RASTER AND VECTOR PROCESSING FOR SCANNED LINEWORK*

David D. Greenlee
U.S. Geological Survey
EROS Data Center
Sioux Falls, South Dakota
57198

ABSTRACT

Recent advances in scanning technology have made it possible for linework to be scanned on relatively inexpensive systems and output as raster images. These images can be edited, automatically thinned to connected lines, and converted to vector representations using conventional image processing techniques.

An investigation of raster editing techniques, including thinning, skeletonizing, filling, and node detecting was performed by using software tests implemented on a microprocessor. The technique was based on encoding a three-by-three neighborhood surrounding each pixel into a single byte. This byte has a range of 0-255, and stores any possible surround condition for a pixel. It was found that various methods for thinning, filling, and node detection could be quickly implemented and tested using surrounding topology. It was possible to efficiently develop decision tables for raster editing, and to assure symmetry and consistency in the decisions.

A prototypical method was also developed for converting the edited raster linework into vectors. Once vector representations of the lines were formed, they could be formatted as a Digital Line Graph, and further refined by deletion of nonessential vertices and by smoothing with a curve-fitting technique.

INTRODUCTION

The conversion of rasterized linework to vector chains or arcs has provided researchers with a challenging set of problems for several years. Peuquet (1981) gives a technical overview of these methods. Based on a large body of available techniques, this paper describes the use and refinement of several methods for processing linework in raster and vector form. Automated aspects of the raster-to-vector process are described, but interactive editing steps are beyond the scope of this study.

This paper also attempts to bridge a gap between theoretical work on raster-to-vector processing and the many turn-key production systems that are in use but are often not well understood. During this project, techniques were developed that created gaps, spikes, spurs, snow, node slippage, and various other unsavory artifacts of automated processing. Through this experience has come a better understanding of the automated techniques that work best for a given application and some insight on the correct mix of automated versus manual editing techniques.

BACKGROUND

In support of applications project work, techniques have recently been developed by Jenson (1985) for extracting hydrologic basins and for delineating drainage networks from raster-formatted Digital Elevation Models (DEM's). It was

---

* Publication authorized by Director, U.S. Geological Survey

determined that a conversion of drainage lines to vector format might better allow for efficient overlay with other mapped data, for measuring lengths and areas, and for ordering of streams. Techniques were available for converting classed hydrologic basins into polygonal form (Nichols, 1983), but not for converting rasterized drainage lines into vector form.

A method was needed to convert rasterized linework, as extracted from the DEM, into vector chains or arcs that could be input to a vector-based geographic information system (GIS). It was observed that this problem is analogous to the problem of converting scanned map linework into vectors. By developing a generic solution to this problem, it was hoped that future scanning of map data could be facilitated by an understanding of these processing and conversion steps.

ENCODING OF TOPOLOGICAL SURROUNDS

Rasterized linework can be visualized as a bilevel or binary image having pixels represented by ones if a line is present, and represented by zeros if they are in the space between lines. In this form, many techniques are available for processing and refining the lines (Rosenfeld and Kak, 1982; Pavlidis, 1982). Most techniques utilize a set of rules for operating on each pixel based on its state and that of its eight contiguous neighbors. Because neighborhood operations are time consuming and relatively troublesome to perform, this was a subject that needed special attention.

The method described below was inspired by Golay (1969), who described 64 surround conditions for a hexagonal data structure. Since image data are usually in a regular raster format, the technique used here encodes for each pixel a single byte of eight bits that completely describe the state of adjacent neighbors. For raster-processing operations to be performed (for example, filling, thinning, or node detection), a set of decision rules is first developed and placed in a table with one entry for each possible surround condition. The raster image is then processed in the following order:

a) encode the surround condition for each pixel, and store it

b) for each pixel,
   1. find the decision rule referenced by the surround condition
   2. make changes to the pixel where indicated by the rule
   3. if changed, update the surround values for neighbor pixels

c) if any changes were made on this pass, go to b)

By encoding the surround values on the first pass and then updating them only when changes have been made, it is not necessary to encode the value on subsequent passes. This can greatly improve the speed of processing, since many of the processing functions are applied iteratively, until no changes are made in an entire pass.

Encoding can be performed by adding together the values shown on the following diagram for the neighbor pixels that have a line present.

64	128	1
32	X	2
16	8	4

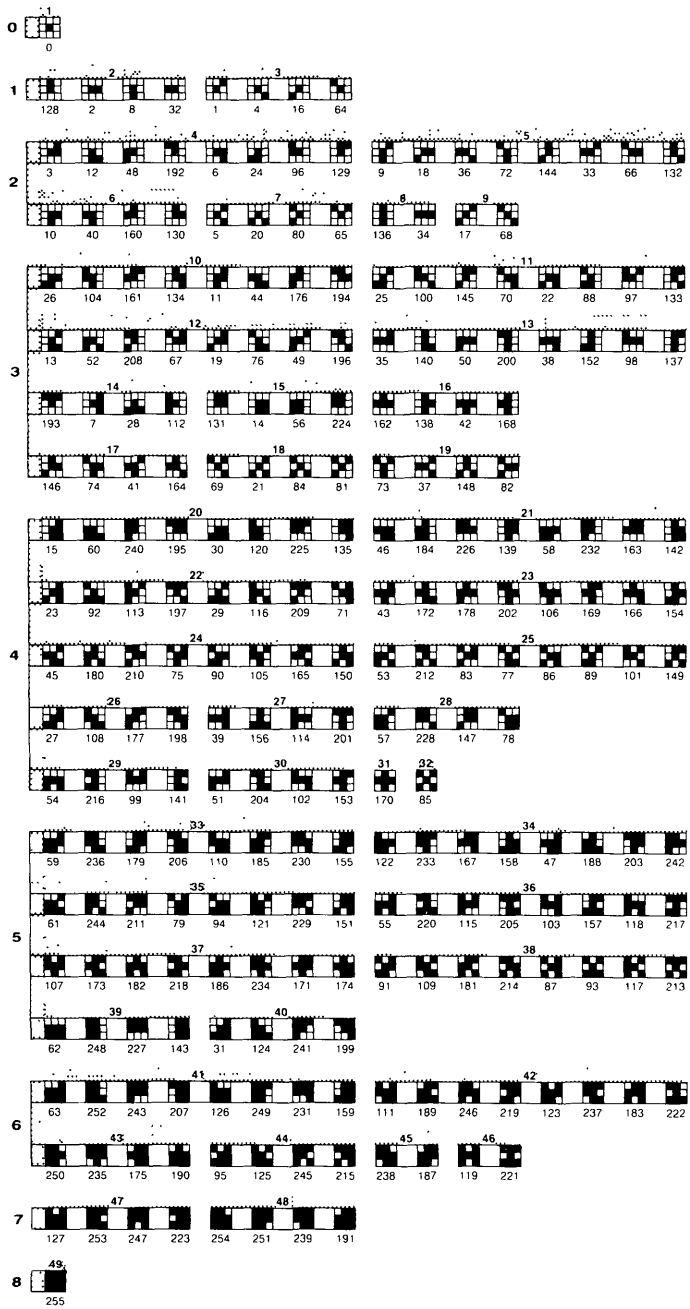

Figure 1 - The 256 possible *surround conditions* for a pixel and its eight adjacent neighbors

PAT GRP NUM	NUMBER OF 8 NGHBRS	PATS IN GRP	NUMBER OF TRANS	NUMBER OF GAPS	FILL RULE	THIN RULE STEP1	THIN LANDY STEP1	THIN RULE STEP2	THIN LANDY STEP2	NODE MARK RULE	NODE MARK SAKAI
1	0	1	0	0				Y		T	
2	1	4	1	1				Y		T	T
3	1	4	1	1				Y		T	T
4	2	8	1	1				Y		T	T
5	2	8	2	2							
6	2	4	2	1				Y	Y		
7	2	4	2	2							
8	2	2	2	2							
9	2	2	2	2							
10	3	8	2	1				Y	Y		C
11	3	8	2	2							C
12	3	8	2	2							C
13	3	8	2	2							C
14	3	4	1	1		Y	Y	Y	Y	T	
15	3	4	1	1		Y	Y	Y	Y		
16	3	4	3	1	Y			Y		C	M
17	3	4	3	2						C	M
18	3	4	3	3						C	M
19	3	4	3	3						C	M
20	4	8	1	1		Y	Y	Y	Y		
21	4	8	2	1	Y		Y	Y			C
22	4	8	2	2							C
23	4	8	3	1	Y			Y	Y	C	M
24	4	8	3	2						C	M
25	4	8	3	3						C	M
26	4	4	2	1				Y	Y		C
27	4	4	2	2							C
28	4	4	2	2						C	C
29	4	4	2	2							C
30	4	4	2	2							C
31	4	1	4	1	Y				Y	C	C
32	4	1	4	4						C	C
33	5	8	2	1	Y			Y	Y		C
34	5	8	2	1	Y			Y	Y		C
35	5	8	2	2						C	C
36	5	8	2	2							C
37	5	8	3	1	Y			Y	Y	C	M
38	5	8	3	3						C	M
39	5	4	1	1	Y	Y	Y	Y	Y		
40	5	4	1	1		Y	Y	Y	Y		
41	6	8	1	1	Y			Y	Y		
42	6	8	2	1	Y			Y	Y	C	C
43	6	4	2	1	Y				Y	C	C
44	6	4	2	2						C	C
45	6	2	2	1	Y				Y	C	C
46	6	2	2	2						C	C
47	7	4	1	1	Y						
48	7	4	1	1	Y						
49	8	1	1	1	Y						

Figure 2 - Pattern groupings of topologically similar surround conditions and their corresponding decision rules

Digitally, encoding can be performed by setting the corresponding binary bits of an eight-bit byte. In either case, the resulting value occupies the range of values 0-255, or 256 possible states. An example of a Fortran subroutine to perform this encoding would look like the following:

```
      SUBROUTINE ENCODN (NBR8,IVAL)
C
C     IVAL = output value (byte)
C     NBR8 = input array of eight    MASK = bit masks corresponding
C            neighbors ordered as:          to NBR8 as follows:
C               1  2  3                     64  128  1
C               4     5                     32       2
C               6  7  8                     16   8   4
C
      DIMENSION NBR8(8),MASK(8)
      LOGICAL*1 IVAL
      DATA MASK /64,128,1,32,2,16,8,4/
C
      IVAL = 0
      DO 100 K=1,8
           IF (NBR8(K).EQ.0) GO TO 100
           IVAL = IVAL.OR.MASK(K)
100   CONTINUE
      RETURN
      END
```

The complete set of 256 possible surround conditions is shown in Figure 1. In this figure, surround conditions have been grouped to allow decisions to be developed easily and consistantly. To accomplish this, the 256 surround conditions were combined into pattern groups by sorting on a set of attributes. Within each of the count groups (labelled 1-8 and organized vertically), are subgroups that may contain as many as eight patterns. These are referred to as pattern groups and may consist of a pattern, rotations of 90^{o}, 180^{o}, and 270^{o}, and its mirror image and rotations. Because many of the patterns are symmetrical, they generate non-unique or duplicate patterns when rotated. As a result, some subgroups will contain only four, two, or as few as one unique pattern. Grouping allows the 256 possible surround conditions to be aggregated into a more manageable 49 pattern groups. None of the operations described in this paper requires subdivision below the pattern group level.

Each of the pattern groups can be represented by a set of attributes that allows decision tables to be efficiently developed. Figure 2 is a table of pattern groups and associated attributes. One basic attribute is the number of neighbors that have a line present. Also, for thinning operations to be performed, it is useful to know whether the pattern is classed as simple, (Rosenfeld and Kak, 1982) or non-multiple (Pavlidis, 1980). For this condition to be true, a pixel must have exactly one pixel or a connected set of pixels adjacent to it. Landy and Cohen (1985) use similar information, that they refer to as the number of black to white transitions present. In addition, they determine the number of lines that result when the center pixel is deleted. This is referred to as the number of gaps, because it measures whether a gap in connectivity would be created by thinning. These characteristics are constant for each pattern group, and were used to define and order the groups shown in figure 1.

RASTER PROCESSING

The following section describes raster operations that were performed as a part of this study, as applied to two sample data sets. These examples demonstrate the

various raster and vector operations that were performed in the process of converting raster-linework to vector arcs.

Figure 3a is a Digital Elevation Model expressed as a shaded-relief display. Figure 3b shows rasterized linework that has been extracted from the DEM as described by Jenson (1985). The linework is portrayed as a binary image with lines as ones and background as zeros.

Figure 4a is a small portion of a topographic map, scanned using commercially available raster scanning equipment. The image is expressed in 256 possible shades with the lines being the lower values and the background as the higher values. In Figure 4b, a convolution filtering process has been performed to enhance the local contrast of the lines against the background. This is commonly referred to as edge enhancement or high-pass filtering, and is well described by Moik (1980). Figure 4c has been converted to a binary valued image by selecting a threshold value that separates lines from background.

Filling Operations

The purpose of a fill operation is to eliminate small voids in a line that add unnecessary complexity. The term fill is also used in computer graphics applications, where it is synonomous with flooding or painting of the interiors of polygons. While somewhat similar, this is not the process we require in processing linework. In the scanning of linework, small voids are sometimes created within the lines as artifacts of the scanning process. In some cases, we wish to eliminate valid inclusionary voids that are too small to be considered significant. In addition to filtering out small voids, filling also tends to generalize and fill in acute angular features on the exterior edges of the linework. This may sometimes be appropriate, especially because subsequent thinning is performed from the line edge inward and is greatly affected by a rough line edge. Figure 3c shows how filling can be used to eliminate small holes or void inclusions in drainage lines that have been extracted from a DEM.

Thinning Operations

Several alternatives exist for thinning or skeletonizing rasterized linework to create a single connected line. Most are iterative, and several use a two-step processing approach. One method performs the bulk of the thinning in a first step with a second step that eliminates all but the single connected lines (Landy and Cohen, 1985). The decision tables for thinning in the first step must not allow thinning to less than two pixels wide. This is to ensure that gaps are not formed by thinning on both sides of a line during the same pass. A similar approach is described by Rosenfeld and Kak (1982) that also requires two steps, and alternately thins first the top and left sides of the linework and then the bottom and right sides, in order to avoid the creation of gaps.

In general, thinning can be performed on all patterns where gaps would not be created by deletion of the center pixel (that is, where 8-way connectivity is maintained), and where the pattern is simple (that is, where only one black-white transition is found as the neighbors are tested in circular order). A special case is posed by the "T" connection (pattern group 16), that will become a "Y" connection, unless given consideration. Also, single or isolated pixels (pattern group 1), may be deleted during thinning, provided they are considered erroneous or insignificant. Endpoints (pattern groups 2-4) may or may not be preserved, as some may be valid lines and some may be spurs that are created as artifacts of thinning of thick lines (more than 4 pixels wide). Spurs may also be deleted after vector conversion, when length may be calculated and used as an editing parameter. Figure 3d shows drainage lines after filling and two-step thinning

Figure 3a - DEM displayed as shaded relief

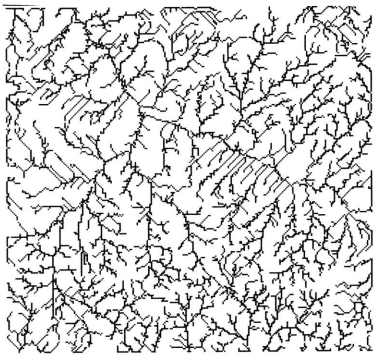

Figure 3b - Raw linework extracted from DEM

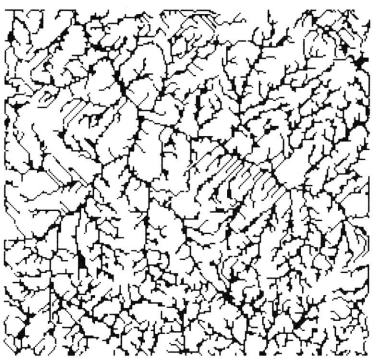

Figure 3c - Linework after *filling*

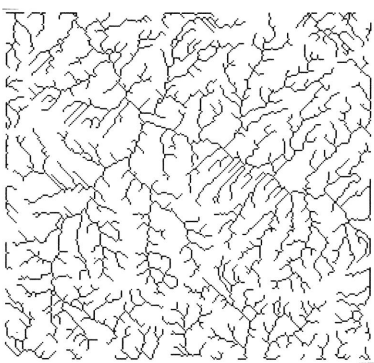

Figure 3d - Linework after two step *thinning*

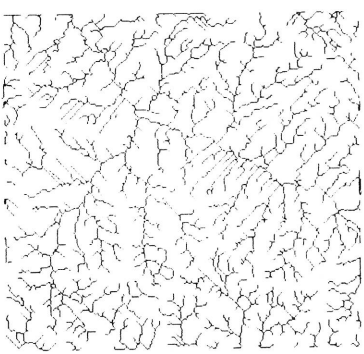

Figure 3e - Linework after conversion to vectors

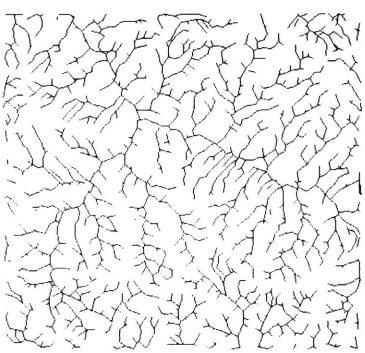

Figure 3f - Vector linework after *spline* smoothing

Figure 3 - Processing steps for linework extracted from Digital Elevation Model

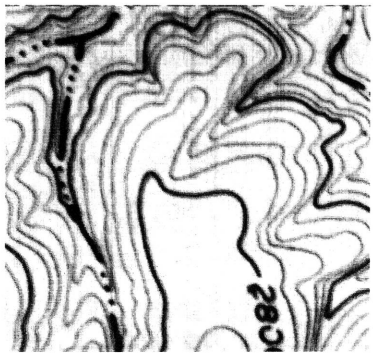

Figure 4a - Portion of scan-digitized USGS topographic map

Figure 4b - After edge enhancement by *convolution* filter

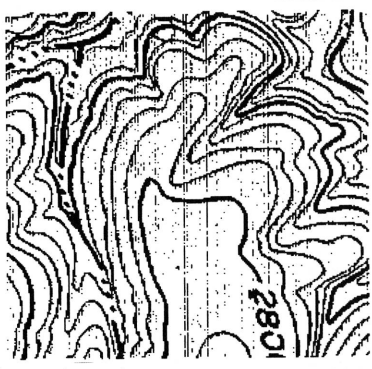

Figure 4c - Linework after *thresholding* to a bilevel (binary) image

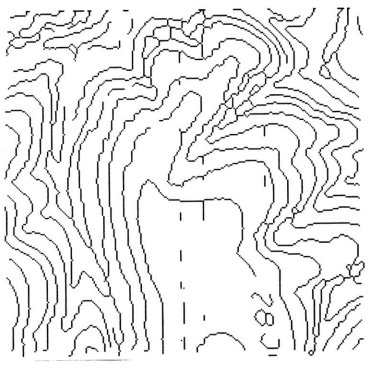

Figure 4d - Linework after *filling* and two-step *thinning*

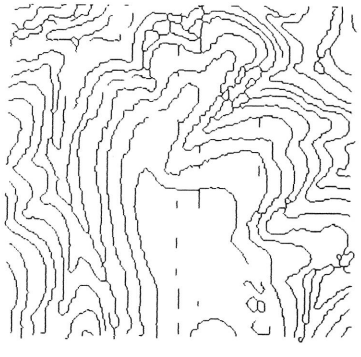

Figure 4d - Linework after conversion to vector format

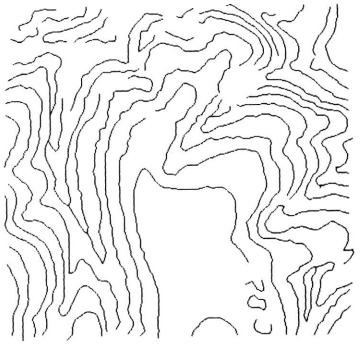

Figure 4f - Linework after *splining* and deleting short arcs

Figure 4 - Processing steps for scanned topographic linework

have been performed. Figure 4d shows scanned linework after filling and two-step thinning. In both examples, the thinning operation has reduced the original binary image to skeletal connected lines, and has eliminated many unnecessary pixels.

Node Marking

Once lines have been filled and thinned, node marking may be performed. Nodes that are endpoints can be detected as 1-count pixels, or pixels with a single occupied neighbor. Connecting nodes, or nodes where lines join, can be detected as patterns that have three or more transitions, indicating three lines eminating from the node. A special case is the "+" (pattern group 31), that is a connector, even though the gap value is one (that is, no gap would be created if the center pixel were removed). Of special concern is the problem of multiple connecting nodes, or node clusters, that make it difficult to properly place the intersection of lines. More than that, unless reduced to a single point, multiple connectors can create a web of short interconnected arcs to be created during vector generation. Rosenfeld and Kak (1982) suggest that one solution is to perform a second pass through the image, test for adjacent connectors, and calculate a single point (a centroid) that can be used as the connector.

Creation of Digital Line Graph (DLG)

With nodes marked, the process of converting linework into vector arcs can begin. The format for the resulting lines, chains, or arcs may be expressed in relative coordinates (for example, above, above and left), or converted to absolute measure (for example, cartesian coordinates). The USGS Digital Line Graph (DLG) stores absolute coordinates, usually as units of digitizer inches, or as UTM meters. For this project, the DLG was chosen because several available systems could be used to read and write data in this form. Conceptually, the procedure is simple. We begin on an endpoint or connector node and simply follow the skeletonized lines that are guaranteed to have only one path to follow until another node is found. To avoid following a line twice, the pixels are deleted as their coordinates are added to the line. This approach works well when the entire image can be resident in computer main memory, or in an image display memory, as in the case of the system used for this study.

VECTOR OPERATIONS

Point Reduction

Linework that has been converted to vectors is made up of short line segments that have a length of one pixel unit for four-way adjacent neighbors, or of 1.414 units for diagonal neighbors. Straight lines can be reduced by eliminating intermediate points on the line. In addition, lines may be further reduced by using a generalization method such as described by Douglas and Peucker (1973). For the examples used in this study, point reductions of 50-80 percent were possible with little perceivable difference in the quality of the lines.

Line Smoothing

Point reduction does not reduce the appearance of sharp jagged arcs that show remnants of their raster derivation. In fact, point reduction, when used exclusively, will remove blunt angles and leave sharper angles in the lines. Sharp angles that exist at points of inflection can be treated by fitting a curve through

the set of points. This will introduce new points but will also yield arcs with a more acceptable appearance. One such technique is cubic spline interpolation and is described by Monmonier (1982). In figure 3f, spline interpolation has been performed on the drainage lines with the result showing a smoother appearance. In figure 4f, spline interpolation has been performed, and in addition, short arcs have been deleted. In this case, short arcs were largely made up of erroneous spurs (generated in thinning) and bridges (between lines).

SUMMARY AND CONCLUSIONS

This study was performed in order to convert rasterized lines to vector arcs in a format appropriate for geographic information system processing. This was accomplished by assembling several image-processing and vector-editing techniques into a prototype system. The documentation of surround conditions may have broader applicability than that required for this project, and it is felt that the use of pattern groups can provide an efficient and consistent framework for many neighborhood operations that are performed in the raster domain. The experiences gained in this project have helped to objectively characterize the steps required in raster-to-vector processing. The understanding of these processes will become more valuable as scanning systems become more affordable and turn-key systems that are tailored to specific requirements become available.

REFERENCES

Douglas, D.H. and Peucker,T.K. (1973), Algorithms for the reduction of the number of points required to represent a digitized line or its caricature: Canadian Cartographer, v.10 , p.112-122.

Golay, M.J.E. (1969), Hexagonal parallel pattern transformations: IEEE Transactions on Computers, v. C-18, no. 8.

Jenson, S.K. (1985), Automated derivation of hydrologic basin characteristics from digital elevation model data: Proceedings of Auto-Carto VII, 1985, Washington, DC.

Landy, M.S., and Cohen, Y. (1985), Vectorgraph coding: efficient coding of line drawings: Computer Graphics and Image Processing, v. 30, p.331-344.

Moik, J.G. (1980), Digital processing of remotely sensed images: NASA SP-431, p.130-140.

Monmonier, M.S. (1982), Computer assisted cartography: Prentice Hall, p. 108-110.

Nichols, D.A. (1981), Conversion of raster coded images to polygonal data structures: Proceedings of Pecora VII Symposium, 1981, Sioux Falls, SD.

Pavlidis, T. (1980), A thinning algorithm for discrete binary images: Computer Graphics and Image Processing, Vol 18, p.142-157.

Pavlidis, T. (1982), Algorithms for graphics and image processing: Computer Science Press, Chapters 7-9, p.128-214.

Peuquet, D.J. (1981), An examination of techniques for reformatting digital cartographic data, part 1: the raster-to-vector process: Cartographica 18: vol.1, p.34-48.

Rosenfeld, A., and Kak, A.C. (1982), Digital picture processing [Second Ed.]: Academic Press, v. 2, p.232-240.