Arab Gulf Journal of Scientific Research, 22 (1), pp 28 - 33 (2004)

ORIGIONAL PAPER

Mohamed Abdulsalaam Ali and Kasmiran Bin Jumari

Skeletonization Algorithm for Arabic Handwriting

Abstract: In this paper, we propose a thinning algorithm for Arabic handwriting using color coding for both thinning and gap recovery in the final output skeleton. This algorithm is designed so that it accepts unconstrained Arabic handwriting. Different colors have been given to different pixels of interest on the original image in the beginning and during the process of skeletonization. Color coding gives good optimization and demonstration and yields an efficient skeletonization. The algorithm preserves very well the shape of the original image and yields a skeleton that can be effectively incorporated in an Arabic OCR system

Keywords: Character recognition, image processing, thinning algorithm, color coding, skeletonization, connectivity preservation and Arabic handwriting

خوارزمية تنحيف للنصوص العربية المكتوبة بخط اليد

محمد عبدالسلام علي و كسمران بن جوماري

المستخلص: تهدف الورقة الى تقديم خوارزمية معدلة لتنحيف خط اليد العربي. وقد تم إستخدام تشفير الألوان، في حالتي التنحيف وملىء الفراغات في الصورة الهيكلية الأخيرة . وتم تصميم هذة الخوارزمية ليكون مدخلها هو ملف صورة لنص عربي مكتوب بخط اليد، على ورقة. ثم يتم إختيار عدد من الألوان لتمثيل عدد من النقيطات ذات الإهتمام، والموجودة على ملف الصورة في بداية وخلال عملية التنحيف. وتمنح تقنية التشفير بالألوان ذات الجودة العالية والإيضاح ، وتساعد وبشكل اسرع في إنتاج صورة هيكلية عالية الكفاءة. وتضمن الخوارزمية الحفاظ على الشكل العام للنص الأصلي المكتوب بخط اليد، وتنتج صورة هيكلية يمكن إستخدامها بكفاءة في منظومات التعرف على خط اليد العربي.

كلمات مدخلية: خوارزمية، خط اليد العربي، تنحيف، تشفير الألوان، مسودة هيكلية، منظومات، التعرف.

Introduction

Character recognition is a field of pattern recognition that has been subjected to considerable work during the past three decades (Mohamed, AA, 2002). Although the designing of thinning algorithms has been an important research area, only a few researchers have considered the thinning of Arabic writing (Altuwaijiri, MM, 1998). Thinning plays a major role in the OCR system, and since recognition is dependent in part on the effectiveness of the thinning algorithm, attention is given in this paper to the development of an effective thinning algorithm for the purpose of developing an Arabic OCR.

The thinning algorithms have been studied extensively in regard to image processing and pattern recognition (Altuwaijiri, MM, 1998, Flores, E, 1995, Sabri, AM,1991, El-Desouky, AI, 1992, Gonzalez, RC, 1987 and Mori, SH, 1999). Skeletonization has been effectively proven in a wide range of image processing usages, for instance character recognition, fingerprint recognition, inspection of printed circuit boards and chromosome shape analysis (Altuwaijiri, MM, 1998). In general, an effective skeletonization algorithm should ideally remove all redundant pixels and retain the significant aspects of the pattern under process. In addition, a good algorithm should fulfill certain requirements, namely:

i) preserving of skeleton connectivity and shape

ii) obtaining the approximate medial axis

iii) outputting a skeleton of unity pixel width

Thinning algorithms can be classified into two types; sequential algorithms (Zainodin, ID, 1994), and parallel algorithms (Jang,. 1BK, 1992). Sequential algorithms have two approaches: the iterative approach and non iterative approach. In the iterative approach, pixels on the boundary are examined (either sequentially or in parallel) and successively deleted until a skeleton of one pixel width is obtained. On the other hand, the non iterative approach produces a medial line of the

Mohamed Abdulsalaam Ali and Kasmiran Bin Jumari*

* *Department of Electronics*
Faculty of Engineering
Universiti Kebangsaan Malaysia
Bangi, Selangor, 43600
Malaysia
e.mail: fadeel1@eng.ukm.my

original image directly (in one pass) without examining all pixels individually. In fact, our algorithm falls under the first approach of the first type, which is that of a sequentially iterative algorithm, in order to achieve simplicity and effectiveness.

In the proposed algorithm we use color coding in a bitmap file of sixteen colors. Different colors have been chosen for different types of pixels throughout the steps of the thinning process (e.g., mark, examine, preserve or delete and pixel recovery) to achieve thinning and solve the problem of discontinuity. Using this technique has yielded a very fine skeleton of the original image of Arabic handwritten text and in turn will facilitate the objective of feature extraction and recognition stages of any character recognition system.

Algorithm Procedure

Our algorithm utilizes a windows color bitmap file format. Six colors, black, white, yellow, blue, red and green, were chosen to represent on-pixel, off-pixel, noise pixel, start or end point pixel, deletable pixel and recovered pixel respectively. The input image file is a monochrome (black and white) bitmap file; however, as the algorithm starts to assign colors for different types of pixels the input file is converted to a windows color bitmap file. There are seven main steps to achieve the task of skeletonization and they are as delineated below.

Start and end point marking

This is done by scanning the whole image from the top-left to bottom-right corner allocating all pixels in the inner and outer border of the image and distinguishing deletable from undeletable pixels. The algorithm considers all black pixels "on-pixel", which are surrounded by six or seven white pixels "off-pixel" (in directions according to the Freeman's code diagram shown in Figure 1), as undeletable and assigns blue color to them. These pixels are expected to be start or end points of the image, which must stay undeletable for the sake of image shape preservation, and they should not be examined in all iterations coming afterward as shown, in Figure 2.

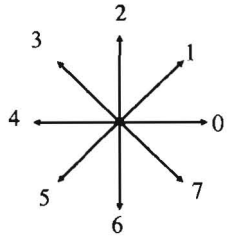
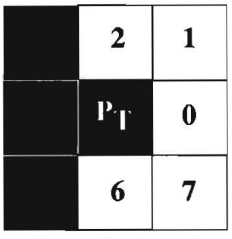


Fig. 1: Freeman Chain code

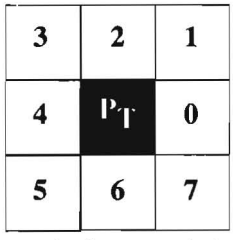
Pixel expected to be start or end-point

Figure 2: Start and end point detection

In the same manner, the algorithm considers all black pixels "on-pixels", which are surrounded by five or eight white pixels "off-pixels", as noise and assigns yellow color to them during scanning and then deletes them as shown in the Figure 3 (a) and Figure 3 (b)



(a) test-pixel surrounded by five white pixels

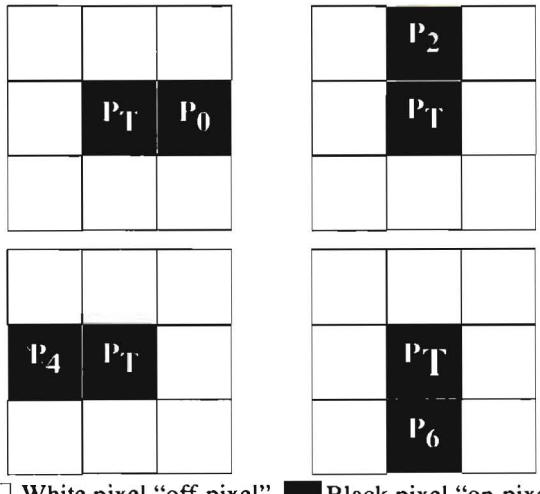


(b) test-pixel surrounded by eight white pixels

Fig. 3: Pixels that are considered as noise

Allocation of deletable pixels

In this step we need to allocate all pixels on the boundary of the image that can be deleted for the sake of thinning. The algorithm marks these pixels with red color. Allocation of these pixels follows the rules (template) shown in Figure 4.



White pixel "off-pixel" Black pixel "on-pixel"

Fig. 4: Templates for allocation of deletable pixel

Where P_T is a pixel under test and P_0 , P_2 , P_4 and P_6 are the four neighbor pixels of P_T in four directions according to Freeman's Code, the conditions that make P_T deletable are as follows:

If $\{(P_2=on)$ & $(P_6=off)$ or
$(P_0=on)$ & $(P_4=off)$ or
$(P_2=off)$ & $(P_6=on)$ or
$(P_0=off)$ & $(P_4=on)\}$

So P_T in all four above mentioned cases is a deletable pixel provided that it should be connected to at least two other black pixels. Subsequently they will be temporary turned red before the algorithm will finally decide whether to delete or retain them depending on the fulfillment of other conditions.

Now to avoid discontinuity there are three more rules to apply before starting to delete all pixels marked as deletable (red) pixels.

i) The first rule that we put to avoid discontinuity is that the deletable pixel should not follow any pattern shown in the Figure 5. If any of deletable pixels do fall under any of patterns shown in Figure 5, one of deletable pixels should be retained. The priority of retaining a pixel goes to the deletable pixel which has more other deletable pixels connected to it than the other; however, if both of the deletable pixels have the same number of other deletable pixels, the priority goes to the one that leads the other according to the direction of scanning the image from top-left to bottom-right, and that pixel is marked as a black pixel (retained).

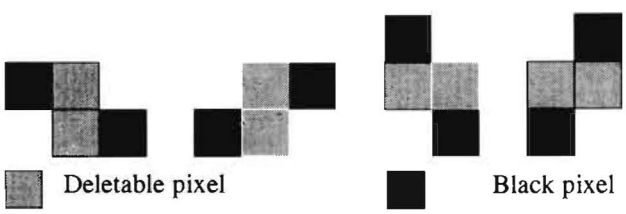


Fig. 5: First rule for discontinuity prevention

ii) The second rule states that if a deletable pixel is connected to another three deletable pixels in a manner shown in Figure 6 (a), the algorithm marks the medial pixel as a black pixel as shown in Figure 6 (b).

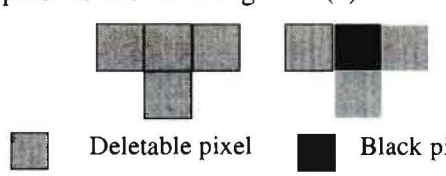


Fig. 6: Second rule for discontinuity prevention

iii) The third rule states that any pixel which has been marked as deletable red and has two white pixels "off-pixel" at the direction of (P_2 & P_6) or (P_0 & P_4) as shown in Figure 7, it should be reverted to a black pixel.

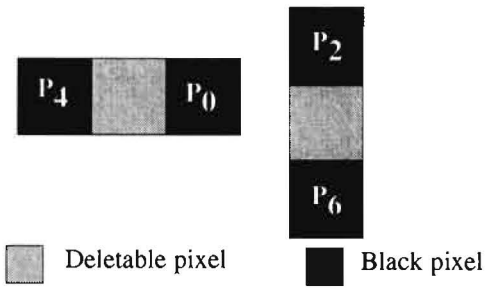


Figure 7: Third rule for discontinuity prevention

Deletion process

We shall now delete all pixels that are still marked as deletable pixels (red pixels) and turn them to white pixels. Pixel deletion follows the scanning of the image from the top-left corner to bottom-right corner. As a result of this deletion, we have noticed that some discontinuities have occurred and hence we make the algorithm finish this process without any interruption and make it iterate as described in the next section till there are no more pixels deleted. (In other words, the number of deleted pixels after each iteration is the same). Only then the algorithm starts checking for discontinuities and deals with them as we shall see later in Section 2.5.

Iteration

The algorithm now will iterate, repeating steps in the *allocation* and *deletion* processes until there are no more red pixels to delete. In other words, the templates in Figure 4 are no longer applicable. The number of iterations depends mainly on the thickness of the handwriting in the input image. For instance the handwritten character (ha), shown in Figure 8 (a), took five iterations to reach its final skeleton whereas the Arabic character (dal), shown in Figure 8(b), took six iterations. We could make notice of this by taking snapshots after each iteration. Using this technique can also help in monitoring the thinning process by following (step by step) the marking of pixels by different colors as explained above, so any process malfunction can easily be detected

Fig. 8: Two Arabic handwritten characters of different thickness and their skeletons

Discontinuity detection and recovery

After making the last deletion we noticed that there are some discontinuities in one place or another in the output skeleton, and accordingly we propose a technique which involves recovering those deleted pixels which cause this type of discontinuity, as follows.

We move a window of 3 x 3 on the whole thinned image and if one of the templates shown in Figure 9 is found, we check the missed pixel so that if it is proven that this pixel was there and, because of the thinning algorithm has been deleted, we just recover that pixel (make it a black pixel) so that we solve the problem of discontinuity; otherwise, we shall consider that as a deliberate discontinuity (i.e., it is one of the character features) and keep it as it is. Referring to Figure 9, P_T is a pixel to be checked whether it was there before applying the algorithm or not, so if it was there we just convert this off-pixel back to an on-pixel; otherwise, we leave it as it is.

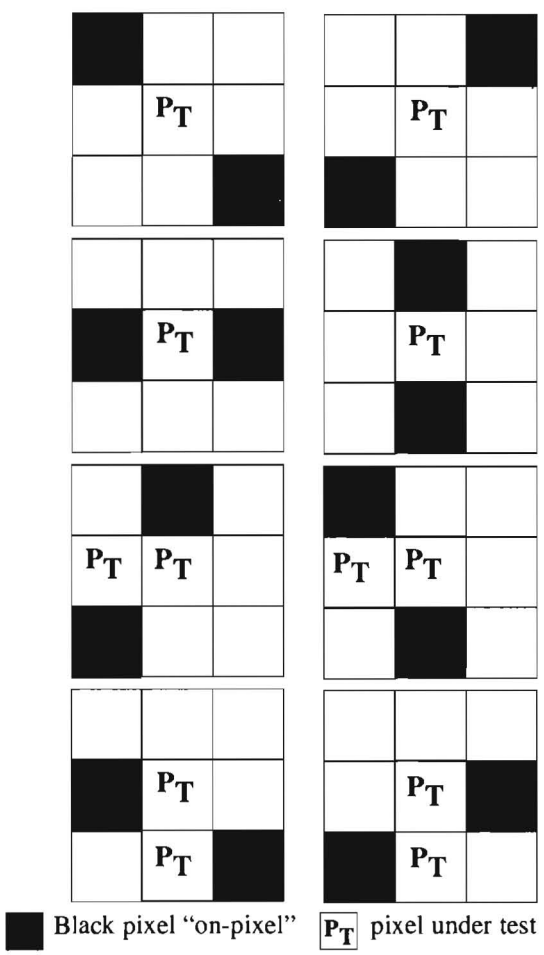


Fig. 9: Templates for recovery of deleted pixel and preserve connectivity

Solving this type of discontinuity does not prevent other types of discontinuity from occurring, such as the one shown in the Figure 10 where none of those templates is applicable and the length of discontinuity is more than two pixels. That has notably happened in the line or stroke which inclined diagonally in the direction of P_3 or P_7 (i.e., the lines go northwest or southeast). In Figure 10 we can clearly notice (from left to right) original image of the Arabic character (lam-alif), skeleton with discontinuity and the skeleton with discontinuity being recovered.

Fig. 10: Type of discontinuity more than one pixel long

The measures taken to recover this type of discontinuity are as follows. The algorithm sweeps the whole skeleton image looking for those black pixels which are connected to one black pixel only (excluding those pixels marked as start and end point blue pixels) and checks its neighbor at P_3 or P_7, so if the tested pixel is connected to either P_3 or P_2 and that P_7 is white and used to be black before deletion, then P_7 is converted back to black. Likewise, if the tested pixel is connected to either P_6 or P_7 and that P_3 is white and used to be black before deletion, then P_3 is converted back to black. Figure 11 illustrates this mechanism. This mechanism is repeated till there are no more pixels (excluding those blue pixels) connected to one black pixel only. In this way it is verified that our algorithm is effectively capable of solving this type of discontinuity

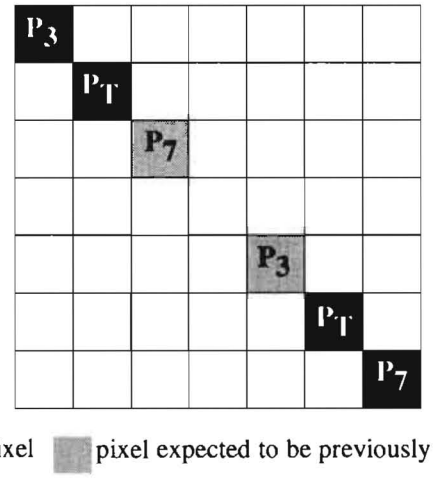


Fig. 11: Mechanism applied for a discontinuity of more than one pixel long

Removal of redundant pixels

One of the main features of our algorithm is that of removing the redundant pixels in the final skeleton. In Figure 12, although the skeleton is one pixel width, it has one or more pixels which can be removed without causing any discontinuity. On the contrary, the removal of those redundant pixels will enhance the processes of feature extraction and character recognition in an OCR system. This is due to the fact that the number of possibilities in the decision tree will be dramatically reduced and hence it will speed up the process.

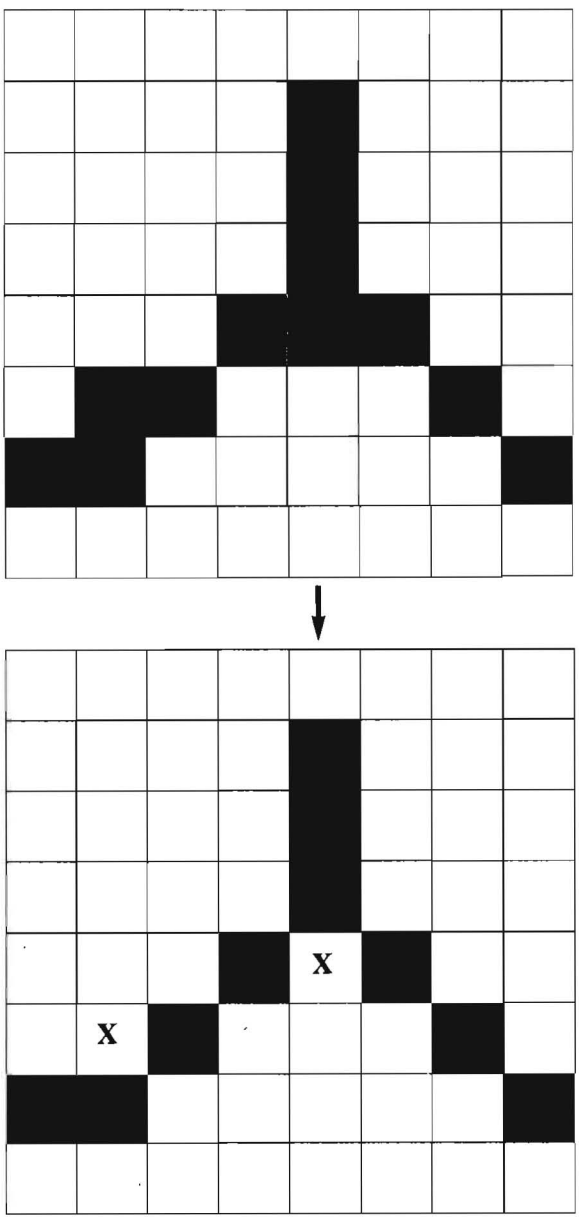


Fig. 12: Removal of redundant pixels

Experiments and results

The algorithm was tested on different Arabic handwritten samples, in both cases discrete and cursive, using an HP-scanner (with 1200dpi resolution) for image capturing. A preserved smooth skeleton was obtained. Figure 8 and Figure 13 show examples of tests carried out on different Arabic handwriting images along with their output skeletons. Figure 13 clearly shows how, when we superimpose the output skeleton on the original image, a skeleton of an image has a preserved shape, smoothness, intermediacy and a one-pixel line width.

Fig. 13: Samples of original Arabic handwritten images more than one pixel long and their skeletons

Optimization

To confine the algorithm to a minimum number of pixels for testing in each iteration so that we reduce the run-time and make it faster, we made the algorithm (in the first scan) assign the location of the first and last black pixels found as pixels of origin so that for the next iterations the algorithm starts and ends at these pixels rather than scanning the whole image area as defined by bit map file format. On the other hand, to avoid inefficient iteration the algorithm is designed so that the process of deletion (thinning) is stopped and the final output image (skeleton) is saved when either there are no more pixels to delete or the number of deleted pixels in two successive iterations is same. Subsequently, the excessive iterations are avoided and program run-time is minimized.

Conclusion

The main goal of this work is to develop a reliable thinning algorithm to be used in an Arabic

handwritten character recognition system. The proposed algorithm has used color coding to mark, delete and recover pixels in an image of Arabic handwritten so that a fine reliable skeleton of that image is produced in a very simple and effective manner compared with those algorithms which are based on a complex morphology and mathematical calculations which make the overall time consumption relatively high. Color coding gives better optimization and demonstration and yields an efficient skeletonization. Using this technique can also help in monitoring the thinning process by following (step by step) the marking of pixels using different colors, so any process malfunction can easily be detected. Through the analysis of the skeletons produced, it can be clearly noticed that they are very representative of the original shape of the handwritten image.

This paper introduces an interactive thinning algorithm for Arabic handwriting in particular; nevertheless, this algorithm can be used for Latin handwriting as well.

Acknowledgment

We would like to thank the Libyan Secretariat of Education for the grant it offered for research on an Arabic handwriting OCR system.

References

Altuwaijiri, MM and **Bayoumi, MA** (1998) A thinning Algorithm for Arabic characters using ART2 Neural Network. *IEEE Trans. Circuits & Systems, Analogue & Digital Signal Processing* **45** (2) 260-264.

El-Desouky, AI, Salem, MM, Abd El-Gwad, AO and **Arafat, H** (1992) A handwritten Arabic character recognition technique for machine reader. *Int. J. of Mini and Microcomputers* **14** (2): 155-161.

Flores, E, Rezende, EN, Gilberto A, Carrijo, Joao B and **Yabu-tti, T** (1995) A Fast Thinning Algorithm for Characters. *1995 IEEE Workshop On Nonlinear Signal And Image Processing,* NM. Greece.

Gonzalez, RC and **Wintz, P** (1987) *Digital Image Processing,* 2nd ed. Addison-Wesley, Canada.

Jang, BK and **Chin, RT** (1992) One-pass parallel thinning analysis, properties, and quantitative evaluation. *IEEE Trans. On Pattern Analysis and Machine Intelligence* **18** (3): 267-278.

Ali, Mohamed A and **Bin Jumari, K** (2002) A Survey and Comparative Evaluation of Selected Off-line Arabic Handwritten Character Recognition Systems. *J. Teknologi* **36:** 1-18.

Mori, S, Nishida, H and **Yamada, H** (1999) *Optical Character Recognition.* John Wiley & Sons.

Sabri A M, AbuHaiba, I and **Green, RJ** (1991) Skeletonization of Arabic characters using clustering based skeletonization algorithm. No periodical name supplied by the authors **24** (5): 453-464.

Zainodin, I, Khairuddin, D and **Horani, S** (1994) Sequential thinning of binary images. *Sains Malaysiana* **32** (4): 35-57.

Ref. 2264
Received 19/08/2003
In revised form 30/12/2003