A HYBRID LINE THINNING APPROACH

Cixiang Zhan
Environmental Systems Research Institute
380 New York Street
Redlands, CA 92374, USA
Telephone (909) 793-2853 Fax (909) 793-5953
Email czhan@esri.com

ABSTRACT

The proposed hybrid thinning approach consists of preprocessing, distance-transform based skeleton extraction, sequential thinning and post-processing. The preprocessing smooths line edges fills holes. The Euclidean distance transform is then performed, and skeletons are extracted via established lookup tables to produce unbiased center lines. Sequential thinning, which deals with nearly-thinned lines better than other approaches, is then applied to thin skeletons to single-pixel width. The post-processing removes spurs, connects disconnected lines caused by skeleton extraction, and extends eroded line tips. Large data sets can be handled. Experiments on contour, parcel and land use data are presented.

INTRODUCTION

Line thinning is important to data compression, raster to vector conversion and pattern recognition. The general requirements of line thinning include the quality of results, speed and ability to handle large images with limited memory. The quality of results include the preservation of geometric and topological properties [Lam et al, 1992], and visual acceptance for problem domains. To preserve the geometric properties the thinned lines should be the median axes of the original line features, maintain the original line lengths, and be clean without additional spurs. To preserve the topological properties, the thinned lines must preserve the connection of the original lines without disconnection and additional loops. The visual acceptance is highly dependent on applications, and may include line smoothness and junction appearance.

Various thinning algorithms (Peuquet, 1984, Lam et al, 1992) have been developed to satisfy these requirements with some requirements being emphasized for a particular problem domain. Thinning algorithms can be divided into the iterative and the distance-transform based. with the iterative further divided into sequential and parallel classes. The speed of iterative approaches, which iteratively peel the contours of thick lines based on the local properties within a moving window, are generally dependent on line width, and its performance in geometric preservation depends on scan direction. The sequential algorithms, which do peeling based on the line patterns in the current iteration, are generally

faster on sequential machines and preserve connection better than the parallel algorithms, but their results are often biased away from the scan directions. Parallel algorithms, which peel contours based on patterns of the previous iteration, makes using parallel processors possible, and are less sensitive to scan direction than the sequential. But to maintain connection, they are forced to use sub-iterations or a large moving window. The distance transform based approaches, which normally perform Euclidean distance transformation on line network and extract skeletons based on the global information of distances from edges, may produce well centered thinned lines of width of one or two pixels at once. The resulted skeletons, however, may not preserve connection, and are sensitive to noise.

Hybrid approaches may be adopted to take advantages of different approaches. Arcelli and Sanniti (1985) combined distance transform and sequential thinning, with capability of reconstruction of original features.In this study, a hybrid thinning approach, uses lookup tables for skeleton extraction based on the Euclidean distance transform and perform sequential thinning and extensive post-processing to solve the problems inherent to the distance-transform based approaches, is described.

OVERVIEW OF THE HYBRID APPROACH,

In the hybrid approach, the Euclidean distance transform is performed first to produce x and y displacements of pixels from line edges. Skeletons are identified using lookup tables by checking the x and y displacements of pixels. Because the distance transform is sensitive to noise, a morphological dilation/erosion filter is optionally used before distance transform to smooth ragged edges and remove small holes within lines. Sequential thinning is followed to further thin skeletons to single-pixel width. The thinned lines are further processed by removing spurs, connecting broken skeletons, extending eroded line tips within the boundaries of the original line, and removing some false junction pixels. The program allows the control of the output line type being either smooth lines or lines with sharp corners. Large images are processed in strips with proper overlap between stirpes. The maximum thickness of input line features, as an input parameter, is used to determine the length limit of spurs and overlapping size of image strips. We will refer to line features in the context of a foreground consisting of black pixels, and a background consisting of white pixels.

DISTANCE TRANSFORM

In the distance transform, the white pixels in the background are used as the source, and the proximity of the black pixels on line features to source pixels are measured. The Euclidean distance transform calculates for each black pixel its X and Y displacements to its nearest source pixel. Actual distance calculation is avoided to reduce computation.

Danielson's algorithm [Danielsson, 1980] is used in the distance transformation. Two passes are required in distance mapping and five neighbors need to be visited in each pass. According to Danielsson, the errors from the true Euclidean distances with eight neighbors is sparsely distributed, and errors are bounded to be less than 0.076 pixel.

SKELETON EXTRACTION

We first briefly explain what we mean by the skeleton of a raster feature. In the raster domain, the disk of radius R centered at a pixel is the set of all pixels whose centers are within the distance R from the center of the pixel. Pixels that are exactly at distance R from the disk center are excluded from the disk. From the center of each feature pixel, there exists a disk of maximum radius among all disks which lie within the feature. The skeleton of a feature consists of those pixels whose maximum disk within the feature can not be covered by the maximum disk of another pixel in the feature.

For a disk of distance R, perform the Euclidean distance transform. Since the skeleton of the disk is its center pixel, it can be assumed the maximum disk of the neighbors of the center pixels is covered by that of the center pixel. If a pixel in the input image has X and Y displacements that match those of one of neighbors of the center pixel in the disk, we may assume the pixel is not a skeletal pixel in the input image. By checking the X and Y displacements of each black pixel and its neighbors it is possible to identify if the black pixel is a skeletal pixel of a line feature. Danielsson [1980] shows that the error by using this local neighborhood checking of X and Y displacements is extremely small due to raster points available.

To quickly identify skeletal pixels with this approach, a set of look-up tables with a pair of X and Y displacements as inputs, and the X and Y displacements of pixels, whose disk are covered by the disk in the diagonal or orthogonal direction, as outputs are established. To build those look-up tables, distance transform is performed on each of the disks with integer squares of radii from 1 to N, where $\sqrt{N}$ is the width of the thickest line the program is capable of processing. The quadrants of possible X and Y displacements of the disk of squares of radii 1 to 25 are shown in Fig. 1. The X and Y displacements of the disk centers become the input to the look-up tables and the X and Y displacements of its orthogonal or diagonal neighbors becomes the output of the orthogonal or diagonal look-up tables, respectively. Given the X and Y displacements of a pixel and its neighbors, the pixel can be identified as a non-skeletal pixel if its X and Y displacements are the outputs of either the orthogonal or diagonal look-up table with the X and Y displacements of one of its neighbors as the inputs.

To make the extracted skeletons adapted for line thinning, attention has been given to the following four situations in

building these look-up tables.

a. At a radius, the X and Y displacements may have multiple pairs of values, excluding the swap of X and Y displacements. For instance, at radius 5 (Fig. 1), both X and Y displacements (5,0) and (4,3) are valid, and may result from different scan directions. All multiple pairs of X and Y displacements at a radius should be identified.

b. A few pair of X and Y displacements at a disk center have different pairs of X and Y displacements at their diagonal neighbors. For instance X and Y displacements (6,0) at the disk center may have both pairs of X and Y displacements (5,0) and (4,3) at its diagonal neighbors depending on scan directions. For some pairs of input X and Y displacements is may be necessary to have two pairs of X and Y displacements as their outputs.

c. Border pixels (1,0 or 0,1) need special handling in order to maintain connection of thin lines. As shown in Fig. 2 the border pixel of (1,0), marked by a thick box, is not normally a skeletal pixel, but should not be removed in order to maintain necessary connection. Those border pixels, which have no neighbor pixel with X and Y displacements (1,1) or (2,0) on one side and another neighbor pixel with X and Y displacements (0,0) on the opposite side, should be removed.

d. To further reduce the disconnection caused by skeleton extraction, the disk of X and Y displacements (2,0) is made not being covered by that of X and Y displacements (3,0). This pixel of (2,0), marked with a thick box in Fig. 2, together with the pixel of X and Y displacements (1,0), also marked with a thick box, maintain the connection in a line with a narrow portion (Fig. 2). Further modifications in look-up tables may improve connection, but tends to increase noise and blur the major line features in skeletons.

It is well known that the skeleton of a line feature extracted using the median axis transform in the discrete space may be of two-pixels width and disconnected, and may have spurs at turns and line ends. For the purpose of line thinning, the erosion of line ends during skeleton extraction need to be recovered. The following four sections describe the processing steps to solve these problems.

SEQUENTIAL THINNING

The major purpose of the step is to further thin the resulted skeletons into thin lines of one-pixel width. In addition single black pixels resulted from skeleton extraction are dropped, and gaps of one-pixel width between disconnected skeletons are filled. The basic idea of the sequential thinning algorithm by Greenlee [1987] is used since the algorithm takes into account of complete 256 junction patterns and is flexible in adjusting the rules of pixel elimination. First the resulted skeleton image is encoded.

Each neighbor of a pixel are assigned an direction number of N^2 where (1 <= N <= 8) depending on the direction of the neighbor. The code of a pixel is the sum of the direction numbers of its black neighbors. Black pixels of code 0 is single pixel and are dropped during encoding. Decision rules for pixel filling and peeling are developed based on the coding. When a pixel is filled or removed, the codes of its neighbors are updated. The procedure is iterated until no change occurs. Peeling rules are set differently for smooth and sharp-turned output lines. Since the skeletons are of no more than two pixel width, the bias of thinned lines that may be caused by sequential thinning are negligible. One pass is enough to thin all skeletons to one-pixel width. At least four passes are required in the step, one for encoding, one for filling, one for thinning and one for checking code changes. Approaches based on contour processing [Kwok, 1988], [Naccache, 1984] may be used for those nearly thinned lines to improve efficiency.

SPUR REMOVAL

Before spur removal, all black pixels are encoded using the number of its black neighbors, and white pixels are labeled 0. A Pixel of code 1 is a tip of a line, a pixel of code 2 is generally the intermediate line pixel, and a pixel of code 3 or larger generally is a junction pixel. Sometimes, a pixel of code 2 may be a spur pixel (Fig. 3(d), and a pixel of code 3 may forms a false junction, as shown in Fig. 3(e).

There are two types of spurs often occur in the resulted skeletons, 1) middle spurs that occur in the middle of a line when the line direction or thickness changes and 2) end spurs that occur at the line ends (Fig. 3) Middle spurs can be processed individually. End spurs, however, appear as pairs, and each pair must be processed simultaneously. Otherwise, after the first spur is removed, the second spur becomes a portion of a line, and can no longer be recognized. The spurs may also be classified as code 1 spurs whose tip pixels are coded 1, and code 2 spurs whose tip pixels are coded 2 and whose length is one pixel. We will discuss the removal of code 2 spurs in the next section.

To identify code 1 spurs, the resulted output is scanned to find pixels of code 1 (tips). From a line tip the next black pixel along the line is searched. When a pixel of code 2 is found, its another black neighbor is searched. The search is restricted in a limited sector without visiting the neighbors of the preceding black pixel. Searching ends when reaching a junction pixel (code > 3), or an end pixel (code 1 or 0), or the defined maximum spur length are reached. The maximum spur length can be defined as 0.8 times the maximum line thickness. When a search ends at a pixel of code 3 or larger, we call the pixel the junction-end pixel of the search or the spur that is identified. The major difficulty is to identify false junctions in Fig. 3(e).

Removing spurs that does not end at a junction-end pixel is to simply to remove all pixels visited during search. When a spur ends at a junction-end pixel and spur is identified, however, one must to decide if the junction-end pixel of the spur should be removed. When the junction-end pixel constitutes a necessary connection for the remaining line, it should not be removed. On the other hand, leaving an extra pixel at a junction means that the spur is not completely removed.

To identify false junctions and determine the removal of junction-end pixels of spurs, all possible patterns when search reaches a pixel of code 3 or larger (a true or false junction) are studied.

Fig. 4 shows the all possible junctions patterns when reaching a junction pixel of code 3 from the upper-left and from the top. For reaching junctions of codes larger than 3, the pattern can be obtained similarly. Based on the analysis of those patterns the following rules are developed to identify spurs and to handle the junction-end pixels. In pattern 1, 2, 3, 5 and 7 of Fig. 4 true junctions are reached, and spurs are identified. The junction-end pixel J should not be deleted, because the other two neighbors of pixel J, (except for the neighbor preceding pixel J in the search) are not contiguous, i.e. at least a white neighbor is between those neighbors.

In pattern 4, 6 and 8, the other two neighbors of pixel J are contiguous, and among them one and only one is the direct (orthogonal) neighbor of pixel J (labeled D in Fig. 4). The following rules are developed to detect false junctions. If the code of D is less than 3, a false junction is detected; if the code of D is larger than 3, a true junction is identified. When the code of D is 3, J and D form a false junction if the rest of neighbors of D are not contiguous, and J reaches a true junction otherwise. Whenever a true junction is reached, the junction-end pixel J can be removed.

When the code of a junction-end pixel is larger than 3, a true junction is always reached, and a spur is identified. To determine if the junction-end pixel is to be removed, the rules for junction end pixels of code 3 can be similarly applied. If the rest of neighbors of a junction-end pixel are not all contiguous, the junction-end pixel is not removed, otherwise, it can be removed.

Although some junction-end pixels, such as in pattern 5 of Fig. 4, can be removed without creating disconnection, it is better to maintain them for the simplicity of rules, and for better junction handling by considering the rest of junction pixels and different junction handling requirements.

When a spur is removed, the codes of neighbors of the last pixel of the spur should be updated. when the code of one of its neighbor becomes 1, a new line tip is created and the need

to be extended

REMOVAL OF EXTRA PIXELS OF CODE 2

After the previous processing some pixels of code 2, which are actually spurs of one-pixel length or the corners of sharp turns that are redundant for smooth line requirement, need to be removed. There are two types of pixels of code 2 that need to be removed, as shown in Fig. 3(d) and (e). The one in Fig. 4(e) has two adjacent direct neighbors, and is not subject to removal when sharp turns are demanded. The spur in Fig. 4(d) can be easily identified since this type of pixels of code 2 always has two adjacent neighbors. The code 2 spurs may also appear as pairs at line ends (Fig. 3(c)), and each pair needs to be handled together. In all other cases pixels of code 2 form necessary line connections, and should not be removed. When a pixel of code 2 is removed, the codes of its neighbors should be updated. When the updated code of a neighboring pixel becomes 2, the pixel needs to be examined for removal, and the process becomes recursive. When the updated code becomes 1, the new line tip is to be extended, as shown in the next section.

EXTENDING LINE TIPS

The purpose of this step is to extend the tips of the thinned lines, to the boundaries of the original lines, or to connect to other lines. In the first case, we try to recover the line length eroded in the previous processing, and in the second case, we try to link the disconnected skeletons. The extending directions are determined in increments of 22.5 degrees by using the last 3 pixels from the line tips. Extending in the directions of angles 22.5+45*N (n=0,7) is realized by alternatively extending in the orthogonal and diagonal directions. Sixteen extending directions makes extended lines coincide with the original line direction better than eight extending directions do. The process that extends line ends are embedded in the process of spur removal and is invoked when the updated code of any pixel becomes 1.

HANDLING LARGE IMAGES

To be efficient many processing steps require the input and output images to be entirely held in memory. When a large image can not be held in memory, it can be processed strip by strip. Each strip spans the width of the image, and overlaps its adjacent strips by a number of rows, which is no less than the maximum thickness of line features. No significant problems have been found in the output by processing in strips. Only a shift of one pixel may be found when some vertical lines cross the border between strips.

CONCLUSION

The paper proposes a hybrid thinning approach, in which the distance-transform based approach is combined with the

sequential approach. The hybrid approach takes the advantages of both approaches: the speed independent of line width and fine median axes from the Euclidean distance transform, and flexibility to handle the fine detail of nearly thinned lines from the sequential approach. The look-up tables for skeleton extraction enable skeletons to be quickly and properly extracted for further processing. The lookup tables have been established to be able to handle thick lines of width up to 60 pixels. To solve the problems left mainly by the skeleton extraction based on distance transform, such as spurs and disconnection, extensive post-processing procedures are developed. Simultaneously handling of pairs of spurs, and recursively processing new spurs are essential to ensure the quality. Extending line tips in proper directions not only restores the line length, more importantly, re-links the disconnected line skeletons.

The approach performed well on contour lines, parcel map, roads and various land-use and land-cover data, where the range of line widths may not be regular, features may be relatively noisy, and sharp-turned or smooth lines may be required. Some of the results are show in Fig. 5. The major problems with the approach exist at junctions of thick lines, where it produces dimples at T junctions, and creates two junctions at a X junction. These junction problems can be better handled in vector structures if vertorization is performed after thinning.

REFERENCES

Arcelli, Carlo and Gabriella, Sanniti DI Baja, 1985, A Width-Independent Fast Thinning Algorithm, *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.7:463-474.

Danielsson, Per-Erik, 1980, Euclidean Distance Mapping, *Computer Graphics and Image Processing*, Vol. 14:277-248.

Greenlee, David D, 1987, Raster and Vector Processing For Scanned Line Wrok, *Photogrammetric Engineering and Remote Sensing*, Vol. 53:1383-1387.

Kwok, Paul C.K., 1988, A Thinning Algorithm By Contour Generation, *Communication of the ACM*, Vol. 31 1314-1324.

Lam, Louisa, Seong-Whan Lee, and Ching Y. Sun, 1992, Thinning Methodologies - A Comprehensive Survey, *IEEE Trans. on Pattern Recognition and Machine Intelligence*, Vol. 14:869-885.

Naccache, N.J. and Shinghal, R, SPTA: A proposed algorithm for the Thinning Binary Patterns, *IEEE Trans. on System, Max and Cybernetics*, SMC-14:409-418.

Peuquet, Donna J., 1981, An Examination of Techniques for Reformatting Digital Cartographic Data /Part 1: The Rater-toVector Process, *Cartographica*, Vol. 18:34-48.

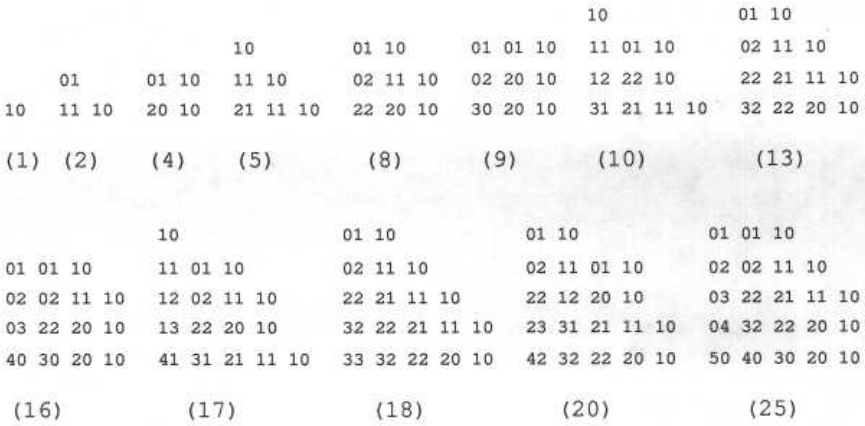

Fig. 1 X and Y Displacements of Disks (The numbers in parentheses are the squares of radii of disks)

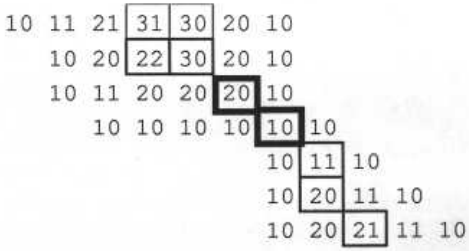

Fig. 2 Modifications to Normal Skeleton Extraction. Normal skeleton pixels are marked with thin boxes, Pixels with thick boxes are put into skeletons.

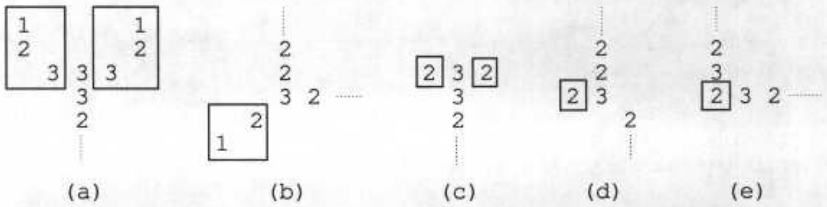

Fig. 3 Types of Spurs. (a) and (b): code 1; (c), (d) and (e) code: 2. (a) and (b): end spurs; (b) and (d): middle spurs. (e) middle spur in smooth lines. Pixels surrounded by boxes are spur pixels.

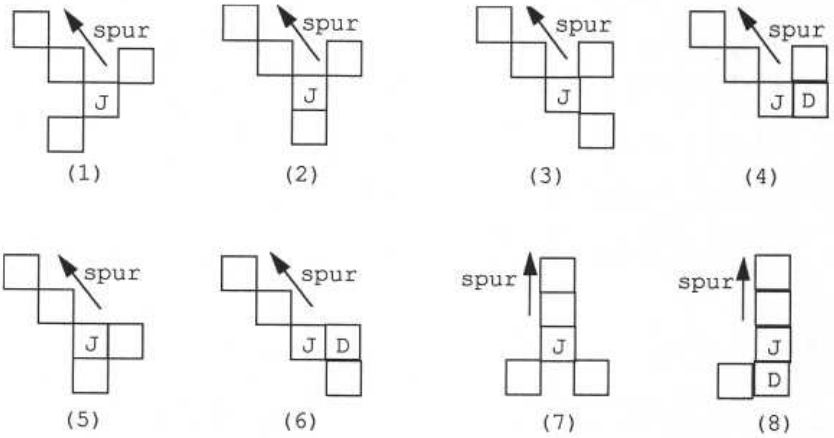

Fig. 4 Patterns When Search from Tips Reaches Pixels of Code 3 (Labeled J) from the Upper-Left (1)-(6), and the Top (7)-(8). D is a direct neighbor of J.

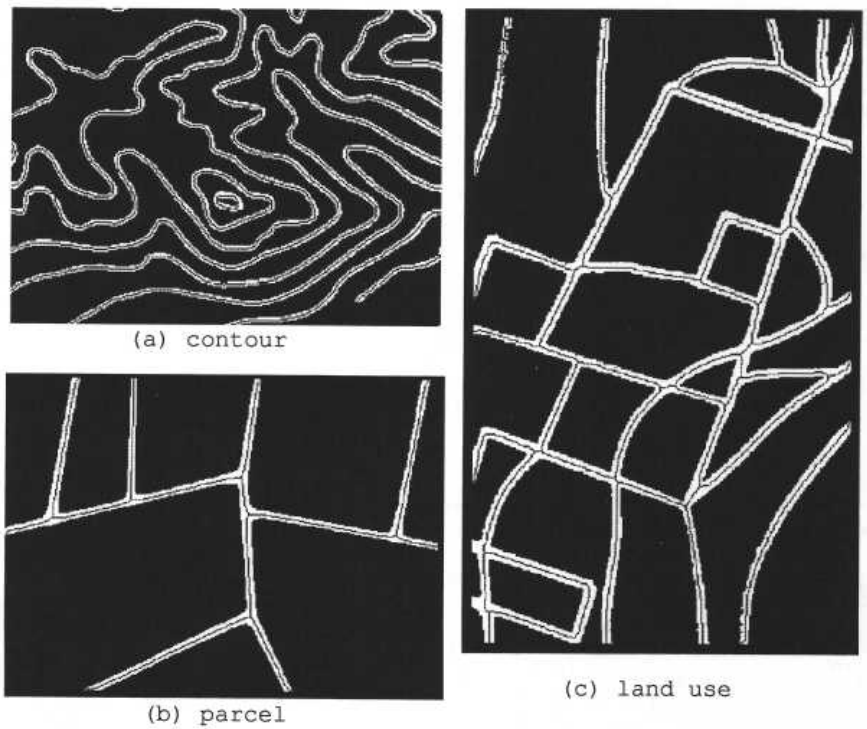

Fig. 5 Thinning Results of Line Features