Vision-Based Image Processing of Digitized Cadastral Maps

Liang-Hwei Lee and Tsu-Tse Su

Abstract

This paper presents an automatic method for processing digitized images of cadastral maps. The method includes two major algorithms: a segmentation and a Raster-to-Vector conversion. Those algorithms use a simple data-list structure for recording data created during single-pass, row-majority scanning and line tracing. The segmentation algorithm obtains the positions and sizes of symbols and characters, in addition to completing map segmentation and proving useful for pattern recognition. The Raster-to-Vector conversion algorithm obtains topological information necessary to relate cadastral map spatial data to line start points, midpoints, intersection points, and termination points. It consists of four integrated sub-algorithms that remove noise, unify run-length coordinates, and perform synchronous line approximations and logical linkage of line breaks. Straight, angled, and curved lines can then be completely reconstructed for display. Also presented are six indices that verify algorithm and experimental results.

Introduction

Many conventional maps are not digitized, making them difficult to store and maintain, and they may also come from different sources and may be drawn to different scales, making them difficult to accurately measure and exchange. This limits the amount of qualitative data that maps can express, and complicates translation to the more useful digital data needed for complex spatial analysis. As society changes, non-digitized maps cannot satisfy versatile and modularized requirements for access to spatial data. Therefore, the need for digital maps is increasing, and they are replacing conventional non-digitized maps(Peuquet, 1981; Musavi *et al.*, 1988; Nagasamy and Langrana, 1990).

Digital maps may be obtained in two ways: by normal survey methods, or by conversion from non-digitized maps. Conversion generally consists of digitization and vectorization, processes that constitute major bottlenecks in the production of digital maps.

Depending on the level of feature extraction obtained, digitization and vectorization can be classified into the following types:

- coding and compression: which thins and vectorizes bi-level images directly and concentrates on coding and compression of data needed for storage or communication (Ramachandran, 1980; Landy *et al.*, 1985);
- features extraction: which in addition to coding and compressing data, concentrates on extracting some features — mainly line segments — but not enough to provide detailed recognition of symbols or characters (it considers them noise) (Musavi *et al.*, 1988; Nagasamy and Langrana, 1990; Kaneko, 1992); and
- segmentation and recognition: which in addition to coding and compressing data, labels all patterns such as symbols, characters, and lines as topological features, recognizing that they are not noise but features that must be extracted by precise location; this approach is capable of providing enough information for detailed symbol and character recognition (Wu, 1990; Yamada *et al.*, 1991; Yamada *et al.*, 1993).

Depending on whether or not thinning is performed, digitization and vectorization can be classified into direct tracking and vectorization, such as the track-follow approach (Yen, 1989) which is effective for curves but is only fit for contour lines, and the run-length approach (Ramachandran, 1980) which, on the other hand, retains line widths and needs only a single scan, leading to reduced storage requirements and better performance. The main problem with this approach is production of zigzag features and variable line widths, leading to difficulty in finding exact coordinates of intersection points.

The other main digitization and vectorization approach (Musavi *et al.*, 1988; Wu, 1990), thinning-based vectorization, produces one-pixel line widths, making it easier to track the direction geometric relationships such as line start points, midpoints, termination points, and intersection points (Peuquet, 1981; Clarke, 1990).

To focus on processing images of cadastral maps, this paper presents an automatic process suitable for computer vision to recognize patterns on map. The process includes two major algorithms. The first, segmentation, records the positions and sizes of symbols and characters in a data-list structure as segmentation is done. It has provisions for eliminating noise and for clipping sub-images from source images to facilitate more detailed recognition of symbols and characters.

The differences between this algorithm and similar approaches are that we update data-list variables instead of classifying with complex statistics, and we use a single-pass scan instead of extra labeling with a two-pass scan. The second algorithm, raster-to-vector conversion, is similar in concept to the segmentation algorithm, but the recorded data are changed into cadastral map spatial data that express topological relationships such as line start points, midpoints, intersection points, and termination points.

Four thinning-based algorithms that remove noise, unify run-length coordinates, perform synchronous line approximations, and perform logical linkage of line-breaks are embedded in the raster-to-vector conversion algorithm. Straight, angled, and curved lines can then be completely re-

Department of Surveying and Mapping Engineering, Chung Cheng Institute of Technology, Tashi, Tao-Yuan, 33509, Taiwan, Republic of China.

Photogrammetric Engineering & Remote Sensing,
Vol. 62, No. 5, May 1996, pp. 533–538.

0099-1112/96/6205–533$3.00/0

```
012345678901234567890123456789   RUN1    RUN2    RUN3
1               *               (16,16)
2             *****             (14,18)
3           **********          (12,21)
4         **************        (10,23)
5        ***************        ( 9,23)
6       ****** ***  *******     ( 8,13) (15,17) (20,26)
7       *****   **   *********  ( 8,12) (16,17) (21,29)
8      *****    **    ********  ( 7,11) (16,17) (22,29)
9     ******    **     ******   ( 6,11) (16,17) (23,28)
10    *****    ***     *****    ( 6,10) (15,17) (23,27)
11    *****     **      *****   ( 6,10) (16,17) (24,28)
12    ***** *********   ******  ( 6,10) (12,20) (24,29)
13    ************************  ( 6,29)
14    **********************    ( 6,27)
15    *****     ***     ****    ( 6,10) (16,18) (24,27)
16    *****     ***     ****    ( 6,10) (16,18) (24,27)
17     *****    ***    *****    ( 7,11) (16,18) (23,27)
18     ******   **    *****     ( 7,12) (16,17) (22,26)
19      ******  ***   ****      ( 8,13) (16,18) (22,25)
20       *****************      ( 9,25)
21        ***************       (10,24)
22          **********          (12,21)
23           ********           (13,20)
24              **              (16,17)
```

Figure 1. ⊕ (center peg) symbol and run-length coding.

constructed for display. The difference between these algorithms and similar approaches are

- Noise Removal. Only line-tracing and detection of line start points or termination points in the non-boundary regions of an image are necessary. Our method does not require point-to-point pixel searching or complex image calculation through whole image.
- Run-Length Coordinates Unification. This is an effective approach to resolving exact pixel coordinates of run-length segments, a difficult problem for most researchers (Ramachandran, 1980; Wu, 1990).
- Synchronous Line Approximation. Our approach does not require any complex fitting functions and does not need to acquire points before fitting.
- Logical Linkage of Line Breaks. Our method records where the two points of a line break are, and doesn't actually perform linkage until any reconstruction for display has been done.

All algorithms proposed in this paper use a simple data-list structure to store the dynamic data produced during single-pass, row-majority scanning and line-tracing. These techniques have the advantage of being able to avoid sortings and can simultaneously handle all lines in a single scan. Therefore, performance speed-up and memory space-down can be expected, making them appropriate for use on small computers (Lumia *et al.*, 1983; Ronse and Devijver, 1984; Haralick and Shapiro, 1992). In addition, there are four indices — coding, parameter-setting, noise immunity, and effectiveness — to assess all presented algorithms. In experimental testing, our approach performed well.

Segmentation

Preprocessing must be performed in segmentation in order to enhance gray-scale images into clear images and to divide a gray-scale image into a two-level scale image. Because the source-image histogram is zigzag, leading to uncertainty about peaks and valleys and making it is difficult to decide how to separate objects from background, enhancement must be done before thresholding. The reason is that most objects in cadastral maps are line drawings, and, after enhancement, the differences between the background and objects can be ascertained more easily, because the gray-scale histogram becomes smoother, and peaks and valleys are seen more clearly. Finding a better thresholding gate is then also easier.

Pointer	Leftmost X of previous row segment	Top-leftmost X coordinate of covered rectangle	Top-leftmost Y coordinate of covered rectangle
	Rightmost X of previous row segment	Top-rightmost X coordinate of covered rectangle	Top-rightmost Y coordinate of covered rectangle

Figure 2. Data structure of connected component.

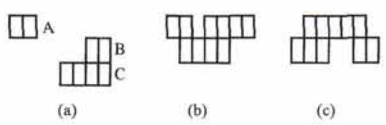

Figure 3. Line-tracing possibilities.

Before segmentation, a rectangle size (height and width) for marking symbols and characters must be determined. The size depends on the largest and smallest symbols and characters used in the cadastral map and must be carefully chosen to avoid covering other objects or covering part of an object.

Data Structure

A cadastral map contains many connected components that are composed of segments. Each segment is an elementary unit that can be described by run-length coding (Figure 1).

The position of a symbol or character is marked when any connected component is tracked during line tracing. The marked information is stored in a data-list structure (Figure 2) which records the smallest covered rectangle.

Run-length coordinates from the previous row segment must also be kept in this data structure and compared with the current row segment to analyze situations that occur during line tracing.

Algorithm

Line tracing involves three elements: (1) line growth start, (2) line growth termination, and (3) line growth continuation. When (1) occurs, a new list element must be inserted into the list. When (2) occurs, a list element must be deleted from the list and a decision must be made regarding differentiation of what attribute (line or character or symbol) the connected component can be. The decision depends on the criterion described below: Let the connected component size be T, the upper bound of symbol and character size be T1, and the lower bound of symbol and character size be T2.

Then,

$$T < T1 \text{ noise; } T1 <= T <= T2 \text{ symbol or character; } T > T2 \text{ line.}$$

When (3) occurs, a list element must be updated and a new value used to completely cover connected components.

In Figure 3(a), Line A begins growing as the first row is scanned, then stops growing. Line B begins growing when the second row is scanned, and keeps growing as the third row is scanned. Line C stops growing when the fourth row is scanned, and so on.

In Figure 3(b), two segments in the first row lie atop segments in the second row, forming a merge case that must be unified into one line, and the corresponding element must be deleted from the data-list structure.

In Figure 3(c), one segment in the first row splits into two segments in the second row, creating a new list element for the right-hand line, and updating the the left-hand line.

Figure 4 is a partial data-list handling result that uses the symbol in Figure 1 as an example to provide a more detailed description of the segmentation algorithm. In Figure 1, row 5 to row 6 and row 14 to row 15 show two splits where the left-hand line keeps growing and the right-hand line is

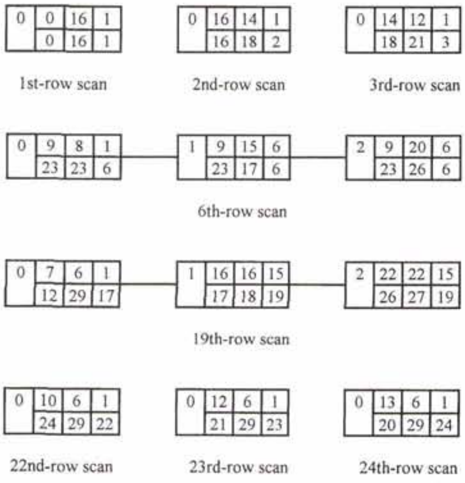

Figure 4. Partial data-list handling result.

viewed as a newly created line. In Figure 1, row 12 to row 13 and row 19 to row 20 show two merges where two lines are unified into one line. After progressive processing (i.e., insertion, deletion, and updating of list strings), the final symbol coordinate output is in the top-leftmost corner (6,1) and bottom-rightmost corner (29,24), and can be used to clip off a sub-image 24 pixels wide and 24 pixels in high.

Noise-Removal

When extracting an object from an image using computer vision techniques, that object must exhibit the following characteristics (Haralick and Shapiro, 1992):

- Neighborhood Spatial Coherence. Extracted objects are inclined to cluster in the spatial domain.
- Neighborhood Pixel Intensity Homogeneity. The gray difference between extracted objects must be small or none.

If any object violates these characteristics, it is viewed as noise and must be removed.

Approaches to noise removal can be classified into two types: (1) detecting lack of coherence and replacing the incoherent pixel, and (2) averaging or smoothing the pixel along with others in its immediate neighborhood. Generally, Type (1) locates a noise by determining a threshold before the noise is removed. Type (2) doesn't care where the noise is but needs to reduce the gray difference between pixels.

In segmentation, the Type (1) approach is used to remove noise because all non-background objects besides symbols, characters, and lines are noise. After thinning, the Type (1) approach is also used to remove the noise because some vectorization can be affected by any remaining noise. To summarize, the noise-removal approaches in this paper are based on knowing what the noise is, where the noise is, when the noise is removed, and how the noise is removed.

Raster-to-Vector Conversion

Two preprocessing topics are first presented in this section. The first concerns line-thinning of cadastral map symbols, characters, and noise clipped during segmentation. Thinning keeps topological relationships and geometric attributes from the source image, and lets the thinned objects preserve a connection skeleton, one pixel wide and robust to rotation. After thinning, some noise may be left so that recorded points include unnecessary data.

The second topic is our noise-removal approach, which detects whether the central pixel in the 8-neighborhood of a 3 by 3 window is a start point or a termination point. If a start point or a termination point is detected, then it detects whether there is another point in a larger search window. In Figure 5(a), marked squares are start or termination pixels on the image boundaries and not noise locations, so they are ignored. In Figure 5(b), however, the marked squares are not start or termination points on image boundaries; therefore these points must be noise. Additionally, the central pixel in the 8-neighborhood information about line direction can be used to remove noise up to the next intersection. Our noise removal approach does not consider every possible 3 by 3 window in a cadastral map, but, rather, is embedded in line-tracing and operates in synchronization with it to improve performance.

Noise removal can also be handled synchronously during line tracing, but some noise might have been recorded in the location that is an intersected pixel between noise and a correct line; thus, that increments the vectorized data total. Noise removal is done during preprocessing to avoid this potential problem.

Data Structure

Two data-list structures are used in raster-to-vector conversion. The first (Figure 6: the *X* direction is in the scan-line direction; the *Y* direction is perpendicular to the *X* direction) is used during line-tracing and synchronous line-approximation. The second (Figure 7) is used to record the vectorized point coordinates along each line. Pixel classification then decides which attribute (start, midpoint, termination, etc.) the vectorized point is, but it can't acquire information about how many points will be needed for approximation, and vectorized points along each line must also be recorded in serial order. We use an array to handle these tasks, an approach with some disadvantages: it is difficult to insert or delete dynamic data created during line-tracing, the largest space must be reserved for lines but most lines need only a little space, and sorting must be executed for vectorized points along each line. If a list is used, it requires only enough space to store the dynamic data created during line-tracing and needn't sort any vectorized points along each line.

Algorithm

Raster-to-vector conversion is similar to the algorithm in segmentation that uses tracing to track line-growth start, line-growth continuation, and line-growth termination. In addition, pixels are classified as start points, midpoints, intersection points, and termination points. For the topological reconstruction and display of these points in vectorized form, each line must be recorded in serial linkage and assigned a line number.

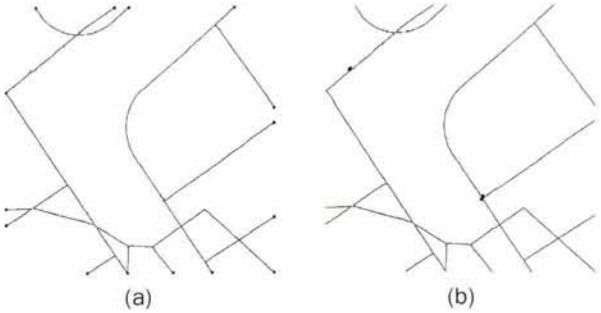

Figure 5. (a) Boundary vectorized pixels. (b) Non-boundary noise pixels.

Pointer	Leftmost X coordinate of previous row segment	Start point X coordinate of approximation	Start point Y coordinate of approximation
		Largest X coordinate of positive direction	Largest Y coordinate of positive direction
Line number	Rightmost X coordinate of previous row segment	Largest X coordinate of negative direction	Largest Y coordinate of negative direction
		Termination point X coordinate of approximation	Termination point Y coordinate of approximation

Figure 6. Data structure of line tracing and approximation.

The greatest difficulty in run-length coding during row-majority, scanning, and line tracing is a segment that is parallel to the *X*-axis. The segment may have many *X*-coordinates, so it is difficult to decide where the exact pixel coordinate on the segment is. The longer a segment is, the greater the shift can be. In Ramachandran's (1980) run-length approach, the line width is variable and the intersection point coordinates are not exact. Wu's (1990) approach takes half the length of the segment as the *X*-coordinate location, which means the longer the segment the longer the error.

We propose an approach that tracks the relationship between the previous row and the current row of a thin image in order to analyze where the exact pixel coordinate is. After thinning, one-pixel line width is reserved and two criteria must be considered: i.e., whether the previous row segment is larger or smaller than the current one. In Figure 8, the X-coordinate may occur in any non-overlap line growth (A or B for example) or any overlap line growth between two rows (C or D for example) because only a one-pixel connection between two rows can occur in the 8-neighborhood line-tracing.

The unification of X coordinates consists of calculating the exact pixel positions and deciding on line direction by analyzing two overlapping row segments. Care must be taken, however, to be sure that the termination coordinates of the longer segment are also recorded in order to avoid ignoring this segment in topological reconstruction.

Synchronous Line Approximation

Synchronous line approximation consists of selecting critical curve points to approximate the curve by linking segments point-by-point between each pair of points. The principle of critical-point selection is choosing the points as representative as possible and maintaining the shape of a curve with the smallest possible number of critical points. Related research can be classified into two main types: (1) angle or corner detection schemes (Rosenfeld and Weszka, 1975; Freeman *et al.*, 1977; Teh and Chin, 1989), and (2) piecewise linear polygonal approximation. Synchronous line approximation belongs to the second type and is embedded in line-tracing in order to avoid having to wait until all points have been acquired (Ramer, 1972; Freeman *et al.*, 1977; Williams, 1978; Pavlidis, 1982; Wall and Danielsson, 1984; Roberge, 1985; Wu, 1990).

In Figure 9(a), we use the line segment SC to approximate the curve SC. The pixels of curve SC can appear on both sides of line segment SC, so the largest distance from the curve to the line segment SC is P and the largest on the other side of the line segment is N. First, we track to the next pixel T and use a distance threshold to decide whether P and N must be updated. We continue tracking to the next pixel, or record the critical point and select a new start pixel for the next approximation. In Figure 9(b), the length of line segment SC is *l*, and vectors $\mathbf{a}=(a_1,a_2)$, $\mathbf{b}=(b_1,b_2)$ and $\mathbf{X}$ are the possible curve pixels. The distance *d* can be calculated by using the following formula:

$$d = \frac{\begin{vmatrix} a_1 & a_2 \\ b_1 & b_2 \end{vmatrix}}{l}.$$

Pointer	Line number	Vectorized X coordinate
		Vectorized Y coordinate

Figure 7. Line output data structure.

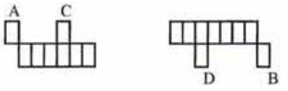

Figure 8. Thin line relationships between two rows.

Synchronous Line-Break Logical Linkage

During segmentation, a marked rectangle may cover a line if a large area of noise is situated near the line. This might cause the line to be cut during noise removal. To compensate for line breaks created during segmentation, synchronous line-break logical linkage is proposed (Pratt, 1991).

Synchronous line-break logical linkage is similar to the noise-removal algorithm. The difference is that the former detects whether any pair of points is located inside the detection search window and the latter detects where non-boundary noise pixels are. If only one start or termination point is in the detection search window, we view this start or termination point as noise which might be removed during noise removal. Otherwise, a line-break logical linkage must be taken by recording the pair of points in the detection search window.

Synchronous line-break logical linkage is also embedded in line tracing to detect where break points are and to output their coordinates to the vectorized data file provided for reconstruction and display. The advantage of this approach is that synchronous processing and logical linkage need not spend any time on line connection. Therefore, this approach is better than the conventional approach which needs complex calculations and detects all the pixels in a cadastral map.

Experimental Result

Figures 10 to 21 show results obtained from a 512 by 512 by 8 (width, height, and bits/pixel) image executed on an

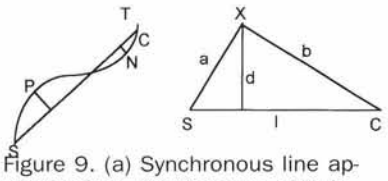

Figure 9. (a) Synchronous line approximation. (b) Distance calculation.

80486DX personal computer. Except for the time required for edge enhancement and thresholding, segmentation took about 6 seconds. Noise removal took about 12 seconds and raster-to-vector conversion, except for the time required for thinning, took about 9 seconds, both acceptable performance results.

Table 1 shows the size comparison of the three files of Figure 20 used to store 4-byte vectorized data (*X*, *Y* coordinates). The total bytes ratio is 262144:8000:560 (about 470: 14:1), a great reduction in storage space required (approximation means that only critical points are recorded; non-approximation means that each pixel of every line is stored).

Discussion and Conclusions

To assess an algorithm, six indices — (1) speed, (2) memory requirement, (3) coding, (4) parameter setting, (5) noise immunity, and (6) effectiveness — are generally considered in digital image processing. The major algorithms proposed in this paper are segmentation, raster-to-vector conversion, noise removal, synchronous line approximation, and synchronous line-break logical linkage. All are implemented with row-majority scanning and line tracing, and they all use a data-list structure to store dynamically created data, thus reducing memory requirements and avoiding sorting. These algorithms are only run-length coding integrated with list insertion, deletion, and update operations. To summarize the advantages of our approach, it can be concluded from indices (1), (2), and (3) that all algorithms provide a concise approach to processing digitized images of cadastral maps.

Segmentation uses two thresholding values: the upper and lower bounds of symbol and character size. Noise removal and synchronous line-break logical linkage use one thresholding value: the upper bound of symbol and character

Table 1. File Size Comparison

Types	Pixels Recorded	Bytes/Pixels	Total Bytes
Raster data format	512×512	1	262144
Vector (non-approximation)	2000	2×2	8000
Vector (approximation)	140	2×2	560

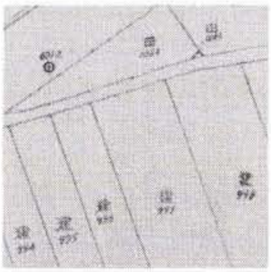
Figure 10. Source image.

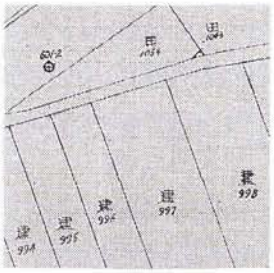
Figure 11. After edge enhancement.

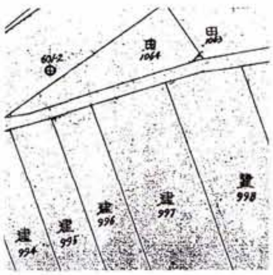

Figure 12. After bi-leveling.

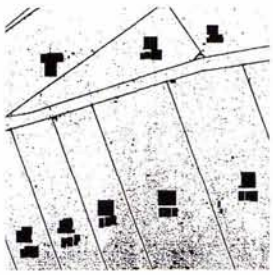
Figure 13. After pattern clipping.

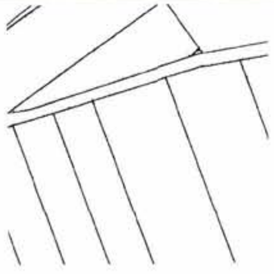
Figure 14. After segmentation.

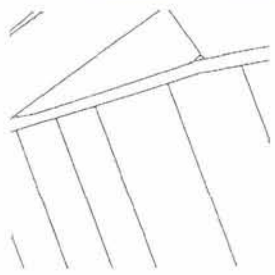
Figure 15. After thinning.

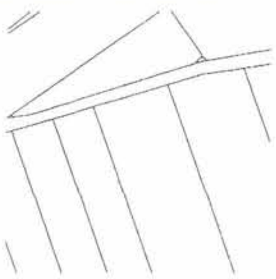
Figure 16. After noise removal.

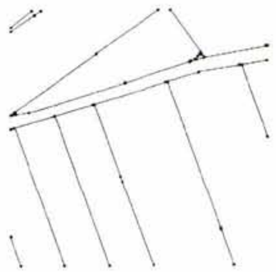
Figure 17. Vectorized points.

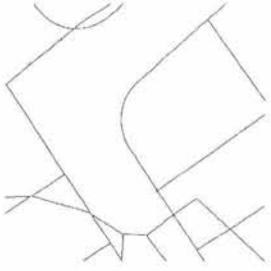
Figure 18. Curves with noise.

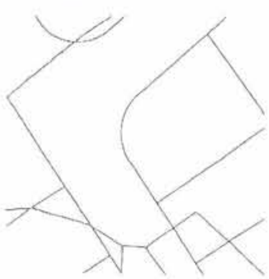
Figure 19. Curves after noise removal.

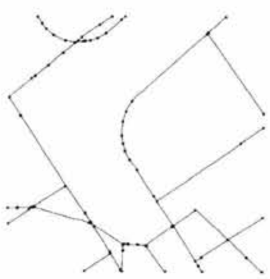
Figure 20. Vectorized points.

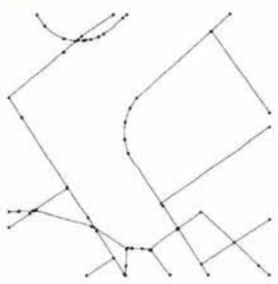
Figure 21. Larger thresholding gate.

size. Synchronous line-approximation uses one thresholding value: the line-approximation distance tolerance. It can be concluded from index (4) that each algorithm uses few parameters and the corresponding criteria are certain of physical meaning. Although segmentation can cause some line breaks, synchronous line-break logical linkage is provided to compensate, and it can be concluded from index (5) that line breaks can be reduced or even unaffected by noise removal and clipping of symbols and character.

Difficulty separating intersections between symbols, characters, and lines remains. Another problem is that most existing cadastral maps were created by manual drawing, and some symbols and characters are connected to each other. Some Chinese characters, such as 旱, 林, and 池 (dryland, woodland, and pond) to mention a few, can be separated but those elements need be viewed as Chinese characters for pattern recognition.

It can be concluded from index (6) that all algorithms can be executed effectively and can provide more information for pattern recognition. Therefore, using computer vision techniques, all algorithms can provide a feasible solution to automating digitization of cadastral maps.

References

Clarke, K.C., 1990. *Analysis and Computer Cartography.* Prentice Hall, Englewood Cliffs, New Jersey, pp. 177–203.

Freeman, H., and L.S. Davis, 1977. A corner-finding algorithm for chain-coded curves, *IEEE Transaction on Computers,* C-26(2): 297–303.

Haralick, R.M., and L.G. Shapiro, 1992. *Computer and Robot Vision,* Volume 1, Addision-Wesley Publishing Company, pp. 303–333.

Kaneko, T., 1992. Line structure extraction from line-drawing images, *Pattern Recognition,* 25(9):963–973.

Landy, M.S., and Y. Cohen, 1985. Vectorgraph coding: efficient coding of line drawings, *Computer Vision, Graphics, and Image Processing,* 30(3):331–334.

Lumia, R., L. Shapiro, and O. Zuniga, 1983. A new components for virtual memory computers, *Computer Vision, Graphics and Image Processing,* 22(2):287–300.

Musavi, M.T., M.V. Shirvaikar, E. Ramanathan, and A.R. Nekonei, 1988. A vision based method to automate map processing, *Pattern Recognition,* 21(4):319–326.

Nagasamy, V., and A.N. Langrana, 1990. Engineering drawing processing and vectorization system, *Computer Vision, Graphics, and Image Processing,* 49(2):379–397.

Pavlidis, T., 1982. *Algorithms for Graphics and Image Processing,* Computer Science Press, Potomac, Maryland.

Peuquet, D., 1981. An examination of techniques for reformatting digital cartographic data/part 1: The raster-to-vector process, *Cartographica,* 18(1):34–48.

Pratt, W.K., 1991. *Digital Image Processing,* Wiley Interscience Publication, pp. 612–614.

Ramachandran, K., 1980. Coding method for vector representation of engineering drawings, *Proceeding of the IEEE,* 68(7):813–817.

Ramer, U., 1972. An iterative procedure for the polygonal approximation of plane curves, *Computer Vision, Graphics, and Image Processing,* 1(2):244–256.

Roberge, J., 1985. A data reduction algorithm for planar curves, *Computer Vision, Graphics and Image Processing,* 29(2):168–195.

Ronse, C., and P.A. Devijver, 1984. *Connected Components in Binary Images: The Detection Problem,* Research Studies, Letchworth, Herts, England.

Rosenfeld, A., and J.S. Weszka, 1975. An improved method of angle detection on digital curves, *IEEE Transactions on Computers,* C-24(9):940–941.

Teh, C.H., and R.T. Chin, 1989. On the detection of dominant points on digital curves, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-11(8):859–872.

Wall, K., and P.E. Danielsson, 1984. A fast sequential method for polygonal approximation of digitized curves, *Computer Vision, Graphics and Image Processing,* 28(2):220–227.

Williams, C.M., 1978. An efficient algorithm for the piecewise linear approximatation of planar curves, *Computer Vision, Graphics and Image Processing,* 8(2):286–293.

Wu, Tse-Chen, 1990. *The Automatic Processing of Computerized Cadastral Map,* M.S. Thesis, Institute of Computer Science and Electronic Engineering, National Central University, Chung-Li, Taiwan, R.O.C.

Yamada, H., K. Yamamoto, T. Saito, and S. Matsui, 1991. Map:multiangled parallelism for feature extraction from topographic maps, *Pattern Recognition,* 24(6):479–488.

Yamada, H., K. Yamamoto, and K. Hosokawa, 1993. Directional mathematical morphology and reformalized Hough transformation for the analysis of topographic maps, *IEEE Transactions on Pattern Analysis and Machine Intelligence,* PAMI-15(4):380–387.

Yen, Hei-Jin, 1989. *Automatic DTM Generation by Contour Lines Digitization,* M.S. Thesis, Department of Resource Management, Defence Management College, Chung-Ho, Taiwan, R.O.C.

(Received 21 February 1995; accepted 17 July 1995; revised 29 August 1995)