A New Thinning Algorithm for Arabic Script

Mansoor A. Alghamdi
Department of Computer Science
Community Collage
Tabuk University
Tabuk, Saudi Arabia
malghamdi@ut.edu.sa

William J. Teahan
School of Computer Science
Bangor University
United Kingdom
w.j.teahan@bangor.ac.uk

***Abstract*—Thinning is one of the critical processes for different applications in image analysis, in particular for Optical Character Recognition (OCR) applications. The accuracy performance of OCR relies on the effectiveness of thinning algorithms. However, previously there has been little attention paid for proposing thinning algorithms for Arabic script. Also, there is a lack of quantitative performance measures of thinning techniques for Arabic script. Consequently, it is unclear which thinning algorithms are more appropriate for Arabic script. In this paper, a new thinning algorithm for Arabic script is proposed with several new performance metrics. An experiment is conducted to evaluate the proposed algorithm against two well established thinning algorithms with respect to the several proposed objective performance metrics. The experimental results show that the new algorithm has the best performance among the other two thinning algorithms.**

Keywords-thinning; skeletonization; Arabic; performance evaluation; Arabic OCR; thinning algorithm

I. INTRODUCTION

Optical Character Recognition (OCR) is a technique that converts a machine-printed or handwritten text image into an editable computer format. OCR is considered a challenging task in the field of pattern recognition. Many OCR approaches have been proposed for Latin and non-Latin scripts. However, Arabic OCR still poses great challenges because of Arabic script characteristics [1].

Generally, the process of developing Arabic OCR systems consists of three main stages: pre-processing, feature extraction and classification. One of the key and initial stages for developing Arabic OCR systems is the pre-prepossessing stage which is a combination of algorithms that are applied to the input pattern images for facilitating the subsequent phases of OCR development process [2].

Producing skeletons is a critical pre-processing operation for OCR in which extracting features from the skeleton of a character is essential [3]. The method for producing the skeleton of a pattern image is called thinning. Thinning "skeletonization" can be defined as the process of unpeeling as many pixels as possible without distorting the general shape of the character [4]. In other words, it involves operations that can be implemented in order to produce the skeleton of object images. Thinning techniques are classified into iterative approaches and non-iterative approaches [5]. The former can be either sequential methods, which perform by peeling the counter pixels individually, or parallel methods which perform simultaneously on all the counter pixels until obtaining a skeleton [6]. The latter utilize other techniques, such as distance transforms, to produce a skeleton without examining all pixels [6].

In general, thinning is usually applied in OCR systems as a method for reducing the amount of data of a pattern that needs to be considered for the next processing stage, thereby saving storage space for the structural information of the pattern [7], [8]. Furthermore, thinning algorithms have contributed to facilitate feature extraction of a pattern which is the core task in OCR to identify the pattern from another, since the relevant information of a pattern is not related to the thickness of the pattern [9], [10]. Therefore, it is claimed that the effectiveness of an OCR performance is heavily relying on how effective the thinning algorithm is [11]. The authors in [4], [12] agree that the main features of a desirable thinning algorithm are: ensuring the peeling is as thin as possible, connected and

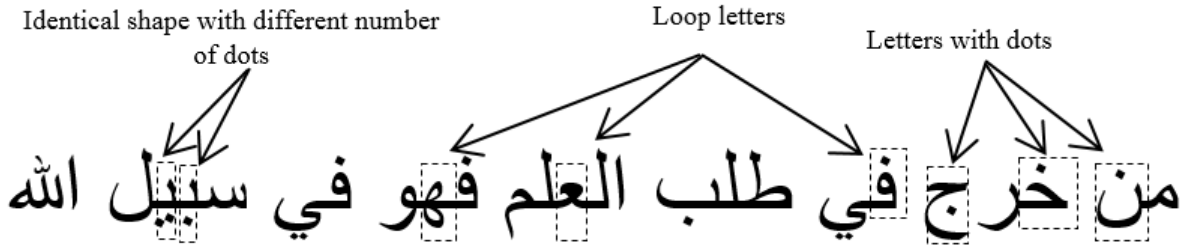


Figure 1. Arabic script characteristics.

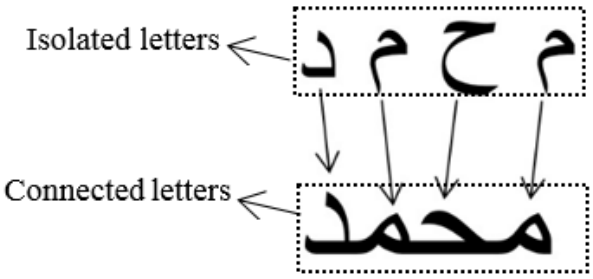


Figure 2. Arabic connectivity characteristic.

preserving the topology of a pattern.

A number of thinning algorithms have been proposed, such as in [13], [14], [15]; for a comprehensive survey refer to [16]. However, these algorithms have not been proposed specifically for Arabic script; rather they are proposed for general purposes. Additionally, direct adoption of such thinning algorithms, which are developed for other languages, may not be applicable to Arabic because of its characteristics [17] [18], [19]. Fig. 1 illustrates the characteristics of Arabic printed script that contribute to the challenge for thinning algorithms of Arabic script. Arabic script contains loop shaped letters, and identical characters that have different number of dots. Also, it is written cursively. In other words, isolated characters are connected to form a word, as shown in Fig. 2.

According to Cowell and Hussain [17], various problems have arisen when applying thinning algorithms to Arabic characters. One of the main obstacles with applying thinning algorithms to Arabic text is recognizing the number of dots for similar characters, where the number of dots can differentiate between them [2], [18]. It is believed that any deletion of character dots will result in misrepresenting these characters [19]. Moreover, another problem of thinning algorithms when considering Arabic script concerns preserving the connectedness of Arabic text. Some thinning approaches may fail in persevering the Arabic text connectivity which will lead to challenges in text recognition [18]. Owing to these reasons, it is claimed that thinning is responsible for many recognition errors in OCR systems [2], [18]. Therefore, thinning algorithms must be capable of both preserving dots and the connectedness of Arabic script.

As stated, there has been relatively few publications on developing thinning algorithms for Arabic [11], [8], [20]. For instance, [17], [21] introduce thinning algorithms for Arabic script. However, the proposed algorithms can only deal with isolated Arabic characters. Furthermore, one study by Ali [11] provides a thinning algorithm for Arabic handwritten script. Unfortunately, none of previous studies have considered the challenge of Arabic script discussed above, such as dots and connectivity preservation, when developing the thinning algorithm.

This paper will provide an efficient thinning algorithm for printed Arabic script. Then evaluation of thinning algorithms for Arabic will be discussed. An evaluation experiment is conducted on the proposed algorithm comparing it with a number of the most common thinning algorithms. Compared to the previous studies [22] and [23], this paper introduces several new objective performance metrics for thinning algorithms. Several performance metrics have been adapted to the evaluation using the evaluation dataset that contains a variety of Arabic font sizes and styles. In the last section, future works and conclusions are addressed.

II. THE PROPOSED METHOD

The proposed algorithm is based on an approach by Kocyigit [24]. However, this algorithm was only investigated for English characters. The method aims to produce a skeleton with one width pixel by detecting the neighborhood connectivity of any given pixel. Each pixel is directly connected with eight neighboring pixels; the pixel is considered as "encased" if only all the black neighbor pixels are interconnected with each other. In other words, if each black neighbour pixel of a considered pixel is connected vertically or diagonally to at least one other black pixel, the given pixel will achieve a state of encasement. For example, if the center pixel (P) is the pixel being processed, it is considered as "encased" in Fig. 3: (a), (b) and (c). In contrast, for (b), (c) and (d), the pixel (P) is not considered as "encased".

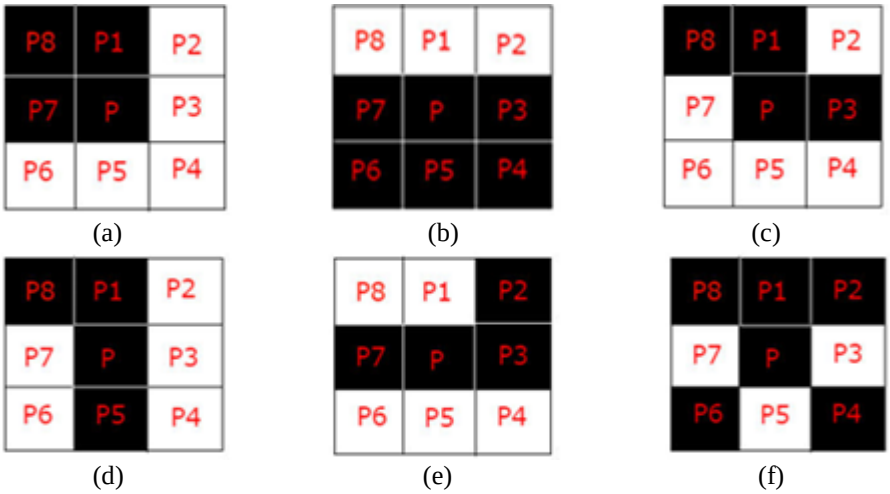


Figure 3. Examples of pixels (a), (b), (c) with an encasement and (d), (e), (f) without an encasement.

The proposed method utilizes a 3x3, 2D array of pixels which is obtained by thresholding the image. The descriptions of the rows and columns are equivalent to the pixel's coordinates and are set to either value "1" (black) or "0" (white) depending on the character. Deletion of a pixel will change the value of the pixel from 1 to 0.

Whenever any pixel is encased, the algorithm cleans up the defined pixel. The algorithm observes sequential and iterative principles in that whenever any pixel is removed, the entire algorithm restarts the iterative process, continuing until a state of consistency is achieved where no further pixel is deleted.

A flowchart of the proposed method is provided in Fig. 4. Initially, the algorithm finds a black pixel. Then, the black pixel is deleted if it has an encasement.

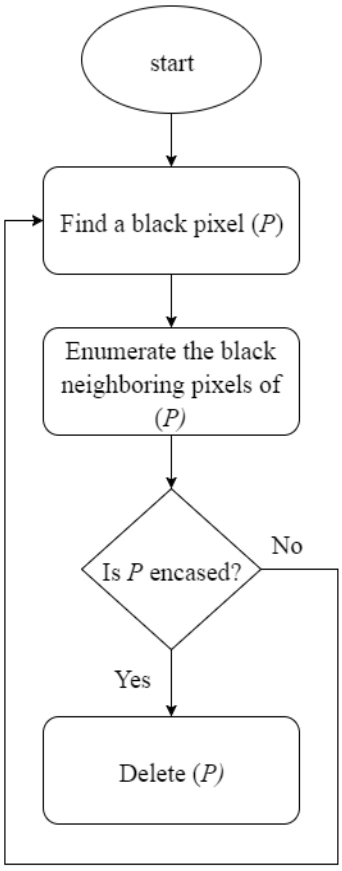


Figure 4. A flowchart of the proposed algorithm

III. EVALUATING THINNING ALGORITHMS FOR ARABIC SCRIPT

In respect to performance evaluation of thinning algorithms, most of studies evaluate their proposed thinning algorithm in terms of computation execution time and compression ratio, such as [7], [25] and [26]. Furthermore, some researchers evaluate thinning algorithms by only illustrating the output images of the algorithms, regardless of application requirements, such as [27], [11], [5], [28]. However, from an OCR perspective, these metrics are not sufficient to provide us with insight into which algorithm performs better for OCR.

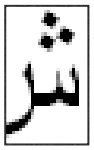
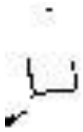
(a)  (b) 

Figure 5. An example of the problem of the connectiviy metric. (a) An Arabic letter with four components and (b) A skeleton for (a) with four components.

The authors in [29] provide other general metrics for assessing thinning algorithms, such as the thinness metric, which assesses the level to which a pattern image is thinned, and the connectivity metric that measures the connectivity of the thinned pattern images.

As mentioned before, Arabic script uses a cursive writing style and therefore, the connectivity measure is considered as a critical metric in order to evaluate the connectivity of the output skeleton of an Arabic text image. For instance, the study in [22] utilizes the connectivity measure to assess the performance of different thinning algorithms for Arabic. The idea of this metric is to consider the number of connected components in the original image and in the output image of the thinning algorithm. Namely, if the number of components is equal in both images, then this is a sign that the thinning algorithm is preserving text connectivity. However, some thinning algorithms may remove some dots of Arabic characters and they may spilt the body of a character into several parts. Consequently, an issue will arise when adopting the connectivity metric to evaluate the connectivity of thinned Arabic text. In particular, when thinning algorithms remove the characters dots or break the connectivity, then a performance metric which is relying on the number of connected components, will not be reliable.

To illustrate this problem, Fig. 5 shows that the number of component of the original image (a) is four and the number of components in the thinned image (b) is also four. Thus, according to the connectivity metric, the algorithm will be assessed as preserving the text connectivity, where in fact it is not. This problem occurs owing to the removing of one dot of the letter and making a gap of the letter's shape. Therefore, it is difficult to obtain statistical evaluation results of connectivity preservation by relying on this measurement.

To overcome this problem, we propose new performance metrics for evaluating thinning algorithms for Arabic text in terms of connectivity and preservation of dots. Moreover, other further objective performance metrics for evaluating thinning algorithms will be discussed below.

A. *Connectivity preservation metric*

Typically, an image for a pattern is constructed from a set of vertices (nodes) and edges (links) some of which might be

connected or disconnected [30]; see Fig. 6 for illustration. For thinning algorithms to preserve text connectivity, they must not divide a pattern in an image into incorrect pieces. A sophisticated thinning algorithm therefore must not delete edges of an image in order to maintain the connectivity characteristic of Arabic script.

Basically, there are two possible cases that will affect preserving the connectivity when producing skeletons for Arabic script. The first case is deletion of edges. For example, considering Figure 7 (a), it is clear that the word in the original image has fourteen (14) vertices and seven (7) edges. In contrast, in the thinned image in Fig. 7 (b), there are fourteen (14) vertices and six (6) edges. The lower number of edges is because of the deletion of one edge. The second case is insertion of a gap which results in the further insertion of edges and vertices. This can be seen in the case shown in Fig. 8 where the thinned image has sixteen (16) vertices and eight (8) edges, compared to the original image in Fig. 6 (a) which has fourteen (14) vertices and seven (7) edges.

Graph Edit Distance (GED) is a distortion measure on graphs that determines the differences between two pattern images [31]. The differences between the two images are obtained with a number of edit operations that are required to convert one image into another image. The authors here will utilize the concept of GED to assess the effectiveness of a thinning algorithm in preserving the connectivity of Arabic text.

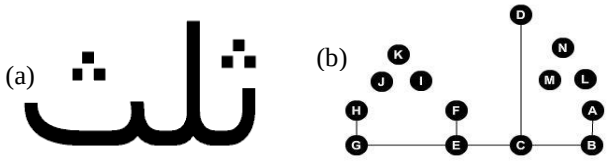


Figure 6. Using a graph to represent a text image: (a) a text image and (b) a graph for (a).

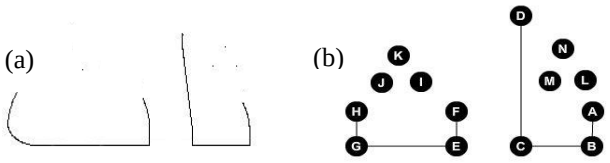


Figure 7. One thinning example for text image in Fig. 6(a). (a) Its skeleton and (b) a graph for (a).

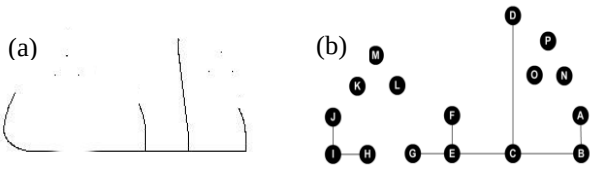


Figure 8. Another thinning example for text in Fig. 6(a). (a) Its skeleton and (b) a graph for (a).

The edit operations are: insertion of edges, insertion of vertices and deletion of edges. In order to measure GED, the minimum number of edit operations (edit distance) needed to convert the thinned image into the original image will be considered. Then, accuracy of thinning algorithms in preserving Arabic connectivity is determined by:

$$\frac{g-e}{g}\times 100 \qquad (1)$$

where g is the number of edges and vertices in the original image, and e is the edit distance. The cost of each edit operation will be defined as 1. For example, if a thinning algorithm inserted a gap, it will result in adding two vertices and delete one edge. Thus, the edit distance cost will be three, as three operations are required to transform the thinned image into the original – two deletions of vertices and one deletion of an edge, each having a cost of 1.

B. *Dot preservation metric*

In order for thinning algorithms to preserve dots of Arabic script, they must not remove isolated vertices which present the dots of Arabic text (see Fig. 9 for illustration). Thus, to statistically measure the effectiveness of thinning algorithms in preserving Arabic dots, the following equation is utilized:

$$\frac{v-e}{v}\times 100 \qquad (2)$$

where v is the number of isolated vertices in the original image, and e is the edit distance. The cost for each edit operation will be defined to be 1.

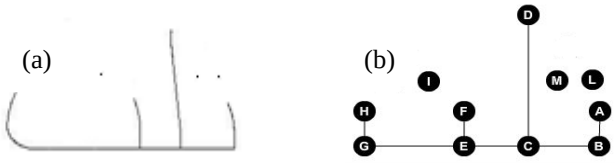


Figure 9. Example of dot perservation for thinning. (a) A skeleton and (b) a graph for (a).

C. *Topology preservation metric*

The topology preservation metric is the performance measurement utilized to assess the degree of thinning algorithm in preserving the visual information of the original image [12], [32], [33] [34] [35]. As reported by the researchers in [22], [18], producing spurious tails is considered a common problem of thinning algorithms for text images. Spurious tails can change the shape of a pattern, thereby affecting the accuracy of OCR output. Accordingly, it is essential to evaluate the effectiveness of thinning algorithms in terms of producing spurious tails.

The accuracy of thinning algorithms in preserving topology is determined by [12] :

$$1-\left[\frac{1}{2}-\frac{o}{c}\right] \quad (3)$$

where o is the number of object pixels in thinned image and c is the number of counter pixels in the thinned image. Note that in order to state that a thinning algorithm maintains visual information of the original image, the algorithm needs to have a preserving topology value close to 1.

D. Thinning rate (unit pixel width)

It is critical to evaluate the thinning algorithms in term of thinness of the skeleton, since one of the main principles of thinning algorithms is to ensure the peeling is as thin as possible. The thinness of the skeleton can be determined by computing the thinning rate as the following [23]:

$$1-\frac{tc}{tr} \quad (4)$$

where tc and tr refer to the number of triangles in the thinned image and in the original image, respectively. A thinning rate of 1 indicates that the skeleton is thinned to one-pixel wide, whereas a 0 value indicates that the skeleton is not thinned.

IV. EXPERIMENTAL RESULTS & DISCUSSION

We have used performance metrics described in the previous section for evaluating the proposed thinning algorithm. For comparison purposes, we also produce results for two other commonly used thinning algorithms for Arabic script, namely the Zhan-Suen (Algorithm 1) [36] thinning algorithm and the Hilditch algorithm (Algorithm 2) [37]. Note that all the three thinning algorithms were implemented in the Java programing language. The performance evaluation will provide us with insight into which algorithm performs better for Arabic script. In particular, a good thinning algorithm for Arabic should have a high performance accuracy in preserving connectivity and dots along with a value of preserving topology and thinning rate both close to 1.

Three datasets were used in this experiment. Specifically, the first dataset [38] is the printed Arabic word dataset, which consists of 50 word images, differing in fonts, sizes and styles. Also, two further sets of images were created for this evaluation experiment: images of Arabic characters and images of Arabic digits images.

In this experiment, all images from the three datasets were sent to each thinning algorithm to obtain thinned images. Then, the output images of each thinning algorithm were utilized to compute the quantities of each performance metric, corresponding to the original images for the datasets. Note that the performance metrics are implemented using the Matlab programming language. Samples of the evaluation experiment from the three datasets are visually illustrated in Table I. Table II presents the experimental data on the visual samples shown in Table I.

As discussed previously, connectivity preservation is one of the most essential features expected of thinning algorithms. From the data presented in Table I, it is apparent that the connectivity characteristic of Arabic script is almost maintained by the new algorithm. In particular, Algorithm 1 and Algorithm 2 failed in preserving the connectivity of sample 1 and sample 5, whereas the connectivity is preserved by the new algorithm in both samples.

There were major differences in the preservation of the dots of Arabic script between the three thinning algorithms. However, the new algorithm preserves the dots of Arabic script on all samples images that contain dots, as illustrated in Table I.

Furthermore, it is instructive to note that Algorithm 1 and Algorithm 2 were not able to preserve the loop shape of the sample 5 image, as shown in Table I and Table II, whereas it has been preserved by the new algorithm

The average scores for each metric for the three thinning algorithms are presented in Table III. The first column compares the three thinning algorithms in terms of the connectivity preservation which shows that the new algorithm obtained the highest accuracy (94.6%) of connectivity preservation among the other two algorithms.

The second column of Table III compares the thinning algorithms in terms of preservation of the dots of Arabic. The results show that there are significant differences between the performance accuracy for dot preservation by Algorithm 1, Algorithm 2 and the new algorithm, 78.2%, 85.2% and 99.4% respectively. Thus, it can be stated that the new algorithm is the most effective at preserving the dots.

The third column of Table III lists the values for topology preservation. As mentioned previously, a value of topology preservation close to 1 indicates that the algorithm is successful in preserving the shape of the original images. The results in the third column of Table III show that the value of topology preservation is significantly better for the new algorithm (0.9599), compared to Algorithm 1 (0.9398) and Algorithm 2 (0.9411).

The results of the thinning rate analysis are presented in the fourth column in Table III. From this column, it clear that the thinning rate value achieved by the new algorithm (0.9887) is superior to the other two algorithms. This indicates that the skeletons produced by the new algorithm is thinner than skeletons produced by Algorithms 1 and 2.

TABLE I. VISUAL SAMPLES USED IN THE EVALUATION.

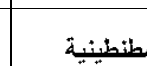
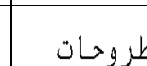
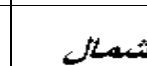
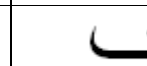
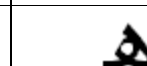
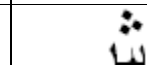
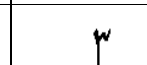
Sample No.	Original image	Algorithm 1	Algorithm 2	Proposed method
Sample 1				
Sample 2				
Sample 3				
Sample 4				
Sample 5				
Sample 6				
Sample 7				

TABLE II. COMPARING THE THREE THINNING ALGORITHMS ON THE SAMPLES.

Image	*Connectivity Preservation*			*Dots Preservation*			*Topology Preservation*			*Thinning rate*		
	Algo1	*Algo2*	*Proposed Algorithm*	*Algo1*	*Algo2*	*Proposed Algorithm*	*Algo1*	*Algo2*	*Proposed Algorithm*	*Algo1*	*Algo2*	*Proposed Algorithm*
Sample 1	90.9 %	90.9 %	100 %	90 %	90 %	100 %	0.9955	0.9288	0.9511	0.9473	0.9259	0.9766
Sample 2	100 %	95.8 %	100 %	100 %	100 %	100 %	0.9751	0.9729	0.9875	0.8711	0.8813	0.9796
Sample 3	94.1 %	94.1 %	100 %	100 %	100 %	100 %	0.9630	0.9983	0.9244	0.9695	0.9390	1
Sample 4	100 %	100 %	100 %	0 %	100 %	100 %	0.9247	0.9513	0.9513	0.9716	0.8865	0.9929
Sample 5	83.3 %	83.3 %	100 %	N/A	NIA	N/A	0.8614	0.8614	0.9045	1	0.9752	0.9886
Sample 6	50 %	50 %	83.3 %	0 %	33.3 %	100 %	0.9695	0.9434	0.9782	0.9389	0.9061	0.9859
Sample 7	80 %	80 %	100 %	N/A	N/A	N/A	0.9426	0.9590	0.9918	1	0.9606	0.9859

TABLE III. SUMMARY OF RESULTS COMPARING THE THINNING ALGORITHMS.

Algorithms	Connectivity Preservation	Dot Preservation	Topology Preservation	Thinning Rate
Algorithm 1	84.4 %	78.2 %	0.9398	0.9552
Algorithm 2	83.7 %	85.2 %	0.9411	0.9266
Proposed Algorithm	94.6%	99.4 %	0.9599	0.9887

In summary, the evaluation experiment results show that the new algorithm is more effective at dealing with the challenges of thinning for Arabic script, namely, connectivity and dots preservation. Moreover, the experimental analysis indicates that the skeletons of Arabic text produced by the new algorithm are better than those produced by the two other algorithms in terms of topology preservation and thinning rate. Also, the evaluation experiment of thinning algorithms by utilizing the proposed performance metrics provides a more in-depth analysis of which algorithm will perform better for Arabic OCR systems.

V. CONCLUSION AND FUTURE WORKS

An efficient thinning algorithm for Arabic script is described. Furthermore, the authors describe several objective performance metrics for assessing statistically the effectiveness of thinning algorithms including two new metrics for assessing topology preservation and dots preservation. By utilizing these metrics, a more robust evaluation of the effectiveness of different thinning algorithms can be carried out. Consequently, these metric will simplify the choice of thinning algorithms for Arabic OCR developers.

An evaluation experiment is conducted to evaluate the performance of the new proposed algorithm against other two well established thinning algorithms. In all performance tests, the new algorithm obtains the best results.

For future work, it will be interesting to measure the impact of the new thinning algorithm on the performance accuracy of Arabic OCR systems.

REFERENCES

[1] M. A. Alghamdi, I. S. Alkhazi, and W. J. Teahan, "Arabic OCR evaluation tool," in *2016 7th International Conference on Computer Science and Information Technology (CSIT)*, 2016, pp. 1–6.

[2] B. Al-Badr and S. A. Mahmoud, "Survey and bibliography of Arabic optical text recognition," *Signal Processing*, vol. 41, no. 1, pp. 49–77, 1995.

[3] J. R. Parker, "Algorithms for Image Processing and Computer Vision," *Vasa*, p. 504, 2011.

[4] M. Cheriet, N. Kharma, C. Liu, and C. Suen, *Character recognition systems: a guide for students and practitioners*. John Wiley & Sons, 2007.

[5] W. Abu-Ain, S. N. H. S. Abdullah, B. Bataineh, T. Abu-Ain, and K. Omar, "Skeletonization Algorithm for Binary Images," *Procedia Technol.*, vol. 11, no. 1, pp. 704–709, 2013.

[6] K. Saeed, M. Tabędzki, M. Rybnik, and M. Adamski, "K3M: A universal algorithm for image skeletonization and a review of thinning techniques," *Int. J. Appl. Math. Comput. Sci.*, vol. 20, no. 2, pp. 317–335, 2010.

[7] N. J. Naccache and R. Shinghal, "SPTA: A Proposed Algorithm for Thinning Binary Patterns," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-14, no. 3, pp. 409–418, 1984.

[8] A. M. Al-Shatnawi, F. H. Al-Zawaideh, S. Al-Salameh, and K. Omar, "Offline Arabic Text Recognition – An Overview," *World Comput. Sci. Inf. Technol.*, vol. 1, no. 5, pp. 184–192, 2011.

[9] H. Devi, "Thinning: A Preprocessing Technique for an OCR System for the Brahmi Script," *Anc. Asia*, vol. 1, 2006.

[10] Y. Alginahi, "Preprocessing Techniques in Character Recognition," INTECH, pp. 1–20, 2010.

[11] M. A. Ali, "An efficient thinning algorithm for arabic ocr systems," *Signal Image Process. An Int. J.*, vol. 3, no. 3, pp. 31–38, 2012.

[12] H. Chatbri and K. Kameyama, "Using scale space filtering to make thinning algorithms robust against noise in sketch images," *Pattern Recognit. Lett.*, vol. 42, no. 1, pp. 1–10, 2014.

[13] Z. Guo and R. W. Hall, "Fast fully parallel thinning algorithms," *CVGIP Image Underst.*, vol. 55, no. 3, pp. 317–328, 1992.

[14] C. Hilditch, "Comparison of thinning algorithms on a parallel processor," *Image Vis. Comput.*, vol. 1, no. 3, pp. 115–132, 1983.

[15] P. S. P. Wang and Y. Y. Zhang, "A Fast and Flexible Thinning Algorithm," *IEEE Trans. Comput.*, vol. 38, no. 5, pp. 741–745, 1989.

[16] L. Lam and S. W. Lee, "Thinning methodologies—a comprehensive survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 9, pp. 869–885, 1992.

[17] J. Cowell and F. Hussain, "Thinning Arabic characters for feature extraction," in *Proceedings of the International Conference on Information Visualisation*, 2001, vol. 2001-Janua, pp. 181–185.

[18] A. M. Al-shatnawi and K. Omar, "The Thinning Problem in Arabic Text Recognition - A Comprehensive Review," *Int. J. Comput. Appl.*, vol. 103, no. 3, pp. 35–42, 2014.

[19] H. M. M. Hosseini, *Analysis and Recognition of Persian and Arabic Handwritten Characters*. University of Adelaide, Department of Electrical and Electronic Engineering, 1997.

[20] H. Al-ani, N. Ban, and H. M. Abass, "Printed Arabic Character Recognition using Neural Network," Journal of Computing, vol. 5, no. 1, pp. 64–66, 2014.

[21] M. Altuwaijri and M. Bayoumi, "A new thinning algorithm for Arabic characters using self-organizing neural network," in *Circuits and Systems, 1995. ISCAS'95., 1995 IEEE International Symposium on*, 1995, vol. 3, pp. 1824–1827.

[22] A. M. Al-Shatnawi, B. M. Alfawwaz, K. Omar, and A. M. Zeki, "Skeleton extraction: Comparison of five methods on the Arabic IFN/ENIT database," in *Computer Science and Information Technology (CSIT), 2014 6th International Conference on*, 2014, pp. 50–59.

[23] P. TARÁBEK, "Performance measurements of thinning algorithms," *J. Information, Control Manag.*, vol. 6, no. 2, pp. 125–132, 2008.

[24] P. Kocyigit, "Agent Based Optical Character Recognition," MSc. thesis, Dept. CS. Bangor University, Bangor, 2012.

[25] A. K. J. Saudagar and H. V. Mohammed, "OpenCV Based Implementation of Zhang-Suen Thinning Algorithm Using Java for Arabic Text Recognition," in *Information Systems Design and Intelligent Applications*, Springer, 2016, pp. 265–271.

[26] Y. Y. Zhang and P. S. P. Wang, "A modified parallel thinning algorithm," *[1988 Proceedings] 9th Int. Conf. Pattern Recognit.*, 1988.

[27] M. Ali and K. Bin Jumari, "Skeletonization algorithm for an Arabic handwriting.," *WSEAS Trans. Comput.*, vol. 2, no. 3, pp. 662–667, 2003.

[28] W. Abu-Ain, S. N. H. Sheikh Abdullah, and K. Omar, "A simple iterative thinning algorithm for text and shape binary images," *J. Theor. Appl. Inf. Technol.*, vol. 63, no. 2, pp. 274–281, 2014.

[29] R. W. Zhou, C. Quek, and G. S. Ng, "A novel single-pass thinning algorithm and an effective set of performance criteria," *Pattern Recognit. Lett.*, vol. 16, no. 12, pp. 1267–1275, 1995.

[30] X. Gao, B. Xiao, D. Tao, and X. Li, "A survey of graph edit distance," *Pattern Anal. Appl.*, vol. 13, no. 1, pp. 113–129, 2010.

[31] M. Neuhaus and H. Bunke, "A Quadratic Programming Approach to the Graph Edit Distance Problem," *Graph-Based Represent. Pattern Recognit.*, pp. 92–102, 2007.

[32] B. K. Jang and R. T. Chin, "One-pass parallel thinning: analysis, properties, and quantitative evaluation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 11, pp. 1129–1140, 1992.

[33] L. Huang, G. Wan, and C. Liu, "An improved parallel thinning algorithm," *Proc. Int. Conf. Doc. Anal. Recognition, ICDAR*, vol. 2003-Janua, no. Icdar, pp. 780–783, 2003.

[34] G. Goyal and M. Dutta, "Design of Pixel Neighborhood Based Offline Handwritten Thinning Framework for Devnagri Numeral Script using Elman Neural Network," *Int. J. Comput. Sci. Inf. Secur.*, vol. 14, no. 7, p. 369, 2016.

[35] G. Goyal and M. Dutta, "Experimental Approach for Performance Analysis of Thinning Algorithms for Offline Handwritten Devnagri Numerals," *Indian J. Sci. Technol.*, vol. 9, no. 30, 2016.

[36] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[37] C. J. Hilitch, "Linear Skeletons From Square Cupboards," in *Machine Intelligence 4*, B. Meltzer and D. Michie, Eds. Edinburgh University Press, 1969, p. 403.

[38] H. Luqman, S. A. Mahmoud, and S. Awaida, "KAFD Arabic font database," *Pattern Recognit.*, vol. 47, no. 6, pp. 2231–2240, 2014.

AUTHORS PROFILE

Mansoor A. Alghamdi I am currently pursuing a Ph.D. in Computer Science at Bangor University, Bangor, UK. In 2011, at Otago University, Dunedin, New Zealand, I completed a Master of Applied Science in Software and Knowledge Engineering with distinguishing. In 2007, at Tabuk University, Tabuk, Saudi Arabia, I completed a B.Sc. Computer Science. I also was a Lecturer in the Department of Computer Science, Community College, Tabuk University, Tabuk, Saudi Arabia from 2012 to 2015.

William J. Teahan I am currently a Lecturer in the School of Computer Science at the University of Wales at Bangor. My work involves research into Artificial Intelligence and Intelligent Agents. Ongoing research has also specifically focused on applying text compression-based language models to natural language processing (NLP), text categorization and text mining. Before I came to Bangor, I was a research fellow with the Information Retrieval Group under Prof. David Harper at The Robert Gordon University in Aberdeen, Scotland from 1999-2000; an invited researcher in the Information Theory Dept. at Lund University in Sweden in 1999; and a Research Assistant in the Machine Learning and Digital Libraries Labs at the University of Waikato in New Zealand in 1998. At Waikato, I completed my Ph.D. in 1998 on applying text compression models to the problem of modelling English text.