A Medial Axis Based Thinning Strategy for Character Images

Soumen Bag, Gaurav Harit

Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur, Kharagpur-721 302, India
{*soumen, gharit*}@*cse.iitkgp.ernet.in*

Abstract

Thinning of character images is a big challenge. Removal of strokes or deformities in thinning is a difficult problem. In this paper, we have proposed a medial axis based thinning strategy used for performing skeletonization of printed and handwritten character images. In this method, we have used shape characteristics of text to get skeleton of nearly same as the true character shape. This approach helps to preserve the local features and true shape of the character images. The proposed algorithm produces one pixel width thin skeleton. As a by-product of our thinning approach, the skeleton also gets segmented into strokes in vector form. Hence further stroke segmentation is not required. Experiment is done on printed English and Bengali characters and we obtain less spurious branches comparing with other thinning methods without any post processing.

1. Introduction

Thinning of shape has a wide range of application in image processing, machine vision, and pattern recognition. But removal of spurious strokes or shape deformation in thinning is a difficult problem. In the past several decades many thinning algorithms have been developed considering all these problems [6]. They are broadly classified into two groups: raster scan based and medial axis based. Raster scan based methods are classified into two other categories: sequential and parallel. Sequential algorithms consider one pixel at a time and visit the pixel by raster scanning [2] or contour following [1]. Parallel thinning algorithms are based on iterative processing and they consider a pixel for removal based on the results of previous iteration only [3, 5, 7, 10, 11]. Many of the raster scan based character thinning methods can not preserve the local properties or features of the character images properly. As a result, they give slightly shape distorted output. Medial

axis based methods generate a central or median line of pattern directly in one pass without examining all the individual pixels [8]. They also give slight distorted result at some local regions. Here we attempt to minimize local distortions by making use of shape characteristics of text. Next we begin a brief description about few prominent raster scan based thinning algorithms.

Datta and Parui [3] have proposed a parallel thinning algorithm which preserves connectivity and produces skeleton of one pixel thickness. Each iteration of the algorithm is divided into four sub-iterations. These sub-iterations use two 1 x 3 and two 3 x 1 templates for removing boundary pixels: east, west, north, and south respectively. The method uses one 3 x 3 window to avoid the removal of critical point (which alters the connectivity) and end point (which shortens a leg of the skeleton).

Leung *et al.* [7] have introduced a contour following thinning method having no sub-iteration. They have used a lookup table to avoid the use of multiple templates for removing boundary pixel. The lookup table has 256 entries and each entry contains three fields (neighbor number, weight number, and connection number). The number of entries depends on the different possibilities of 8-connectivity of a pixel. The determination of the possibility of pixel removal is depends on the values of the patterns of the spatial contour pixels.

To improve the pixel connectivity of thinned character images, Huang *et al.* [5], have introduced a new set of templates (three 4 x 3, one 4 x 4, and three 3 x 4). This algorithm considers all possible patterns of 8-connectivity of a pixel (similar to [7]) and generates template based elimination rule for deleting boundary pixels. The pixel deletion is done based on the number of black pixels in the 8-neighbor connectivity. Additionally, it compensates information loss by integrating the contour and skeleton of pattern. The information loss is detected based on the ratio of the skeleton and contour pixels. If the value is less than a predefined

threshold then the thinned image is replaced by the contour image.

The main contribution of Zhu's method [11] is that it does not consider all pixels equally for performing character thinning. It is based on the substitution of pixels of strokes or curves which are most valuable part for character recognition. This method uses a set of thinning templates, similar to [3], for boundary pixel removal and handling corners or junction points properly. This method has tried to preserve the shape of character images such as Chinese characters, English alphabets, and numerals after thinning. All the above discussed algorithms perform boundary pixel deletion using various thinning templates or values of lookup table. Now we are discussing a medial axis based thinning algorithm.

For performing thinning of regular shape, Martinez-Perez *et al.* [8] have introduced a medial axis based thinning method. This approach does not use thinning templates or lookup table for performing thinning. They have generated medial points in between two parallel contour segments and repeat the procedure for all remaining parallel contour segments. But they have not applied this concept for curve. Finally, we give a brief description of a thinning method which is used for skeleton simplification.

Telea *et al.* [9] have introduced a method to simplify skeleton structure of an image by removing few skeleton branches. The simplification is done by analyzing the quasi-stable points of the Bayesian energy function, parameterized by boundary of contour and internal structure of the skeleton. The experimental results show that it gives multi-scale skeleton at various abstract levels.

All the above mentioned algorithms can not retain the shape of the character images properly. From that point of view, we have addressed a medial axis based thinning method for generating proper skeleton of character images. The algorithm is non iterative and template free. The main advantage is that it uses shape characteristics of text to determine the areas within the image region to stop thinning partially. This approach helps to preserve the local features and true shape of the character. Additionally, it produces a set of vectorized *strokes* with the thin skeleton as by-product. These resultant skeletons have stronger ability to oppose shape deformation and more convenience to feature extraction and classification for OCR.

This paper is organized as follows. Section 2 describes the methodology of the proposed algorithm. Section 3 contains the experimental results and there comparison with other thinning methods. This paper concludes with some remarks on the proposed method and future work in section 4.

2. Medial axis based thinning algorithm

In this section we describe our proposed thinning strategy based on contour extraction and medial axis extraction for text alphabets. The advantages of our proposed technique are:

- It provides a vectorized output of the thin skeleton.

- The vectorized output is a collection of strokes. Hence further stroke segmentation is not required.

- Many of the spurious thinning branches which inevitably occur when applying other rasterized thinning algorithms do not occur in our proposed algorithm.

- Our approach is most suitable for thinning text alphabets, printed or handwritten. Particularly it gives correct output even in the presence of changing width of the strokes.

2.1. Preprocessing and contour extraction

Given a scanned document page we binarize it using the Otsu's algorithm [4]. Currently we are working with documents with all text content, hence we identify the alphabets as connected components in binary images. This works for English alphabets, however, for Bengali or Hindi documents the entire word gets identified as a single connected component because of the *matra/shiro-rekha* which connects the individual characters. For this case we separate out the individual *akshara* within a word by making use of vertical profile, and delineating adjacent *akshara* within a word at the minima points of the profile.

Given an isolated alphabet, its boundary contour is extracted. We detect a boundary pixel on the alphabet using a 3×3 mask centered on every black pixel of the character. If there is even a single background (white) pixel within the mask it would imply that the center black pixel is flagged as a boundary pixel. Starting from this first boundary pixel we traverse the boundary using a connected component aggregation algorithm, i.e. we go on aggregating the black pixels which are adjacent and also on the boundary. This produces a vector representation of the boundary contour. This boundary contour is then segmented into straight line or curved segments which we refer to as *contour strokes*.

2.2. Contour segmentation

In this section we describe our algorithm to convert the vectorized boundary contour into component segments − straight lines or curves. The novelty of this

algorithm lies in its *search*-based identification of contour segmentation points. Given a vectorized contour the candidate segmentation points are identified by analyzing the incremental changes in the orientation as pixels are added at one end of a hypothesized contour segment. The orientation of a contour segment is the slope of the line joining its hypothesized end points. A hypothesized contour segment initiated from a *start point s* is denoted C_s. The other end i.e. the *end point* is allowed to extend to the adjacent contour pixel x_1, then to the next one x_2 in sequence and so on. Our objective is to find a suitable contour boundary pixel for the end point of C_s. For each possible *end point* location e_i we get a hypothesized contour segment $\widehat{se_i}$. The orientation θ_i of this segment is the slope of the line joining the points s and e_i. We analyze the change $\Delta\theta_i = \theta_i - \theta_{i-1}$. For a sequence of hypothesized end point positions $e_i = \ldots, x_{k-1}, x_k, x_{k+1}, \ldots$ we note the trend of $\Delta\theta_i$ being positive, negative, or zero. We form groups of successive pixels for which $\Delta\theta_i$ remains positive, negative, or zero. In other words the pixels forming a group are successive and have a common trend for $\Delta\theta_i$. For each such group, say G_p, which spans pixels, say x_{p1} to x_{p2}, we compute the total change in orientation, $\Delta\theta_{Gp} = \theta_{p2} - \theta_{p1}$. We also note the total number of pixels N_{Gp} in the group G_p.

Since our objective is to identify a candidate end point e for the hypothesized contour segment C_s which attempt to choose a suitable position for e from the available end points of the pixel groups G_p's which have been identified with respect to the start point s of the contour segment C_s.

- If $\Delta\theta_{Gp} > \boldsymbol{\theta}_{\text{threshold}}$ (value is set to 0.7) for any group G_p, then take the end point of C_s as the end point of G_{p-1}. If G_p is the first pixel group from the start point s, then restrict G_p to extend up to only that pixel where $\Delta\theta_{Gp} = \boldsymbol{\theta}_{\text{threshold}}$.

- If $N_{Gp} > \boldsymbol{N}_{\text{threshold}}$ (value is set to 5) for any group G_p, then count the total number of pixels N_s in C_s up to the end pixel of G_{p-1}. If $N_s > \boldsymbol{N}_{\text{threshold}}$ then take the end point of G_{p-1} as the end point of C_s.

In the above formulation G_{p-1} denotes the group just preceding G_p. Once we get the candidate end point for the hypothesized contour segment C_s, we do a refinement of its location on the contour. In other words we search for its most appropriate position in the vicinity of the identified candidate position. This search is based on analyzing local variance of orientations in the forward and backward direction from each contour pixel position in the vicinity of the candidate end point.

The position where both the forward and the backward variances are minimum/low is taken as the most appropriate position of the end point of C_s.

The result of contour segmentation for Fig. 1(a) is shown in Fig. 1(b) where the identified contour strokes are shown in a different color. Each contour stroke corresponds to a straight line stroke or a curved stroke within the character. (Note that our notion of a stroke is different from the one used in handwriting recognition). Given that the character boundary contour has been now
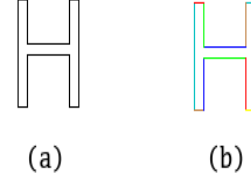
Figure 1. (a) Contour image (b) Contour strokes (see in color)

segmented into contour-strokes, as the next step we get the medial axis for the character.

2.3. Getting the medial axis

We obtain the medial axis using the pixels on the boundary contour partitioned into contour strokes. To obtain the medial axis we need to identify the parallel contour-strokes on the boundary contour. While doing so we incorporate the most obvious text-specific knowledge: the distance between the parallel contour-strokes should be small (since the pen-width is generally small), and all the pixels in-between the two parallel contour-strokes should be black i.e. belong to the character itself and not the background.

Processing starts from any contour-stroke, say C_s. We process the pixels $\{x_1, x_2, \ldots x_{Ns}\}$ on the contour segment C_s.

1. For a pixel x_i on C_s compute the local orientation.

$$\theta_{\text{local}^{x_i}} = \frac{1}{2}\left[\operatorname*{Avg}_{j=1 \text{ to } 5}\theta(x_i, x_{i+j}) + \operatorname*{Avg}_{j=1 \text{ to } 5}\theta(x_i, x_{i-j})\right]$$

where $\theta(x_i, x_{i+j})$ refers to the orientation of the line segment joining pixel x_i to the jth pixel following down the contour (i.e. forward direction), and $\theta(x_i, x_{i-j})$ refers to the orientation of the line segment joining pixel x_i to the jth pixel preceding up the contour (i.e. backward direction). The averaging operation is denoted by Avg.

2. Compute the direction $\perp^{x_i}$ perpendicular to the local orientation $\theta_{\text{local}^{x_i}}$.

3. Starting from pixel x_i on C_s, traverse pixels along the direction of $\perp^{x_i}$ till a border pixel (not on contour C_s) is reached. Let this border pixel be denoted as x_i'.

4. Compute the local orientation $\theta_{\text{local}^{x_i'}}$ at the pixel x_i'.

5. Compute the distance $d_{x_i x_i'}$ between the points x_i and x_i'.

6. If $\theta_{\text{local}^{x_i}}$ is nearly same as $\theta_{\text{local}^{x_i'}}$ ($\left| \theta_{\text{local}^{x_i}} - \theta_{\text{local}^{x_i'}} \right| < 0.45$) and $d_{x_i x_i'} < pen\ width$ (value is set to 12), then mark the mid point of the line joining x_i and x_i' as a pixel on the medial axis. If the two conditions are not satisfied then the medial pixel is *not* marked since this is likely to be the junction point of two or more pen strokes, hence the ambiguity has to be resolved later.

7. Flag off the pixels x_i and x_i' as been processed and move to the pixel x_{i+1} and repeat the above steps.

The above steps are repeated for all contour strokes. For each contour stroke we consider only the pixels which have not been flagged as already processed.

The output of this step is given in Fig. 2. We see from the result that the medial axis of the character has come up as segments. These medial axis segments now need to be extrapolated to join the neighboring segments while ensuring that the extrapolation is within the character region.
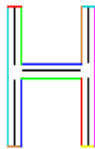
Figure 2. Medial axis segments (see in color)

2.4. Extrapolating thin segments in ambiguous regions

We consider the medial axis segments of a character as forming the set $\mathcal{M} =$ $\{(M_1^s, M_1^e), (M_2^s, M_2^e), ..., (M_m^s, M_m^e), \},$ where M_i^s and M_i^e denote the start and end points of the ith medial axis segment. The start/end points now need to be extended so that the neighboring medial axis segments can be joined together. The steps to identify the neighboring medial axis segments are as follows:

1. For each start/end point of a medial axis segment find the close start/end points belonging to other medial axis segments. For this purpose we consider that the distance between the two start/end points should be less than a threshold and that the line joining the two start/end points should be within the character region.

2. If a given start/end point has two or more start/end points of other segments close enough then we make a proximity set, e.g. $\mathcal{P} = \{M_a^s, M_b^s, M_c^e\}$ would indicate start/end point of three medial axis segments a, b, and c.

Each proximity set would correspond to start/end points which belong to the junction region belonging to multiple pen strokes. Identifying the medial axis points in the junction region of pen strokes give skeleton points which tend to distort the true shape of the character. Our approach avoids identifying the medial axis points in such ambiguous regions and instead tries to extrapolate the start/end points of the medial axis segments in a proximity group such that the result would be very close to the true shape of the character. This is the major contribution of this work.

The steps for extrapolation are as follows:

1. In a given proximity set $\mathcal{P}$ if any two medial axis segments have the same orientation then any one of the two close start/end points is extended to meet the other one.

2. If there is a medial axis segment whose orientation does not match with any other segment in the same proximity set, then its start/end point is extended till it meets the medial axis (extrapolated from other segments) or a junction (formed by extrapolating from other segments in $\mathcal{P}$) on the medial axis or to the start/end point of the other medial axis segment. Of the three cases mentioned which one applies depends on the number of start/end points which are part of a proximity set.

The result of extrapolation is showed in Fig. 3. Note that extrapolation result is almost the true shape of the character and does not produce spurious segments. Our approach gives correct thinning results even for very thick strokes.

Figure 3. Skeleton image

3. Experimental results

At the time of evaluating the performance of thinning algorithm, we need to take care of the following criteria:

- The algorithm must preserves the connectivity.

- The width of resultant skeleton is one pixel.

- The skeleton edges are not shorten.

- The skeleton is very close to medial axis.

- The local properties or features of the character images are preserved after the completion of thinning.

- The thinning result is very close to the true shape of the input image.

We have tested our proposed algorithm on printed English and Bengali character images. All the programs are written in C++ using Opencv 1.0.0 in the UNIX platform. The *pen width* has to be provided so as to be proportional to the average height of the characters.

3.1. Test results of the proposed algorithm

There are two outputs of the algorithm. First one is the set of vectorized medial axis segments and last one is the final skeleton image formed by performing the extrapolation of the medial axis segments. In Fig. 4, there are four type of images: contour of the input image (Fig. 4(a)), contour strokes of different colors (Fig. 4(b)), medial axis segments (Fig. 4(c)), and skeleton image (Fig. 4(d)).

3.2. Comparison of test results with other thinning methods

We have applied Datta's [3], Huang's [5], and Telea's [9] algorithm on printed English and Bengali character images (Fig. 5(a)) and compare the results with our proposed algorithm. We have seen that Datta's

(a)   (b)   (c)   (d)
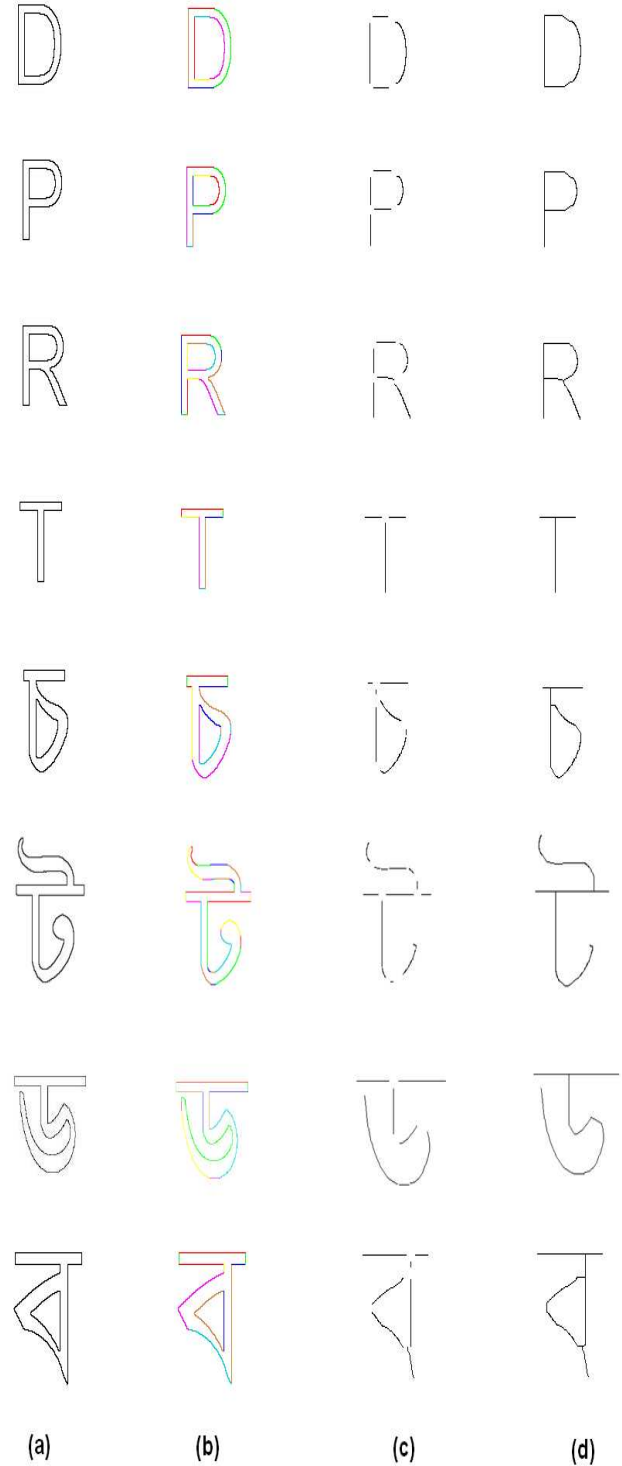
Figure 4. Experimental result (a) Contour image, (b) Contour strokes of different colors (see in color), (c) Medial axis segments, (d) Skeleton image.

Figure 5. Comparison of results of different methods (a) Printed character, (b) Datta's algorithm, (c) Huang's algorithm, (d) Telea's algorithm, (e) Our algorithm.

result (Fig. 5(b)) produces few unwanted strokes which are not acceptable for stroke segmentation and character recognition. Huang's result (Fig. 5(c)) suffers from the distortion at the junction points of the thinned image. Telea's result (Fig. 5(d)) suffers from shortage of skeleton edges. Finally, after comparing the test results of our algorithm (Fig. 5(e)) with all other test results, we have seen that our method gives much better result for thinning character images and maintains all the basic thinning properties as well. The major improvement is that the deformation of the skeletal structure at the junction point of the skeletal branches is not there with our results.

4. Conclusion

This paper improves the performance of existent character thinning algorithms. The main challenge of thinning character images is to preserve the shape of characters after thinning. In spite of slight speed dis-

advantage, our algorithm gives quite promising result. The proposed algorithm avoids shape distortion by detecting the ambiguous regions and avoiding the identification of medial axis points within these regions. The resultant skeleton maintains the pixel connectivity and is very close to the medial axis. Additionally, our algorithm provides *strokes* of character images as an intermediate result during the thinning process. We have compared our test results with three other thinning algorithms and have observed better performance compared to all other algorithms. The proposed thinning approach has a good potential for applications to improve the performance of OCR in Indian script.

References

[1] C. Arcelli. Pattern thinning by contour tracing. *Compuer Graphics and Image Processing*, 17(2):130–144, Oct. 1981.

[2] C. Arcelli and G. S. di Baja. A one-pass two-operation process to detect the skeletal pixels on the 4-distance transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(4):411–414, Apr. 1989.

[3] A. Datta and S. K. Parui. A robust parallel thinning algorithm for binary images. *Pattern Recognition*, 27(9):1181–1192, Apr. 1994.

[4] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*, volume 3rd. Prentice Hall, USA, 2008.

[5] L. Huang, G. Wan, and C. Liu. An improved parallel thinning algorithm. In *Proceedings of the International Conference on Document Analysis and Recognition*, volume 2, pages 780–783, Edinburgh, Scotland, Aug. 2003.

[6] L. Lam, S. W. Lee, and C. Y. Suen. Thinning methodologies—a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(9):869–885, Sept. 1992.

[7] W. Leung, C. M. Ng, and P. C. Yu. Contour following parallel thinning for simple binary images. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics*, volume 3, pages 1650–1655, Nashville, TN, USA, Oct. 2000.

[8] M. P. Martinez-Perez, J. Jimenez, and J. L. Navalon. A thinning algorithm based on contours. *Computer Vision, Graphics, and Image Processing*, 39(2):186–201, Aug. 1987.

[9] A. Telea, C. Sminchisescu, and S. Dickinson. Optimal inference for hierarchical skeleton abstraction. In *Proceedings of the IEEE International Conference on Pattern Recognition*, volume 4, pages 19–22, Cambridge, UK, Aug. 2004.

[10] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communications of the ACM*, 27(3):236–239, Mar. 1984.

[11] X. Zhu and S. Zhang. A shape-adaptive thinning method for binary images. In *Proceedings of the International Conference on Cyberworlds*, pages 721–724, Hangzhou, China, Sept. 2008.