TR-88-008

# A NEW CHINESE CHARACTER THINNING ALGORITHM BASED ON TRACING THE BOUNDARY

# A NEW CHINESE CHARACTER THINNING ALGORITHM BASED ON TRACING THE BOUNDARY

Ching-Sung Lin, Chiou-Feng Wang and Jun S. Huang

Computer Vision Laboratory

Institute of Information Science

Academia Sinica

Taipei, Taiwan, Rep. of China

June, 1988

Correspondence: Jun S. Huang 黃俊雄，中央研究院

Abstract :

Machine printed Ming-font Chinese characters usually have large black areas in the sharp turns of strokes, and most conventional thinning algorithms will produce some spurious branches in these sharp turns. In this paper we have developed a new thinning algorithm based on tracing and stripping the boundary pixels of a character, and this algorithm will in general produce no spurious branches in these sharp turns of strokes. Thus we can use the thinning results to extract the topological features such as end-points, break-points and cross-points, etc. These features will be stable under rotation and noise corruption.

Key words: Image processing, pattern recognition.

## 1. Introduction

If we want to extract the topological features of a character, thinning seems to be a necessary step. This is why many recent papers on character recognition [1-4] use thinning as an essential step. However there are many thinning algorithms [5-8] and each has its own deficiency and imperfection for some cases. In this report we want to do Chinese character thinning and so the domain is limited in some sense. In the beginning we try an algorithm developed by Zhang and Suen [5], which is the simplest one for implementation, but the result is unsatisfactory. The problem comes from that a Ming-font character generally has a large black area in a sharp turn of a stroke, and this will produce a spurious branch and thus produce an extra endpoint and also an extra forkpoint. Later on we try another algorithm developed by Chu and Suen[6], where they have made several comparisons with other typical thinning algorithms and have shown that their method is better than others for characters. Unfortunately we find that in their paper the description of the algorithm contains a lot of errors and in some important steps the description is unclear. Finally we have to develop our own algorithm.

The idea of stripping the boundary pixels in Chu and Suen's paper is nice and we follow this idea. We first fill the holes and define some 3 x 3 window patterns as one-pixels

wart, corner point, break point, end point, crosss point, trace point and base point(see section 2). Then we trace the boundary of a character and strip the boundary pixels by checking each pixels's 3 x 3 window patterns. After the end of stripping we delete those too short branchs. Experiments show that the thinning results for Ming-font characters are satisfactory, and in general will produce no spurious branches at the sharp turns of strokes.

## 2. The New Thinning Algorithm

This algorithm consists of two parts used to obtain the skeletons of Chinese characters. The first part includes following three steps : (1) fill-delete (2) label (3) strip. The three steps are iterated for several times until the pattern cannot be thinned anymore. Thus we can obtain the rough sketelon of the character. The second part modifies the skeleton by deleting the error branches caused by thinning.

The following definitions are used to represent special figures of points and their 8-connected neighbors.

A. one-pixel hole

```
X 1 X
1 0 1   X represents don't care condition
X 1 X
```

B. one-pixel wart

```
1 0 0   0 0 0   0 0 1   1 1 1
1 1 0   0 1 0   0 1 1   0 1 0
1 0 0   1 1 1   0 0 1   0 0 0
```

C. coner point

```
1 1 0    0 0 0    0 0 0    0 1 1
1 1 0    1 1 0    0 1 1    0 1 1
0 0 0    1 1 0    0 1 1    0 0 0
```

D. break point

```
A A A    1 0 A    A 0 1    A A A
0 1 A    0 1 A    A 1 0    A 1 0   at least one "A" is "1"
1 0 A    A A A    A A A    A 0 1

B B B    B 0 C
0 1 0    B 1 C   at least one "B" is "1",
C C C    B 0 C   at least one "C" is "1"
```

E. end point

```
1 0 0    0 1 0    0 0 1    0 0 0
0 1 0    0 1 0    0 1 0    0 1 1
0 0 0    0 0 0    0 0 0    0 0 0

0 0 0    0 0 0    0 0 0    0 0 0
0 1 0    0 1 0    0 1 0    1 1 0
0 0 1    0 1 0    1 0 0    0 0 0
```

F. cross point

```
0 1 0
1 1 1
0 1 0
```

G. trace point

```
X A X
A 1 A   X represents don't care condition,
X A X         at least one "A" is "0"
```

H. bare point - when the trace point is stripped, its nonzero 4-connected neighbors are called bare points. e.g.,

```
X 0 X   0 0 X
1 0 1   0 0 1    these 1's are bare points
X 1 X   X 1 X
```

G. non-deleting point - includes end points and break points

I. contour point - during the stipping process,

contour points consist of trace points,

bare points and non-deleting points.

This thinning algorithm consists of following three steps which are done repeatedly until the pattern is reduced to skeleton.

A. fill-delete

This step smooths the pattern by (1) filling one-pixel holes (2) deleting one-pixel warts and (3) deleting corner points. e.g.,

```
(1)  X 1 X          X 1 X
     1 0 1  ---->   1 1 1
     X 1 X          X 1 X

(2)  1 0 0          1 0 0          0 0 0          0 0 0
     1 1 0  ---->   1 0 0          0 1 0  ---->   0 0 0
     1 0 0          1 0 0          1 1 1          1 1 1

(3)  1 1 0          1 1 0          0 0 0          0 0 0
     1 1 0  ---->   1 0 0          1 1 0  ---->   1 0 0
     0 0 0          0 0 0          1 1 0          1 1 0
```

B. label

This step label the outer and inner contour points to be the trace points "T". e.g.,

```
1 0 0      1 0 0      0 0 0      0 0 0
1 1 1 ---> 1 T 1      0 1 1 ---> 0 T 1
1 1 1      1 1 1      1 1 1      1 1 1
```

If all points remained are trace points except cross points, then, after stripping,the final pattern is just the skeleton.

C. strip

This step is more complicated in order to strip the trace points along the outer and inner contours. At

first we define the 8-connected neighbors of a point, and the scanning sequences as the following :

```
H A B
G X C    X is the concerning point
F E D
```

Table 1 Start scanning sequence

| Type of scanning | Start scanning sequence |
|---|---|
| Outer clockwise scanning | C D E F G H A B |
| Inner clockwise scanning | E F G H A B C D |
| Outer counter clockwise scanning | G F E D C B A H |

Table 2 Next scanning sequence

| Position of X in previous window | Clockwise scanning | Counter clockwise scanning |
|---|---|---|
| A | F G H A B C D E | D C B A H G F E |
| B | G H A B C D E F | E D C B A H G F |
| C | H A B C D E F G | F E D C B A H G |
| D | A B C D E F G H | G F E D C B A H |
| E | B C D E F G H A | H G F E D C B A |
| F | C D E F G H A B | A H G F E D C B |
| G | D E F G H A B C | B A H G F E D C |
| H | E F G H A B C D | C B A H G F E D |

Now we begin to strip the pattern by following processes :

(1) Search the first trace point by system scanning. If there is no trace point remained, return to fill-delete pass.

(2) Look for the neighboring contour point by the outer clockwise start scanning sequence. If the first nonzero point we found is contour point, then go to (6).

(3) Look for the neighboring contour point by the inner clockwise start scanning sequence. If the first nonzero

point we found is contour point, then go to (6).

(4) Look for the neighboring contour point by the outer counter clockwise start scanning sequence. If the first nonzero point we found is contour point, then go to (7)

(5) If we cannot find the contour point by these sequences, strip the point concerning.

(6) Use clockwise next scanning sequence to find next contour point and check it to see :

(a) If it is the starting trace point and is not non-deleting point, then delete it and return to (1). Whenever we delete a trace point, we mark its nonzero 4-neighbors with "B" as bare points.

(b) If it is non-deleting point, mark it with "N", otherwise, delete it and mark its bare points. Search the neighboring contour point by the next scanning sequence. If the first nonzero point we found is "1", reset the starting scanning neighbor and go to (7), otherwise return to (6)-(a).

(7) The same as (6) except that we use counter clockwise next scanning sequence and we will always find contour point by counter clockwise scanning.

3. Modifying

There may be some error branches occurred during the thinning process. We will delete them according to their length by the following steps.

A. Find all end-points.

B. Trace the branches from end-points and check the following conditions :

(1) If the point is neither fork point nor end point, go on tracing until the length exceeds the threshold value, then trace next branch.

(2) When meet another end-point within threshold value, then trace next branch.

(3) When meet fork-point within threshold value, delete the whole branch, then trace next branch.

The modifying process is finished when all end points have been traced.

4. Experimental Results

Some experimental results for Ming-font characters are shown in Fig. 1, where at the sharp turns of strokes there are no spurious branches. But Zhang and Suen's algorithm will produce spurious branches as shown in Fig. 2. This shows that our new algorithm is far better than Zhang and Suen's and is suitable for Chinese characters.

REFERENCES


1. Hideo Ogawa and Keui Taniguchi, Thinning and stroke segmention for handwritten Chinese character recognition. Pattern Recognition, Vol. 15, 299-308, 1982.
2. Louisa Lam and C. Y. Suen, Structural classification and relaxation matching of totally unconstrained handwritten Zip-code numbers. Pattern Recognition, Vol. 21, 19-31, 1988.
3. F. H. Cheng and W. H. Hsu, Fuzzy approch to solve the recognition problem of handwritten Chinese characters. Pattern Recognition (to appear in 1988)
4. F. H. Cheng and W. H. Hsu, Solving the recognition problem of handwritten chinese character by stroke's information. The Transaction of the IEICE (to appear in 1988, Japan)
5. T. Y. Zhang and C. Y. Suen, A fast parallel algorithm for thinning digital patterns. Comm. of ACM, 1984.
6. Y. K. Chu and C. Y. Suen, An alternate smoothing and stripping algorithm for thinning digital binary patterns. Signal Processing 11, 207-222, 1986.
7. Y. S. Chen and W. H. Hsu, A modified fast parallel algorithm for thinning digital patterns. Pattern Recognition Letters 7, 99-106, 1988.
8. J. S. Huang and W. J. Lee, A new thinning algorithm for removing noise-spurs and retaining end-points, Information Science and Engineering 1, 45-57, 1985.(Taiwan)

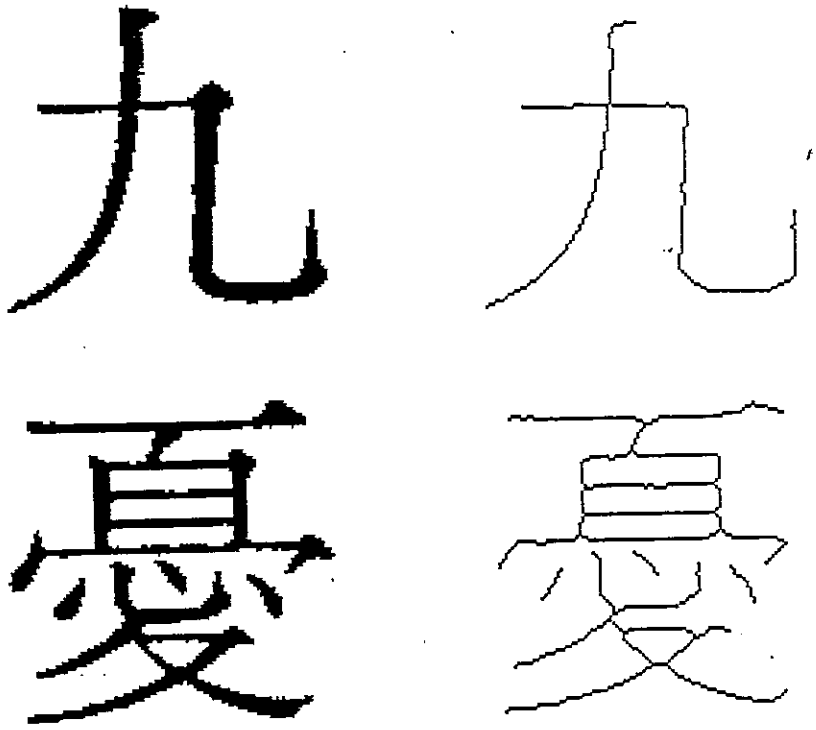

Fig. 1 Some thinning results based on our new algorithm.

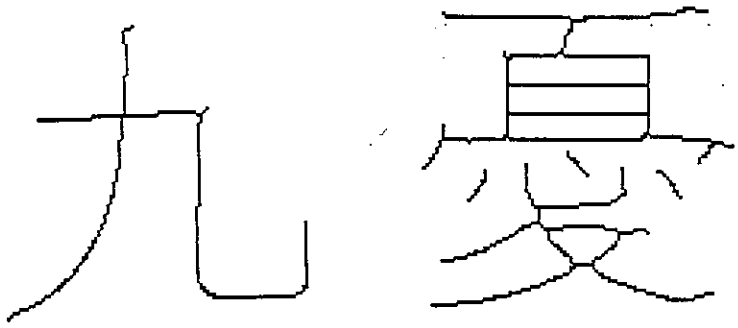

Fig. 2 Some thinning results ( from Fig. 1) based on Zhang and Suen's algorithm.