axes of the original matrices. The first measure is the deviation between the number of pixels deleted and the number of pixels that should have been deleted. The second measure is the number of

```
 11111
122221
 12321
  123211
  123321
   123321
   1234321
   123321
  123321
 122221
  1111
```

a. Distance Transformation

```
 -----
------
 --3--
  --3---
  --33--
   --33--
   ---4---
   --33--
  --33--
 ------
  ----
```

Number of Pixels = 11

Desired Number = 7

Total Value = 34

Desired Value = 22

b. Medial Axis

Figure 20. Determining Medial Axis Values

pixels in the final matrix that fall on the medial axis. The third is the number of pixels retained that are not on the medial axis.

After assessing the usefulness of the medial axis value in evaluating raster results, it is determined that this value provides relatively the same information as the number of pixels on the medial axis. This value does provide indirect information concerning the quality of the remaining line. For example, the Chen-Hsu algorithm's medial axis value for Matrix 3 is only 95 out of 149. This implies either that portions of the final raster line fall off the medial axis and/or that the line has been drastically shortened. The lack of information concerning the number of pixels off the axis limits the use of this measure, especially on complex lines that are more likely to stray from the center and/or generate line spurs. Since the medial axis value only reinforces information already available from the number of pixel on the medial axis, it is not incorporated into the evaluation statistics.

The deviation in the number of cells deleted is a misleading statistic when analyzed alone. It is possible for an algorithm to delete the exact number of cells desired without having a single pixel fall on the medial axis. In such a case, the quantity is correct, but the selection is wrong. The number of cells deleted gives incomplete information as to the quality of the pixels that remain. Therefore, this statistic has little value in determining the accuracy of the

final raster lines without the qualifying information provided by the number of pixels on and off the medial axis.

To evaluate the different methods based on the results obtained by thinning the seven test matrices, the computer results are compared to the manual results. A program has been written to compute the number of pixels deleted and the number of pixels in each final matrix that falls either on or off the medial axis of the original matrix defined by the manual deletion process. The value of the desired number of pixels retained is compared to the actual number of pixels that are on the medial axis. Each method receives demerits based on the type and severity of the errors. There are four possible errors that can occur:

1. Pixel is correctly retained, but its position is shifted off the medial axis.

2. Pixel is wrongly deleted from the medial axis.

3. Pixel is wrongly retained, and its position is off the medial axis.

4. Pixel is wrongly retained, and its position is on the medial axis.

Shifting pixel position (Error Code 1) results in one demerit per pixel. Deleting or retaining extra pixels (Error Codes 2, 3 & 4) results in two demerits per pixel.

Table 1 outlines the results based on this demerit system of thinning the seven test matrices with the nine thinning methods described in this chapter. To illustrate how each algorithm is evaluated, examine the results for Matrix 1. The Hilditch algorithm deletes 4 extra pixels. All of these pixels are removed from the medial axis, resulting in 8 demerits. The Chen-Hsu algorithm deletes 2 extra pixels from the medial axis, for 4 demerits. Plus, 6 pixels are shifted off the medial axis, resulting in a total of 10 demerits. The modified Landy-Cohen algorithm retains 28 extra pixels. One pixel falls on the medial axis and 27 fall off the axis, resulting in 56 demerits. The Suetens et. al. algorithm retains 3 extra pixels which fall off the medial axis, for 6 demerits. Plus, 2 pixels are shifted off the axis, resulting in a total of 8 demerits.

Table 2 summarizes the demerits accumulated by each algorithm. Five of the thinning methods have been eliminated from further testing based on these results. The original and modified Landy-Cohen algorithms receive the most demerits, as expected from initial observations. The Deutsch algorithm that retained 90 degree angles is in seventh place, which is mainly due to the number of extra pixels retained in the final matrix that both fall on and off the medial axis. Line spurs generated by the Deutsch and Landy-Cohen algorithms contribute to the number of pixels off the medial axis. The modified Landy-Cohen algorithm does produce better results over the original algorithm, but improvements are not substantial. The modified

Table 1. Results From Thinning Test Matrices

THINNING METHOD	ERROR	CELLS ON / OFF AXIS	ERROR CODES	DEMERITS	METHOD RANK
Matrix 1		89 / 0		0	1
HILDITCH	D 4	85 / 0	2	8	2
M ROSENFELD-STEFANELLI	D 11	74 / 4	1 & 2	26	7
DEUTSCH-1	R 11	89 / 11	3	22	6
DEUTSCH-2	R 5	89 / 5	3	10	4
CHEN-HSU	D 2	81 / 6	1 & 2	10	4
LANDY-COHEN	R 17	59 / 47	1 & 3	64	9
M LANDY-COHEN	R 28	90 / 27	3 & 4	56	8
M HILDITCH	D 3	86 / 0	2	6	1
SUETENS*	R 3	87 / 5	1 & 3	8	4
Matrix 2		88 / 0		0	1
HILDITCH	D 8	77 / 3	1 & 2	19	5
M ROSENFELD-STEFANELLI	R 6	84 / 10	1 & 3	16	4
DEUTSCH-1	R 10	90 / 8	3 & 4	20	7
DEUTSCH-2	R 4	84 / 8	1 & 3	12	2
CHEN-HSU	- -	82 / 6	1	6	1
LANDY-COHEN	R 24	39 / 73	1 & 3	97	9
M LANDY-COHEN	R 38	97 / 29	3 & 4	76	8
M HILDITCH	D 8	77 / 3	1 & 2	19	5
SUETENS*	R 4	81 / 11	1 & 3	15	3
Matrix 3		42 / 0		0	1
HILDITCH	D 7	34 / 1	1 & 2	15	3
M ROSENFELD-STEFANELLI	D 2	38 / 2	1 & 2	6	1
DEUTSCH-1	R 6	32 / 16	1 & 3	22	6
DEUTSCH-2	R 4	32 / 14	1 & 3	18	5
CHEN-HSU	D 9	26 / 7	1 & 2	25	7
LANDY-COHEN	R 13	14 / 41	1 & 3	54	8
M LANDY-COHEN	R 29	43 / 28	3 & 4	58	9
M HILDITCH	D 7	34 / 1	1 & 2	15	3
SUETENS*	D 2	38 / 2	1 & 2	6	1

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

D - Deleted R - Retained

Table 1. Results From Thinning Test Matrices (cont.)

THINNING METHOD	ERROR	CELLS ON / OFF AXIS	ERROR CODES	DEMERITS	METHOD RANK
Matrix 4		32 / 0		0	1
HILDITCH	D 1	28 / 3	1 & 2	5	1
M ROSENFELD-STEFANELLI	R 9	35 / 6	3 & 4	18	6
DEUTSCH-1	R 11	35 / 8	3 & 4	22	7
DEUTSCH-2	R 5	31 / 6	1 & 3	11	5
CHEN-HSU	R 3	31 / 4	1 & 3	7	3
LANDY-COHEN	R 9	19 / 22	1 & 3	31	8
M LANDY-COHEN	R 19	34 / 17	3 & 4	38	9
M HILDITCH	D 1	28 / 3	1 & 2	5	1
SUETENS*	R 4	31 / 5	1 & 3	9	4
Matrix 5		116 / 0		0	1
HILDITCH	D 22	91 / 3	1 & 2	47	4
M ROSENFELD-STEFANELLI	- -	106 / 10	1	10	1
DEUTSCH-1	R 4	90 / 30	1 & 3	34	2
DEUTSCH-2	D 16	83 / 17	1 & 2	49	6
CHEN-HSU	D 20	74 / 22	1 & 2	62	7
LANDY-COHEN	D 8	26 / 82	1 & 2	98	9
M LANDY-COHEN	R 32	107 / 41	1 & 3	73	8
M HILDITCH	D 22	91 / 3	1 & 2	47	4
SUETENS*	D 18	94 / 4	1 & 2	40	3
Matrix 6		220 / 0		0	1
HILDITCH	D 17	193 / 10	1 & 2	44	4
M ROSENFELD-STEFANELLI	R 3	208 / 15	1 & 3	18	1
DEUTSCH-1	R 48	218 / 50	1 & 3	98	7
DEUTSCH-2	R 23	211 / 32	1 & 3	55	5
CHEN-HSU	D 7	190 / 23	1 & 2	37	2
LANDY-COHEN	R 23	77 / 166	1 & 3	189	9
M LANDY-COHEN	R 80	221 / 79	3 & 4	160	8
M HILDITCH	D 17	193 / 10	1 & 2	44	4
SUETENS*	R 21	224 / 17	3 & 4	42	3

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

D - Deleted R - Retained

Table 1. Results From Thinning Test Matrices (cont.)

THINNING METHOD	ERROR	CELLS ON / OFF AXIS	ERROR CODES	DEMERITS	METHOD RANK
Matrix 7		77 / 0		0	1
HILDITCH	D 3	74 / 0	2	6	5
M ROSENFELD-STEFANELLI	R 20	95 / 2	3 & 4	40	9
DEUTSCH-1	R 11	85 / 3	3 & 4	22	7
DEUTSCH-2	R 1	78 / 0	4	2	3
CHEN-HSU	R 2	79 / 0	4	4	4
LANDY-COHEN	R 1	76 / 0	2	2	1
M LANDY-COHEN	R 1	76 / 0	2	2	1
M HILDITCH	D 3	74 / 0	2	6	5
SUETENS*	R 18	93 / 2	3 & 4	36	8

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

D - Deleted R - Retained

Table 2. Total Demerits and Final Ranks

THINNING METHOD	LIST OF DEMERITS FOR THE SEVEN TEST MATRICES							TOTAL DEMERITS	FINAL RANK
HILDITCH	8	19	15	5	47	44	6	144	3
MODIFIED ROSENFELD-STEFANELLI	26	16	6	18	10	18	40	134	1
DEUTSCH-1	22	20	22	22	34	98	22	240	7
DEUTSCH-2	10	12	18	11	49	55	2	157	6
CHEN-HSU	10	6	25	7	62	37	4	151	4
LANDY-COHEN	64	97	54	31	98	189	2	535	9
MODIFIED LANDY-COHEN	56	76	58	38	73	160	2	463	8
MODIFIED HILDITCH	6	19	15	5	47	44	6	142	2
SUETENS*	8	15	6	9	40	42	36	156	5

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

Rosenfeld-Stefanelli algorithm unexpectedly receives the least demerits. Unfortunately, this algorithm occasionally deletes entire lines. Since the stability of the endpoints in not ensured, it is not used in the remainder of this study. Finally, since output from the original and modified Hilditch algorithms are identical except for one pixel for two demerits, the modified algorithm is discarded due to increased processing time. Therefore, four algorithms used in the final test are Hilditch, Chen-Hsu, Suetens-Dierckx-Piessens-Oosterlinck, and Deutsch that removes the 90 degree angles.

Among the four selected methods, Hilditch deletes extra pixels from the medial axis for all matrices. The other algorithms are not as consistent. The Deutsch algorithm that removes the 90 degree angles usually retains extra pixels off the medial axis, partly due to line spurs. Spurs are not as numerous as with the first Deutsch method. The Chen-Hsu and Suetens-Dierckx-Piessens-Oosterlinck algorithms vary in error types.

CHAPTER 3

LINE FOLLOWING

Algorithms that utilize the line following principle are not as abundant in literature as the successive layer removal method. Also unlike the layer removal methods, line following operations have little in common besides the main purpose of determining the center of the line. The input data for the four algorithms, as described in this chapter, also differ. The binary and distance matrices introduced previously are represented. In addition, the use of an adjacency matrix and a film negative is also investigated.

The Canada Geographic Information System (CGIS), established in 1967, uses one of the most well-known line following systems. Designed to use a system similar to the distance method described by Rosenfeld and Pfaltz (Peuquet, 1981), a map area of approximately 1x2 inches is used, instead of the standard 3x3 pixels processing window. During the first pass through each map patch, v-values are assigned to each line pixel. V-values range from 0 to 4 based on the number of non-zero axial neighbors of the pixel in question (Figure 21a). With this method, the center raster line always consists of codes of 3 or 4 with the value of 2 designating potential endpoints (Figures 21b,c). In the second pass through the patch, the CGIS follows the raster line by moving along the path of maximum codes. The location of each selected

```
order of neighbors:    -- P1 --
                       P4 P0 P2
                       -- P3 --
```

P0 = P1 + P2 + P3 + P4

a. V-values

```
1111
 1111
  1111
   111
   1111
    111
    1111
     1111
```

b. Binary Matrix

```
1332
 2442
  2442
   343
   2442
    343
    2442
     2331
```

c. Adjacency Matrix

Figure 21. Canada Geographic Information System's Code Assignment

pixel is saved to a vector file. After all patches are processed, the line segments are linked together.

The FASTRAK digitizing system described by Fulford (1981) has been developed specifically for extracting cartographic information. The original source material is photographically reduced, and the laser deflection system scans a film negative of the map. A local raster-scan pattern is used as the laser moves across the width of the line (Figure 22). Each time the line is crossed, the width and the center coordinate are recorded. The data points are then structured into lines based on their relative locations. No raster output is generated.

The FASTRAK track-following system depends on human interaction, since it is completely command-driven. The operator selects each new line system to be followed, instructs the laser to skip over gaps created by attribute data, changes scanning procedures when digitizing small, closed features, and specifies the need for corner reconstruction (Figures 23a,b).

I. BINARY MATRIX: The Baruch Algorithm

Baruch (1988) has created a line following method based on his interpretation of how a person would thin a raster image given only pencil, paper, and manual skills. Baruch has attempted to automate this human process of line thinning. Therefore, thinning is not

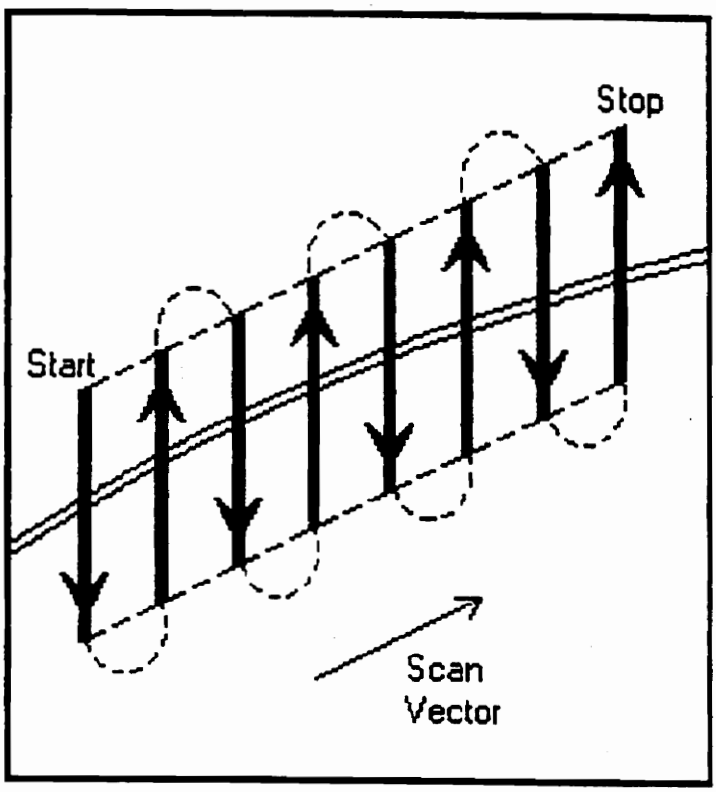
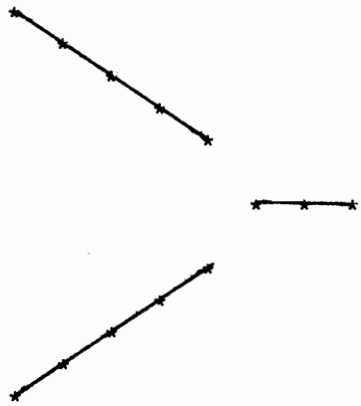
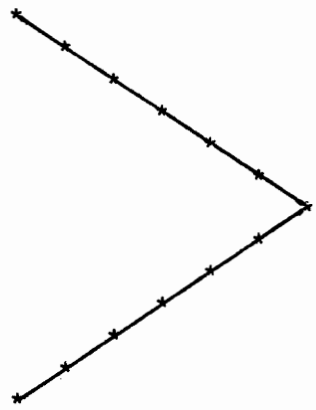


Figure 22. FASTRAK Scan Vector



a. Initial Results



b. Corrected Image

Figure 23. FASTRAK Corner Reconstruction

accomplished by removing layers of pixels but by following the observed center of the raster line.

The Baruch algorithm is based on a dynamic window, two pointers, and two tracers. One pointer (LP) follows the left or upper edge of the raster line; the other (RP) follows the right or lower edge. The window adjusts its size to contain both pointers and the section of the line being followed. Figure 24 illustrates the flexibility of the dynamic window when the distance between the window and the pointers is two pixels. The larger this distance, the faster is the algorithm, but this can cause the final line to deviate from the medial axis.

Three situations are possible within the processing window. First, the line can end. Second, the line can cut straight through. Third, the line can split. To identify the different cases, a process similar to a crossing number is used. The perimeter of each new window is traversed simultaneously in both the clockwise and counterclockwise directions, beginning at the position of the current pointers. These tracers are designated by LLP and RRP and the current pointer positions are shown as Xs in Figure 24. If the tracers meet before encountering line pixels, the first situation is present. When the tracers encounter their first line pixels, the location is marked. Then the perimeter between these two locations is examined for possible background pixels. If no background pixels are located, the second situation is present. The pointers are moved to the marked pixels. If background pixels are encountered, an intersection is present and each

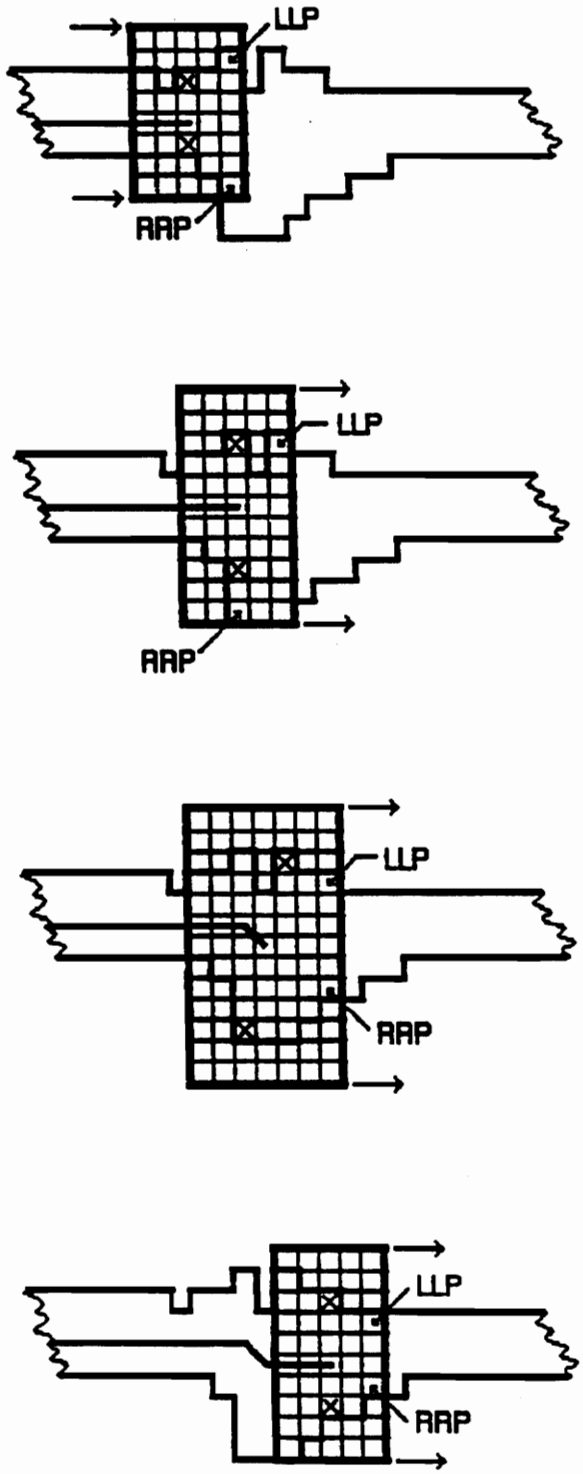


Figure 24. Baruch's Dynamic Processing Window

branch is followed recursively until a line end is reached. In each case, the vector coordinates for the center pixel of each window is saved to a final file, the pointers are moved to where the window intersects the raster line, and the portion of the line that has been followed is erased from the current file. No raster output is generated.

II. DISTANCE MATRIX: The Bush Algorithm

The development of my automatic line thinning method is influenced by the writings of Baruch (1988), Peuquet (1981), and Rosenfeld and Pfaltz (1966). Baruch has influenced the development of a line following method verses a successive layer removal method. Baruch has attempted to automate the human process of line thinning. Consequently, my algorithm also replicates the manual processes.

Since the Canada Geographic Information System uses scanned images of scribed lines, the raster lines are assumed to be 3 to 4 pixels wide. The coding system is not effective on lines with a greater width, since the center of the raster line can no longer be distinguished (Figure 25). Hence, in order to accommodate raster images that are not restricted in line width, I am designing a line following method that moves along the medial axis designated by distance values instead of adjacency values.

```
233332
244443
 244442
  344442
  2444442
   244443
    244442
     233331
```

Figure 25. V-values of a Thick Raster Line

The three basic components of my automatic line following program are two lengths and one pointer. Lengths consist of numbers of pixels along row and column radiating from a given pixel. The pointer is moved along the ridge of the medial axis as the line is being followed. In Figure 26, the length along the row in which the pointer rests is 28, and the length along the column is 5. From this information, it is obvious that the immediate line has a horizontal orientation. Therefore, the pointer is incremented along the maximum horizontal pixels, until it is determined that the direction has changed.

The raster lines must be at least three pixels wide. Ideally, the lines should average five pixels in width along relatively straight segments. The vector coordinates are calculated as the line is being followed and are saved in a MOSS format. All coordinates fall in the center of the raster line, as opposed to the center of the selected pixel (Section E).

A. Starting Pixel

Once the distance transformation of the input matrix is completed, the starting pixel is located by scanning the file in standard raster order for the first pixel with the value 3. If the line is consistently 3 pixels in width (i.e. contains no value of three), the first pixel with the value of 2 is used as the starting point. It is preferred that the line following process begin along a

```
                              |
                              |
                              |
                              |
                              |
                              |
                              |
                              |
             1111      11     11111
           112222111111221111122222111
--------- 122+3332222223322222*33332221 ------------ Row
          12344433333334333333223333321
         1234433222333332222221122233321
         123432211122222111111  1112221
        12344211   11111      |    111
        123321                |
       123321                 |
       123321                 |
       123321                 |
      123321                  |
      122221                  |
       11111                  |
                              |
                              |
                           Column
```

* - Pointer / Starting Pixel

\+ - First 3-Pixel

raster length along the row = 29
pixels left of pointer = 20
pixels right of pointer = 8

raster length along the column = 5
pixels above pointer = 2
pixels below pointer = 2

Figure 26. Basic Components of the Bush Algorithm

relatively straight line segment. Since the raster lines used are not of uniform width, the medial axis values tend to increase along curves (Figure 26). Once a 3-pixel is encountered, the values of its eight neighbors are checked for possible 4-pixels. If no 4-pixel is found, the pixel is accepted as the starting point. If one is found, the pixel could possibly, but not necessarily, fall within a curve. This can cause difficulties in determining line direction. Therefore, the pixel is discarded as the starting point, and each proceeding 3-pixel is evaluated until a suitable one is found. In Figure 26, the starting pixel is actually the seventh 3 encountered.

B. Processing Direction

There are only four possible processing directions: up, down, left, and right. The four diagonal directions are not necessary due to the movement of the pointer, which will be described later. As outlined in the beginning of this section, pixel lengths along the row and column from the current pixel are counted to determine orientation of the line. In the case of Figure 26, the orientation is horizontal. Since there are more pixels to the left of the starting pixel than there are to the right, the initial processing direction is to the left. Once the initial processing direction is selected, the program checks only for a change in direction. For Figure 26, the possibility for a directional change is from horizontal (left) to vertical (up or

down). If the lengths along the row and column are equal, the processing direction does not change.

C. Pointer

When the starting pixel is located, the pointer is set at that position. If the processing direction is horizontal, the maximum pixel along the column is selected as a center line pixel. The pointer is moved to that position. In Figure 26, the starting pixel and the center column pixel are identical; so, the pointer does not move. If the processing direction is vertical, the maximum pixel along the row is selected, and the pointer is moved to that location. The pointer is free to move diagonally, as well as horizontally and vertically.

Once the given row or column has been processed, the pointer is incremented in the processing direction. If the processing direction is right, one column position is added to the pointer. If the direction is up, one row position is subtracted from the pointer, and so on. Then the new row or column is processed, depending on the current direction. Hence, the processing window has dimensions of one by the width of the raster line at any given section.

D. Change in Direction

The raster line is not erased as it is being followed, only recoded. Unfortunately, this quality makes it possible for the pointer

to switch back and forth between two pixels in an infinite loop. To prevent this from occurring, a check is added whenever it is determined that the direction should change. Before the direction is changed, the suggested new direction is compared to the most recent old direction. If these two directions are the same, the direction is not changed. For example, if the direction sequence is up-left-up (old, present, new), the direction remains left. Also if the suggested new direction is the opposite of the old direction, the present direction is not changed. For example, if the sequence is left-down-right, the direction remains down.

A change in processing direction causes some pixels to remain at their original values, instead of being flagged for deletion. For instance, if the processing direction changes from right to down, a wedge-shaped section of pixels to the northeast of the pivot point will remain unprocessed. These pixels are recoded as soon as the directional change is completed (Figure 27). Sometimes a gap can also form after the processing direction has changed. This mainly occurs when the raster line has an even width, which forces the program to select between two equally appropriate pixels. When a gap is detected, the space is filled before the following process continues.

E. Point Selection

In the case of Figure 26, the starting pixel is selected as the center line pixel, since it is the maximum pixel along the column being

```
111111
---*--
 ---*--
  ---*--      vertical processing direction
  ---*--
  ---*---
   ---*---
   ----P---
    123|G|||||
    112|**|||||
      1|||*|||||
       ||||**|||
        |||||*|||       horizontal processing
          ||||*||||        direction
           |||*||||
            |||*||||
             |||**||||
              ||||*|||1
               |||*||21
                ||*|321
                  |||P---        vertical
                  ||||*---       processing
                   |||-*---      direction
                   ||--*---
                     ---*---
                      ---P--
                      111|11
```

- Deleted Pixels By Row
| Deleted Pixels By Column
* Selected Pixels
P Selected Pivot Point
G Filled Gap

Figure 27. Filling Gaps and Deleting Unflagged Pixels

processed. If there is an even number of maximum column pixels during horizontal processing, the pixel below the exact center is saved. If there is an even number of maximum row pixels during vertical processing, the pixel to the right of the exact center is saved.

For the vector coordinates, the exact center of the raster line is calculated. When there is only one maximum pixel, the coordinate represents the center of the pixel. When there is an even number of maximum pixels, the coordinate falls between the two central pixels, not in the center of the selected pixel (Figure 28). This allows the vector coordinates to move in half steps, as opposed to being restricted to only whole steps.

When the center of the pixel is the only option in vector coordinate placement, the angles of the line segments are restricted to 45 degree increments. Using a thinning method with this limitation on a straight, 20-degree-angle line produces a stair-stepped line consisting of tiny segments with 0 to 45 degree angles. When the vector coordinates are allowed to fall between two pixels, the angles of these line segments can increment 22.5 degrees. This half-step selection process greatly reduces the amount of vector error introduced when processing lines with 11 to 33 degree angles for all corresponding rotations.

The selected pixel is recoded. All other connected pixels in the column or row being processed are flagged for deletion. A pixel that is flagged for deletion can not be selected as a line pixel. A

```
1 2 3 2 1
1 2 3 2 1
1 2 3 3 2 1
  1 2 3 3 2 1
    1 2 3 2 1
    1 2 3 3 2 1
    1 2 3 2 1
```

a. Raster Output

```
1 2 * 2 1
1 2 * 2 1
1 2 3*3 2 1
  1 2 3*3 2 1
    1 2 * 2 1
    1 2 3*3 2 1
    1 2 * 2 1
```

b. Vector Output In Center Of Raster Line

```
1 2 * 2 1
1 2 * 2 1
1 2 3 * 2 1
  1 2 3 * 2 1
    1 2 * 2 1
    1 2 3 * 2 1
    1 2 * 2 1
```

c. Vector Output In Center Of Selected Pixel

Figure 28. Raster and Vector Output

previously selected pixel can only be re-selected if the processing direction changes. When a pixel is re-selected, the values of its eight neighbors are checked. If the pixel has neighbors that are not flagged for deletion, then it falls on the border between the processed and unprocessed sections of the raster line. In this case the previously selected pixel acts as a pivot point as the line is being followed. If all of the neighbors are flagged for deletion, than the processing of that line is complete. This check permits the following of closed features, such as contour lines, by ending the following process once closure has been made, and prevents infinite processing.

Since the raster lines are assumed to be at least three pixels wide, processing ends if the maximum value is 1. In this case, the end of the raster line has been reached. To prevent the raster and vector lines from being too long in comparison to the original line, no pixel or coordinate is saved for the final row or column in the raster line. The pointer is returned to the starting pixel and the initial processing direction is reversed (i.e. left to right, up to down, etc). A marker is placed in the vector file to designate the beginning of the second section of the line. The same method is then used to follow the second portion of the line.

F. Final Output

Since the raster lines are assumed to average 5 to 6 pixels in width along relatively straight segments, ignoring only the last row or

column in the raster line is not enough to ensure that the final lines are not too long. One pixel is flagged for deletion on the ends of the thinned raster line. When the final raster file is being saved to the disk, all selected pixels are changed to line pixels of value 1. All other pixels are changed to background pixels with value 0.

Before saving the final vector file, the two sections of the line are properly joined, one coordinate is deleted from the endpoints, and unnecessary points are deleted. The first section of the vector line is read from the temporary raster file. The last point, which is an endpoint, is discarded. The same is done for the second section. The first point of each section is the same value (i.e. the starting point). Hence, the order of the first section is reversed before the two sections are combined into one file.

Once this is done, all intermediate points that fall along a straight line are removed from the vector file. The first point in the temporary vector file is saved. Using the first two points in the temporary file, the initial direction of the line is determined. Then, using the second and third points, the direction is checked for a change. If the line has changed direction, the second point is saved to the final vector file. The comparison is then made using the third and fourth points, and so on, until the last point in the temporary file is saved. The resulting vector file is saved in a MOSS format (feature type, geocode, number of points). The feature type is either

1 for a line or 0 for a point. The geocode remains zero and is added outside of this program.

G. Limitations and Modifications

The Bush algorithm is not complete. The program cannot handle sharp curves or intersections. Intersections have never been addressed directly since the final tests are conducted on individual lines. Adjustments needed to correct the curve problem should also help the intersection problem.

When creating an outline for my line following method, I have neglected to add a variable to monitor the width of the line as the line is being followed. For example, there is nothing in the program to recognize a change in line width from 5 pixels to 10 pixels. Such a jump can be an indication for either of the two problem situations (Figure 29a,b). Figure 30a shows the results after thinning Figure 29a with the current Bush algorithm. The program ignores the presence of the curve, cuts off the portion of the line beyond the curve, and ends the line following process within the curve. Figure 30b shows one possible result from thinning Figure 29a manually, allowing for width.

Attempting to automate the manual thinning process for sharp curves and intersections introduces many unanswered questions. Is it necessary to determine if a curve or an intersection has been encountered? If so, how is this distinction made, and how will the thinning processes differ? Recognizing that there is an increase in

```
                    Column Width = 5
                            :
                            :
                            :
111111111111111111111111
122222222222222222222222211
12*******************332211
1222222222222222222223433221
1111111111111111111112344321          *   Selected Pixel
                         123321
                       11233321
                     112233221
                    122332211
                    1222211
                   111111
                          |
                          |
                          |
                  Column Width = 10
```

a. Sharp Curve

```
          Column Width = 5
                  :
                  :
                  :
111111111111111111111111111111
122222222222222222222222222221
12*************3333333333321
1222222222222224444442222221
111111111111115544551111111           *   Selected Pixel
                  123321
                  123321
                  123321
                  122221
                  111111
                  |
                  |
                  |
           Column Width = 10
```

b. Intersection

Figure 29. Problem Situations

```
1----------------------
1------------------------
1********************------
1---------------------**-----
1-----------------------****1        *   Selected Pixel
                      ------         -   Deleted Pixel
                    11------
                  1122-----
                 12233----
                 12222--
                111111
```

a. Results Received

```
-----------------------
-------------------------
-********************-----
-----------------------**----
------------------------*---        *   Selected Pixel
                      ---*--         -   Deleted Pixel
                    ----*---
                  ----**---
                 ---**----
                 --*----
                ------
```

b. Results Desired

Figure 30. Thinning A Sharp Curve

line width is only the beginning of solving the problem. What other variables need to be considered? Do all eight neighbors need to be evaluated when determining line direction? Can steps be added to the existing Bush thinning method to handle the problem? If not, can a subroutine be developed tangent to the existing method, or does the method need to be completely redesigned to accommodate these situations from early stages? These are not easy questions to answer. Due to the extent of additional research and development required and time restraints, these questions will remain unsolved. (Appendix H)

CHAPTER 4

FINAL TESTING

According to observations recorded in Chapter 2, the preliminary results among the four successive layer removal methods selected for further testing are (demerits in parentheses): Hilditch (144), Chen-Hsu (151), Suetens-Dierckx-Piessens-Oosterlinck (156), Deutsch (157). The Hilditch algorithm does an excellent job on line segments, but it does tend to delete extra pixels from the medial axis. The Deutsch algorithm has minor problems with line segments and major problems with intersections and extra line spurs. The Suetens et. al. algorithm appears to have slight problems in all these areas. The Chen-Hsu algorithm excels in aligning intersections but has problems with some line segments. In view of these initial observations, it is expected that the Bush algorithm will surpass the Suetens et. al. and Deutsch algorithms, because it does not generate line spurs. Since Deutsch creates excessive line spurs, the relative tie between the Deutsch and Suetens et. al. methods will be broken with Suetens et. al. surpassing Deutsch. Since no intersections will be used in the final tests, all algorithms will surpass the Chen-Hsu algorithm. Therefore, the sequence of the final combined results is anticipated to be as follows (from best to worst): Hilditch, Bush, Suetens-Dierckx-Piessens-Oosterlinck, Deutsch, Chen-Hsu.

I. DIGITAL LINE INPUT

The raster images used in the preliminary testing of the successive layer removal algorithms consist mainly of hand-drawn lines. Lines used in the final testing phase are digitized from existing maps. This line extraction is done to shift the emphasis of line thinning away from character and figure recognition and toward cartographic considerations, such as line representation and accuracy.

Twenty-five lines are manually digitized using point mode from six topographic maps. Four maps are from the USGS 7.5 Minute Quadrangle Series at 1:24,000 (Arapahoe, Broad Brook, Coleman Gap, Toms Brook). One map is a USGS-DMA Topo at 1:50,000 (Tenino), and one is a USGS 15 Minute Quadrangle at 1:62,500 (Tyrone). The lines collected cover a wide range of cartographic features. Man-made features include roads and irrigation ditches. Natural features include rivers and contours. Table 3 matches features to map sources. Broad Brook (CT) provides most roads. Arapahoe (WY) is the main source for irrigation ditches. Tenino (WA), Coleman Gap (TN-VA) and Tyrone (PA) provides the complicated contours. Toms Brook (VA), Broad Brook (CT), and Arapahoe (WY) each supplies a river bank.

To limit processing time and the amount of computer storage needed, the dimensions of the lines are confined to a 2x2.5 inch map region. As discussed in Chapter 1, human errors can be introduced into the data during the manual digitizing process. Therefore, no comparisons are made between the vector input or output and the

Table 3. Measurements of the Original Lines

GROUP	LINE	FEATURE	MAP STATE	LINE LENGTH*	ANCHOR LINE LENGTH*	FRACTAL DIMENSION
	1	road	CT	123.002	123.002	1.000
	2	road	CT	182.142	182.142	1.000
	3	road	WY	137.017	136.912	1.001
	4	road	CT	175.871	174.160	1.002
	5	ditch	WY	128.959	126.524	1.005
	6	contour	VA	127.120	124.250	1.005
	7	ditch	WY	187.482	177.737	1.010
A	8	contour	WA	201.356	185.634	1.018
	9	contour	WY	100.005	89.399	1.023
	10	contour	WY	150.095	128.428	1.035
	11	ditch	WY	82.754	66.521	1.051
	12	ditch	WA	197.300	155.636	1.061
	13	contour	TN-VA	88.853	66.521	1.061
	14	ditch	WY	158.742	120.544	1.068
	15	contour	VA	135.784	90.166	1.088
	16	river bank	VA	301.923	172.791	1.097
	17	contour	VA	196.362	89.137	1.170
	18	contour	WY	267.984	88.648	1.196
	19	river bank	CT	223.325	91.839	1.211
B	20	river bank	WY	588.100	183.781	1.214
	21	contour	WY	183.610	57.286	1.251
	22	contour	TN-VA	200.251	46.757	1.278
	23	contour	WA	490.528	99.563	1.331
C	24	contour	VA	429.243	N/A	N/A
	25	contour	PA	273.661	N/A	N/A

* 1 inch = 75 line units

original map lines. The vector input is assumed to be completely error free. The comparisons for the tests are made between the "accurate" vector input and the corresponding vector output for each thinning method.

Using a program from Young (1991), three calculations are made: line length, anchor line length, and fractal dimension. Line length is simply the sum of the lengths of all line segments in the line. Anchor line length is the distance between the endpoints, which measures the relative positions of the endpoints. Fractal dimension measures the irregularity of the line. If the fractal dimension (D) is 1, then the line is regular, such as a straight line or perfect circle. The closer the value of D gets to 2, the greater the irregularity. Fractal dimension provides a means to determine if the thinned line has increased complexity or has been generalized when compared to the original line (Goodchild and Mark,1987; Longley and Batty,1989).

Table 3 lists the values obtained for the three measures of line character for each of the lines. The lines are separated into three classes. Group C contains the two closed contours. The remaining lines are divided based on their fractal dimensions. Group A contains lines with a fractal dimension less than 1.100. The remaining lines are placed in Group B. Most of the lines (16) fall in Group A. This imbalance is deemed representative of map features.

Figures 31 through 35 display the lines selected and their corresponding fractal dimensions, if calculated. According to Table 3,

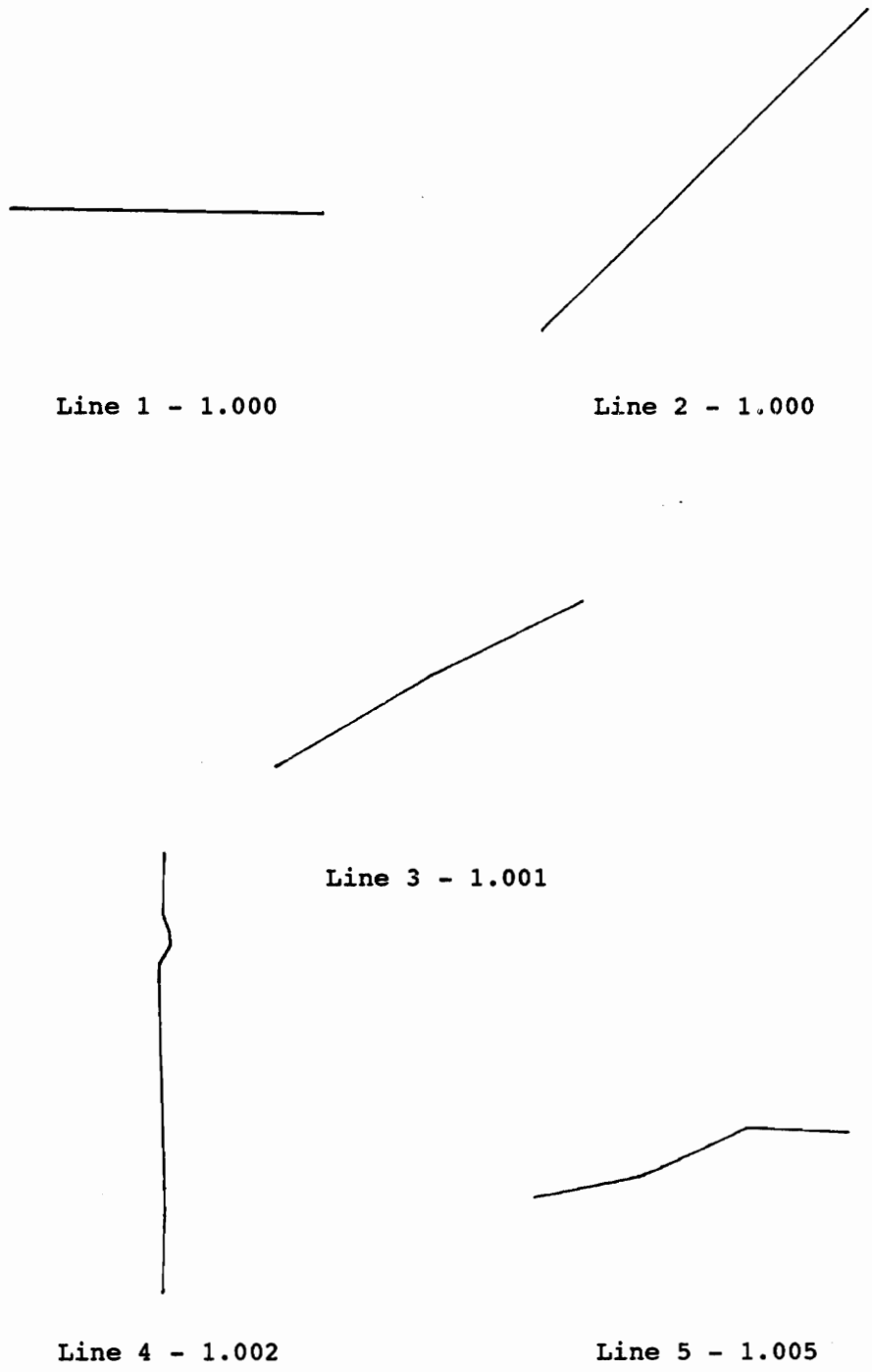
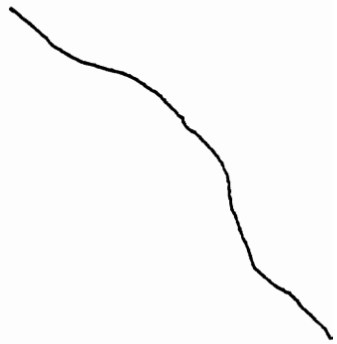
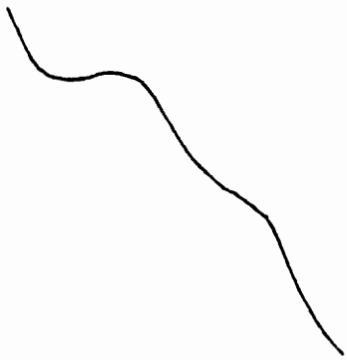


Figure 31. Input Lines - 1 through 5

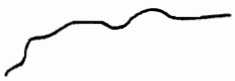
Line 6 - 1.005



Line 7 - 1.010



Line 8 - 1.018

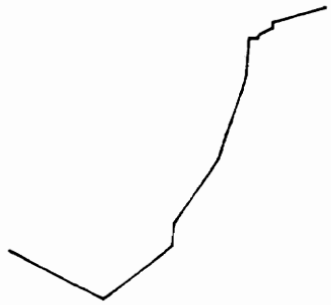


Line 9 - 1.023

Line 10 - 1.035

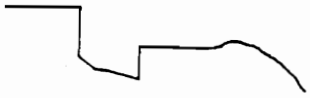
Figure 32. Input Lines - 6 through 10

Line 11 - 1.051



Line 12 - 1.061

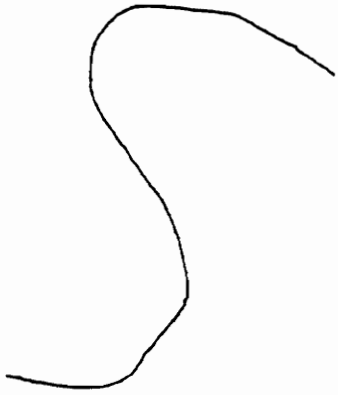
Line 13 - 1.061



Line 14 - 1.068

Line 15 - 1.088

Figure 33. Input Lines - 11 through 15

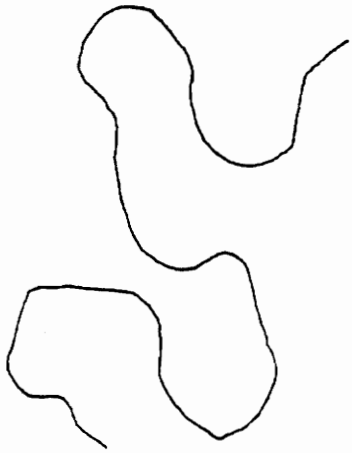


Line 16 - 1.097

Line 17 - 1.170

Line 18 - 1.196

Line 19 - 1.211



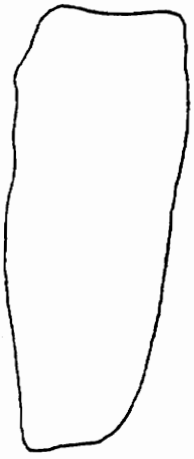
Line 20 - 1.214

Figure 34. Input Lines - 16 through 20

Line 21 - 1.251

Line 22 - 1.278

Line 23 - 1.331



Line 24

Line 25

Figure 35. Input Lines - 21 through 25

these lines represent 3 river banks, 4 roads, 5 irrigation ditches, and 13 contours. It may seem unbalanced that there are more irrigation ditches represented than rivers or roads. Of the ditches, line 5 has the same pattern as many roads, and lines 7 and 11 could be mistaken for contours or streams. Line selection focuses more on covering a wide range of fractal dimensions than on diversifying feature types. The ratio of contours to non-contours is intentional since contours compose more than 50 percent of topographic maps.

II. RASTER LINE INPUT/OUTPUT

The lines are plotted on vellum to prevent excessive bleeding of the ink. Rapidograph plotting pens (No. 0 or No. 3) are used to draw distinct lines that do not collapse upon themselves along curves. The lines are then scanned (300 dpi or 75 dpi) so that the raster lines average 5 pixels in width along relatively straight line segments. A Hewlett-Packard 7475A Plotter and ScanJet Scanner are used. The raster lines are thinned using the five remaining methods: Hilditch, Chen-Hsu, Deutsch, Suetens-Dierckx-Piessens-Oosterlinck, and Bush.

A. Raster Results

Table 4 outlines the final raster results based on the demerit system. As described in Chapter 2, there are four possible errors that can occur:

Table 4. Raster Output for Final Testing Phase

THINNING METHOD	ERROR	CELLS ON / OFF AXIS	ERROR CODES	DEMERITS	METHOD RANK
L1		247 / 0		0	1
HILDITCH	D 2	245 / 0	2	4	2
CHEN-HSU	D 3	243 / 1	1 & 2	7	4
DEUTSCH	R 6	246 / 7	1 & 3	13	5
SUETENS*	D 2	245 / 0	2	4	2
BUSH	- 0	247 / 0	-	0	1
L2		272 / 0		0	1
HILDITCH	D 1	270 / 1	1 & 2	3	1
CHEN-HSU	- 0	263 / 9	1	9	4
DEUTSCH	R 6	263 / 15	1 & 3	21	5
SUETENS*	R 2	274 / 0	4	4	2
BUSH	R 3	275 / 0	4	6	3
L3		240 / 0		0	1
HILDITCH	D 2	235 / 3	1 & 2	7	2
CHEN-HSU	D 4	218 / 18	1 & 2	26	4
DEUTSCH	R 4	219 / 25	1 & 3	29	5
SUETENS*	D 2	235 / 3	1 & 2	7	2
BUSH	D 1	235 / 4	1 & 2	6	1
L4		350 / 0		0	1
HILDITCH	D 1	349 / 0	2	2	3
CHEN-HSU	D 3	325 / 22	1 & 2	28	5
DEUTSCH	R 5	345 / 10	1 & 3	15	4
SUETENS*	- 0	349 / 1	1	1	2
BUSH	- 0	350 / 0	-	0	1
L5		245 / 0		0	1
HILDITCH	D 1	244 / 0	2	2	3
CHEN-HSU	D 3	203 / 39	1 & 2	45	4
DEUTSCH	R 40	242 / 43	1 & 3	83	5
SUETENS*	- 0	244 / 1	1	1	2
BUSH	- 0	245 / 0	-	0	1

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

D - Deleted R - Retained

Table 4. Raster Output for Final Testing Phase (Cont.)

THINNING METHOD	ERROR	CELLS ON / OFF AXIS	ERROR CODES	DEMERITS	METHOD RANK
L6		247 / 0		0	1
HILDITCH	D 3	244 / 0	2	6	3
CHEN-HSU	D 5	241 / 1	1 & 2	11	4
DEUTSCH	R 8	242 / 13	1 & 3	21	5
SUETENS*	R 1	246 / 2	1 & 3	3	2
BUSH	D 1	246 / 0	2	2	1
L7		317 / 0		0	1
HILDITCH	D 5	312 / 0	2	8	3
CHEN-HSU	R 3	302 / 18	1 & 3	21	4
DEUTSCH	R 51	309 / 59	1 & 3	110	5
SUETENS*	- 0	316 / 1	1	1	1
BUSH	R 5	321 / 1	3 & 4	9	2
L8		350 / 0		0	1
HILDITCH	D 4	346 / 0	2	8	3
CHEN-HSU	R 1	328 / 23	1 & 3	24	4
DEUTSCH	R 11	346 / 15	1 & 3	26	5
SUETENS*	- 0	350 / 0	-	0	1
BUSH	R 2	352 / 0	4	4	2
L9		183 / 0		0	1
HILDITCH	D 2	181 / 0	2	4	3
CHEN-HSU	- 0	171 / 12	1	12	4
DEUTSCH	R 21	172 / 32	1 & 3	43	5
SUETENS*	R 1	183 / 1	3	2	2
BUSH	- 0	183 / 0	-	0	1
L10		271 / 0		0	1
HILDITCH	D 8	263 / 0	2	16	3
CHEN-HSU	D 1	252 / 18	1 & 2	20	4
DEUTSCH	R 32	253 / 50	1 & 3	82	5
SUETENS*	R 1	269 / 3	1 & 3	4	1
BUSH	D 3	265 / 3	1 & 2	9	2

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

D - Deleted R - Retained

Table 4. Raster Output for Final Testing Phase (Cont.)

THINNING METHOD	ERROR	CELLS ON / OFF AXIS	ERROR CODES	DEMERITS	METHOD RANK
L11		292 / 0		0	1
HILDITCH	D 2	289 / 1	1 & 2	5	1
CHEN-HSU	- 0	247 / 45	1	45	4
DEUTSCH	R 23	270 / 55	1 & 3	78	5
SUETENS*	R 3	292 / 3	3	6	3
BUSH	R 1	289 / 4	1 & 3	5	1
L12		350 / 0		0	1
HILDITCH	D 2	347 / 1	1 & 2	5	1
CHEN-HSU	R 2	330 / 22	1 & 3	24	4
DEUTSCH	R 22	339 / 33	1 & 3	55	5
SUETENS*	R 4	352 / 2	3 & 4	8	3
BUSH	R 1	347 / 4	1 & 3	5	1
L13		301 / 0		0	1
HILDITCH	D 3	296 / 2	1 & 2	8	1
CHEN-HSU	R 5	280 / 26	1 & 3	31	3
DEUTSCH	R 36	294 / 43	1 & 3	79	4
SUETENS*	R 6	301 / 6	3	12	2
BUSH	-----	--------	-----	--	-
L14		293 / 0		0	1
HILDITCH	D 6	287 / 0	2	12	2
CHEN-HSU	D 3	277 / 13	1 & 2	19	3
DEUTSCH	R 18	287 / 24	1 & 3	42	4
SUETENS*	R 3	292 / 4	1 & 3	7	1
BUSH	-----	--------	-----	--	-
L15		479 / 0		0	1
HILDITCH	D 2	471 / 6	1 & 2	10	1
CHEN-HSU	R 11	419 / 71	1 & 3	82	3
DEUTSCH	R 60	455 / 84	1 & 3	144	4
SUETENS*	R 8	481 / 6	3 & 4	16	2
BUSH	-----	--------	-----	--	-

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

D - Deleted R - Retained

Table 4. Raster Output for Final Testing Phase (Cont.)

THINNING METHOD	ERROR	CELLS ON / OFF AXIS	ERROR CODES	DEMERITS	METHOD RANK
L16		543 / 0		0	1
HILDITCH	R 1	543 / 1	3	2	1
CHEN-HSU	R 2	500 / 45	1 & 3	47	4
DEUTSCH	R 28	519 / 52	1 & 3	80	5
SUETENS*	R 5	545 / 3	3 & 4	8	3
BUSH	R 2	543 / 2	3	4	2
L17		692 / 0		0	1
HILDITCH	D 2	686 / 4	1 & 2	8	1
CHEN-HSU	R 5	622 / 75	1 & 3	80	3
DEUTSCH	R 43	675 / 60	1 & 3	103	4
SUETENS*	R 10	687 / 15	1 & 3	25	2
BUSH	-----	--------	-----	--	-
L18		931 / 0		0	1
HILDITCH	R 4	924 / 11	1 & 3	15	1
CHEN-HSU	R 13	855 / 88	1 & 3	101	3
DEUTSCH	R 55	901 / 85	1 & 3	140	4
SUETENS*	R 18	929 / 20	1 & 3	38	2
BUSH	-----	--------	-----	--	-
L19		779 / 0		0	1
HILDITCH	D 9	765 / 5	1 & 2	23	2
CHEN-HSU	R 13	691 / 101	1 & 3	144	3
DEUTSCH	R 68	749 / 98	1 & 3	166	4
SUETENS*	R 7	778 / 8	1 & 3	15	1
BUSH	-----	--------	-----	--	-
L20		1058 / 0		0	1
HILDITCH	D 11	1046 / 1	1 & 2	23	3
CHEN-HSU	R 5	1004 / 59	1 & 3	64	4
DEUTSCH	D 91	1023 / 126	1 & 3	217	5
SUETENS*	R 2	1057 / 3	1 & 3	5	1
BUSH	D 1	1052 / 5	1 & 2	7	2

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

D - Deleted R - Retained

Table 4. Raster Output for Final Testing Phase (Cont.)

THINNING METHOD	ERROR	CELLS ON / OFF AXIS	ERROR CODES	DEMERITS	METHOD RANK
L21		643 / 0		0	1
HILDITCH	D 3	632 / 8	1 & 2	33	2
CHEN-HSU	R 12	558 / 97	1 & 3	109	3
DEUTSCH	R 54	607 / 90	1 & 3	144	4
SUETENS*	R 10	644 / 9	3 & 4	20	1
BUSH	-----	--------	-----	---	-
L22		699 / 0		0	1
HILDITCH	D 9	686 / 4	1 & 2	22	2
CHEN-HSU	R 9	610 / 98	1 & 3	107	3
DEUTSCH	R 63	657 / 105	1 & 3	168	4
SUETENS*	R 6	692 / 13	1 & 3	19	1
BUSH	-----	---------	-----	---	-
L23		1421 / 0		0	1
HILDITCH	R 85	1387 / 119	1 & 3	204	1
CHEN-HSU	R 113	1178 / 356	1 & 3	469	3
DEUTSCH	R 415	1288 / 548	1 & 3	963	4
SUETENS*	R 162	1411 / 172	1 & 3	334	2
BUSH	-----	----------	-----	---	-
L24		831 / 0		0	1
HILDITCH	D 2	829 / 0	2	4	1
CHEN-HSU	- 0	778 / 53	1	53	4
DEUTSCH	R 32	806 / 57	1 & 3	89	5
SUETENS*	- 0	826 / 5	1	5	2
BUSH	D 3	827 / 1	1 & 2	7	3
L25		969 / 0		0	1
HILDITCH	- 0	966 / 3	1	3	1
CHEN-HSU	R 6	837 / 138	1 & 2	144	4
DEUTSCH	R 53	950 / 72	1 & 2	128	3
SUETENS*	R 8	969 / 8	3	16	2
BUSH	-----	-------	-----	---	-

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

D - Deleted R - Retained

1. Pixel is correctly retained, but its position is shifted off the medial axis.

2. Pixel is wrongly deleted from the medial axis.

3. Pixel is wrongly retained, and its position is off the medial axis.

4. Pixel is wrongly retained, and its position is on the medial axis.

Shifting pixel position (Error Code 1) results in one demerit per pixel. Deleting or retaining extra pixels (Error Codes 2, 3 & 4) results two demerits per pixel.

The Hilditch, Chen-Hsu, Deutsch, and Suetens et. al. algorithms thin the 25 lines with varying levels of success. The Bush algorithm follows 15 of the 25 lines, due to problems with sharp curves (Chapter 3). As expected, the majority of the lines followed (13) fall in Group A, with one line from each of the other two groups. The Deutsch algorithm produces line spurs on all lines tested. The spurs are not limited to one or two pixels in length, often reaching lengths of four or five pixels. The degree of subsequent processing required to delete the numerous spurs makes Deutsch the least efficient method. The Suetens et. al. algorithm also produces an occasional line spur. These spurs are usually one pixel in length and can easily be deleted by a program designed to decrease line noise. All algorithms used generate

spurs for lines 18 and 23, due to their complexity. In Group C, all layer removal algorithms thinned both contours with the Bush algorithm following the less complicated one. A group of two contours is too small of a sample to form any conclusions beyond the observation that the algorithms are capable of thinning such features.

After evaluating the accuracy of the final raster lines, the results in Table 4 lead to a division of the five methods into two groups. In one group, the Chen-Hsu and Deutsch algorithms deviate from the medial axis more frequently than the other methods. These line shifts cause both methods to rank last on every line. From Group A, the average number of pixels that are shifted, deleted, or retained incorrectly is 30. This average triples to 90 pixel errors for Group B. The other three methods, Hilditch, Suetens et. al., and Bush, fare well. From Group A, the average number of pixel errors is 3. Suetens et. al. equals the average; Hilditch is slightly higher (3.7); Bush is slightly lower (2.5). From Group B, both Suetens et. al. and Hilditch average approximately 11 pixel errors.

The strong point of the Hilditch method is the weak point for the Suetens et. al. method, and vice versa. The Hilditch algorithm tends to delete more pixels from the raster line than any of the other algorithms. For 21 lines, extra pixels are deleted from the medial axis (Error Code 2 in Table 4), as opposed to 2 lines for the Suetens et. al. method. This extra deletion means that the line is being shortened with this method. Hilditch also tends to discard more

unnecessary pixels than the other algorithms. For 3 lines, extra pixels are retained in positions off the medial axis (Error Code 3), as opposed to 17 lines for Suetens et. al.

According to Table 4, there are only five cases where a perfect score is achieved. For a perfect score, the number of cells on and off the medial axis must match the desired statistics listed. Consequently, the algorithm receives no demerits for that line. The Bush algorithm accomplishes this for four lines: L1, L4, L5, and L9. The Suetens et. al. algorithm is successful with L8. All of these lines fall in Group A, which is to be anticipated. The higher the fractal dimension, the more difficult it becomes for any method to match the expected results.

B. Raster Summary

Table 5 summarizes the final demerits and ranks of each method. Suetens et. al. demerits for line 23 alone are greater than its demerits for all the other lines combined, 334 and 227 respectively. The other algorithms also show a drastic increase for line 23. Obviously there is a breaking point between fractal dimensions of 1.278 and 1.331 that makes it impractical and unreasonable to scan and process lines of extreme complexity. Since incorporating the results for line 23 would dramatically skew the data, its influence is removed from the final results. Therefore, the algorithms rank as follows

Table 5. Summary of Final Raster Demerits and Ranks

LINE	HILDITCH	CHEN-HSU	DEUTSCH	SUETENS*	BUSH
1	4	7	13	4	0
2	3	9	21	4	6
3	7	26	29	7	6
4	2	28	15	1	0
5	2	45	83	1	0
6	6	11	21	3	2
7	10	21	110	1	9
8	8	24	26	0	2
9	4	12	43	2	0
10	16	20	82	4	9
11	5	45	78	6	5
12	5	24	55	8	5
13	8	31	79	12	-
14	12	19	42	7	-
15	10	82	144	16	-
16	2	47	80	8	4
17	8	80	103	25	-
18	15	101	140	38	-
19	23	114	166	15	-
20	23	64	217	5	7
21	33	109	144	20	-
22	22	107	168	19	-
23^	204	469	963	334	-
24	4	53	89	3	7
25	3	144	128	2	-
Demerits & Rank for 24 lines	235 2	1223 3	2076 4	227 1	N/A -
Demerits & Rank for 15 lines	101 2	436 4	962 5	59 1	62 2

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

^ Line 23 not included in Demerits & Rank

(total demerits): Suetens-Dierckx-Piessens-Oosterlinck (227), Hilditch (235), Chen-Hsu (1223), and Deutsch (2076).

The total number of demerits for the 24 lines evaluated emphasizes the gap between the 'better' algorithms, Suetens et. al. and Hilditch, and the 'worse' algorithms, Chen-Hsu and Deutsch. Hilditch with 235 demerits and Suetens et. al. with 227 do not appear to produce drastically different overall results when looking at the total value. When evaluating the less complicated lines, as defined by the 15 lines that the Bush algorithm can follow, differences do arise. Suetens et. al. receives more than 40 percent fewer demerits than Hilditch. The Bush algorithm ranks second, with only a 3-demerit gap between it and Suetens et. al. For the nine more complicated lines, Suetens et. al. receives almost 20 percent more demerits than Hilditch.

Figure 36 illustrates the gap present between these two algorithms for lines within 1.001 and 1.051 fractal dimensions. Suetens et. al. excels in thinning lines with fractal dimensions less than 1.051 (lines 1 thru 10). For fractal dimensions between 1.051 and 1.196 (lines 11 thru 18), Hilditch surpasses Suetens et. al. For fractal dimensions greater than 1.196 (lines 19 thru 22), Suetens et. al. again excels. Both algorithms have one bad line, line 18 for Suetens et. al. and line 21 for Hilditch. For Group B, often wide gaps form between the abilities of each of these methods to thin individual lines. Based on this division in ability along fractal dimension values, an appropriate thinning algorithm for a given map could be

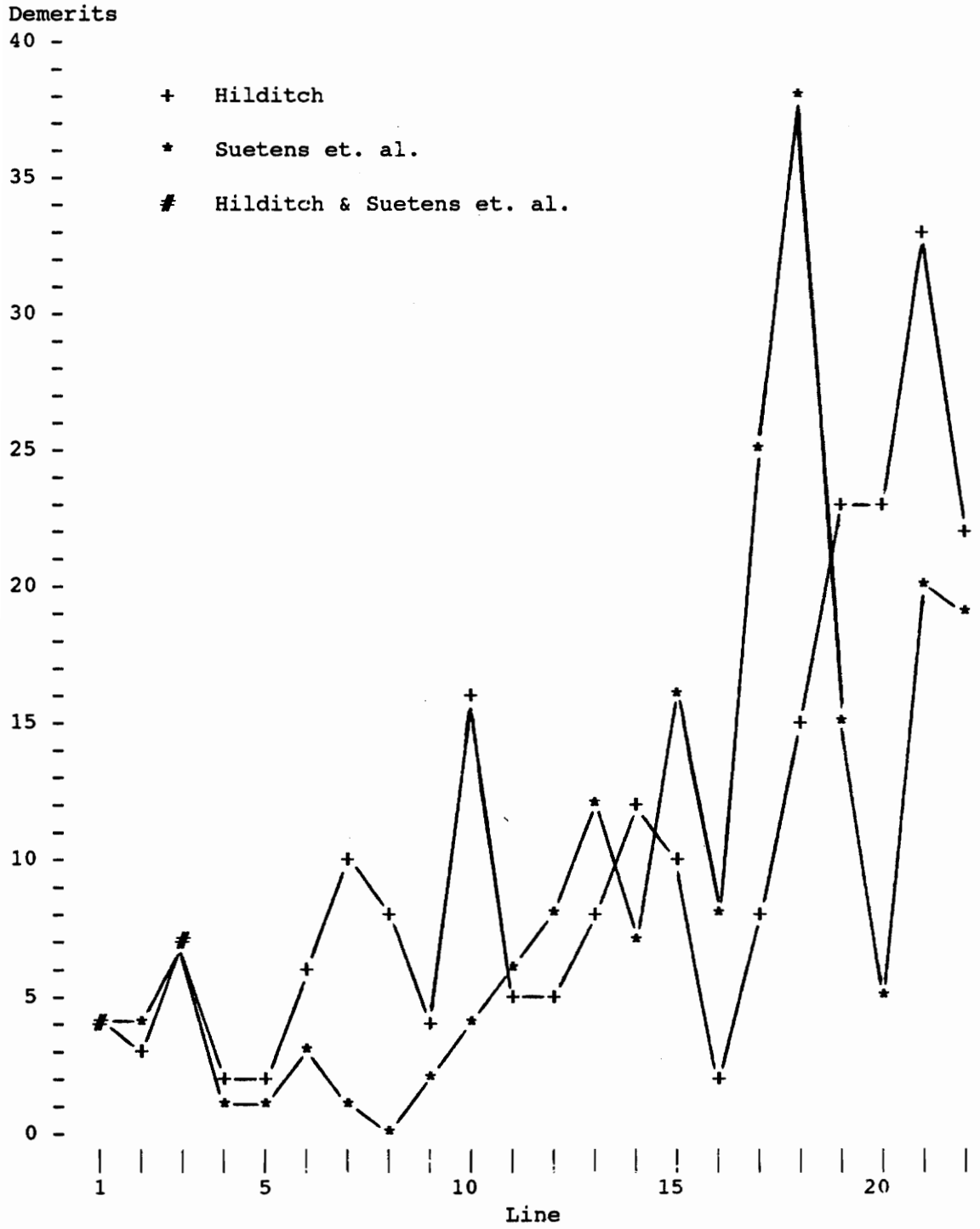


Figure 36. Demerits of Hilditch & Suetens-Dierckx-Piessens-Oosterlinck

selected based on the mean fractal dimension for the entire map or for localized regions if the map is not uniform.

Whereas Figure 36 shows the relationship between Hilditch and Suetens et. al. for each individual line, Table 6 converts the demerits to a percent deviation, which allows the relationship between individual lines to be compared. The percent deviation for lines 10 and 21 for the Hilditch method exceeds the percent deviation recorded for line 18 for Suetens et. al., despite the high peek shown in Figure 36.

III. DIGITAL LINE OUTPUT

A. Vector Results

Of the three vector measurements used to evaluate the vector results, the statistics for line length and anchor line length can be deceiving. This test does not provide a direct means to 'fit' the output lines to the input lines. Distance can be measured but not position. Therefore, a near zero deviation for these two lengths implies that the positions of the vertices are near the original locations; it does not prove this point.

1. Line Length

The results for line length deviations have the most distinct pattern among the measures used to evaluate the digital line output,

Table 6. Percent Deviation from Desired Raster Line Based on Demerits

LINE	EXPECTED PIXELS ON MEDIAL AXIS	PERCENT DEVIATION				
		HILDITCH	CHEN-HSU	DEUTSCH	SUETENS*	BUSH
1	247	1.62	2.83	5.26	1.62	0.00
2	272	1.10	3.31	7.72	1.47	2.21
3	240	2.92	10.83	12.08	2.92	2.50
4	250	0.80	11.20	6.00	0.40	0.00
5	245	0.82	18.37	33.88	0.41	0.00
6	247	2.43	4.45	8.50	1.21	0.81
7	317	3.15	6.62	34.70	0.32	2.84
8	350	2.29	6.86	7.43	0.00	0.57
9	183	2.19	6.56	23.50	1.09	0.00
10	271	5.90	7.38	30.26	1.48	3.32
11	292	1.71	15.51	26.71	2.05	1.71
12	250	2.00	9.60	22.00	3.20	2.00
13	301	2.65	10.30	26.25	3.99	----
14	293	4.10	6.48	14.33	2.39	----
15	479	2.09	17.12	30.06	3.34	----
16	543	0.37	8.66	14.73	1.47	0.74
17	692	1.16	11.56	14.88	3.61	----
18	931	1.61	10.85	15.04	4.08	----
19	779	2.95	14.63	21.31	1.93	----
20	1058	2.17	6.05	20.05	0.47	0.66
21	643	5.13	16.95	22.40	3.11	----
22	699	3.15	15.31	24.03	2.72	----
23	1421	14.36	33.00	67.77	23.50	----
24	831	0.48	6.38	10.71	0.60	0.84
25	969	0.31	14.86	13.21	1.65	----

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

among the algorithms. Table 7 depicts the differences between the vector output and the original line length before the scanning process. The results are divided between the Bush and Hilditch algorithms. The Bush method is superior regarding 12 of the 15 lines that it currently is capable of following. This achievement is due to the ability of the vector coordinates to increment in half steps when lines have even width, instead of falling within the center of the selected pixel (Chapter 3, Section E). For the remaining 10 lines, Hilditch deviates the least, due to the lack of pixels falling off the medial axis. The Chen-Hsu algorithm is the third most effective method, scoring better than Suetens et. al. and Deutsch in 18 incidences. Chen-Hsu also surpasses Hilditch for 7 of the 16 lines in Group A. All of the methods are more likely to lengthen the scanned lines.

Of the 115 line measurements recorded in Table 7, only 9 of them are shorter than the original line length. The lines could be longer because the endpoints are extended and/or the body of the line is exaggerated due to the stair-step effect of the pixel selection process. This statistic alone does not provide enough information to determine the cause.

The line lengths and deviations, displayed in Table 7 in line units, are converted to inches (75 line units equal 1 inch) and presented in Table 8 as lengths and percent deviations. It seems logical to expect that the more complicated the line the greater the percentage of error, but this logic does not hold true. There are

Table 7. Deviation from Original Line Length by Line

LINE	ORIGINAL LINE LENGTH^	DEVIATION				
		HILDITCH	CHEN-HSU	DEUTSCH	SUETENS*	BUSH
1	123.002	0.488	-0.014~	1.578	0.075	2.019
2	182.142	2.884	3.592	4.387	3.764	1.044~
3	137.017	9.606	8.602	10.608	9.606	3.745~
4	175.871	7.300	3.174~	5.174	5.924	4.774
5	128.959	6.423	6.452	7.763	6.924	4.889~
6	127.120	2.271	1.267~	2.228	4.070	1.631
7	187.482	7.733	10.078	14.640	9.406	6.028~
8	201.356	10.864	12.125	12.918	12.246	5.828~
9	100.005	2.085	2.468	5.471	3.175	1.133~
10	150.095	2.643	3.460	7.879	5.494	0.519~
11	82.754	1.929	2.326	3.459	2.665	0.105~
12	197.300	9.477	9.194	11.493	9.694	4.754~
13	88.853	-0.012~	0.022	1.142	0.600	-----
14	158.742	-1.902	-1.213	1.794	-0.329~	-----
15	135.784	0.793~	1.350	1.470	1.075	-----
16	301.923	15.576	15.037	15.630	15.545	7.213~
17	196.362	5.757~	5.833	7.612	6.292	-----
18	267.984	9.116~	9.601	11.176	10.189	-----
19	223.325	5.084~	7.482	9.450	6.912	-----
20	588.100	25.994	29.239	34.973	30.229	15.265~
21	183.610	2.560~	4.344	5.432	5.432	-----
22	200.251	3.696~	4.061	6.185	3.994	-----
23	490.528	-9.757~	-14.968	-12.296	-9.851	-----
24	429.243	21.144	16.987	20.110	18.400	10.525~
25	273.661	11.003	9.265	10.265	4.678~	-----

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

^ 1 inch = 75 line units

~ Best results for the given line

Table 8. Percent Deviation from Original Line Length by Line

LINE	ORIGINAL LINE LENGTH (inches)	PERCENT DEVIATION				
		HILDITCH	CHEN-HSU	DEUTSCH	SUETENS*	BUSH
1	1.6400	0.40	-0.01	1.28	0.06	1.64
2	2.4286	1.58	1.97	2.41	2.07	0.57
3	1.8269	7.01	6.28	7.74	7.01	2.73
4	2.3449	4.15	1.80	2.94	3.37	2.71
5	1.7194	4.98	5.00	6.02	5.37	3.79
6	1.6949	1.79	1.00	1.75	3.20	1.28
7	2.4998	4.13	5.37	7.81	5.02	3.22
8	2.6847	5.40	6.02	6.42	6.08	2.89
9	1.3334	2.08	2.47	5.47	3.17	1.13
10	2.0013	1.76	2.30	5.25	3.66	0.35
11	1.1034	2.33	2.81	4.18	3.22	0.13
12	2.6307	4.80	4.66	5.83	4.91	2.41
13	1.1847	-0.01	0.02	1.28	0.67	----
14	2.1166	-1.20	-0.76	1.13	-0.21	----
15	1.8104	0.58	0.99	1.08	0.79	----
16	4.0256	5.14	4.98	5.18	5.15	2.39
17	2.6181	2.93	2.97	3.88	3.20	----
18	3.5731	3.40	3.58	4.17	3.80	----
19	2.9777	2.27	3.35	4.23	3.09	----
20	7.8413	4.42	4.97	5.95	5.14	2.60
21	2.4481	1.39	2.37	2.96	2.96	----
22	2.6700	1.84	2.03	3.09	1.99	----
23	6.5404	-1.99	-3.05	-2.51	-2.01	----
24	5.7232	4.93	3.96	4.68	4.29	2.45
25	3.6488	4.02	3.39	3.75	1.71	----

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

nine lines that receive worst scores for all methods than line 23, including lines 3 and 5. The orientation of the line sometimes has more influence on the results than the overall complexity (Chapter 3, Section E).

Twenty-five lines do not provide a large enough sample to draw any conclusions concerning trends within the results for each algorithm. An average of these percentages by group does serve as a reinforcement for previous observations. Table 9 shows the Bush algorithm with a large margin over the other methods for Group A. In Group B, Hilditch surpasses the remaining methods, followed closely by the Chen-Hsu algorithm.

In three cases in Table 9, the percent deviation drops from Group A to Group B, instead of the anticipated increase. When scanning and processing a completely straight line, it is not unusual for the resulting thinned line to contain more than 2 vertices, which is a measurable deviation. For more complicated lines, deviation is not as easy to distinguish. On one hand, the scanning process tends to smooth changes in line character, which is more prevalent for complicated lines. On the other hand, line thinning methods add extra vertices to the line due to the stair-stepped pixel selection process. The smoothing action of scanning overshadows and appears to negate some of the deviation introduced during thinning. The drop in the average percent deviation does not mean that the line thinning methods are more accurate on more complicated lines. It does emphasize the difficulty

Table 9. Average Percent Deviation from Original Line Length by Group

GROUP	AVERAGE PERCENT DEVIATION				
	HILDITCH	CHEN-HSU	DEUTSCH	SUETENS*	BUSH
A	2.96	2.90	4.11	3.37	1.94~
B	2.61	3.18	3.83	3.17	N/A
C	4.48	3.68	4.22	3.20	N/A
A,B,C	2.98	3.04	4.04	3.29	N/A

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK
~ based on the 13 lines followed

in separating and measuring the different error factors. Therefore, comparisons should not be made between groups; they should only be made between the different methods within each group.

2. Anchor Line Length

Deviation from the original anchor line length, as seen in Table 10, produces an unexpected outcome. Surprisingly, Deutsch achieves the best results in this category, but this feat is only accomplished after the line spurs are deleted from the raster output. In most cases, the Deutsch algorithm generates a forked intersection at the ends of the raster lines. These forks are manually removed down to the common pixel before the raster line is vectorized. The Hilditch algorithm appears to have the most success with lines having a fractal dimension between 1.06 and 1.20, which partially reinforces raster results (Section II,B). The Bush method excels only in two cases.

Unlike results for line length, there are numerous negative values for anchor line length. This situation is particularly true in the Hilditch and Chen-Hsu results for Group A, whereas Deutsch receives mostly positive values. Table 11 shows the distribution of average positive and negative deviations. The number of lines incorporated is shown in parentheses. From Group A, Hilditch shortens the anchor line length for 14 lines an average of -0.751, as opposed to lengthening 2 lines an average of 0.091, with mean of -0.645. Since Hilditch removes extra pixels off the medial axis, this uneven distribution is expected.

Table 10. Deviation from Original Anchor Line Length by Line

LINE	ORIGINAL ANCHOR LINE LENGTH^	DEVIATION				
		HILDITCH	CHEN-HSU	DEUTSCH	SUETENS*	BUSH
1	123.002	-0.544	-1.046	-0.034~	-0.544	0.962
2	182.142	-1.257	-0.549	-0.213~	-1.257	0.363
3	136.912	-0.396	-1.270	0.287~	-0.396	0.602
4	174.160	-0.114~	-1.124	0.875	0.375	-0.125
5	126.524	-0.313	-1.411	0.172~	0.172~	0.657
6	124.250	-2.229	-3.233	1.691	-0.688~	-0.707
7	177.737	-0.276	-0.276	1.496	0.078~	0.964
8	185.634	-0.304	0.061~	0.426	0.769	1.134
9	89.399	-1.776	-1.035	0.197~	-0.678	-0.255
10	128.428	-1.343	-0.962	-0.578	-0.683	0.509~
11	66.521	-0.232	-0.133	0.097	-0.232	-0.050~
12	155.636	-0.392	-0.050~	0.712	-0.392	0.712
13	66.528	-0.197~	-0.299	0.239	0.261	-----
14	120.544	-1.134	-2.134	-0.185~	-1.160	-----
15	90.166	0.109~	0.277	0.630	0.445	-----
16	172.791	0.072~	0.079	0.775	1.128	0.953
17	89.237	0.068~	0.279	0.351	0.351	-----
18	88.648	0.115~	0.125	0.365	0.608	-----
19	91.839	-0.766	0.234~	-0.421	0.365	-----
20	183.781	-0.733	-0.286	0.833	0.161~	-0.267
21	57.286	0.454	0.097~	0.097~	0.275	-----
22	46.757	-0.307	-0.307	-0.069~	-0.628	-----
23	99.563	10.117	1.695	1.187~	5.987	-----
24	N/A	N/A	N/A	N/A	N/A	N/A
25	N/A	N/A	N/A	N/A	N/A	N/A

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK
^ 1 inch = 75 line units
~ Best results for the given line

Table 11. Average and Mean Deviations for Anchor Line Length

METHOD	NEGATIVE AVERAGE	POSITIVE AVERAGE	MEAN
GROUP A			
HILDITCH	-0.751 (14)	0.091 (2)	-0.645
CHEN-HSU	-1.040 (13)	0.139 (3)	-0.819
DEUTSCH	-0.253 (4)	0.633 (12)	0.412
SUETENS*	-0.670 (9)	0.461 (7)	0.175
BUSH	-0.397 (4)	0.762 (9)	0.440
GROUP B			
HILDITCH	-0.602 (3)	0.212 (3)	-0.195
CHEN-HSU	-0.297 (2)	0.147 (4)	0.024
DEUTSCH	-0.245 (2)	0.412 (4)	0.193
SUETENS*	-0.628 (1)	0.352 (5)	0.189

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

The Deutsch algorithm has the opposite effect. Deutsch lengthens the anchor line length of 12 lines an average of 0.633 and shortens 4 lines an average of -0.253, with mean of 0.412. Suetens et. al. is the only method that provides relative balance between positive and negative values. Suetens et. al. shortens the anchor line length for 9 lines an average of -0.670 and lengthens 7 lines an average of 0.461, with mean of 0.175. This stability makes Suetens et. al. the best algorithm in endpoint placement for lines in Group A. Suetens et. al. is second to Chen-Hsu for lines in Group B.

Table 12 displays percent difference between original and final anchor line lengths after they have been converted to inches. The complexity of the body of the lines should not effect the placement of the endpoints, in most cases. Consequently, percentage errors recorded for Group B are not drastically greater than those of Group A, with the exception of line 23. The percent deviation for Hilditch and Suetens et. al. increase sharply for line 23.

3. Fractal Dimension

Deviation from original fractal dimension is outlined in Table 13. As with errors in line length, it is expected that intricate lines would generate more error than the simple ones. Proving this assumption, the overall trend shown in Table 13 is that the greater the original fractal dimension the greater the deviation by the final output. In addition, for lines with fractal dimensions greater than

Table 12. Percent Deviation from Original Anchor Line by Line

LINE	ORIGINAL ANCHOR LINE LENGTH (inches)	PERCENT DEVIATION				
		HILDITCH	CHEN-HSU	DEUTSCH	SUETENS*	BUSH
1	1.6400	-0.44	-0 85	-0.03	-0.44	0.78
2	2.4286	-0.69	-0.30	-0.12	-0.69	0.20
3	1.8255	-0.29	-0.93	0.21	-0.29	0.44
4	2.3221	-0.07	-0.65	0.50	0.21	-0.07
5	1.6870	-0.25	-0.11	0.14	0.14	0.52
6	1.6567	-1.79	-2.60	1.36	-0.55	-0.57
7	2.3698	-0.15	-0.16	0.84	0.04	0.54
8	2.4751	-0.16	0.03	0.23	0.41	0.61
9	1.1920	-1.02	-1.16	0.22	-0.76	-0.29
10	1.7124	-1.04	-0.75	-0.45	-0.53	0.40
11	0.8869	-0.35	-0.20	0.15	-0.35	-0.08
12	2.0751	-0.25	-0.03	0.45	-0.25	0.45
13	0.8870	-0.30	-0.45	0.36	0.39	-----
14	1.6073	-0.94	-1.77	-0.15	-0.96	-----
15	1.2022	0.12	0.31	0.70	0.50	-----
16	2.3039	0.04	0.05	0.45	0.65	0.55
17	1.1898	0.08	0.31	0.39	0.40	-----
18	1.1820	0.13	0.14	0.41	0.69	-----
19	1.2245	-0.83	0.25	-0.55	0.40	-----
20	2.4504	-0.40	-0.16	0.45	0.09	-0.15
21	0.7638	0.79	0.17	0.17	0.48	-----
22	0.6234	-0.66	-0.66	-0.15	-1.34	-----
23	1.3275	10.16	1.70	1.19	6.01	-----
24	N/A	N/A	N/A	N/A	N/A	N/A
25	N/A	N/A	N/A	N/A	N/A	N/A

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

Table 13. Deviation from Original Fractal Dimension by Line.

LINE	ORIGINAL FRACTAL DIMENSION	DEVIATION				
		HILDITCH	CHEN-HSU	DEUTSCH	SUETENS*	BUSH
1	1.000	0.000~	0.000~	0.000~	0.000~	0.000~
2	1.000	0.001	0.001	0.001	0.001	0.000~
3	1.001	0.004	0.004	0.004	0.004	0.000~
4	1.002	0.001~	0.001~	0.001~	0.001~	0.001~
5	1.005	0.001	0.003	0.002	0.001	0.000~
6	1.005	0.000~	0.001	0.001	0.001	-0.001
7	1.010	0.001	0.002	0.003	0.001	0.000~
8	1.018	-0.002	-0.001~	-0.001~	-0.002	-0.002
9	1.023	0.000~	0.003	0.003	0.002	-0.003
10	1.035	-0.004	-0.003	0.000~	-0.003	-0.006
11	1.051	-0.005~	-0.005~	-0.006	-0.006	-0.008
12	1.061	-0.031	-0.033	-0.018~	-0.031	-0.023
13	1.061	-0.014	-0.012~	-0.013	-0.013	-----
14	1.068	-0.008	-0.006~	-0.006~	-0.006~	-----
15	1.088	-0.012~	-0.013	-0.013	-0.013	-----
16	1.097	-0.013	-0.014	-0.010	-0.013	-0.003~
17	1.170	-0.024	-0.023~	-0.024	-0.024	-----
18	1.196	-0.035	-0.073	-0.073	-0.034~	-----
19	1.211	-0.034~	-0.039	-0.039	-0.034~	-----
20	1.214	-0.042	-0.039	-0.032	-0.040	-0.024~
21	1.251	-0.055	-0.029~	-0.029~	-0.055	-----
22	1.278	-0.045	-0.040~	-0.043	-0.046	-----
23	1.331	-0.057	-0.056	-0.052~	-0.055	-----
24	N/A	N/A	N/A	N/A	N/A	N/A
25	N/A	N/A	N/A	N/A	N/A	N/A

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

~ Best results for the given line

1.05, every algorithm removes detail from the lines. The majority of this generalization probably occurs during the scanning process. The more complicated the line, the higher the probability that the scanning process will cause seemingly minor details to blend together or that curves will lose some of their sharpness. Once detail is lost during scanning, it can not be regained during the thinning process. The remainder of the generalization problem is due to the inability of the thinning methods to distinguish between detail and noise within the raster lines. Plus, there is a tendency for the successive layer removal methods to further smooth curves.

Also in Table 13, between the dimensions of 1.05 and 1.06, there is a noticeable shift in the abilities of the thinning algorithms. This displacement is more conspicuous when the error is represented as percent deviation (Table 14). This distinct break in the values leads to a division within Group A alone the fractal dimension of 1.055. Table 15 depicts the average deviations from the original fractal dimensions for Groups A and B. It also divides Group A into two parts. Group A1 contains lines 1 through 11, and Group A2 contains lines 12 through 16. This division reinforces observations recorded for raster and anchor line length results for the Hilditch algorithms (Sections II,B and III,A,2). There is a minor fluctuation among the average deviations within Groups A and B. When Group A is split, there is no difference among the averages within Group A1. Overall, the Deutsch algorithm does do slightly better than the others with lines in

Table 14. Percent Deviation from Original Fractal Dimension by Line.

LINE	ORIGINAL FRACTAL DIMENSION	PERCENT DEVIATION				
		HILDITCH	CHEN-HSU	DEUTSCH	SUETENS*	BUSH
1	1.000	0.00	0.00	0.00	0.00	0.00
2	1.000	0.10	0.10	0.10	0.10	0.00
3	1.001	0.40	0.40	0.40	0.40	0.00
4	1.002	0.10	0.10	0.10	0.10	0.10
5	1.005	0.10	0.30	0.20	0.10	0.00
6	1.005	0.00	0.10	0.10	0.10	-0.10
7	1.010	0.10	0.20	0.30	0.10	0.00
8	1.018	-0.20	-0.10	-0.10	-0.20	-0.20
9	1.023	0.00	0.29	0.29	0.20	-0.29
10	1.035	-0.39	-0.29	0.00	-0.29	-0.58
11	1.051	-0.48	-0.48	-0.57	-0.57	-0.76
12	1.061	-2.92	-3.11	-1.70	-2.92	-2.17
13	1.061	-1.32	-1.13	-1.23	-1.23	-----
14	1.068	-0.75	-0.56	-0.56	-0.56	-----
15	1.088	-1.10	-1.19	-1.19	-1.19	-----
16	1.097	-1.19	-1.28	-0.91	-1.19	-0.27
17	1.170	-2.05	-1.97	-2.05	-2.05	-----
18	1.196	-2.93	-6.10	-6.10	-2.84	-----
19	1.211	-2.81	-3.22	-3.22	-2.81	-----
20	1.214	-3.46	-3.21	-2.64	-3.29	-1.98
21	1.251	-4.40	-2.32	-2.32	-4.40	-----
22	1.278	-3.52	-3.13	-3.36	-3.60	-----
23	1.331	-4.28	-4.21	-3.91	-4.13	-----
24	N/A	N/A	N/A	N/A	N/A	N/A
25	N/A	N/A	N/A	N/A	N/A	N/A

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

\- Best results for the given line

Table 15. Average Deviation from Original Fractal Dimension by Group

GROUP	AVERAGE DEVIATION				
	HILDITCH	CHEN-HSU	DEUTSCH	SUETENS*	BUSH
A	0.006	0.006	0.005	0.006	0.004~
B	0.042	0.043	0.042	0.041	N/A
A1	0.002	0.002	0.002	0.002	0.002
A2	0.016	0.016	0.012	0.015	0.013^
B	0.042	0.043	0.042	0.041	N/A
A,B	0.017	0.017	0.016	0.017	N/A

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK
~ based on the 13 lines followed
^ based on the 2 lines followed

Group A2. Based on the first 23 lines, there is not any overpowering proof that one method preserves line character better than the others.

B. Vector Summary

The results pertaining to line length, anchor line length, and fractal dimension do not paint a clear picture as to which method produces the best vector output. Fractal dimension has not proven itself to be an effective tool in comparing the various thinning methods. All methods preserve or remove line character to approximately the same degree as fractal dimension increases. Line length and anchor line length often contradict one another, as illustrated by Table 16. The value within the parentheses is the sum of the rounded, absolute value of percent deviation measures displayed in Table 8 for line length and Table 12 for anchor line length. Since it is difficult to find map lines with a fractal dimension over 1.3, it is not deemed unreasonable to remove line 23 from the final vector analysis. Plus, the raster results displayed in Table 5 illustrate that it is impractical to expect any method to thin such complicated lines. Therefore, Group B refers to lines 17 to 22 in Table 16 and all subsequent tables.

If an algorithm adequately preserves line length, it is the least efficient in dealing with anchor line length. Due to the inability to statistically fit the output lines to the input lines, these variables do not measure directly the accuracy of the final vector line. This

Table 16. Contradictory Vector Results - Successive Layer Removal

Group A	best			worst
line length	Chen-Hsu (46)	Hilditch (47)	Suetens* (54)	Deutsch (66)
anchor line length	Deutsch (6)	Suetens* (7)	Hilditch (8)	Chen-Hsu (10)
Group B~	**best**			**worst**
line length	Hilditch (16)	Chen-Hsu (19)	Suetens* (20)	Deutsch (24)
anchor line length	Chen-Hsu (1.6)	Deutsch (2.1)	Suetens* (3)	Hilditch (6)
Groups A & B~	**best**			**worst**
line length	Hilditch (64)	Chen-Hsu (66)	Suetens* (74)	Deutsch (90)
anchor line length	Deutsch (8)	Suetens* (11)	Chen-Hsu (12)	Hilditch (14)

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

~ Group B - lines 17 to 22

fact makes it difficult to determine the significance of the observed contradiction or to objectively judge the value of the variables as evaluation tools. The intuitive judgement is that anchor line length results provide more pertinent information than line length.

According to the raster results (Table 4), the Hilditch method deletes extra pixels from the ends of the raster lines, thereby shortening the lines. This observation is confirmed by the anchor line length results in Table 16. Anchor line length measures the relative position of the endpoints. Line length does not divulge any information concerning position, relative or otherwise. Since no smoothing process is incorporated before measuring line length, the step-like character imposed by the pixel selection process adds length. Consequently, a line that is too long but has the desired anchor line length is more accurate than a line that is the desired length but has a short anchor line length, assuming that error in line length is not extreme. Based on this subjective premise, Suetens et. al. produces the best results from Group A, followed by Hilditch. From Group B, Chen-Hsu produces the best results, followed by Suetens et. al. And for both groups combined, Suetens et. al. and Chen-Hsu produce the best results. The line length error for Deutsch is always extreme.

As mentioned, the sequence of affectivity within Group A in Table 16 shows are complete reversal between line length and anchor line length for Group A and for both groups combined. When the performance of the Bush algorithm is considered, this pattern is broken. Table 17

Table 17. Contradictory Vector Results - All Algorithms - 13 Lines

	best				worst
line length	Bush (25)	Chen-Hsu (45)	Hilditch (46)	Suetens* (52)	Deutsch (62)
anchor line length	Deutsch (5.2)	Suetens* (5.3)	Bush (5.5)	Hilditch (6.5)	Chen-Hsu (7.8)

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

shows the comparative results for the 13 lines that the Bush algorithm can follow. Based on the above premise, the error in line length for all of the successive layer removal methods can be considered extreme compared to the line length error for Bush. Hence, Bush produces the best results followed by Suetens et. al. and Hilditch.

To combine the vector results, the absolute values of percent deviations from Tables 8 (line length), 12 (anchor line length), and 14 (fractal dimension) are added together to show an accumulative percent deviation for each method within each group. Since it has been determined that mean anchor line length provides insight into the stability of endpoint placement, the absolute value of means from Table 11 are added to the anchor line length variable. Since the anchor line length variable is only one that measure the relative placement of the line, its contribution to the total error is doubled. The accumulative error measure are presented in Table 18.

When the three measures are combined the above premise is almost completely reinforced by actual values. The expected results for Group A are reversed. Hilditch (74) ranks first, with Suetens et. al. (77.9) and Chen-Hsu (78.4) in a relative tie. For Group B, the results do satisfy expections. Chen-Hsu (43) ranks first; Suetens et. al. (46) is second, followed closely by Hilditch (47). When the three measures and three groups are combined, the results again are as anticipated. The sequence of final vector results are: Suetens-Dierckx-Piessens-Oosterlinck (127), Chen-Hsu (128), Hilditch (130), and Deutsch (144).

Table 18. Accumulative Error Measures for Vector Results

GROUP	METHOD	ACCUMULATIVE ERROR MEASURES				RANK
		LINE LENGTH	ANCHOR LINE LENGTH & MEAN	FRACTAL DIMENSION	TOTAL ERROR	
A	HILDITCH	47.34	8.55	9.15	73.59	1
	CHEN-HSU	46.44	11.17	9.63	78.41	3
	DEUTSCH	65.77	6.77	7.75	87.06	4
	SUETENS*	53.96	7.34	9.25	77.89	2
	BUSH-	25.24	5.94	4.47	41.59	-
B	HILDITCH	16.25	5.98	19.17	47.38	3
	CHEN-HSU	19.27	1.69	19.95	42.60	1
	DEUTSCH	24.28	2.31	19.69	48.59	4
	SUETENS*	20.18	3.59	18.99	46.35	2
C	HILDITCH	8.95	-	-	8.95	4
	CHEN-HSU	7.35	-	-	7.35	2
	DEUTSCH	8.43	-	-	8.43	3
	SUETENS*	6.00	-	-	6.00	1
A,B,C	HILDITCH	72.54	14.53	28.32	129.92	3
	CHEN-HSU	73.06	12.86	29.58	128.36	2
	DEUTSCH	98.48	9.08	27.44	144.08	4
	SUETENS*	80.14	9.53	28.24	127.44	1

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

\- 13 out of 16 lines

CHAPTER 5

DISCUSSION

This study was developed to analyze various line thinning methods on scanned line data in respects to both raster and vector outputs. A correlation between the raster and vector results was anticipated that would provide distinction among the methods tested and establish guidelines for testing additional methods.

Among the variables used to evaluate the raster results, the medial axis value provides indirect information concerning the quality of the remaining line, but the lack of direct information concerning error limits its usefulness. It is not incorporated into this study due to limitations. The total number of pixels deleted from the scanned line has little value in determining the accuracy of the final raster line. To clarify the significance of this variable, it should be accompanied by another variable that measures the quality of the line. The number of pixels both on and off the medial axis provides qualifying information concerning placement of the raster line. Therefore, the number of pixels on and off the medial axis combined with the total number of pixels deleted creates a powerful mechanism by which to gage raster line accuracy.

Among the variables used to evaluate the vector results, fractal dimension did not emphasize any major differences among the methods tested. The algorithms generalized the lines to similar degrees as the

fractal dimensions increased. When there is no means to mathematically fit the output line to the input line, line length and anchor line length only furnish indirect information concerning the accuracy of the vector line. When no smoothing routine is implemented before measuring the vector output, line length increases due to the centralized pixel selection process. This factor makes the significance of this variable difficult to interpret. Anchor line length provides information about the relative placement of the endpoints. The distribution of positive and negative anchor line lengths is vital to determining the overall stability of this endpoint placement. The combination of anchor line length and mean anchor line length supplies the best source of vector line accuracy.

Table 19 combines the raster and vector error measures. The accumulative raster error is computed by adding together the percent deviations based on the demerit system listed in Table 6. The accumulative vector error comes from Table 18. For Group A, the raster and vector results reinforce one another so that the methods continue to rank in the following order: Hilditch (110), Suetens et. al. (115), Chen-Hsu (224), Deutsch (390). The Bush algorithm can only follow 81 percent of the lines in Group A, but the error received for those lines is only 53 percent of Hilditch's total value. This observation is very encouraging for the principles behind the Bush line following method. For Group B, the diversity in the raster results overshadows the contribution of the vector results, so the sequence is: Suetens et.

Table 19. Combined Final Results

GROUP	METHOD	ACCUMULATIVE ERROR MEASURES			RANK
		RASTER	VECTOR	TOTALS	
A	HILDITCH	36.14	73.59	109.73	1
	CHEN-HSU	146.08	78.41	224.49	3
	DEUTSCH	303.41	87.06	390.47	4
	SUETENS*	37.36	77.89	115.25	2
	BUSH-	16.70	41.59	58.29	-
B	HILDITCH	16.17	47.38	63.55	2
	CHEN-HSU	75.35	42.60	117.95	3
	DEUTSCH	117.71	48.59	166.30	4
	SUETENS*	15.92	46.35	62.27	1
C	HILDITCH	0.79	8.95	9.74	2
	CHEN-HSU	21.24	7.35	28.59	3
	DEUTSCH	23.92	8.43	32.35	4
	SUETENS*	2.25	6.00	8.25	1
A,B,C	HILDITCH	53.10	129.92	183.02	2
	CHEN-HSU	242.67	128.36	371.03	3
	DEUTSCH	445.04	144.08	589.12	4
	SUETENS*	45.53	127.44	172.97	1

* SUETENS-DIERCKX-PIESSENS-OOSTERLINCK

\- 13 out of 16 lines

al. (62), Hilditch (64), Chen-Hsu (118), Deutsch (166). For all the groups combined, the anticipated results outlined at the beginning of Chapter 4 are altered. As opposed to Hilditch, Suetens et. al., Deutsch, Chen-Hsu, the order of final combined results for this line thinning test are: Suetens-Dierckx-Piessens-Oosterlinck (173), Hilditch (183), Chen-Hsu (371), and Deutsch (589).

If only one method is to be used to thin a scanned map containing line features encompassing the full range of fractal dimensions, than Suetens-Dierckx-Piessens-Oosterlinck performs slightly better than Hilditch. If the map is divided into regions of similar line complexity, than the thinning method can be selected based on the mean fractal dimension of the area. If the mean fractal dimension is less than 1.055, than use Suetens et. al. If it is between 1.055 and 1.200, than use Hilditch. If the mean fractal dimension is greater than 1.200, than use Suetens et. al. If the area consists of a grid pattern with straight line segments and many intersections, than maybe Chen-Hsu would be the method of choice. Since it is unrealistic to expect one method to handle all situations, it is important to know the strong points of the methods available and use them accordingly.

The benefits of the Bush algorithm are more for the researcher than for the future of line thinning methods. Attempting to develop my own near perfect method illustrated clearly how difficult it is to design one method that can handle all possible situations. It is personally rewarding to see that the ideas behind my method form

improvements over many features of the successive layer removal methods, especially the adjustment to vector coordinate placement. In addition, the Bush algorithm also represents an attempt to shift the focus within the development of line thinning methods toward cartographic considerations.

In the study of line thinning methods, there needs to be a division between methods that are designed for pattern recognition and those designed for patterned line recognition. The concerns of character recognition and the concerns of map accuracy are simply different. When scanning a map, the names are one of the least important features to be collected; whereas in character recognition they are the features being collected. Thinning lines to achieve accuracy needs to receive greater attention, since computer mapping continues to gain dominance. Unfortunately, for this study, there has not been enough information available in literature to rely solely on studies of scanned map accuracy. Hopefully, that will change in the future.

APPENDIX A

Hilditch Algorithm

```
Order of neighbors:  7 0 1
                     6   2
                     5 4 3

begin
      do (until no pixels can be deleted)
           do (until pass through matrix is complete)
                for (pixel = 1)
                     read eight neighbors
                     count axial 0-neighbors (edge)
                     count all line neighbors (endpt)
                     count 1-neighbors (circ)
                     calculate crossing number (cross)

                     if (p[0] = 2  and/or  p[6] = 2)
                          let p[0] = 0 temporarily
                            and/or
                          let p[6] = 0 temporarily
                          calculate second crossing number (cros2)
                     endif

                     if (edge >= 2  and  endpt >= 2  and  circ > 0
                         and  cross = 1  and
                         (p[0] != 2  or  cros2 = 1)  and
                         (p[6] != 2  or  cros2 = 1))
                               pixel = 2  (flagged for deletion)
                     endif
                endfor
           enddo
           delete flagged pixels
      enddo
end


Hilditch's crossing number:

           for (n = 0,2,4,6)
                if (p[n] = 0  and
                    (p[n+2] >= 1  or
                     (p[n+1] >= 1 and p[n+2] = 0)))
                          cross = cross + 1
                endif
           endfor
```

APPENDIX B

Modified Stefanelli-Rosenfeld Algorithm

```
Order of neighbors:  7 0 1
                     6   2
                     5 4 3
begin
     do (until no pixels can be deleted)
          do (until pass through matrix is complete)
               for (pixel = 1)
                    read eight neighbors
                    calculate crossing number (cross)
                    count line neighbors (nabrs)
                    N = p[6] * p[0] * p[2]
                    E = p[0] * p[2] * p[4]
                    S = p[2] * p[4] * p[6]
                    W = p[4] * p[6] * p[0]

                    if (N > 0)
                         shift window up one pixel
                         calculate crossing number (crosN)
                    endif
                    if (E > 0)
                         shift window right one pixel
                         calculate crossing number (crosE)
                    endif
                    if (S > 0)
                         shift window down one pixel
                         calculate crossing number (crosS)
                    endif
                    if (W > 0)
                         shift window left one pixel
                         calculate crossing number (crosW)
                    endif

                    if (2 <= nabrs <= 6  and  cross = 1  and
                         ((N = 0  or  crosN != 1)  and
                          (E = 0  or  crosE != 1))  and
                         ((S = 0  or  crosS != 1)  and
                          (W = 0  or  crosW != 1)))
                              pixel = 2  (flagged)
                    endif
               endfor
          enddo
          delete flagged pixels
     enddo
end
```

crossing number for cross:

```
for (n = 0,1,...,7)
     if (p[n] = 0  and  p[n+1] >= 1)
          cross = cross + 1
     endif
endfor
```

crossing number for crosN, crosE, crosS, and crosW:

```
for (n = 0,1,...,7)
     if (p[n] = 0  and  p[n+1] = 1)
          cros? = cros? + 1
     endif
endfor
```

APPENDIX C

Deutsch Algorithm

```
Order of neighbors:  5 6 7
                     4   0
                     3 2 1

begin
      do (until no pixel can be deleted)
           do (until pass through matrix is complete)
                for (pixel = 1)
                     read eight neighbors
                     calculate crossing number (cross)
                     count line neighbors (nabrs)

                     if (pass = 1  and  nabrs > 1)
                          south = p[0] * p[2] * p[4]
                          east = p[0] * p[2] * p[6]
                          NE = p[0] + p[6]
                          SW = p[2] + p[3] + p[4] + p[7]
                          SE = p[0] + p[2]
                          NW = p[1] + p[4] + p[5] + p[6]

                          if (p[1] = 1  or  p[5] = 1)
                               diag1 = 1
                          endif
                          if (p[3] = 1  or  p[7] = 1)
                               diag2 = 1
                          endif

                          if (cross = 2)
                               if (south = 0  and  east = 0)
                                    pixel = 2  (flagged)
                               endif
                          else
                          if (cross = 4)
                               if (south = 0  and  east = 0  and
                                   ((NE = 2  and  SW = 0  and
                                     diag1 = 1)  or
                                    (SE = 2  and  NW = 0  and
                                     diag2 = 1)))
                                         pixel = 2  (flagged)
                               endif
                          endif
                     else
```

```
                    if (pass = 2  and  nabrs > 1)
                        north = p[0] * p[4] * p[6]
                        west = p[2] * p[4] * p[6]
                        SW = p[2] + p[4]
                        NE = p[0] + p[3] + p[6] + p[7]
                        NW = p[4] + p[6]
                        SE = p[0] + p[1] + p[2] + p[5]

                        if (p[1] = 1  or  p[5] = 1)
                             diag1 = 1
                        endif
                        if (p[3] = 1  or  p[7] = 1)
                             diag2 = 1
                        endif
                        if (cross = 2)
                             if (north = 0  and  west = 0)
                                  pixel = 2  (flagged)
                             endif
                        else
                        if (cross = 4)
                             if (north = 0  and west = 0  and
                                 ((SW = 2  and  NE = 0  and
                                   diag1 = 1)  or
                                  (NW = 2  and  SE = 0  and
                                   diag2 = 1)))
                                      pixel = 2  (flagged)
                             endif
                        endif
                    endif
               endfor
          enddo
          delete flagged pixels
          switch pass number
     enddo
     (optional)
     do (until no pixels can be deleted)
          do (until pass through matrix is complete)
               for (pixel = 1)
                    read axial neighbors
                    count axial line neighbors
                    corner = p[n] + p[n+1]   (n=1,2,3,4)
                    if (corner = 2  and  nabrs = 2)
                         delete pixel
                    endif
               endfor
          enddo
     enddo
end
```

APPENDIX D

Chen-Hsu Algorithm

```
Order of Neighbors:  7 0 1
                     6   2
                     5 4 3

begin
      do (until no pixels can be deleted)
           do (until pass through matrix is complete)
                for (pixel = 1)
                     read eight neighbors
                     calculate crossing number  (cross)
                     count line neighbors  (nabrs)

                     if (pass = 1  and  2 <= nabrs <= 7)
                          if (cross = 1)
                               east = p[0] * p[2] * p[4]
                               south = p[2] * p[4] * p[6]
                               if (east = 0  and  south = 0)
                                    pixel = 2  (flagged)
                               endif
                          else
                          if (cross = 2)
                               NE = p[0] * p[2]
                               SW = p[4] + p[5] + p[6]
                               SE = p[2] * p[4]
                               NW = p[0] + p[6] + p[7]
                               if ((NE = 1  and  SW = 0)  or
                                   (SE = 1  and  NW = 0))
                                         pixel = 2
                               endif
                          endif
                     else

                     if (pass = 2  and  2 <= nabrs <= 7)
                          if (cross = 1)
                               west = p[0] * p[4] * p[6]
                               north = p[0] * p[2] * p[6]
                               if (west = 0  and  north = 0)
                                    pixel = 2  (flagged)
                               endif
                          else
```

```
                    if (cross = 2)
                         NW = p[0] * p[6]
                         SE = p[2] + p[3] + p[4]
                         SW = p[4] * p[6]
                         NE = p[0] + p[1] + p[2]
                         if ((NW = 1  and  SE = 0)  or
                             (SW = 1  and  NE = 0))
                                   pixel = 2
                         endif
                    endif
               endif
          endfor
     enddo
     delete flagged pixels
     switch pass number
enddo
end
```

In the Chen and Hsu article, there is a misprint in the algorithm outline. The crossing number is defined as the number cf 0-1 (background to unflagged line) patterns encountered when tracing the eight neighbors. In order to achieve the same results as Chen and Hsu, I also had to include the 0-2 (background to flagged line) patterns in this count.

APPENDIX E

Landy-Cohen Algorithm

```
Order of neighbors:  7 0 1
                     6   2
                     5 4 3

begin
      do (until no pixels can be deleted)
           do (until pass through matrix is complete)
                for (pixel = 1)
                     read eight neighbors
                     calculate crossing number  (cross)
                     count line neighbors  (nabrs)
                     if (cross = 2  and  3 <= nabrs <= 5)
                          delete pixel
                     endif
                endfor
           enddo
      enddo

      do (until no pixels can be deleted)
           do (until pass though matrix is complete)
                for (pixel = 1)
                     read eight neighbors
                     calculate crossing number  (cross)
                     count line neighbors  (nabrs)
                     count gaps

                     if (gaps = 1)
                          if ((nabrs = 2  and  cross = 4)  or
                               (nabrs = 3  and  (cross = 2 or 4))
                               or  (4 <= nabrs <= 6))
                                    delete pixel
                          endif
                     endif
                endfor
           enddo
      enddo
end
```

APPENDIX F

Suetens Algorithm

```
Order of neighbors:  5 6 7
                     4   0
                     3 2 1

begin
     do (until no pixels can be deleted)
          px = 1
          do (until no pixels can be deleted)
               do (until pass through matrix is complete)

                    for (pixel = px)
                         read eight neighbors
                         calculate crossing number (cross)
                         count line neighbors (nabrs)

                         if (cross != 2  or  nabrs != 2)
                              val = 1
                         endif

                         if (pass = 1  and  nabrs != 1)
                              south = p[0] * p[2] * p[4]
                              east = p[0] * p[2] * p[6]

                              if (p[0] > 0  and  p[6] > 0)
                                   NE = 2
                              endif

                              NW = p[1] + p[4] + p[5] + p[6]
                              SW = p[2] + p[3] + p[4] + p[7]

                              if (p[0] > 0  and  p[2] > 0)
                                   SE = 2
                              endif


                              if (p[1] > 1  or  p[5] > 1)
                                   diag1 = 1
                              endif

                              if (p[3] > 1  or  p[7] > 1)
                                   diag2 = 1
                              endif
```

```
     if (cross = 2  and  val = 1)
          if (south = 0  and  east = 0)
               pixel = 99  (flagged)
          endif
     else

     if (cross = 4  and  val = 1)
        if (south = 0  and  east = 0  and
            ((NE = 2  and  SW = 0  and
              diag1 = 1)  or
             (SE = 2  and  NW = 0  and
              diag2 = 1)))
                 pixel = 99
        endif
     endif
else

if (pass = 2  and  nabrs !=1)
     north = p[0] * p[4] * p[6]
     west = p[2] * p[4] * p[6]

     if (p[2] > 0  and p[4] > 0)
          SW = 2
     endif

     SE = p[0] + p[1] + p[2] + p[5]
     NE = p[0] + p[3] + p[6] + p[7]

     if (p[4] > 0  and  p[6] > 0)
          NW = 2
     endif

     if (p[1] > 1  or  p[5] > 1)
          diag1 = 1
     endif

     if (p[3] > 1  or  p[7] > 1)
          diag2 = 1
     endif
```

```
                    if (cross = 2  and  val = 1)
                         if (north = 0  and  west = 0)
                              pixel = 99
                         endif
                    else

                    if (cross = 4  and  val = 1)
                       if (north = 0  and  west = 0  and
                           ((SW = 2  and  NE = 0  and
                             diag1 = 1)  or
                            (NW = 2  and  SE = 0  and
                             diag2 = 1)))
                                 pixel = 99
                        endif
                    endif
                 endif
            endfor
        enddo
        delete flagged pixels
        switch pass number
    enddo
    px = px + 1
enddo
end
```

APPENDIX G

Bush Algorithm

```
begin
     locate the starting pixel
     determine the initial processing direction
     determine the center of the row/column:
          find center
          vectorize pixel in 0.5 increments
          save vector coordinates into a temporary file
          check connectivity
          adjust raster file
               center or gap pixels = 99
               other line pixels = 11

     do (until end of line)
          increment pointer along processing path
          check for change in processing direction
          check for loop
          determine the center of the row/column (see above steps)
     enddo

     return to starting pixel
     reverse the initial processing direction

     do (until end of line)
          increment pointer along processing path
          check for change in processing direction
          check for loop
          determine the center of the row/column (see above steps)
     enddo

     delete one pixel from each end of the raster line
     save the final raster file

     join the two vector line segments at common starting point
     delete one point from each end of the vector line
     delete insignificant internal points (along straight segments)
     save the final vector file
end
```

APPENDIX H

Curve and Intersection Problems

Attempting to automate the manual thinning process for sharp curves and intersections introduces many unanswered questions. Is it necessary to determine if a curve or an intersection has been encountered? If so, how is this distinction made, and how will the thinning processes differ? Recognizing that there is an increase in line width is only the beginning of solving the problem. What other variables need to be considered? Do all eight neighbors need to be evaluted when determining line direction? Can steps be added to the existing Bush thinning method to handle the problem? If not, can a subroutine be developed tangent to the existing method, or does the method need to be completely redesigned to accommodate these situations from early stages?

To address some of these questions, a limited amount of time has been spent trying to revise the present program to thin sharp curves. The main modifications occurs during the selection of the processing direction (Chapter 3, Section II-B). The adjusted program compares the values of neighboring pixels as they are counted. The original procedure counts all pixels radiating from the pointer regardless of pixel values. In Figure 29a, the pixel order reading down the first column of ten pixels is: 1 2 3 3 2 1 2 3 2 1. Before checking line width, the pointer rests on the first 3. Therefore, the pixel count above the pointer is 2 and below is 7. The sequence 2 1 2 marks the

trough between two adjacent line segments. Using the modified program, the counting stops at the 1, making the count below the pointer 3, not 7. After the central pixel is selected, pixels flagged for deletion extend down to the first 1 pixel, as in Figure 37a, thereby allowing the processing to continue beyond the curve.

In the modified program, pixels flagged for deletion are not counted when determining line widths. In Figure 31a, the processing direction remains horizontal after the increase in line width due to the influence of the straight line segment, which was previously coded for deletion. In Figure 37b, the processing direction automatically switches to vertical when this infuence is removed. Due to this adjustment, instead of checking only for a change in direction, the processing direction must be completely re-evaluated at each step. The final results are shown in Figure 37c.

If a change in processing direction is not automatically induced, then it is forced. If the line width increases to over eight pixels, a variable is set indicating that a curve has been located. If the processing direction does not change before the second row/column in the curve is completely processed, then the direction is changed despite the widths.

Satisfactory results are achieved when thinning the sample curve in Figure 29a, but the sharp curve problem is not solved by this limited effort. The program still can not follow the majority of sharp curves used during the developmental stages. In addition, the changes

```
11||||||||||||||||||||1
12||||||||||||||||||||211
12*******************|32211
12||||||||||||||||||||*433221
11||||||||||||||||||||344321
                      |23321
                   11233321
                 112233221
                122332211
                1222211
               111111
```

|	Deleted Pixel by Column
*	Selected Pixel

a. Adjacent Line Segments

```
11||||||||||||||||||||1
12||||||||||||||||||||211
12*******************|32211
12||||||||||||||||||||P------
11||||||||||||||||||||GG*---
                      |--*--
                   11233321
                 112233221
                122332211
                1222211
               111111
```

|	Deleted Pixel by Column
-	Deleted Pixel by Row
*	Selected Pixel
P	Selected Pivot
G	Filled Gap

b. Pixels Flagged for Deletion

```
1||||||||||||||||||||1          ||||||||||||||||||||-
1||||||||||||||||||||211        ||||||||||||||||||||---
1*******************|32211      |*******************|-----
1||||||||||||||||||||P------     ||||||||||||||||||||*------
1||||||||||||||||||||GG*---      ||||||||||||||||||||***---
                      |--P--                          |--*--
                   ||||*|2|                          ||||*|||
                 1|||**||1                          -|||**|||
                12|**||||                          --|**||||
                --P||||                            --*||||
               111|||                             ---|||
```

c. Revised Results

Figure 37. Correcting Sharp Curve Problem

cause minor problems in lines that were initially thinned properly. Therefore, simply adding a few steps to the existing method probably will not solve the problem efficiently. Possibly, if these additional steps (plus whatever else is needed) are added to a subroutine only envoked to handle sharp curves, a solution can be developed with much cleaner and effective programming steps. Due to an incomplete solution, additional problems, and limits on time, the altered program is not used during the final tests.

REFERENCES CITED

Antenucci, John C., and Mickey Fain, 1991. "From Scanning to Automated Data Conversion," Geo Info Systems, May, pp. 54-57.

Arapahoe USGS 7.5 Minute Quadrangle at 1:24,000, Wyoming, 1953.

Arcelli, Carlo, and Gabriella Sanniti di Baja, 1981. "A Thinning Algorithm Based on Prominence Detection," Pattern Recognition, pp. 225-235.

Arcelli, Carlo, and Gabriella Sanniti di Baja, 1989. "A One-Pass Two-Operation Process to Detect the Skeletal Pixels on the 4-Distance Transform," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. II, No. 4, pp. 411-414.

Baruch, Orit, 1988. "Line Thinning by Line Following," Pattern Recognition Letters, Vol. 8, pp. 271-276.

Baumann, Melissa, 1991. System Engineer, Intergraph Corporation.

Berry, Joseph K., 1987. "Computer-Assisted Map Analysis: Potential and Pitfalls," Photogrammetric Engineering and Remote Sensing, Vol. 53, No. 10, pp. 1405-1410.

Blakemore, M., 1984. "Generalization and Error in Spatial Data Bases," Cartographica, Vol. 21, No. 2-3, pp. 131-139.

Broad Brook USGS 7.5 Minute Quadrangle at 1:24,000, Connecticut, 1964.

Burrough, P. A., 1986. Principles of Geographical Information Systems for Land Resources Assessment. Clarendon, Oxford.

Carstensen, William L., Associate Professor, Virginia Polytechnic Institute and State University. Modified Steffanelli-Rosenfeld FORTRAN program.

Chen, Yung-Sheng, and Wen-Hsing Hsu, 1988. "A Modified Fast Parallel Algorithm for Thinning Digital Patterns," Pattern Recognition Letters, Vol. 7, February, pp. 99-106.

Chrisman, Nicholas R., 1982. "The Role of Quality Information in the Long-Term Functioning of a Geographic Information System," Cartographica, Vol. 21, No. 2, pp. 79-87.

Coleman Gap USGS 7.5 Minute Quadrangle at 1:24,000, Tennessee-Virginia, 1946, photorevised 1969.

Deutsch, E. S., 1972. "Thinning Algorithms on Rectangular, Hexagonal, and Triangular Arrays," Communication of the ACM, Vol. 15, No. 9, pp. 827-837.

Fulford, Martin C., 1981. "The FASTRAK Automatic Digitizing System," Pattern Recognition, Vol. 14, Nos. 1-6, pp. 65-74.

Goodchild, Michael F., and David M. Mark, 1987. "The Fractal Nature of Geographic Phenomena," The Annals of the Association of American Geographers, Vol. 77, pp. 265-278.

Greenlee, David D., 1987. "Raster and Vector Processing for Scanned Linework," Photogrammetric Engineering and Remote Sensing, Vol. 53, No. 10, pp. 1383-1387.

Hilditch, C. Judith, 1969. "Linear Skeletons from Square Cupboards," Machine Intelligence, Vol. 4, pp. 402-420.

Holmes, David D., 1991. "Automated Data Capture for Geographic Information Systems," Surveying and Land Information Systems, Vol. 51, No. 2, pp. 87-92.

Holt, C. M., A. Stewart, M. Clint, and R. H. Perrott, 1987. "An Improved Parallel Thinning Algorithm," Communications of the ACM, Vol. 30, No. 2, pp. 156-160.

Jenks, George F., 1981. "Lines, Computers, and Human Frailities," The Annals of the Association of American Geographers, Vol. 71, No. 1, pp. 1-10.

Leberl, Franz, and Dale Olson, 1982. "Raster Scanning for Operational Digitizing of Graphical Data," Photogrammetric Engineering and Remote Sensing, Vol. 48, No. 4, pp. 615-627.

Landy, Michael S., and Yoav Cohen, 1985. "Vectorgraph Coding: Efficient Coding of Line Drawings," Computer Graphics and Image Processing, Vol. 30, pp. 331-344.

Longley, Paul A., and Michael Batty, 1989. "On the Fractal Measurement of Geographical Boundaries," Geographical Analysis, Vol. 21, No. 1, pp. 47-67.

Peuquet, Donna J., 1981. "An Examination of Techniques for Reformatting Digital Cartographic Data/Part 1: The Raster-To-Vector Process," Cartographica, Vol. 18, No. 1, pp. 34-48.

Rosenfeld, Azriel, and John L. Pfaltz, 1966. "Sequential Operations in Digital Picture Processing," Journal of the ACM, Vol. 13, No. 4, pp. 471-494.

Snyder, John P., 1987. Map Projections - A Working Manual. U.S. Geological Survey Professional Paper 1395, U.S. Government Printing Office, Washington.

Suetens, P., P. Dierckx, R. Piessens, and A. Oosterlinck, 1981. "A Semiautomatic Digitization Method and the Use of Spline Functions in Processing Line Drawings," Computer Graphics and Image Processing, Vol. 15, pp. 390-400.

Sugawara, Sandra, 1989. "Putting GIS on the Map: Geographic Information Is the Latest in Computer Cartography," Washington Post, Business Section, September 25, pp. 1 & 34,35.

Stefanelli, R., and A. Rosenfeld, 1971. "Some Parallel Thinning Algorithms for Digital Pictures," Journal of the Association for Computing Machinery, Vol.18, No.2, pp. 255-264.

Tenino USGS-DMA Topo at 1:50,000, Washington, 1975.

Toms Brook USGS 7.5 Minute Quadrangle at 1:24,000, Virginia, 1966, photo inspected 1972.

Tyrone USGS 15 Minute Quadrangle at 1:62:500, Pennsylvania, 1929.

van Vliet, Lucas J. van, and Ben J. H. Verwer, 1988. "A Contour Processing Method for Fast Binary Neighbourhood Operations," Pattern Recognition Letters, Vol.7, pp. 27-36.

Xia, Yun, 1989. "Skeletonization Via the Realization of the Fire Front's Propagation and Extinction in Digital Binary Shapes," IEEE Transactions on Pattern Analysisand Machine Intelligence, Vol. II, No. 10, pp. 1076-1086.

Young, John, 1991. Effects of Automated Cartographic Generalization on Linear Map Features. Master's Thesis. Virginia Polytechnic Institute and State University.

Zhang, T. Y., and C. Y. Suen, 1984. "A Fast Parallel Algorithm for Thinning Digital Patterns," Communications of the ACM, Vol. 27, No. 3, pp. 236-239.

ADDITIONAL REFERENCES

Brassel, Kurt E., and Robert Weibel, 1988. "A Review and Conceptual Framework of Automated Map Generalization," Int. J. Geographical Information Systems, Vol.2, No. 3, pp. 229-244.

Buttenfield, Barbara, 1985. "Treatment of the Cartographic Line," Cartographica, Vol. 22, No. 2, pp. 1-26.

Chen, Yung-Sheng, and Wen-Hsing Hsu, 1989. "A Systematic Approach For Designing 2-Subcycle And Pseudo 1-Subcycle Parallel Thinning Algorithms," Pattern Recognition, Vol. 22, No. 3, pp. 267-282.

Cheng, Fang-Hsuan, and Wen-Hsing Hsu, 1988. "Parallel Algorithm for Corner Finding on Digital Curves," Pattern Recognition Letters, Vol. 8, pp. 47-53.

Dahlberg, Richard E., 1985. "Combining Data from Different Sources," Challenge Paper for the Workshop on Fundamental Research Needs in Surveying, Mapping, and Land Information Systems, Blacksburg, VA, November, pp. 112-131.

Dyer, Charles R., and Azriel Rosenfeld, 1979. "Thinning Algorithms for Gray-Scale Pictures," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 1, No. 1, pp. 88-89.

Freeman, Herbert, 1974. "Computer Processing of Line-Drawing Images," Computer Surveys, Vol. 6, No. 1, pp. 57-97.

McMaster, Robert B., 1986. "A Statistical Analysis of Mathematical Measures for Linear Simplification," The American Cartographer, Vol. 13, No. 2, pp. 103-116.

Pavlidis, Theo, 1982. Algorithms for Graphics and Image Processing. Chapter 9, Computer Sci., Rockville, Maryland.

Pavlidis, Theo, 1980. "A Thinning Algorithm for Discrete Binary Images," Computer Graphics and Image Processing, Vol. 13, pp. 142-157.

Peuquet, Donna J., 1981. "An Examination of Techniques for Reformatting Digital Cartographic Data/Part 2: The Vector-To-Raster Process," Cartographica, Vol. 18, No. 3, pp. 21-33.

Peuquet, Donna J., 1979. "Raster Processing: An Alternative Approach to Automated Cartographic Data Handling," The American Cartographer, Vol. 6, No. 2, pp. 129-139.

Sugawara, Sandra, 1989. "The Mapping of a GIS Map," Washington Post, Business Section, September 25, pg. 35.

Watson, L. T., K. Arvind, and R. W. Ehrich, 1984. "Extraction of Lines and Regions From Grey Tone Line Drawing Images," Pattern Recognition, Vol. 17, No. 5, pp. 493-507.

Wilkinson, G. G., and P. F. Fisher, 1987. "Recent Development and Future Trends in Geo-Information Systems," The Cartographic Journal, Vol. 24, pp. 64-69.

VITA

Loretta J. Bush was born in Chandler, Arizona, on April 10, 1965. Since her father, Roy H. Bush, Jr., was in the U.S. Air Force, she moved frequently during her first eleven years. These locations included Kentucky, South Carolina, Germany, and California. When her father retired from the military, Loretta and her parents returned to Kentucky where she subsequently graduated from Henderson County Senior High School in 1983. She attended Western Kentucky University where she studied mathematics and geography. For seven months during 1987, Loretta participated in the Cooperative Education Program at Western. She worked as a coop at the U.S. Census Bureau in Suitland, Maryland, where her main duties were to update computerized census maps and write operator's manuals. In May 1988, she received a B.S. in Mathematics and a B.S. in Geography. During the following summer, she was employed by Western Kentucky University and the City-County Planning Commission to collect data for the Comprehensive Plan for the City of Bowling Green and Warren County, Kentucky. Loretta enrolled in the graduate program at Virginia Tech to continue studying geography with a greater emphasis placed on computer cartography. She completed her course work and thesis research within two years. With her thesis 75 percent complete, she accepted a job with Intergraph Corporation in Reston, Virginia, working on the MARK90 Project for the Defense Mapping Agency. In 1993, Loretta returned to Virginia Tech to defend her thesis and finally complete the Master's program in Geography.

Loretta J Bush