Tạp chí Tin học và Điều khiển học T 12, S 2 (1996) (21 - 29)

AN EXAMINATION OF TECHNIQUES FOR RASTER-TO-VECTOR PROCESS AND IMPLEMENTATION OF SOFTWARE PACKAGE FOR AUTOMATIC MAP DATA ENTRY-MAPSCAN

BACH HUNG KHANG, NGO QUOC TAO, PHAM NGOC KHOI, LUONG CHI MAI,
DO NANG TOAN , NGUYEN DUC DUNG, VU VAN THINH

The majority of commonly used manipulative techniques in computer-assisted cartography continue to require that the data be in vector format. This situation has recently precipitated the requirement for fast techniques for converting digital cartographic data from raster to vector format for processing. This article concerns with examining the states in theses conversion techniques. In part one, algorithms to perform all phases of the raster-to-vector process are systematically outlined, and then compared in the general term. Part two will describe the package for automatic map data entry MapScan, the algorithms implemented in which are based on the fast techniques for converting map raster data to vector format. With MapScan users are able to move printed maps or drawing into a mapping system much more quickly and easily compared to using traditional digitizer techniques.

I. TECHNIQUES FOR RASRER-TO-VECTOR PROCESS

Almost all data manipulation in computer cartography today requires that the data be in vector format prior to theire manipulation. This format is necessary because the existing manipulation algorithms are heavily dependent upon the vector data structure. The dependence on vector format is not an optimal situation, but it is one that has been heavily mandated by both cartographic tradition and human intuition. The major difficulty encoutered at present is that the algorithms for performing the various components of raster-to-vector conversion have never been systematically analyzed as parts of an intergrated process. Although individual algorithms are well known, the variouse combinations of algorithms for converting raw, scanned map data into vector form have never been described nor have the interrelationships of the various factors affecting execution times been adequately quantified.

1.1. Algorithms for raster-to-vector conversion

The task of converting raster-formatted map or other line data into vector format can be devided into three basic operations. First is **Skeletonization** or line thinning, the process of reducing lines to unit thickness at a given resolution. The second operation is **Line Extraction** or vectorization - the process of identifying a particular series of data entities or coordinates that constitute an individual line segment as portrayed on the input document. The third operation is **Topology Reconstruction** - the process of

determining the adjacency relationships among all the lines. The individual line segments are joined into whole lines if disired, and maps may be joined together into a continuous areal representation.

Two other operations ancillary to the basic raster-to-vector process are line smoothing and spike and gap removal. These operations are frequently requried to eliminate inaccuracies either present in the input map data or induced by the specific algorithms employed in skeletonization or line extraction.

1.1.1. Skeletonization

There are curently three basic approaches for the skeletonization of arbitrarily shaped lines. The first approache involves "peeling" the sides of a line in an interactive process. The second approach involves expanding the spaces between the lines, as if one where blowing up a bunch of ballons inside a boxe. The third involves canculating the center, or medial axis, of each line directly.

The Peeling Approach. The most commoly used algorithm for skeletonizing line data in raster of matrix form is a "peeling" process whereby each thick line is reduced by one resolution unit at a time on each side of the line until only one unit of line width demain. For the purpose of discution, it is assumed that all data are represented as a binary matrix of a 0 and 1. An 0 represents background and a 1 represents a location occupied by a portion of a line. In the general term, this process examines the surounding pixels of each non-zero pixel or location cell to determine whether or not the location of interest is on the edge of a line. The central pixel and its 8-adjacent cells are examined (see Fig.1). The central pixel must satisfy all of the following criteria or it is removed (i.e., changed to 0) as part of the line.

1. it is not the only occupied cell in the matrix
2. it does not connect neighboring cells
3. the removal of that cell could not alter the continuity of the line (i.e., not a line junction).

6	7	8
5	0	1
4	3	2

Fig. 1. An 8-adjacent matrix

This logic is best implemented as a lookup table containing a "retain" or "delete" answer for each of the 256 possible 8-neighbor configuration of a 3 x 3 matrix.

There are several advantages to this algorithm. First, it is a parallel process, and such , can be implemented utilizing a scan-line approach, i.e. pixels are handled on a raster-by-raster instead of a line-by-line basis. Second, line junction are deleted as an intergral part of this process and can also be indentified by type with only a minor extension of the algorithm. Third the total line length, or line density, has a minimal effect on execution time.

Several disavantages are also associated with this algorithm. First, since it "peels" the line two pixels at a time, the thickness of the original heavily impacts the speed of the algorithm. The number of times each portion of a line needs to be thinned (and the subsequent number of passes through the data) is equal to one-half the width of the thickness line, measured in resolution units. Execution speed is thus a linear function of line thickness. If the width of a line varies by more than one pixel on either side of the original line, the line needs to be smoothned beforhand to avoid local waves, or "kinks", in the thinned line. Third disavantages all result from an inherent characteristic - the algorithm is highly sensitive to overall line quality on a scanned map.

The problem identified above are avoided if the initial data consist of crisp lines with sharp junction that are no more than four pixels wide in either the horizontal or the vertical direction (i.e., the lines are thin to begin with) In this case, the data can be thinned in two passes.

The Ballooning Approach. A variation of the peeling approach to thinning line data is known as the "ballooning" algorithm. In this approach the area between the lines are expanded until the line separating them can not be reduced futher without causing a break in the boundary. The open area between thick lines in the image (are expanded interatively by following the inside edge of each open area, always in the same direction, using the 4-adjacent test, see Fig.2. This test uses the three criteria as the peeling approach but does not take diagonal neighborss into account (i.e., 4 neighbors instead of 8). As the inside edge is folloed, each "expanded" pixel is deleted from the line (i.e., changed to 0). A separeted pass is then needed to locate nodes in the thinned line network for subsequent topology reconstruction by using 8-neighbor (i.e., full 3 x 3 cell) matrix to chek for interserctions.

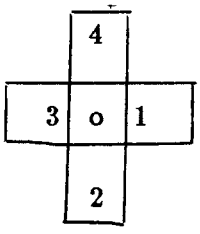


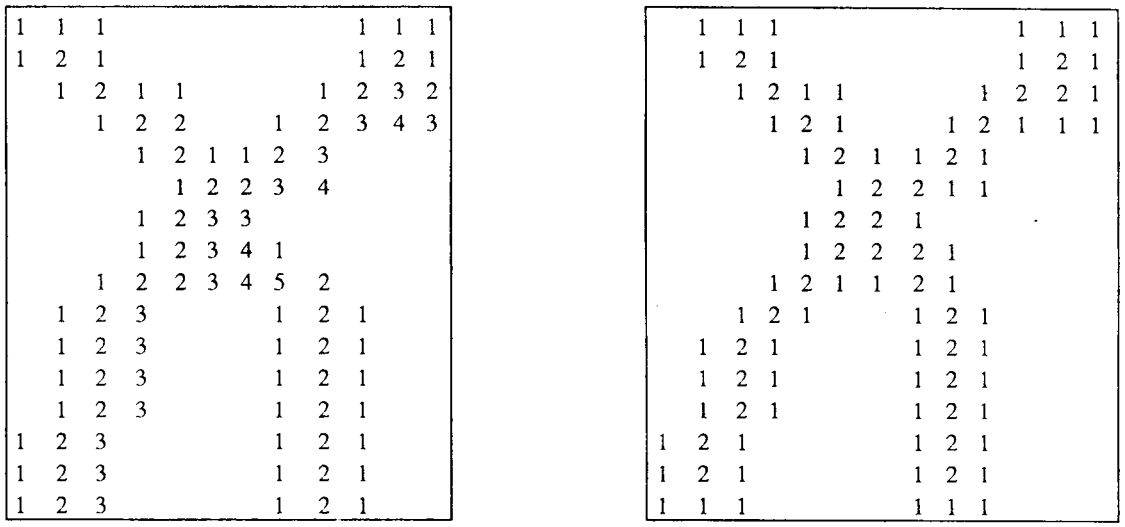
Fig.2. A 4-adjacent matrix

The ballooning approach is the dual or logical "minor image" of the peeling approach. The ballooning algorithm deals with the spaces between the lines instead of the lines themsels. The primary disavantage of the ballooning approach is that it can dislocate node position because the sequential processing of open areas can only "see" one side of a line at a time. This particular approach gains speed by utilizing the faster but less accurate 4-neighbor matrix the bulk of the processing and then utilizing the 8-neighbor matrix for more accurate node extraction. This technique can also be used in the peeling algorithm to gain speed.

The Medial Axis Approach. Each pixel in the original line (i.e., 1) is given a value equal to the number of pixels from its nearest non-line (i.e., background) pixel. All 1 are assigned a distance value in only two passes. The basic algorithm is as follows: first, all pixel are processed in foward raster order (left to right), from the top left corner, processing each complete raster in turn. Each occupied pixel is assigned a value equal to the minimum value of its neighbors directly above and immediately to the left, plus 1. In the second pass, the resulting matrix is processed in the reverse order, assigning to each pixel the minimum value among itself, the neighbor to the right plus 1, and the neighbor below plus 1. Fig.3a and Fig.3b show the results of passes 1 and 2 respectively. The skeletion of the resulting matrix consists of all locally maximum-valued ceels after the second pass. The binary skeletion can be derived by reversing the above logic for an additional two-step process, making the entire skeletonization algorithm a four-pass procedure regardless of the original line thickness. Thes additional two passes can be avoided if the maximum value matrix can be used directly by the line extraction algorithm.

The medial axis algorithm to be the fastest among all the above skeletonization approaches. It is sequential, scan-line oriented process that never requires more than four passes through the data. Overall efficiency is primarily affected by the number of pixels that must be "looked at" (i.e., the number of occupied cells, and thus the overall original thickness of the lines). Speed is not as severely affected by line thickness as in the peeling approach. The primary drawback of the approach is that errors can results from very thick nodes. However, this problem seems to be universal with existing skeletonization algorithms. Another drawback is that this algorithm is sequential in nature. Thus, the strict order in which pixels are processed prohibits streamlining techniques employed in the implementation of parallel algorithm.

1.1.2. Line Extraction



(a) (b)

Fig.3. a. Distance values of pixels after first pass
b. Distance values of pixels after second pass

Currently, two approaches exist for line extraction. The first approach is line following or line tracking, in which each individual line is followed from pixel to pixel, in any direction, until the end of the line is reached. Each line is reconstructed sequentially and ereased in the input file as it is followed so that it is traced no more than once. The second approach involves simultaneous tracking of all lines within each scan line.

The Line-Following Approach. The primary advantage of the line-following approach is that it is conceptually simple. It also takes care of the topology automatically. However, many disavantages are associated with this approach. First, when a node or junction in a line network is encountered, a decision must be made about which branch is to be followed and a marker kept to trace the remaining branches. This complication can be avoided by marking a preliminary pass through the data to detect and remove all junction points, leaving only separated line segments. Second, the execution time is a linear function of the total line length to vectorized. This approach produces a large amount of temporal

overhead when implemented on a computer if the lines tend to wander over a larger number of scan lines, because repeated input and output of the same scan lines is frequently requied. This overhead quickly becomes overwhelming as the size of the map area or the density of the map line increase.

The Scan Line Approach. The scan-line approach to line extraction uses the same basic logic as the line-following approach, but is implemented in scan-line order. Here, all lines intersecting a given scan line are processed simutaneously and each scan line reads only once. The entier map is processed from top to bottom (or bottom to top) in an "eating-down-the sheet" fashion. Individual convoluted lines are detected as multiple line segments that eventually must be connected together, as shown in Fig.4. These segments can either be joined at the time the line junction are detected, or recorded so that all line segments can be reconstructed as a separate step after all scan-lines have been processed.

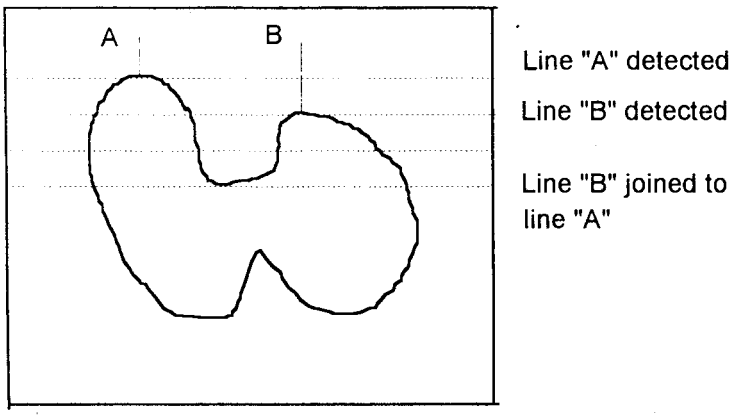


Fig.4. Line extraction via the scan-line algorithm

The price paid for this scan-line approach, however, is a greatly increased level of bookkeeping activity required to keep track of many line segments simutaneously. The volume of bookkeeping overhead neccessary is directly related to the same factors affecting 1/0 in the line-following algorithm - overall line-density and the range of scan lines covered

by individual map lines.

1.1.3. Topology Reconstruction

Topology reconstruction is normally performed partially as a byproduct of either line thininng or line extraction. Since line junctions must be recognized and treated as special situations, particularly in the line extraction process, the common practice is to record the type and location of all line junctions in a separate table when they are first encountered. This information is later used to reconstruct the topology without needing to make an additional pass through the map data itself.

In the peeling and ballooning algorithms for skeletonization, junctions can be located when a lines is reduced to unit thickness. Junctions can then be recognized as "T", "X", "+", or "Y" type intersections and recorded in a separate table for use during the topology reconstruction phase. Junctions can similarly be recognized during line extraction, and junction identification in fact an integral part of any line extraction process.

II. IMPLEMENTATION OF SOFTWARE PACKAGE FOR AUTOMATIC MAP DATA ENTRY - MAPSCAN

MapScan is a software package that accepts various formats of scanned maps or drawing, reads and converts scanned images into vector maps with text references of different formats that can be used by popular mapping systems. With MapScan users are able to move printed maps or drawing into a mapping system much more quickly and easily compared to using traditional digitizer techniques. Information flow of MapScan is shown in Fig.5.

MapScan fuatures

• Raster Image Editing. Some linework editing is much more efficiently accomplished at this stage than after automatic vectorization. Problems more efficiently addressed at this stage include removal of unwanted linework, eliminate unnecessary items, connect broken lines, rotate an image, merge multiple pages to form the entier map image, ...

• OCR (Optical Chracter Recognition). Text in the raster images is specified as polygons to allow separation from line elements. It is possible also to identify batches of text references and perform OCR processing, generate text references files with appropriate format for other specific mapping and GIS software.

• Vectorization. MapScan performs raster-to-vector conversion, used algorithms are based on the fast techniques for converting map raster data to vector format. The process has two steps:

(1) MapScan reads the raster image and performs a thinning process to reduce the width of all lines to only one pixel. This process of skeletonization used the peeling aproach by *contour tracing algorithm.*

(2) MapScan reads and vectorizes the thinned lines by producing and connecting all the nodes and generate a map coordinate file in DXF or appropriate format for other specific mapping and GIS software. Algorithms used here belong to the first approach of line extraction, i.e. the approach of *line following or line tracking.*

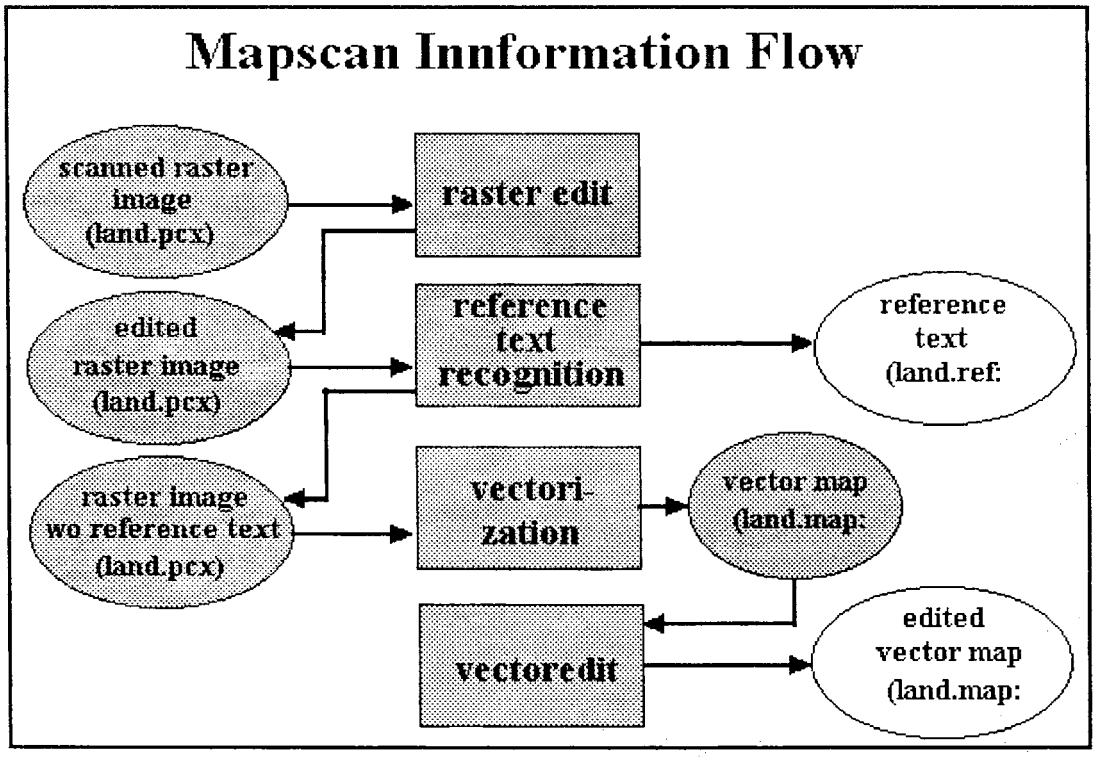


Fig.5. Information Flow of MapScan

• Vectorized Image Editing. To complete the raster-to-vector translation, some post-)st-vectorization editing will be required. Specifically addressed during this step is the problem level of the file which contains elements the system was unable to interpret. Checks and corrections will be required at some intersections, and proper line attribute translations will also be verified during this step.

The MapScan versions supports input of PCX, TIF and IMG formats and output of DXF and PopMap (PopMap is a United Nations intergrated software package for geographical information, maps and geographics databese) formats.

REFERENCES

1. A. Rosenfeld and A.C. Kak, Digital Picture Processing, Academic Press, New York, 1976.
2. T.Pavlidis, *A thinning algorithm for discrete binary images,* Comput. Graphics and Image Processing 13, 1980, 142-157.
3. A.Rosenfeld, *A characterization of parallel thinning algorithms,* Iform. Contr.29, 1975, 286-291.
4. Donna J Peuquet, *An examination of techniques for reformating digital cartographic data,* Cartographica, Vol 18, No.1, 1981, 34-48.
5. T.Pavlidis, *Filling algorithms for raster graphics,* Comput. Graphics Image Proc. 10, 1979, 126-141.
6. U. Montanari, *Continous skeletons from digitized images,* J.ACM 16 1969, 534-549.