AN EFFICIENT THINNING ALGORITHM FOR GREYSCALE IMAGES

Koushik Das and Phalguni Gupta
Department of Computer Science & Engineering
Indian Institute of Technology,
Kanpur 208 016, India
e-mail: pg@iitk.ac.in

ABSTRACT

In this paper an efficient thinning algorithm is p-resented which tries to minimize the problem of the deformation in shape of the skeleton at corner points. The algorithm is a hybrid of the medial axis method [1] and the contour generation method [2]. Experimental results with various greyscale and binary pattern reveal that our method generates better skeletons than the previous well known algorithms.

1. INTRODUCTION

Thinning is an important preprocessing step for many image analysis operations such as optical character recognition, fingerprint recognition and document processing. Thinning involves removing points or layers of an outline from a pattern until all lines or curves are single pixel wide. The resulting set of lines or curves is called the skeleton of the object. There is no mathamatical definition of a skeleton; applying d-ifferent existing thinning methods on patterns lead to different results. The general approach for extracting the skeleton consists of removing, at each iteration, all the edge pixels except those pixels that belong to the skeleton. The edge pixels are those that denote the boundaries (also holes) that are present in a pattern.

Most of the thinning algorithms are iterative. In an iteration, the edge pixels are examined against a set of criteria to decide whether the edge pixels should be removed or not. There exists several thinning algorithms on sequential and parallel computers. A sequential algorithm uses result from the previous iteration and the result obtained so far in the current iteration to process the current pixel. Thus at any point in an iteration a number of pixels have been already processed. These results can be used immediately to process the next pixel. With parallel algorithms, on the other hand, only the results from the previous iteration affects the decision to remove a point in the current iteration, making it suitable for processing by parallel hardware such as an array processor. Most applications make use of one of these two strategies to thin various shapes. One algorithm may generate good skeletons for some shapes but can produce poor skeletons with other. It is very difficult to develop a generalized thinning algorithm which can produce satisfactory results for all varieties of pattern shapes.

Actually thinning is a simple task for human beings. One can thin patterns with a wide variety of shapes without any difficulty. It appears that he first catches the global view of the shapes, then applies different algorithms to thin different shapes of different parts of the same pattern. As a result, the skeletons produced by human are usually considered as reference skeletons, which have always been superior to those obtained from thinning algorithms. This fact suggests that such knowledge can be very helpful to the thinning process. One major defect of thinning algorithms is the deformation in the shape of the skeleton genarated at corner and cross regions as shown in Figure 1. This problem arises due to the fact that while generating the skeleton the two pixels p_1 and p_2 are connected as in Figure 2 (a) and not as in Figure 2 (b).

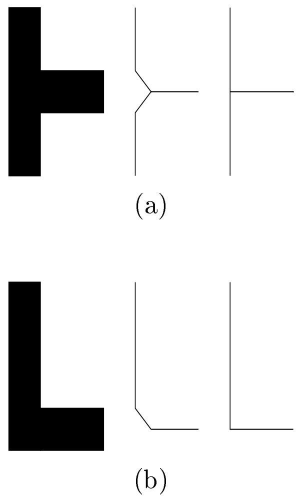


Figure 1. Original, deformed & desired shapes

Another defect is the generation of two pixel wide skeletons for curved regions as shown in Figure 3. This problem arises due to the fact that the pixels shown in Figure 2 (a) are connected as in Figure 2 (b). So it can be seen that the two problems are contradictory to one another and that there should be a compromise

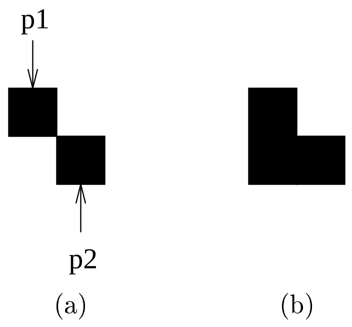


Figure 2. Ways to connect p1 and p2

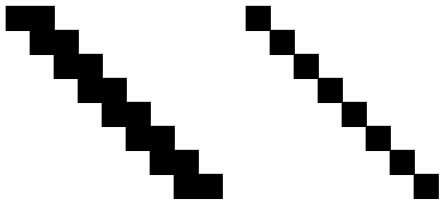


Figure 3. 2-pixel wide & desired skeletons

between the two.

Connectivity problems and thick skeletons experienced by many thinning algorithms often result when the outer layer of pixels of an object is removed and the structure of the resulting object is unknown as far as the current iteration is concerned. In the case of serial algorithms, constraints were introduced to ensure connectivity so that in some instances, a doubly thick skeleton is produced. In the case of parallel algorithms, the solution has been to divide a pass into several subiterations or to obtain information about the neighbours of a pixel.

This paper aims to solve these problems by incoporating a hybrid approach. The approach is to segment the image into straight and curved regions. The straight regions will be thinned by the medial axis method and the curved regions by the contour generation method. After that the skeletons of the various segments will be connected together. The results of experiments on a large variety of patterns show that this technique produces superior results compared to other thinning algorithms and it almost requires the same time as taken by some of the fast methods.

This paper is organized in the following fashion. Section 2 deals with the characteristics of a skeleton. Section 3 deals with a review of the existing thinning algorithms along with some of the problems associated with them. Section 4 describes an efficient the algorithm. Section 5 contains a comparison of the results obtained with other thinning algorithms. Concluding remarks are given in the final section.

2. CHARACTERISTICS OF A SKELETON

This section deals with the general characteristics of a skeleton and some of the terminologies that are frequently used to describe the property of a skeleton. Consider a binary image described by a 2D array of pixels (Figure 4). The object which forms the foreground Q [1] of this image is represented by a set of *dark points* while the background Q' corresponds to a set of *white points*.

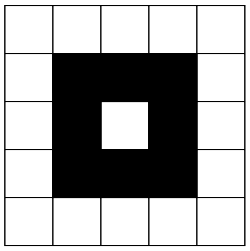


Figure 4. A binary image

For a given pixel p there are eight neighbours n_0, n_1, ... n_7, with the subscript denoting the direction of the neighbour from p, with respect to the x-axis (Figure

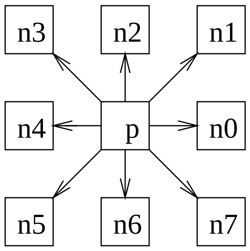


Figure 5. The 8-neighbours of a pixel p

5). Thus for n_i, the direction is $i * 45^o$. The neighbours with even subscripts are known as D-neighbours, which are accessible from p by moving in a horizontal or vertical direction. The other neighbours are called I-neighbours, which are accessible from p by moving along any of the 45^o lines. If p is a dark point and one of its eight neighbours n_i is also dark, p and n_i is said to be 8-connected. On the other hand, if p is dark and one of its four D-neighbours n_{2i} is also dark, p and n_{2i} is said to be 4-connected.

If p_0 and p_m are two dark points that belong to the same object, there exists a path, which can be described as a chain of dark points p_0, p_1, ... p_m, with each consecutive pair of pixels, p_i and p_{i+1} being neighbours of each other. If all eight neighbours are considered, p_0 and p_m are said to be 8-connected. If only D-neighbours are considered, p_0 and p_m are said to be 4-connected.

[1]Q denotes the set of foreground pixels (dark) and its complement denotes the set of background pixels (white)

An object is 8-connected if all pairs of points in the object are 8-connected and 4-connected if all pairs of points in the object are 4-connected. The essential characteristics of a skeleton are as follows:

1. Connectivity should be preserved. If the object is connected, the resulting skeleton should also be connected. In general 8-connectivity should be p-reserved for the foreground, while 4-connectivity should be preserved for the background.

2. Excessive erosion should be prevented. The end points of a skeleton should be detected as soon as possible so that the length of a line or curve that represents a true feature of the object is not shortened excessively.

3. The skeleton should be immune to noises. Noise, or small convexities, which do not belong to a skeleton, will very often result in a tail after thinning. The length of these tails should be minimized.

The preservation of 8-connectivity for the foreground is an important feature of a skeleton. If 4-connectivity was preserved it would result in skeletons two pixel wide and that would go against the properties of an ideal skeleton (single pixel wide). But the preservation of 8-connectivity has also one side effect. The skeletons generated at cross and corner points are distorted especially when the object is very thick.

3. PREVIOUS RESULTS

Thinning algorithms have been studied extensively in the context of pattern recognition and image processing. There exist several thinning algorithms in the literature [[4], [5], [7], [6], [8], [9]]. All of them have their own advantages and disadvantages. The sequential thinning method consists of iterative deletion of the dark points along the edges of a pattern until the pattern is thinned to single-pixel wide line drawing. The edge-points are identified by conducting tests on the eight neighbours of each pixel. The edge-points are deleted in such a way that their deletion *does not*

1. remove end-points (intuitively, they are dark points at the open extremeties of a stroke)

2. break the connectedness of the pattern

3. cause excessive erosion

Different algorithms perform these tests in a differen-t manner. Less memory is required in sequential algorithms. Sequential algorithms examine every pixel in the bitmap to distinguish the foreground from the background. So it can be said that the time complexity depends on the size of the bitmap. The complexity also depends on the operations required for determining the edge-points. These operations are performed in every iteration until the pattern is completely thinned. A significant reduction in time complexity can be achieved by examining only those points that belong to the outline of an object.

In the contour tracing technique the contour describing the edge of an object is traced in every iteration. After the contour has been traced and the sequence of pixels examined to determine whether it is a skeletal pixel or not, the contour is removed. In the next iteration, the new contour is traced and the operation repeats until all the nonsafe points are removed.

Medial axis thinning algorithm [1] is one of the earliest algorithms developed for thinning simple binary patterns. The medial axis transformation of a set S is basically the set of centers and radii of the maximal discs that are contained in S, or equivalently, the set of points of S whose distances to the complement S' are local maxima, together with these distances. It is not hard to see that medial axis points tend to lie midway between opposite borders of S or along the bisectors of angles formed by the borders. Thus these points constitute a kind of skeleton of S. The points thus generated may not be 8-connected. So the connectivity has to be taken care of by looking at the pixel selected in the current row with that of the previous row and if they are not connected an 8-connected path should be constructed to connect the two points. Although this method takes very less time, in most cases the quality of the skeleton obtained is not as superior to those obtained by other known thinning algorithms. It works fine if the regions are straight and does not have too much noise as it is very sensitive to noise.

A thinning algorithm by contour generation [2] is another sequential technique. In this method, a given binary image is first represented by chain codes. A chain code is generated for every closed contour of the object, and the direction of the chain is counter-clockwise for the exterior contour of an object and clockwise for the inner contour of a hole. The contour is also considered as a sequence of a chain of edge points p_0, p_1, p_2,....., p_n, where $p_0 = p_n$. The sequence of pixels is represented by the Freeman code, which is a sequence of directions dir_0, dir_1, ..., dir_{n-1}, pointing to the next pixel in the segment. For 8-connected contours , dir_i is in the range 0 to 7 representing the eight directions shown in Figure 5. As this thinning algorithm is iterative, so after plotting the first contour, the algorithm goes through a number of iterations. In every iteration, the contour describing the edge of an object is traced; meanwhile the edge points are examined against a set of criteria to decide whether the edge point should be removed or not (the pixels are not actually removed but simply marked, this removes the problem of mutual exclusion). The iteration terminates for a particular contour when there are no more unsafe points in that contour. When the operation completes, the skeleton remains.

This is an efficient algorithm as in this method the image is scanned only once to generate the contours for the boundary of the object. After that in every iteration a new contour is generated from the existing contour and the process is repeated until the final skeleton is obtained. But one major problem is that the shape of the skeletons generated are distorted at the corner and cross points. This is due to the fact that 8-connectivity is maintained for the object pixels. If the image is a thin one it gives satisfactory results but as the thickness of the image goes on increasing the shape of the skeleton goes on deteriorating.

Similar to sequential algorithms, parallel algorithms also use the approach of visiting all the pixels in the bitmap to identify the dark points. The dark points are then classified into edge points and non-edge points. Only the edge points are required to be considered. Tests are conducted on each edge point's eight neighbours to determine whether they are break points, end points, or nonsafe points (break points - removal of these points will break the connectivity, end points - points at the end of the contour, nonsafe points - removal of these points will have no effect on the skeleton). The nonsafe points are then removed at the end of the pass. The end and break points are collectively known as safe points and should not be removed. Like sequential algorithm the time complexity of a parallel algorithm consists of three components:

1. In every pass and in every subiteration, every pixel in the bitmap has to examined once to identify the dark pixels. The number of operations is proportional to the area of the bitmap.
2. Every dark pixel has to be examined for edge points. The number of operations is proportional to the area of the objects in every pass.
3. The number of passes is related to the thickness of the object.

But since the steps are performed in parallel the time taken is lesser than the sequential algorithms. Although parallel algorithms are faster than sequential algorithms there are some basic problems associated with them as is the case with any parallel processing architecture. In some parallel algorithms, a 2-pixel wide line will be completely removed because at the beginning of the pass, points on both sides of the line will not break the connectivity of the pattern if they are examined independently. If both sides are examined in parallel using the results from the previous pass, they will be removed simultaneously because the result of removing one side is unknown to the other side during the same pass. This is the problem of mutual exclusion which occurs when several parallel processes share the same memory. In thinning algorithms, when a process examines a certain pixel, it should have exclusive use of that pixel and its eight neighbours. In parallel thinning algorithms this is often not the case.

Evidently the necessity of multiple subiterations in a parallel algorithm and the possibility of a two pixel wide skeleton in contour tracing may be attributed to the problem of mutual exclusion. In parallel algorithm, a pixel is processed on the basis of its previous state so that when pixels are considered in parallel, all the pixels are removed.

When the problem is viewed from another angle, one can see that in both contour tracing and parallel techniques, pixels are removed from the contours without knowledge of what is going to remain in the object. The result is that either all the pixels will have been removed or, to prevent this from happening, a thick curve will remain after the final iteration.

The solution is therefore to consider the results obtained so far for processing the current pixel. If a pixel were to be removed, the new contour which will be exposed to the background can be computed. Thus when the current contour is traversed, a section of the new contour is generated for every pixel in the current contour being visited. The section is checked for break points and this information is available when subsequent pixels in the sequence or those on the next sequence are visited. At the end of the iteration, a new contour will be available for the next iteration without having to remove the old one. At any time, the algorithm will have a complete knowledge of what remains of the object when the current contour is removed.

4. NEW ALGORITHM

The algorithm is a hybrid of the medial axis method and contour generation method to generate the skeleton of the binary image of the object. The algorithm has five steps. In the first step, like any other thinning algorithm, it removes noise from the input image. This helps in avoiding spurious tails and miscellaneous distortions. There exist several noise removal algorithms. We have used the smoothing algorithm [3] which essentially consists of moving a 3x3 window across the binary image and comparing the state of the central pixel of this window with its 8-neighbours to decide whether this state (pixel value) should be retained or modified.

In the next step the image is segmented into horizontal, vertical and curved regions based on the length of each run (a continuous span of dark pixels in a row) and comparing them with the average thickness (average run length of the image). After segmentation, depending on the type of the region the appropriate method is applied to thin the region. The horizontal and vertical regions are thinned using the medial axis method and the curved regions by the contour generation method. At this stage the skeletons for all the regions are separately generated. The skeletons thus generated are connected together to give the final skeleton. Finally, there is a need to do postprocessing to delete some unnecessary points and to keep the skeleton at unit width.

0.1 Terminology

In an image an *edge-point* is a dark pixel that has atleast one white 4-neighbour. An *end-point* is a dark pixel that has atmost one dark 8-neighbour. A *break-point* is a dark pixel, the deletion of which would break the connectedness of the original pattern. A *safe-point* is a dark pixel which is an end-point or break-point but not an edge-point. Removal of a safe point would affect the shape of the skeleton. A *chain* is a linked list of edge pixels defining the boundaries of an object. Each node in the linked list has information about the location of the current pixel(coordinates) and the direction of the next pixel relative to it. The direction is actually a number (0 to 7) depending on which of the eight neighbouring pixels is next in the chain. The chain is counter-clockwise for exterior of an object and clockwise for an interior hole.

0.2 Representation of the Image

The binary image of an object can be represented by horizontal line segments which are coded by run length. Let $f(i,j)$ be the binary image of the object at the ith row and the jth column. Define the kth line segment by the tuple (i, b_k, e_k) where b_k and e_k are the position of the start point and end point of the kth line segment respectively and

$$\begin{aligned} &b_k{=}j, \text{ when } f(i,j-1){=}0 \text{ and } f(i,j){=}1;\\ &e_k{=}j, \text{ when } f(i,j){=}1 \text{ and } f(i,j+1){=}0;\\ &f(i,j){=}1 \text{ for } b_k \leq j \leq e_k \end{aligned}$$

From the set of line segments represented by the tuple (i, b_k, e_k), we can determine the relationship between the current line segment and its neighbours, i.e. the lines above and the lines below it.

0.3 Segmentation

In one top-down scan of the image the start and end of all runs are identified and stored in a separate array. Using the start and end values of a run the run length is computed. The run length of the kth run is given by

$$run\_length_k = e_k - b_k$$

The run lengths are then compared with the threshold (average thickness of the image) to segment the image into curved, horizontal and vertical regions. The threshold is the average value of thickness of the image under consideration. It is computed in the same pass by computing the average value of the run lengths. It is given by

$$threshold = \sum_{k=1}^{n} run\_length_k \div no\_of\_runs$$

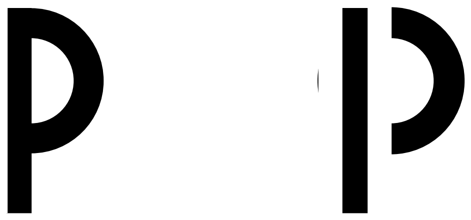


Figure 6. The segmentation of alphabet P

Based on the result of comparison the image is then divided into horizontal, vertical and curved regions. If the run length is greater than the threshold value atleast by a factor of 15 to 20 and the value of the run length remains almost the same along with the start and end of the runs for atleast 5 to 10 rows in continuation then the region is classified as a horizontal one. If the run length is less than or equal to the threshold and the run length, start and end of the runs remains almost the same for atleast 5 to 10 rows in continuation then the region is classified as a vertical one. Remaining parts of the image are classified as curved regions. At the end of this process we are left with horizontal, vertical and curved regions. These regions are identified by the starting row, ending row, starting column, ending column and the region type. The segmentation of the alphabet P is shown in Figure 6.

0.4 Thinning the Regions

After the image is segmented into separate regions, each region is separately thinned using either the medial axis method or the contour generation method. These regions are simply treated as separate objects and are identified by their starting and ending rows, starting and ending columns and type.

The horizontal and vertical regions are thinned using the medial axis method. For the vertical regions thinning is done by simply computing the mid-point of each run from the start and end of the run computed at the starting. For the horizontal regions the same thing is repeated as with vertical regions but in this case instead of calculating the mid-point row wise it is done in a column wise manner. The skeleton obtained is in terms of the pixel locations. The problems associated with medial axis method does not arise as the regions are simple and straight.

The curved regions are thinned using the contour generation method. The entire region is passed to the subroutine which performs thinning by contour generation method. In this method the boundaries of the curved regions are coded in the form of chains. The chains are counter clockwise for outer boundary and clockwise for holes. Each pixel in the chain is tested whether it is a safe point. The pixels which are identified as safe in a particular iteration are marked. Except the safe ones all other pixels are removed from a chain at the end of an iteration and a new chain is constructed. In the process of iteration new chains are generated from the old chains. The process ends when there are no more non-safe points left. The chain that is left out is the skeleton. Since corner and cross points are not thinned by this method the problem of deformation in the shape of the skeleton does not arise.

0.5 Connectivity

After the skeletons are separately generated for each regions, they are connected together to give the complete skeleton. The horizontal and vertical edges are simple to connect as their exact locations are known. The starting point of a skeleton for a vertical region is the mid-point of the starting row and the end point is the mid-point of the end row. For horizontal regions the starting and ending points of the skeleton are the mid-points of starting and ending columns. The method is further eased as the skeletons generated fore these regions are always vertical or horizontal. The regions are connected in the following manner. The entire vertical edge is shifted towards the end of the horizontal edge which is nearest to it as shown in Figure 7. Shifting is not a problem as the skeletons are entirely horizontal or vertical.

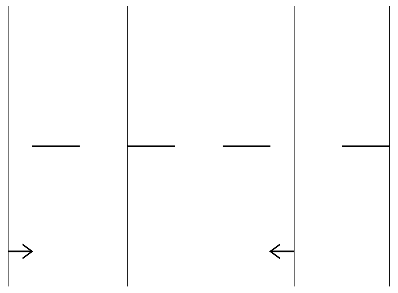


Figure 7. Vertical horizontal connectivity

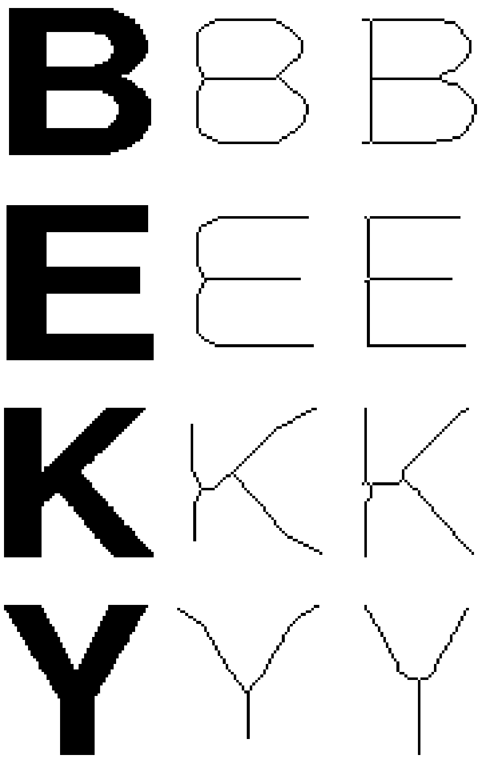
The curved and straight regions are connected together by means of curve fitting between the end points of the curves and straight edges. This is done while thinning the regions separately. A small noise is attached to the curved regions along the directions where they are connected to the straight edges. In doing so the result obtained after thinning the curved regions is such that the end points of the skeletons for the curved regions almost overlaps with the end points of the vertical and horizontal skeletons. There may be a gap of 2 to 3 pixels which are later fixed in the postprocessing step.

0.6 Postprocessing

This step is there to remove some of the unwanted pixels from the final skeleton and also to add some pixels which may not have been taken care of in the connectivity phase. This is done by comparing the skeleton with the original image and adding and deleting some of the pixels.

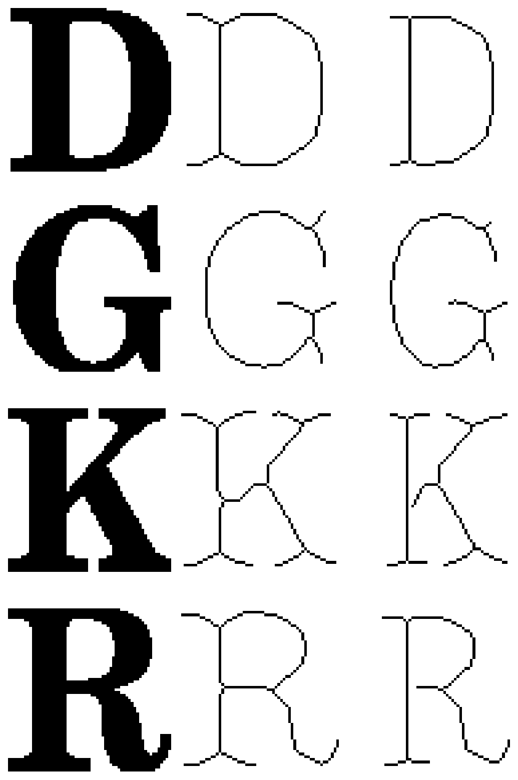
5. COMPARATIVE STUDY

Several experiments were conducted to compare the proposed algorithm and the contour generation algorithm. The test cases are 128x128 greyscale images and consist of different types of fonts. Some results produced by the proposed algorithm are illustrated(Figure 8 and Figure 9). It can be seen that the algorithm produces better results than that produced by the contour generation algorithm, specially at the curved and crossed regions. This is due to the fact that the straight regions are thinned by medial axis method the results for the corner points, T-shaped joints so generated are much superior. Since this gives superior results compared to the conotur generation method, so it can be concluded that this will give better results than SPTA [5], Xu Wang's CGT [7] and other thinning algorithms where it is shown that the contour generation gives better results. Further, the time taken is of similar order to that of the contour generation method.



(a) Original (b) Contour Skeleton (c) New Skeleton

Figure 8. Some Test Results (Helvetica Font)



(a) Original (b) Contour Skeleton (c) New Skeleton

Figure 9. Some Results (Times Roman Font)

CONCLUSION

A new thinning algorithm has been presented in this paper. This algorithm, a hybrid of the medial axis and the contour generation method for thinning, produces a connected skeleton of unit width along the medial axis of the given greyscale image. The time taken is of similar order to that of the contour method which is one of the fastest known thinning algorithms. The skeletons produced are superior to those produced by the contour generation method, SPTA, and many others especially at cross and corner regions for images of varying thickness.

References

[1] S. Wang, A. Rosenfeld, A. Y. Wu, A Medial Axis Transformation for Greyscale pictures. *IEEE trans. PAMI, Jul 1982 Vol PAMI-4 No 4, pp 419- 421*

[2] P. C. K. Kwok, A Thinning algorithm by contour generation. *Communications of the ACM, Nov 1988 Vol 31 No 11, pp 1314-1324*

[3] B. Li, C. Y. Suen, A Knowledge-Based Thinning Algorithm. *Pattern Recognition, 1991 Vol 24 No 12, pp 1211-1221*

[4] Govindan, Shivaprasad, A Pattern Adaptive Thinning Algorithm. *Pattern Recognition, 1987 Vol 20 No 6, pp 623-637*

[5] Naccache, Shinghal, SPTA: A Proposed Algorithm for Thinning Binary Patterns. *IEEE trans. Sys, Man, Cyb. May/Jun 1984 Vol SMC-14 No 3, pp 409-418*

[6] T. Pavlidis, A Thinning Algorithm for Discrete Binary Images. *Comp. Graphics and Image Processing 1980 Vol 13, pp 142-157*

[7] W. Xu, C. Wang, CGT: A Fast Thinning Algorithm Implemented on a Sequential Computer. *IEEE trans. Sys, Man, Cyb. Oct 1987 Vol SMC-17 No 5, pp 847-851*

[8] Stefanelli, Rosenfeld, Some Parallel Thinning Algorithms for Digital Pictures. *Journal ACM Apr 1971 Vol 18 No 2, pp 255-264*

[9] Zhang, Suen, A Fast Parallel Algorithm for Thinning Digital Patterns. *Commun. ACM Mar 1984 Vol 27 No 3, pp 236-239*