A Generic Thinning Algorithm with Better Performance

S. Ahmed, M. Sharmin and Chowdhury Mofizur Rahman
Department of Computer Science and Engineering,
Bangladesh University of Engineering and Technology, Dhaka, Bangladesh
Emails: ashameem@bdonine.com, nisha_moushumi@yahoo.com, **cmr@cse.buet.ac.bd**

Abstract: *This paper reports on a generic sequential thinning algorithm to thin both Bangla and English characters and numerals. In this algorithm the thinned image is obtained by making the whole image uniform at first and then thinning it by deleting the unnecessary pixels by a new approach. To avoid distortion of the original character after thinning, a special technique is adopted here. This algorithm ensures that the basic characteristics of thinning are maintained and also the algorithm remains as simple as possible. This algorithm is effective, fast and simple. The effectiveness of this algorithm is examined both theoretically and experimentally.*

Keywords: Sequential Thinning Algorithm, MakeTwoTone, Uniforming process, Zoom-in process, ThinImage, Line of Points, Skew ness, GrandUniforming process.

1. INTRODUCTION

A wide range of image analysis operations require thinning as a preprocessing step. The huge practical implication that thinning has in various fields enhances its importance and requires faster and simpler algorithms to design. Thinned images are used in the Optical Character Recognition [1], Fingerprint classification [2], printed circuit boards [3] etc.

Different algorithms use different approaches to solve the problem of thinning. The algorithms are broadly divided into two categories: Iterative & Non-Iterative. The Iterative algorithms are divided into two subparts: Sequential and Parallel. Sequential algorithms consider all the pixels to take decision for a single pixel. On the other hand, pixels can be independently judged for decision making in Parallel algorithms [4], [5]. The proposed algorithm belongs to the Sequential group.

The organization of this paper is as follows. Section 2 contains a brief comparison of the existing Sequential algorithms and the proposed algorithm. Section 3 discusses the detailed approach of thinning followed in this algorithm. Experimental results are discussed in section 4. Concluding remarks are contained in section 5.

2. THINNING STRATEGY: PAST AND PRESENT

By using conventional algorithms, different experiments are carried out. The most Conventional algorithms [4], [5] etc. are designed for English, an international language or for some other very renowned languages. Since English character's shape is very simple and those algorithms are only dedicated for that language, those will not be applicable for Bangla Language, a very complicated language full of different features.

Some special and complicated features make Bangla characters different from the characters of other languages such as *matra* (the upper zone of many Bangla characters leads to the connectivity of Bangla words, different kind of bending such as bha ('ভ'), chha ('ছ') etc, different variations of loops such as po ('প'), sha ('শ'), various patterns of spiral structures such as umo ('ঊ'), khondoto ('ৎ') etc. So special attentions are needed for different features and this requires extra effort to apply thinning algorithm on Bangla characters. Some researches have been made in those arenas [6], [7], [8].

One of the important limitations of most conventional algorithms for Bangla character thinning is that they produce disconnected thinned character. Not only that, deformation and skewed versions are also being produced. Moreover *matra* is distorted in some algorithms and results in a deformed character. Finally there are some algorithms that seems fine but very tough to be implemented. Some algorithms may have solved all the previous problems but can't reform a character if the character is skewed.

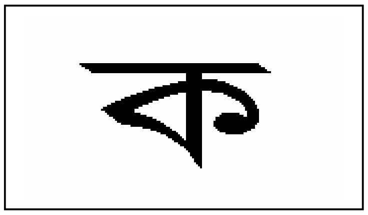
The proposed algorithm is free from any objections described above and shows a way to get perfect Bangla thinned characters comfortably.

Finally the proposed algorithm is unique and generic in the sense that it can almost perfectly thin both Bangla and English characters and numerals perfectly.

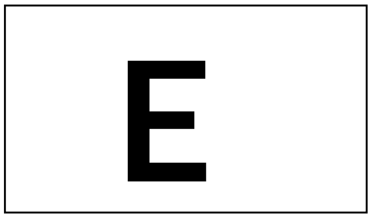
3. THINNING PROCESS

Thinning can be defined as the act of identifying those pixels belonging to an object that are essential for identifying the object's shape.

1) MakeTwoTone process: It is the first process of the thinning procedure. It stores the whole scanned image (see Figure 1) to two dimensional array of cells containing only white and black colored pixels. White (0) represents background and black (1) represents foreground color (see Figure 2).

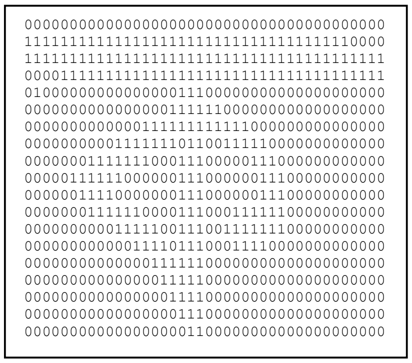


(a)



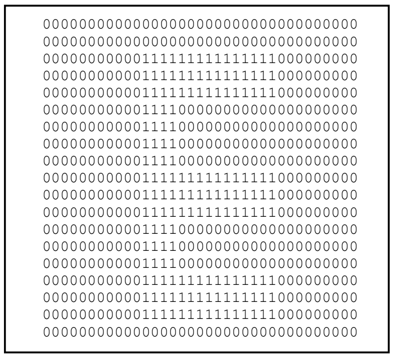
(b)

Figure 1. Before applying MakeTwoTone Process on
(a) a Bangla Character
(b) an English character



```
0000000000000000000000000000000000000000
1111111111111111111111111111111111100000
1111111111111111111111111111111111111111
0000111111111111111111111111111111111111
0100000000000000011100000000000000000000
0000000000000000111111000000000000000000
0000000000000111111111111000000000000000
0000000000111111101100111110000000000000
0000000111111100011100000111000000000000
0000011111100000011100000011100000000000
0000001111000000011100000011100000000000
0000000111111000011100011111100000000000
0000000000111110011100111111100000000000
0000000000001111011100011110000000000000
0000000000000011111100000000000000000000
0000000000000001111100000000000000000000
0000000000000000111100000000000000000000
0000000000000000011100000000000000000000
0000000000000000001100000000000000000000
```

(a)



```
00000000000000000000000000000000000
00000000000000000000000000000000000
00000000000111111111111111000000000
00000000000111111111111111000000000
00000000000111111111111111000000000
00000000000111100000000000000000000
00000000000111100000000000000000000
00000000000111100000000000000000000
00000000000111100000000000000000000
00000000000111111111111111000000000
00000000000111111111111111000000000
00000000000111111111111111000000000
00000000000111100000000000000000000
00000000000111100000000000000000000
00000000000111100000000000000000000
00000000000111111111111111000000000
00000000000111111111111111000000000
00000000000111111111111111000000000
00000000000000000000000000000000000
```

(b)

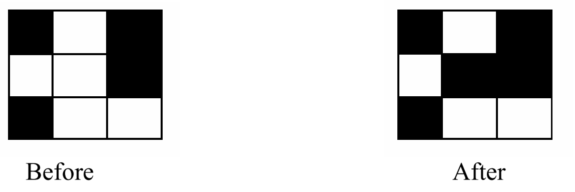
Figure 2. After applying MakeTwoTone process
(a) On a Bangla character
(b) On an English character

2) *Uniforming process*: Next process for thinning is Uniforming process. It makes the image array in such a way that if any position's color is white and at least (threshold+1) positions of this position's surroundings are black colored, then it makes the white colored pixel black. It does nothing if the considered position's color is black (see Figure 3).

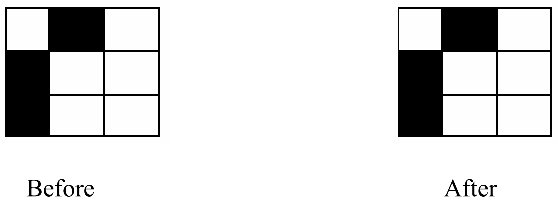
3) *Zoom-in process*: Now to get the zoomed image, zoom-in process is applied. This process checks whether the black color positions are actually eligible to be black or not. For that purpose, if more than threshold color positions are black in the surroundings of considered black color position, then it leaves it black; otherwise it makes the black pixel white (see Figure 4). After applying zoom-in process, the image array would not be the actual image but it will represent the actual image array and would be the zoomed version of the image array.

The previous two processes would be performed on the two tone image array for several times. But the Uniforming process would be performed more than the zoom-in process performed in order to get the uniformity of the image which would be necessary for the next processes.

3) *ThinImage process*: The ThinImage process is applied to the image array which would be described later. This thinning process can make the whole image file skewed (see Figure 5).

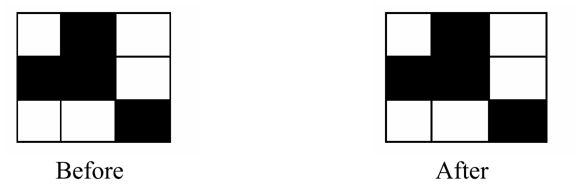


(a)

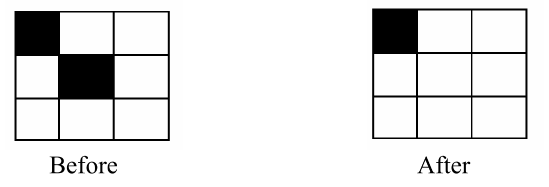


(b)

Figure 3. Uniforming process (threshold=3)
(a) Eligible to be black from white
(b) Not eligible to be black from white

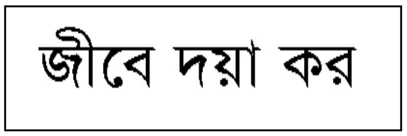


(a)

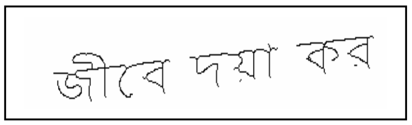


(b)

Figure 4. Zoom-in process (threshold=3)
(a) Eligible to be remaining black
(b) Not eligible to be remaining black

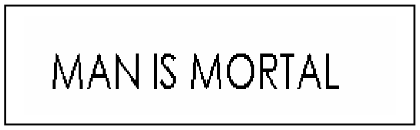


Before skewing

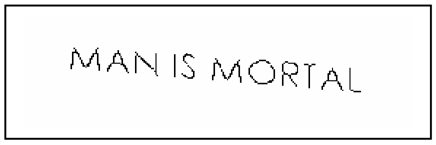


After skewing

(a)



Before skewing



After skewing

(b)

Figure 5. Skewing an Image File on
(a) Bangla characters
(b) English characters

4) *Determine Skew Angle*: To resolve this skew ness problem, it needs to detect the angle by which the image has been skewed and the process named Determine Skew Angle serves that purpose. This process at first determines each line existing in the thinned and zoomed image array which is called Line of Points. Then it determines the angles between each Line of Points and corresponding horizontal line. Average of those angles is the required skew angle. "Equation (1)" reflects the above process.

$$\Psi(x) = \sum_{i=1}^{n} \frac{\Psi(x(i))}{n}, n \geq t\,. \qquad (1)$$

here,

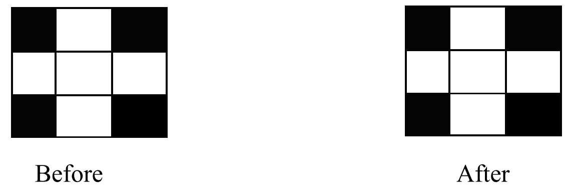
$\psi(x)$	=	angle to rotate
x(i)	=	i-th Line of Points
n	=	Total Number of Line of Points
t	=	Threshold value of Number of Line of Points

Angle between each Line of Points and corresponding horizontal line, $\Psi(x(i))$ can be detected by applying "(2)" considering first pixel (x1, y1) and last pixel (x2, y2) for each Line of Points.

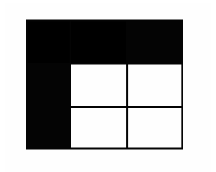
$$\tan(\Psi(x(i))) = \frac{y1 - y2}{x1 - x2}\,. \qquad (2)$$

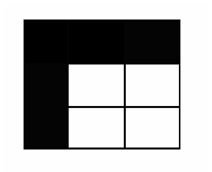
5) *Skew ness correction*: Now in order to get the actual thinned image, previously described MakeTwoTone process would be done again on the given actual image file. Then if the determined skew angle is not so small, the whole two tone image array would be rotated by - ψ(x) to correct skew ness problem.

6) *GrandUniforming*: Then GrandUniforming process would be applied. Actually it is almost similar to the previously described Uniforming process. But an additional constraint is adopted in the later one and it is to check whether there is more than 1 white-black combination around the considered white pixel or not. If number of white-black color combination is more than 1 then the considered white colored pixel would be black (see Figure 6(a)). On the other side, if number of white-black color combination is not more than 1 then the considered white colored pixel would not be eligible to be black although it has black colored pixel around it more than the threshold value (see Figure 6(b)).



(a)





Before After

(b)

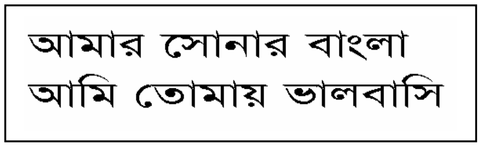
Figure 6. GrandUniforming process (threshold=3)
(a) Eligible to be black
(b) Not eligible to be black

7) *Final ThinImage process*: Lastly the previously depicted ThinImage process is applied. This process actually checks whether any black pixel can be removed (made to be white) safely or not. At first it checks the number of black pixels around the black pixel under consideration. If this number is less than 2 then it is not eligible to be removed. But for more than 2 numbers, it considers the number of white-black color combinations around the considered pixel. If this number is not equal to 1 then it does nothing. But if that number is 1, it performs one of two types of checking named "first type checking" and "Second type checking". If (i,j) is considered black pixel (see Figure 7) then according to the "first type checking", it would be white if (i,j+1) or (i+1, j) or both (i-1,j) and (i,j-1) are white and according to the "second type checking", it would be white if (i-1,j) or (i,j-1) or both (i, j+1) and (i+1,j) are white.

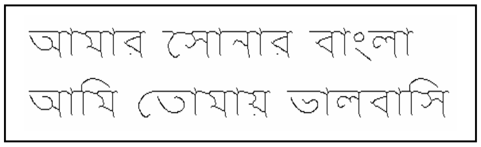
Finally the image becomes thinned image (see Figure 8).

i-1, j-1	i-1, j	i-1, j+1
i, j-1	i, j	i, j+1
i+1, j-1	i+1, j	i+1, j+1

Figure 7. 3X3 Window frame of considered pixel and its surroundings 8 pixels



Before thinning

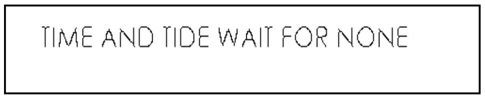


After thinning

(a)

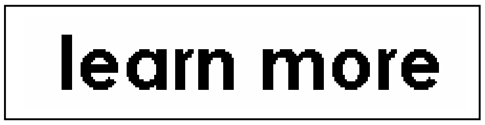
TIME AND TIDE WAIT FOR NONE

Before thinning

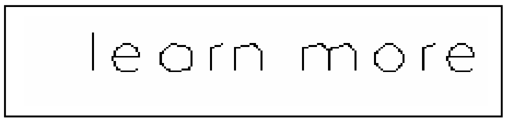


After thinning

(b)

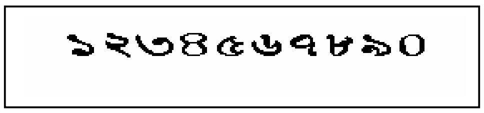


Before thinning

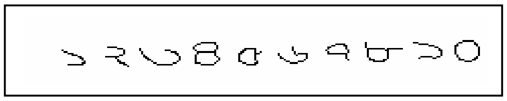


After thinning

(c)

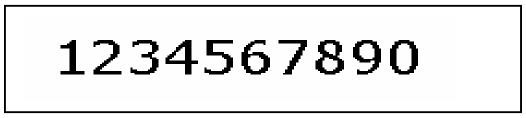


Before thinning

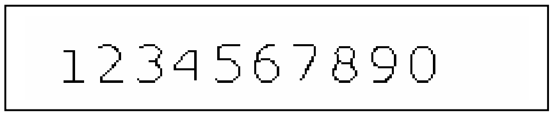


After thinning

(d)



Before thinning



After thinning

(e)

Figure 8. Thinning process on
(a) Bangla Characters
(b) English Capital letters
(c) English small letters
(d) Bangla numerals
(e) English numerals

So the whole thinning algorithm is as follows.

```
1.  Algorithm Thinning
2.  Apply MakeTwoTone process on image format
3.  for i =1 to 3 do
4.  {
5.          for i =1 to 7 do
6.          {
7.                  apply Uniforming  process
8.          }
9.          apply zoom-in process
10. }
11. apply ThinImage process for 3 times
12. Find  every Line of Points
13. Determine Skew angle ψ
14. Again Apply MakeTwoTone process
15. rotate whole image array by (- ψ)
16. Apply GrandUniforming process
17. apply ThinImage process for 20 times
```

3. EXPERIMENTAL RESULT

The proposed algorithm is tested for effectiveness and for that purpose several experiments were conducted on the proposed algorithm and other renowned algorithms applicable for Bangla and English character thinning. First of all Bangla characters and numerals will be considered.

When the proposed algorithm is compared with the existing algorithms from the distortion point of view for Bangla characters, then it is found that the distortion of Hilditch[7] was highest, SPTA[6] produces characters with medium distortion, the FAST[8] algorithm produced nearly undistorted characters and the proposed algorithm produced characters almost without distortion (see Figure 9).

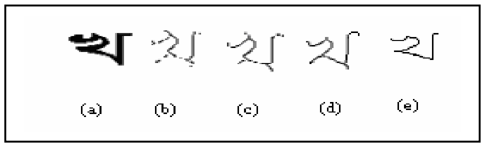


Figure 9. Comparison of algorithms according to distortion point of view
(a) Original Bangla character (kha)
(b) Hilditch algorithm
(c) SPTA algorithm
(d) FAST algorithm
(e) Proposed algorithm

From the point of view of *matra* distortion, SPTA[6] and Hilditch[7] sometimes make the *matra* distorted. FAST[8] keeps the matra but sometimes not in proper order. But proposed algorithm can keep the *matra* accurately (see Figure 10).

If the algorithms are compared from connectivity point of view of Bangla characters, SPTA[6], FAST[8] algorithm and the proposed algorithm produced connected characters but Hilditch[7] sometimes produced disconnected characters(see Figure 11).

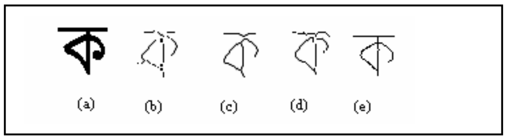


Figure 10. Comparison of algorithms according to matra distortion point of view
(a) Original Bangla character (ka)
(b) Hilditch algorithm
(c) SPTA algorithm
(d) FAST algorithm
(e) Proposed algorithm

.

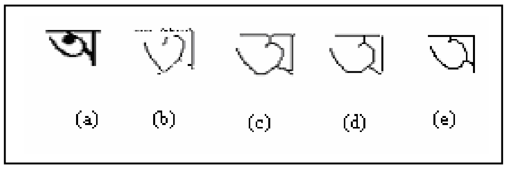


Figure 11. Comparison of algorithms according to connectivity point of view
(a) Original Bangla character
(b) Hilditch algorithm
(c) SPTA algorithm
(d) FAST algorithm
(e) Proposed algorithm

From the implementation point of view, SPTA[6] is very complex. Programming complexity of Hilditch[7] is of medium order. FAST[8] algorithm is easy to understand. The proposed algorithm works on a very simple algorithm and is implemented using very simple logic. All of the comparisons are shown in Table 1 and in Table 2.

A simple inspection of the nested loop structure of thinning algorithm yields a running time of $O(n^2)$ for the algorithm. The loops are nested two deep and each loop index takes on at most n (the number of rows or columns of the image array) values.

Actual cost of the process in cps (characters per second) is found experimentally and this is shown in Table 3.

Now if English characters are considered, it can said that from point of view of correctness, simplicity, Time and Space Complexity, the proposed algorithms is comparable with standard renowned algorithms.

Table 1. Comparison of different thinning algorithms according to curvature property of character

Algorithm	Distortion Produced	*Matra* distortion	Connectivity
Hilditch	Highest	High	Disconnected
SPTA	Medium	Medium	Connected
FAST	Low	Low	Connected
Proposed Algorithm	Lowest	Lowest	Connected

Table 2. Comparison of different thinning algorithms according to implementation and applicability

Algorithm	Implementation Complexity	Applicability
SPTA	Very Complex	English & Bangla
Hilditch	Medium Complex	English & Bangla
FAST	Simple	Bangla
Proposed Algorithm	Simple	English & Bangla

Table 3. Actual cost of the proposed algorithm in cps (characters per second)

Font size	Thinning cost in CPS
8	867
10	650
12	520
14	485

4. CONCLUSION

In case of thinning a Bangla character, from different points of view such as “perfect connectivity maintenance within different characters”, “smooth thinning”, “Time and Space Complexity”, “Perfect thinned formation from normal images” etc, it can be said that the proposed algorithm gives more accurate result in more cases than any other existing algorithms not only theoretically but also experimentally. Every possible aspects of thinning are taken into account and the special features such as *matra* are handled with great care. On the other hand, this algorithm can easily thin English characters and numerals almost without any distortion. The algorithm is efficient in performing all the checks and at the same time managing the algorithm to remain easy to understand and implement. The time complexity of this algorithm is also not high. So, it can be claimed that the proposed algorithm is a generic efficient thinning algorithm for both Bangle and English characters and numerals.

REFERENCES

[1] B. B. Chowdhury and U.Pal, “A complete Printed Bangla OCR System”, vol. 31(5), pp. 531-549, 1998.

[2] B. Moayer and K.S.Fu, *A syntactic approach to fingerprint pattern recognition*, vol. 7, Pattern Recognition, 1975, pp. 1-23.

[3] Q. Z. Ye and P.E Daielsson, *Inspection of printed circuit boards by connecting preserving shrinking*, vol. 10, no. 5, IEEE Trans. Pattern Anal. Mach. Intell, 1988, pp. 737-742.

[4] V. Ubeda, *A Parallel Thinning Algorithm Using KxK Masks*, JPRAI(7) , 1993, pp. 1183-1202.

[5] Z. hang and Y.Y. Wang. *A New Parallel Thinning Methodology*, IJPRAI(8), 1994, pp. 999-1011.

[6] N. J. Naccache and R. Shinghal, *SPTA: A proposed Algorithm for thinning binary patterns*, vol. SMC-14, no.3, IEEE trans. Syst. Man and Cybern, , 1984, pp. 409-418

[7] N. J. Naccache and R. Shinghal, *An investigation into the skeletonization approach of Hilditch*, vol. 17(3) , Pattern Recognition, 1986, pp.279-284.

[8] D. Akhter and M. M. Ali, “A Fast Thinning Algorithm for Bangla Characters”, *Proc.ICCIT 1998*, pp. 132-136, Dhala, Bangladesh, 1998