MVA'90 IAPR Workshop on Machine Vision Applications Nov. 28-30,1990, Tokyo

RASTER TO VECTOR CONVERSION IN A MAP INTERPRETATION SYSTEM

Paul C K Kwok
Department of Computer Science
The University of Calgary
2500 University Dr NW
Calgary, Alberta, Canada T2N 1N4

T John Turner
DataSpan Technology, Inc.
400 - 540 5th Avenue SW
Calgary, Alberta
Canada T2P 0M2

ABSTRACT

This paper describes the two components: thinning and vectorization, at the front end of a Geographic Information System. The raster dataset associated with a map is very large. A small D-sized drawing gives a 120 Mpixel document. The amount of page faults encountered by a typical conventional thinning algorithm is so large that the run time is dominated by disk I/O time. The Contour Generation Thinning Algorithm based on chain codes has been adopted and the amount of page faults are reduced by a factor equal to the total number of iterations.

The skeleton is vectorized by traversing the chain codes and examining the raster. A tagging scheme is used that allows the vectorization process to identify all the end points and multiply-connected nodes.

INTRODUCTION

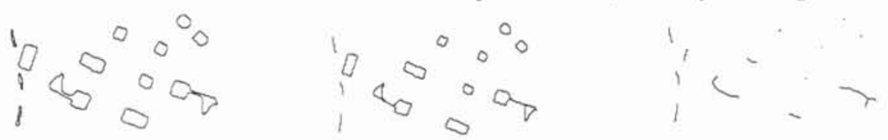
Hard copy graphical documents generated and accumulated by many organizations are becoming a problem because of the lack of storage space; the perishable nature of printed documents and the inflexibility of the medium. They are harder to modify and integrate with other data. Thus there is an urgent need to capture these graphical data and convert them into a digital format for subsequent processing and retrieval.

One such processing and retrieval system is a Geographic Information System (GIS). At the front end of a GIS, the cartographic document interpretation process is responsible for the preparation of input from maps and drawings. It takes the scanned data, extracts the boundaries of line segments and symbols, thins and vectorizes them, and in the case of symbols performs recognition. The result is an interpreted document which is the input to a GIS.

The magnitude of the task is tremendous because of the large dataset associated with the input document. A small D sized (22"x34") drawing digitized at 400 dots per inch gives a 120 Mpixel document. The system is usually implemented on a general purpose workstation environment for the ease of its incorporation in an existing GIS. In the following sections, the thinning and vectorization processes of a document interpretation system developed by DataSpan Technology and the Alberta Research Council are described.

THINNING

Many iterative thinning algorithms have been developed in the past twenty years. Most of them are parallel algorithms [1] that rely on parallel processing hardware, or sequential algorithms [2] that scan the image pixel by pixel during every iteration. A new class of thinning algorithms have received much attention in recent years. They are serial algorithms [3] that process contour pixels from a queue, and by doing so, create or generate new contour pixels to be placed at the end of the queue. Some algorithms [4] store the pointers of the contour pixels. Others [3, 5]store the chain code which enables successive contour pixels to inherit information from their predecessors, whereby reducing the amount of data that



a) Raster b) Partially thinned c) Entirely Thinned

Figure 1. Thinning a Raster

requires examination during the processing.

The Contour Generation Thinning Algorithm based on chain codes [5] has been demonstrated to be faster than many algorithms. In the beginning, the contours of the components are traced to establish the initial queue of chain codes.

BOUNDARY EXTRACTION.
Many contour tracing algorithms are available [6-8]. The algorithm described in [5] is fast but consumes a lot of memory for its efficiency. In this system, Capson's algorithm [6] was adopted, with some modifications. The contour tracer writes the chain codes to a file so that they can be examined later for smoothing and noise removal. The usual convention is that the trace is counterclockwise for an exterior contour and clockwise for an interior contour (hole).

A component can be identified uniquely by a set of chain codes describing its contours. Contour tracing separates the components so that they can be processed independent of each other.

Assume the bitmap is first initialized to zero. The contour is generated by using the chain codes. Every contour pixel generated will have its value incremented.

ITERATION.
A component is thinned by an iterative "traversing" the present contour and "generating" a new contour. Traversal is accompolished using the chain code. Successive contour pixels can be found by using the link. When a contour pixel is visited, its value determines whether it is going to be removed or not. If it is to be removed, the new contour passes through its 4-neighbors in a clockwise direction. If it is to be retained, the new contour passes through it. The four parameters: the values of the contour pixel and its predecessor, and the two links terminating at and originating at the pixel will enable the new contour to be generated. Again, every pixel generated has its value incremented.

The iterative process terminates when all the pixels visited have values of at least 2.

PAGE FAULTS.
With a conventional sequential thinning algorithm, the entire image is scanned once every iteration. With a large bitmap, all the memory pages in the bitmap will be swapped into the main memory one by one during an iteration. If m is the total number of iterations; N the bitmap size and s the page size, the number of page fault encountered is, in the worst case, mN/s. If the components are separated, each component can be thinned independently. When a component is thinned, it is necessary to bring into the main memory those pages occupied by the component. In the case of a contour map, the dark area occupied by an individual curve is sparse compared with the entire bitmap. Even with a curve that spans the entire bitmap, the number of memory pages associated with the curve is still very small and can be accommodated in the main memory of a workstation. If the workstation does not have sufficient memory to accommodate the pages associated with a component, a sparse matrix representation can be used. In this case, disk I/O time is minimized at the expense of a slight increase in computation time for locating pixels in the bitmap.

PARTIAL THINNING.
A typical cartographic map consists of lines, hand-drawn characters and symbols. While it is necessary to thin lines to their skeletons, it is undersirable to thin characters and symbols completely. Hand-drawn characters may have defects such as a filled hole as in the case of the numeral "6". The image should be thinned to the extent of the line width. The unthinned region can then be used as the hole in the digit.

Another example is in the school symbol which is a polygon with a flag on the top (Figure 1). Thinning the symbol completely will eliminate all the polygon data, and thus destroy vital features necessary for recognition.

One problem with thinning an object completely is the creation of artifacts such as "displaced junctions". An example is in a T intersection. It is possible to thin the object partially and determine the actual position of the intersection later [9].

VECTORIZATION

Pavlidis [8] described a method for vectorizing thinned images. In his method, the skeleton is scanned to find a thinned pixel to be used as a starting point. The skeleton is followed until a node of degree other than two is found. A graph traversal algorithm is then used to vectorize the raster. The application of Pavlidis' method is difficult in the case of a partially thinned component. Therefore a new method has been adopted.

DEFINITION OF A NODE.
Consider a fully thinned raster image. Any dark pixel can have up to four thinned dark neighbors. If there are more than 4 dark neighbors at least two of them cannot be thin. If there are zero thinned neighbors, the pixel is a singleton. A dark pixel is n connected if there are exactly n thinned dark neighbors. Most of the pixels will be two connected with each of their neighbors being two connected as well. These strings of two connected pixels will be referred to as edges. When recording an edge it is only necessary to record the point where the edge makes a turn. A one connected pixel will be referred to as an end point as this is where an edge suddenly comes to an end. Pixels that have more than two edges entering it will simply be referred to as a node. A node needs to be further defined as a pixel that has more than two thinned neighbors that have a white space in between each of

them. That is, the number of zero to two transitions should be greater than two. This further requirement is necessary because a pixel may have more than two thinned neighbors and be part of an edge if the pixel is immediately next to a node.

With this definition of a node and exhaustively going through the definition of a multiple pixel, it can be seen that there are only four node types (Figure 2).

```
2   2      2            2   2      2   2
  2          3 2          3          4
  2          2              2      2   2

(a) 8Y node  (b) 4Y node  (c) 8T node  (d) 8X node
```

Figure 2. Possible node configurations.

Nodes such as the 4T and the 4X nodes are not possible the chain codes will cut the corner of the 90 degree turn and not touch the pixel.

DETECTION OF NODES.
Directly following the chain codes is insufficient to detect the nodes as the pixels is encountered. This is because a 4Y node is only touched twice and so the pixel count cannot be used to determine whether the pixel is a node or not. There are three solutions. The pixel can be vectorized as part of an edge and a postprocessing stage can be applied to split these edges and make proper nodes. This would be a fairly expensive operation because it involves a considerable amount of searching. The second solution is to examine the pixel's neighbors. However, examining a pixel's neighbors is time consuming. The third solution is to note that each thinned pixel in the raster is touched at least twice when the chain codes are traversed. A pixel is said to be touched on the first encounter and it can be vectorized on the second encounter. A node can be detected by noting the transition between a vectorized pixel and an untouched pixel or between a touched pixel to a previously touched pixel. Using this method only the current pixel and the next pixel need to be examined. This is a considerable improvement over the former two solutions.

This method works fine except for the 4Y nodes. This is because eight connected chain codes will cut the corner of any 90 degree turn whose initial and final direction is parallel to axes. The solution is to store the previous link in the current pixel as it is touched. This can then be consulted when the pixel gets vectorized to resolve the 4Y nodes.

When two nodes are neighbors of each other, the configuration is called a "dual node" (Figure 3). Dual nodes should be recorded and resolved after the vectorization is complete.

```
  2                       2   2
    3 2                     3|
2 3/                        3
  2                       2   2

(a) An 8Y/4Y dual node    (b) an 8Y/8Y dual node
```

Figure 3. Dual nodes

It can be seen that the information about a pixel can be encoded in a single byte: the lower three bits store the pixel value produced by the thinning process (possible values are 0 to 4). The next two bits are used to indicate the pixel's state (untouched, touched, vectorized or a node). The upper three bits store the value of the previous link when the pixel is touched.

VECTORIZING UNTHINNED REGIONS.
Unthinned regions are touched only once by the chain codes and need to be vectorized when they are first encountered. Two unthinned edges usually meet at a common thinned pixel. The thinned pixel is used to represent the node and the unthinned edges should include this point even though this point is thinned. This implies an examination of the next pixel when vectorizing. If the current pixel is thinned and the next pixel is unthinned, the current pixel should be made a node. Likewise, if the current pixel is unthinned and the next pixel is thinned, the next pixel should be made a node. Occasionally two unthinned edges will meet at thinned pixels residing next to each other (Figure 4). These two pixels should be noted as dual nodes and resolved later.

RESOLVING DUAL NODES.
Pavlidis [8] suggested that pixels in a dual node should be considered as an identical point. However, if there is a string or network of dual nodes (Figure 4), this is not an appropriate solution. These dual node should be thought of as having an edge created between them.

```
  1 1                 2         2
  2-2\                  2 1 2
2     2                  >2<
    2/  1               2 1 2
  2   1               2         2

(a) Strings of        (b) A network of
    dual nodes            dual nodes
```

Figure 4. Complex configurations of dual nodes.

When dealing with both thinned and unthinned edges it is often difficult to determine if the edge between the dual nodes should be thinned or unthinned. Dual nodes can be resolved after all the dual nodes have been recorded. Resolution is similar to the scene labelling problem presented by Waltz [10]. Using the constraint that each node must have an even number of unthinned edges and using Mackworth's [11] arc consistency algorithm most of the cases can be resolved. Note that a network or string of dual nodes is resolved only because both end on nodes that have only one dual node. Solving these nodes will cause the next nodes to become resolvable.

IMPLEMENTATION

The technique has been implemented in C++ on Sun 4 and MIPS workstations [12]. A typical D sized drawing (22 in. x 34 in.) of a contour map contains about 3600 contours of various sizes and about 1200 digits. Scanning takes 25 minutes using a Scitex laser scanner at 400 dots per inch and produces a run length encoded file of 10.2 Mbytes in size. Depending on the individual map, it takes approximately 10 minutes to perform boundary tracing; 10 minutes for thinning; 14 minutes for vectorization and 3 minutes for polygonal approximation [13]. It is used in production at DataSpan as well as being marketed world wide on UNIX based workstations.

CONCLUSION

A thinning and vectorization components in a document interpretation system have been presented. The algorithms were developed and tailored for the problem of a large dataset which cannot be accommodated in the main memory of a contemporary workstation. Some new strategies have been adopted to vectorize a partially thinned bitmap. These techniques have been implemented, tested and used extensively in a production environment.

ACKNOWLEDGEMENT

The authors would like to thank DataSpan Technology Inc. for permission to publish this paper. Paul Kwok is supported by the Natural Sciences and Engineering Council of Canada.

REFERENCES

[1] Zhang, T.Y. and Suen, C.Y. A fast parallel algorithm for thinning digital patterns. *Communications ACM 27*, 3 (1984), 236 -239.
[2] Naccache, N.J. and Shinghal, R. SPTA : a proposed algorithm for thinning binary patterns. *IEEE Trans. Systems, Man and Cybernetics SMC-14*, 3 (1984), 409 - 418.
[3] Arcelli, C. Pattern thinning by contour tracing. *Computer Graphics and Image Processing 17*, (1981), 130 - 144.
[4] Xu, W. and Wang, C. CGT: A fast thinning algorithm implemented on a sequential computer. *IEEE Trans. Systems, Man and Cybernetics SMC-17*, 5 (1987), 847 -851.
[5] Kwok, P.C.K. A thinning algorithm by contour generation. *Communications ACM 31*, 11, (1988), 1314 - 1324.
[6] Capson, D.W. An improved algorithm for the sequential extraction of boundaries from a raster scan. *Computer Vision, Graphics and Image Processing 28*, (1984), 109 - 125.
[7] Grant, G. and Reid, A.F. An efficient algorithm for boundary tracing and feature extraction. *Computer Graphics and Image Processing 17*, (1981), 225 - 237.
[8] Pavlidis, T. *Algorithms for Graphics and Image Processing*. Rockville MD: Computer Science Press, (1982).
[9] Brown, R.M., Fay, T.H. and Walker, C.L. Handprinted symbol recognition system. *Pattern Recognition 21*, 2, (1988), 91 - 118.
[10] Waltz, D. Understanding line drawings of scenes with shadows. in *The Psychology of Computer Vision*, P. Winston, Ed., McGraw Hill, (1975).
[11] Mackworth, A.K. Consistency in networks of relations. *Artificial Intelligence 8*, (1977), 99 - 119.
[12] Melville, R., Michalchuk, J., Turner, T., O'Connor, K., Schack, B., Sidebottom, G. and Liblong, B. A cartographic document interpretation system. *Proceedings of the Seventh Thematic Conference on Remote Sensing for Exploration Geology*, vol. 2, Calgary, (1989), 1125 - 1139.
[13] Wall, K. and Danielsson, P. A fast sequential method for polygonal approximation of digitized curves. *Computer Vision, Graphics and Image Processing 28*, (1984), 220 - 227.