S. Abuhaiba, **Sabri A. Mahmoud**, and R.J.Green, "Cluster number estimation and skeleton refining algorithms for Arabic characters.", The Arabian Journal for Science and Engineering, Vol. 16, Number 4B, pp. 519-530, October 1991

Key words: Arabic Cursive Characters, Character Graph Model, Clustering, Fuzzy Numbers, Skeletonization, Tree Structure,

CLUSTER NUMBER ESTIMATION AND SKELETON REFINING ALGORITHMS FOR ARABIC CHARACTERS

Ibrahim S. Abuhaiba* and Sabri A. Mahmoud

Computer Engineering Department, CCIS
King Saud University
Riyadh, Saudi Arabia

and

Roger J. Green

Electrical Engineering Department
University of Bradford
Bradford, U.K.

الخلاصـــة :

يقـدِّم هذا البحث طريقتين لتقدير عدد مجموعات تجزيء الحروف (العناقيد) اللازمة للخوارزمي المعتمد على عدد هذه المجموعات لتشكيل هياكل الحروف العربية بطريقة آلية (CBSA) . ويشير تحليل النتائج الى أن استخدام عدد تقريبي لهذه المجموعات كاف لهذا النوع من التطبيقات . وتعتمد خوارزميات تقدير عدد العناقيد على تقدير طول هيكل الحرف بطريقة تقريبية ، ويستخدم هذا الطول إمّـا مباشرة وإمّـا بعد معايرته (أي جعله قياسيا) وقد بيَّـنتْ النتائج العملية أنَّ استخدام الطريقة المباشرة أفضل على وجه العموم .

إنَّ استخدام الطرق المذكورة أعلاه قد ينتج عنه هياكل بها حلقات زائدة ولذلك صممت خوارزميات للتخلص من الحلقات الزائدة . وقد أثبتت النتائج العملية أن استخدام تقدير عدد المجموعات الوارد في هذا البحث مع خوارزمي تشكيل هياكل الحروف العربية (CBSA) ، بالاضافة الى خوارزميات تحسين هذه الهياكل سالفة الذكر ، ينتج عنه هياكل مقبولة بدون الحاجة الى توفر معرفة سابقة عن هذه الحروف . وسيثبت أنَّ هذه الهياكل هى خطوة أساسية في النمذجة وكذلك في الطرق الآلية للتعرف على الحروف العربية التي تستخدم هذا الاسلوب .

*Address for correspondence:
P.O. Box 89591
Riyadh 11692
Saudi Arabia

ABSTRACT

This paper presents two techniques for estimating the number of clusters for automating a clustering based skeletonization algorithm (CBSA) for Arabic characters. The analysis indicates that an approximate number of clusters suffices for this type of application. The cluster number estimation algorithms are based on estimating the approximate skeleton length of the character. This length is either used directly or standardized first. Experimental results show that the direct method is, in general, better.

Using the above techniques may produce skeletons with redundant loops. Skeleton refining algorithms are introduced to eliminate any redundant loops. The experimental results show that the combination of the cluster number estimation, the CBSA skeletonization algorithm, and the refining algorithms produce acceptable skeletons that do not require any *a priori* knowledge of the data. These skeletons will prove to be an essential step in modeling and in Arabic optical character recognition techniques using this approach.

CLUSTER NUMBER ESTIMATION AND SKELETON REFINING ALGORITHMS FOR ARABIC CHARACTERS

1. INTRODUCTION

Cluster analysis is based on partitioning a collection of data into a number of subgroups, where the objects inside a cluster (subgroup) show a certain degree of similarity. Given a set of points, A, clustering is partitioning of A into subsets or classes that maximizes both the similarity among members of the same subset, and dissimilarity across classes [1]. Some common algorithms compute the clusters that minimize (or optimize) some objective function. Others, perform clustering using the spatial adjacency information or neighbors of dots. Neighbors in clustering could be the dots that fall in a circular neighborhood [2] or k-nearest neighbors [3, 4]. Other definitions of neighbors exist [5–7]. The clustering algorithms perform partitioning using two criteria: (1) a measure of the similarity indicating if given pixels (or tokens) belong to a single cluster and (2) a criterion to decide when a given clustering is a good fit to the given data. Distance of dots and sum of squared errors are examples of similarity measures and criterion functions, respectively.

Other clustering algorithms exist that do not use the standard optimization procedures (*viz.* the hierarchical and graph-theoretical algorithms) [4, 5, 8, 9]. A review of clustering algorithms can be found in [10–12].

A problem common to all clustering techniques is the difficulty of deciding the number of clusters present in the data. Unfortunately, the available clustering algorithms cannot define the number of clusters in a direct manner. Rather, cluster validity criteria are formulated to aid in building curves, from which the number of clusters is extracted [11, 13, 14]. The process of building such curves is time consuming, especially if just an approximate number of clusters is needed. Actually, there are situations where *a priori* knowledge of internal structure of the data can help in estimating the number of clusters.

Mahmoud *et al.* [15] presented a skeleton generation algorithm for Arabic characters based on an earlier fuzzy clustering algorithm [16, 17]. The character image is clustered into a pre-determined number of clusters. The skeleton of a character is generated by plotting the cluster centers and then connecting adjacent clusters by straight line segments. The selection of the number of clusters was done manually by experience or by trial. The limitation in automating the CBSA skeletonization algorithm is the selection of the approximate number of clusters. The detailed analysis of the CBSA skeletonization algorithm and comparative study of other skeletonization algorithms are found in [15].

Distorted skeletons are generated when improper number of clusters is selected. In fact, applying clustering algorithms to the character image can be considered one of the applications in which general knowledge about data structure is available. The pixels constituting the character are distributed uniformly along the character body. This uniformity of distribution and the fact that the character is an elongated object is helpful in estimating the number of clusters.

This paper presents algorithms for estimating the approximate number of clusters and refining the resulting skeletons. The organization of the paper is as follows: in Section 2 characteristics of Arabic text are presented; classification of Arabic characters is addressed in Section 3; cluster number estimation algorithms are detailed in Section 4. Section 5 includes skeleton refinement algorithms for the elimination of any distortion in the generated skeletons. Section 6 details the experimental results and finally the concluding remarks are given in Section 7.

2. CHARACTERISTICS OF ARABIC TEXT

Arabic script and fonts characteristics are different and more complex than other languages. The following are some of the characteristics of Arabic language script:

(1) Arabic language is cursive (similar to cursive Latin script) and is written from right to left. Generally, an Arabic word consists of one or more connected segments, each consisting of one or more characters. The discontinuities between word segments is a result of the segments ending in characters which are not connectable from the left side with the succeeding character. Some Arabic characters appear only at the tail of a segment. The cursive nature of the Arabic language makes it difficult to segment a cursive word directly into characters.

(2) An Arabic character can have different shapes depending on its position in the word (beginning, middle, end, or alone).

(3) Arabic characters vary in their geometric sizes (particularly their widths), even in the same font of typewritten text.

(4) Many Arabic characters have dots which are positioned at a suitable distance above or below the character. Dots can be single, pairs, or triples. Different Arabic characters can have the same body and differ in the number of dots identifying them. Sometimes, the ambiguity of the position of those dots in handwritten texts brings out different interpretations for one word.

(5) The Arabic language uses another type of special characters as short vowels, which are referred to as diacritics. Although different diacritics on the same characters could lead to different words, an Arabic reader is trained to deduce the meaning of undiacriticized text. When diacritics are used they appear above or below the characters and they are viewed as isolated entities. Some of the Arabic characters use special marks to modify the character accent, such as *Hamza* (ء) and *Madda* (آ) which are positioned at a certain distance from the character.

(6) Some Arabic characters may overlap with neighboring ones. The degree of overlap can vary according to the typeface and the typewriter design or the writer's habits.

3. CLASSIFICATION OF ARABIC CHARACTERS

Arabic characters can be divided into two groups: character-without-dots and character-with-dots. Each of these groups can be subdivided into characters with intersections or loops and characters without intersections or loops. Table 1 shows the Arabic characters assigned to each group.

3.1. Character-without-Dots

Characters in this group have no dots, diacritics, *etc.* This group is subdivided into characters without intersections or loops and characters with intersections or loops. The results of applying the skeletonization technique of [15] to this group is as follows:

3.1.1. Skeletonization of Characters without Intersections or Loops

Figures 1 and 2 show two characters of this type: *Ra* (ر) and *Kaf* (ك). *Ra* (ر) is clustered into 3−14 clusters. As shown in Figure 1, skeletons corresponding to the range of clusters from 4 to 11 are all acceptable as they reflect the structural relationship of the components of *Ra* (ر). Clustering the character to lesser or larger number of clusters yields distorted skeletons as shown in Figures 1($b, k-m$) and 2(b, g, i, j). In one hand, the use of small number of clusters may result in merging of components of the character into one cluster as shown in Figures 1(b) and 2(b). This results in loss of structural information. On the other hand, selecting larger number of clusters may result in segmenting one component of a character into subsegments, hence generating skeletons with redundant loops as shown in Figures 1($k-m$) and 2(g, i, j).

In general, the characters of this category have simple structural relationships with no intersection points or loops, hence simple skeletons are generated. It is noticed that clustering the characters to a wide range of clusters yields skeletons that retain the structural information of the character. Hence, the use of an approximate number of clusters is adequate.

Table 1. The Arabic Character Set.

Arabic Character Set			
Character-with-Dots Group		Character-without-Dots Group	
With Intersections and/or Loops	Without Intersections and/or Loops	With Intersections and/or Loops	Without Intersections and/or Loops
ج خ ح	ب ت ن	ح س ص	ا ل
ش ض	ث ن	ط ع	د ر
غ خ	ذ ز ي	ح ه	ى
ف ق	ي	م لا	
ظ ة		و	

3.1.2. Skeletonization of Characters with Intersections or Loops

This category of characters has intersection points and/or loops. Skeletons for three characters of this category: *Ain* (عـ), *Ta* (طـ), and *Waw* (و) are shown in Figures 3–5, respectively. Clustering the

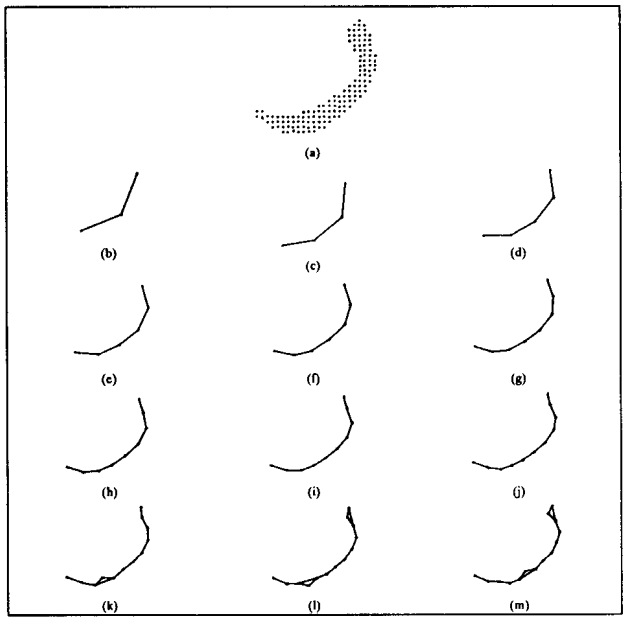


Figure 1. Skeletons of Ra (ر). *(a) Pixels of* Ra (ر), *(b)* NC = *3, (c)* NC = *4, (d)* NC = *5, (e)* NC = *6, (f)* NC = *7, (g)* NC = *8, (h)* NC = *9, (i)* NC = *10, (j)* NC = *11, (k)* NC = *12, (l)* NC = *13, (m)* NC = *14.*

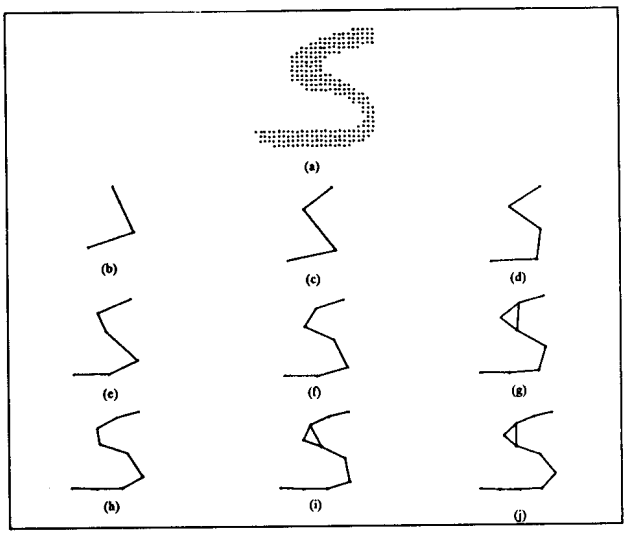


Figure 2. Skeletons of Kaf (ك). *(a) Pixels of* Kaf (ك), *(b)* NC = *3, (c)* NC = *4, (d)* NC = *5, (e)* NC = *6, (f)* NC = *7, (g)* NC = *8, (h)* NC = *9, (i)* NC = *10, (j)* NC = *11.*

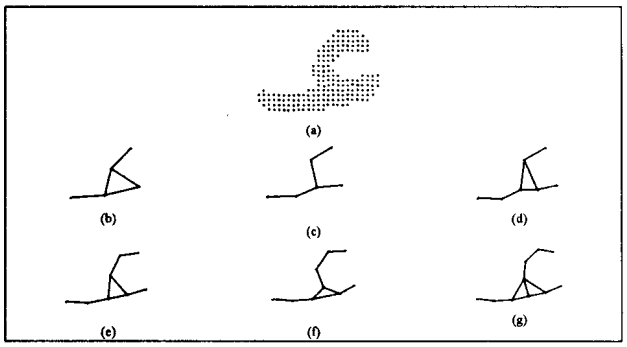


Figure 3. Skeletons of Ain (عـ). *(a) Pixels of* Ain (عـ), *(b)* NC = *5, (c)* NC = *6, (d)* NC = *7, (e)* NC = *8, (f)* NC = *9, (g)* NC = *10.*

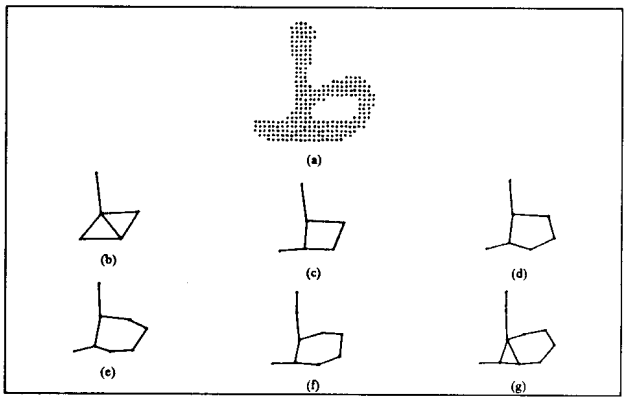


Figure 4. Skeletons of Ta (طـ). *(a) Pixels of* Ta (طـ), *(b)* NC = *5, (c)* NC = *6, (d)* NC = *7, (e)* NC = *8, (f)* NC = *9, (g)* NC = *10.*

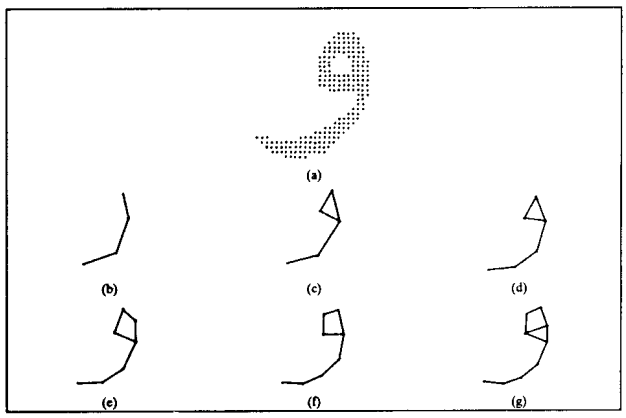


Figure 5. Skeletons of Waw (و). *(a) Pixels of* Waw (و), *(b)* NC = *4, (c)* NC = *5, (d)* NC = *6, (e)* NC = *7, (f)* NC = *8, (g)* NC = *9.*

Ain (عـ) character to 6–9 clusters and the *Ta* (ط) character to 6–10 clusters, produces acceptable skeletons as shown in Figures 3(*c–f*) and 4(*c–g*), respectively. This suggests the use of an approximate method to estimate the number of clusters required to produce acceptable skeletons to represent the character.

In some cases similar skeletons for different characters are obtained using small number of clusters as shown in Figures 1 and 5(*b*). Using higher number of clusters results in skeletons with redundant loops as shown in Figures 3(*d–g*), 4(*g*), and 5(*g*). This is a result of more than two clusters being adjacent to each other. Connecting the adjacent cluster centers produces the loops. The loops are triangles in most of the cases, and are rarely quadrants. These loops are eliminated by the proposed algorithm of the next section.

3.2. Character-with-Dots

These characters may have one, two, or three dots. A character of this type consists of two portions: (1) the main body of the character (it may be similar to the body of another character without dots) and (2) the dots or diacritics. This group is subdivided into characters with intersections or loops and characters without intersections or loops. The skeletons of the main body of the character (not including dots) are similar to those of the first group (*i.e.*, the character-without-dots). Additional problems arise in this group, due to the existence of dots. The following is a description of these problems.

Figure 6 shows the skeletons for the *Zain* (ز). character. It is clear from Figure 6(*b*) that a skeleton of 3 clusters is similar to the skeleton of the non-dotted character *Ra* (ر) as shown in Figure 1. The dot is not isolated with this low number of clusters which results in ambiguity in the recognition stage. To isolate the dot, larger number of clusters is used as shown in Figures 6(*c–j*). The same analysis applies for the *Noon* (ـنـ) and *Tha* (ـثـ) characters as shown in Figures 7 and 8. Sometimes, the improper selection of the number of clusters may lead to skeletons with a single isolated dot, although the original character has more than one dot as shown in Figure 8(*b*). The skeleton obtained is similar to that of another character with a single dot as shown in Figures 7(*c–j*). In this case, it is not clear whether an isolated cluster represents a single dot, or multiple of dots.

Figure 9 shows a dotted character *Za* (ظ), with its skeletons. *Za* (ظ) has a body which is the same as the character *Ta* (ط) with an additional dot as shown in Figures 4(*a*) and 9(*a*). A 6-cluster skeleton, which retains the structural information, is obtained for the character *Ta* (ط) as shown in Figure 4(*c*). It is clear from Figure 9 that although the cluster

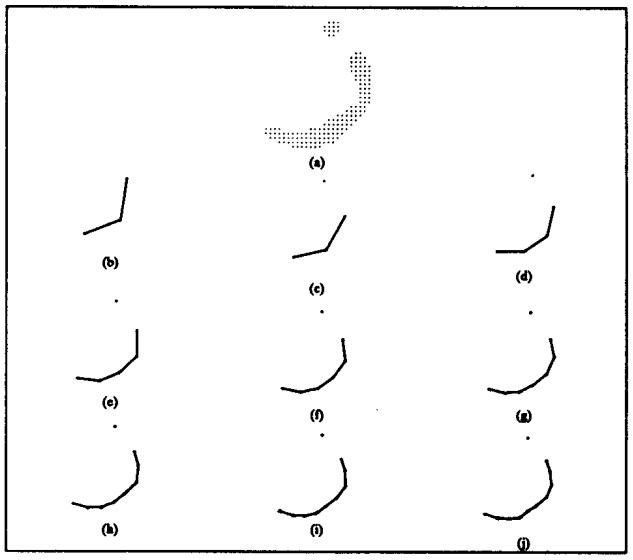


Figure 6. Skeletons of Zain (ز). *(a) Pixels of* Zain (ز), *(b)* NC = *3, (c)* NC = *4, (d)* NC = *5, (e)* NC = *6, (f)* NC = *7, (g)* NC = *8, (h)* NC = *9, (i)* NC = *10, (j)* NC = *11.*

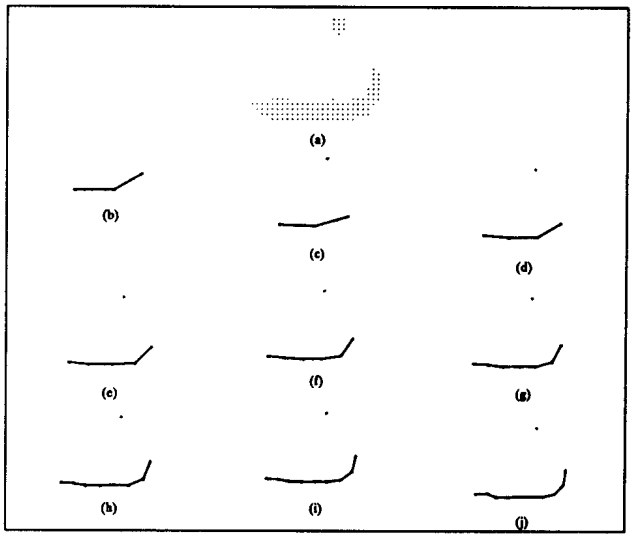


Figure 7. Skeletons of Noon (ـنـ). *(a) Pixels of* Noon (ـنـ), *(b)* NC = *3, (c)* NC = *4, (d)* NC = *5, (e)* NC = *6, (f)* NC = *7, (g)* NC = *8, (h)* NC = *9, (i)* NC = *10, (j)* NC = *11.*

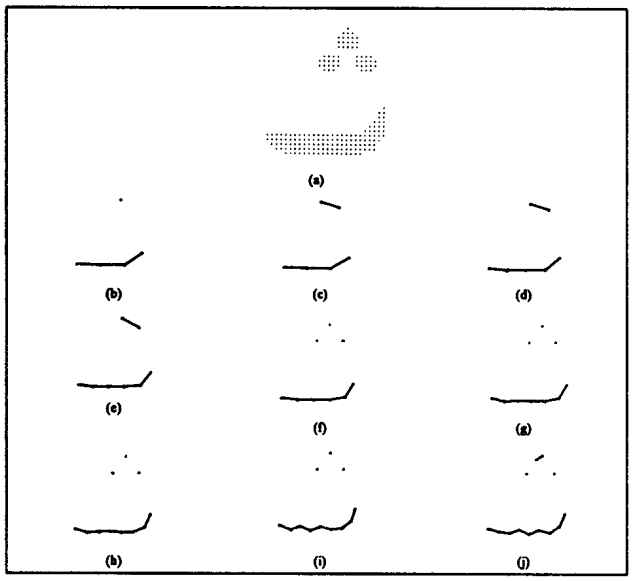


Figure 8. Skeletons of Tha (ث). *(a) Pixels of* Tha (ث), *(b)* NC = *5, (c)* NC = *6, (d)* NC = *7, (e)* NC = *8, (f)* NC = *9, (g)* NC = *10, (h)* NC = *11, (i)* NC = *12, (j)* NC = *13.*

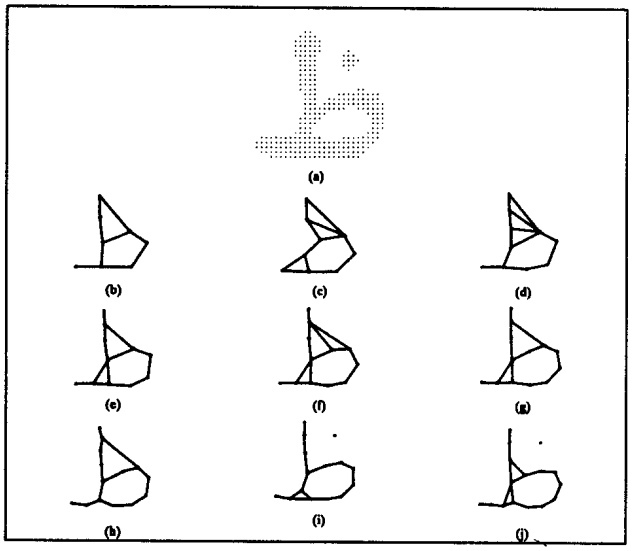


Figure 9. Skeletons of Za (ظ). (a) *Pixels of* Za (ظ), *(b)* NC = *8, (c)* NC = *9, (d)* NC = *10, (e)* NC = *11, (f)* NC = *12, (g)* NC = *13, (h)* NC = *14, (i)* NC = *15, (j)* NC = *16.*

number is increased to 14 clusters, it is not possible to isolate that single dot. This shows the added complexity due to the presence of dots in a character. In Figure 9 the dot is isolated at $NC > 14$.

An approximate function for estimating the number of clusters for dotted characters may not produce acceptable skeletons for all characters. A dot may not be isolated, especially if it consists of a very small number of pixels relative to the total number of pixels of the character. In this case the dot may not constitute a separate cluster. The dot pixels may be distributed among other close clusters. In addition, if the dot is close to the character body, it is very difficult to isolate it, as shown in Figures 9(b–h). The dot–body adjacency in the skeleton may yield a skeleton of another character, or a strange skeleton. This suggests that dots should be detected and isolated in an earlier step. The separation of dots before applying the clustering technique has the advantage of reducing the clustering time as the required number of clusters decreases. In addition, it has the advantage of reducing the number of prototype skeletons required for recognition (since many Arabic characters have the same body with different number of dots). Actually, dots are important discriminators and can be used to narrow down the number of alternative paths in the recognition. Dots can be viewed as the smallest non-divisible unit of a character, which cannot be shared by more than one cluster. Each dot should be an independent entity (cluster) by itself. The rest of the paper assumes that the dots are detected and isolated in an earlier step.

4. ESTIMATION OF THE NUMBER OF CLUSTERS

As shown earlier, one of the problems in automating the clustering technique for characters is finding a suitable cluster number. An improper cluster number does not produce skeletons that reflect the relationships among the character components. The number of pixels in a character is not a suitable measure for cluster number estimation as a character can be written with different line widths, which yields different numbers of pixels for the same character. The following two methods for cluster number estimation are investigated.

4.1. Linear Method for Cluster Number Estimation

When a character is written by a human or typed by a machine, some relation exists between the character script length and the line width of the character. If the line width is small then the character script length is, in general, small and *vice versa.* Assuming the dot to be the smallest non-divisible unit of the character then a character can be segmented into segments similar to dots. In general, there is a certain number of clusters into which a

character can be clustered to obtain a correct skeleton of the character. This number is proportional to the number of dot-like segments into which a character body can be segmented. The number of clusters is estimated as follows:

Let NP be the number of pixels in the character, CL be the approximate contour length of the character, and SL be the character script length.

Step 1: The approximate contour length CL of the character is found. It is approximately taken as the number of pixels on the contour of the character.

The approximate character script length SL is given by

$$SL = CL/2 \,. \tag{1}$$

Step 2: The line width of the character script SW, is given by:

$$SW = NP/SL \,, \tag{2}$$

where NP is the number of pixels in the character.

Step 3: The maximum number of small nondivisible segments NS, into which the character can be segmented, assuming a dot-like segment is approximately square in shape, is given by:

$$NS = SL/SW \,. \tag{3}$$

Step 4: The number of clusters NC is given by:

$$NC = [K_1 \times NS] \tag{4}$$

where K_1 is constant that is found not to exceed one, as shown in Section 6 (For the IBM typewriter character set it is found to be 1) and [ ] is the largest integer $\leq K_1 \times NS$.

4.2. Standard Method for Cluster Number Estimation

The same characters can be written with different script lengths. If the character length is too long relative to the line width, then using the previous method produces large number of clusters. This may generate skeletons with redundant loops and distortions. One way to overcome this problem is to standardize the character, so that it fits in a rectangular box whose length is equal to unity. This is a virtual box whose longest side is equal to unity. The suitable number of clusters is estimated as follows:

Step 1: Find the approximate character script length SL, as shown in Equation (1).

Step 2: Calculate the standardizing factor SF by:

$$SF = 1/\max(x, y) \tag{5}$$

where x and y are the character width and length in pixels, respectively.

Step 3: Calculate the standardized character script length, SSL, by:

$$SSL = SF \times SL \,. \tag{6}$$

Step 4: Calculate the approximate number of clusters NC, by:

$$NC = [K_2 \times SSL] \tag{7}$$

where K_2 is a constant that is found to be 7 as shown in Section 6.

5. SKELETON REFINEMENT ALGORITHMS

The generated skeletons using the previous cluster number estimation techniques suffer from the problems of redundant loops. These loops can be classified into two different types.

1. *Type-1 Loops:* A triangle is considered as a type-1 loop if: (*a*) it has only one free vertex (a vertex is considered free if it is adjacent to only the other two vertices of the triangle); and (*b*) it is not adjacent to any other loop. Figure 10(*a*) shows this type of loop.

2. *Type-2 Loops:* This type consists of a set of adjacent triangles, or of triangles which does not satisfy type-1 loops. The set of vertices of the adjacent triangles are grouped together to form a type-2 loop set. Figure 10(*b*) shows this type of loops.

Each type of the loops described above is manipulated by a different algorithm.

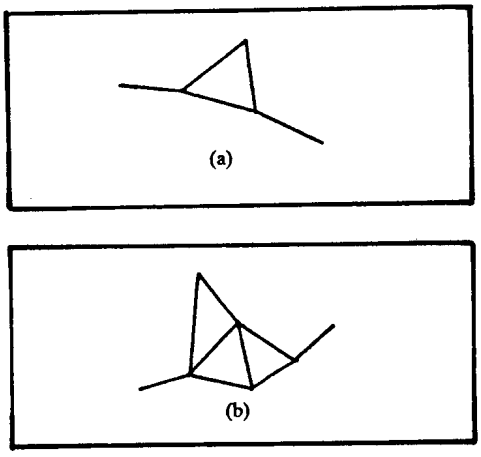


Figure 10. Types of Loops: (a) *Type-1 Loop,* (b) *Type-2 Loop Set.*

5.1. Type-1 Loop Elimination Algorithm

The elimination of this type of loop is done by adding a new vertex. This new vertex is taken as the mean of the three vertices constituting the type-1 loop triangle. The adjacency relation, AM, is modified to show this change. For the elimination of type-1 loops, the following algorithm is applied.

Let LT be a type-1 loop triangle; v_i a free vertex of LT; v_j, v_k the other non-free vertices of LT; and VCC is the vector of cluster centers v_i, v_j, v_k, where $v_l = [x_l, y_l]^T$, $l \in \{i, j, k\}$, and x_l, y_l are the cartesian coordinates of the cluster center v_l.

Step 1: Calculate the center of a new vertex, v_g according to:

$$v_g = (v_i + v_j + v_k)/3 \ . \tag{8}$$

This new vertex is added to the old set of vertices.

Step 2: Modify the adjacency relation, AM, as follows:

$$am_{lm} = 0 \quad \forall l, m \in \{i, j, k\}, \ l \neq m \text{ and}$$
$$am_{gl} = am_{lg} = 1 \quad \forall l \in \{i, j, k\} \ . \tag{9}$$

5.2. Type-2 Loop Elimination Algorithm

In this type of loops, a new vertex is generated. The location of the new vertex is taken as the mean of all the cluster centers that are vertices of the type-2 loop set. The cluster centers that are connected to only vertices in the type-2 loop set are eliminated. The remaining cluster centers are connected to the new vertex. The refinement algorithm is described below.

Let Q be a set of vertices constituting a type-2 loop set.

Step 1: The added vertex v_g is the center of gravity of the vertices in Q and is given by:

$$v_g = \left(\sum_{v \in Q} v\right)/n \tag{10}$$

where n is the number of vertices in Q. This vertex is added to the adjacency matrix.

Step 2: The set of vertices, $W \subseteq Q$, that are adjacent to vertices that are only in Q are deleted from the adjacency matrix.

Step 3: The adjacency matrix is modified according to:

$$am_{gl} = am_{lg} = 1, \quad \forall l \in \{Q - W\} \ . \tag{11}$$

6. EXPERIMENTAL RESULTS

An IBM PS2 model 80, with 16MHz clock and 80287 coprocessor and IBM machine character set were used in the experiments. The images were captured using an HP ScanJet scanner which was connected to the microcomputer *via* an interface card. The Scanning Gallery software was used for scanning images. Binary images were taken directly from the Scanning Gallery. The resolution used is 300 dots per inch, in both the horizontal and vertical directions. The C language (Turbo C 2.0) was used in writing the software.

The clustering algorithm of [15] is used to generate the skeletons of 38 characters (this includes different shapes for some characters as the Arabic character set contains 29 characters only). Cluster numbers are calculated using the linear and the standard methods. In the linear method, the skeletons are generated for $0.8 \leq K_1 \leq 1.2$ in increments of 0.1. Skeletons of *Ha* (ح) and *Ha* (هـ), for $0.9 \leq K_1 \leq 1.2$, are shown in Figures 11 and 12, respectively. The triangle in Figure 11(*c*) is a type-1 loop. The triangles of Figures 11(*d, e*) and 12(*b*–*e*) are type-2 loop sets. The *Ha* (ح) character is without loops. It has a large line

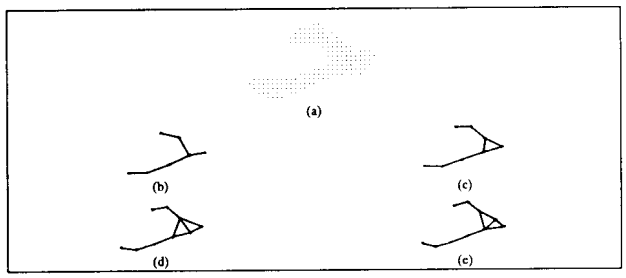


Figure 11. Cluster Number Estimation: Linear Method. (a) Pixels of Ha (ح), (b) $K_1 = 0.9$, NC = 7, (c) $K_1 = 1.0$, NC = 8, (d) $K_1 = 1.1$, NC = 9, (e) $K_1 = 1.2$, NC = 10.

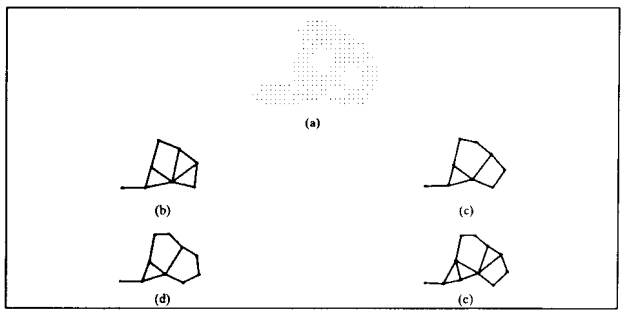


Figure 12. Cluster Number Estimation: Linear Method. (a) Pixels of Ha (هـ), (b) $K_1 = 0.9$, NC = 8, (c) $K_1 = 1.0$, NC = 9, (d) $K_1 = 1.1$, NC = 10, (e) $K_1 = 1.2$, NC = 11.

thickness relative to its script length. The character *Ha* ( ) has a complicated structure relative to other Arabic characters. It is noticed that K_1 can be chosen less than 1 in two cases: (1) if the character is without loops (*i.e.*, it has a simple structure) as shown in Figure 11(*a*) or (2) the line width of the character is small relative to the character script length. If the line thickness of the character is large then distorted skeletons with redundant loops are produced using $K_1 < 1$ as shown in Figure 12(*b*). As K_1 becomes larger than 1 the skeleton may have extra loops as shown in Figures 11(*d, e*) and 12(*e*). In general, the value of K_1 should not exceed 1 to prevent excessive clustering.

Again the cluster numbers were calculated using the standard method for $5 \leq K_2 \leq 9$. Figures 13(*a*) and 14(*a*) show two different characters: *Seen* ( ) and *Ha* ( ). Examples of type-1 loops are shown in Figures 13(*b–d*). Examples of type-2 loops are shown in Figures 13(*e*) and 14(*b–e*). It is noticed that selecting $K_2 < 7$ may not produce correct skeletons of the character as shown in Figures 13(*b*) and 14(*b, c*). For $K_2 > 7$ redundant loops are produced as shown in Figures 13(*e*) and 14(*e*). The best value of K_2 that produces acceptable skeletons for most of the characters under consideration is found to be 7.

Applying the refinement algorithms to the skeletons of the 38 characters produces preferable skeletons.

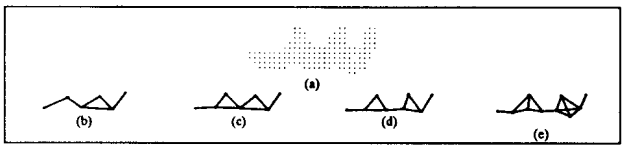


Figure 13. Cluster Number Estimation: Standard Method. (a) *Pixels of* Seen ( ), (b) $K_2 = 5$, NC = *6*, (c) $K_2 = 6$, NC = *7*, (d) $K_2 = 7$, NC = *8*, (e) $K_2 = 8$, NC = *10*.

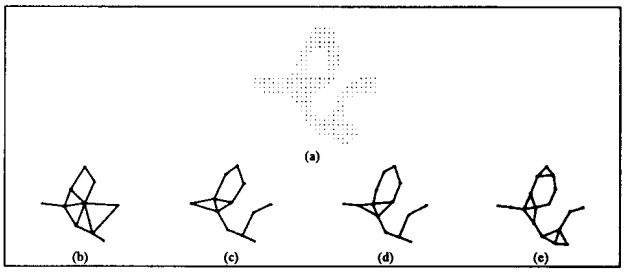


Figure 14. Cluster Number Estimation: Standard Method. (a) *Pixels of* Ha ( ), (b) $K_2 = 5$, NC = *10*, (c) $K_2 = 6$, NC = *12*, (d) $K_2 = 7$, NC = *14*, (e) $K_2 = 8$, NC = *17*.

Figure 15 shows five Arabic characters and their skeletons before and after refinement when the linear method is used, $K_1 = 1$, in cluster number estimation. Applying the vertex number reduction algorithm introduced in [15] produces a considerable reduction in the number of skeleton vertices without introducing skeleton distortion.

It is noticed that the skeletons of 19 of the 38 characters are better when the linear method is used in calculating the number of clusters. Reasons of preference can be one or any combination of the following:

1. The skeleton has no deflection;
2. The skeleton is without extra tails;
3. The skeleton is more accurate;
4. The skeleton is not similar to any other skeleton;
5. The skeleton has less number of clusters.

When the standard method is used to calculate the initial number of clusters, only 10 skeletons are better than those produced using the linear method. Reasons of preference are one or any combination of the above five reasons. For the remaining 9 characters, skeletons produced using the two methods in cluster number estimation are almost the same. In general, skeletons produced using the linear method in estimating the number of clusters are better than those produced using the standard method.

7. CONCLUSIONS

This paper presents the characteristics of the Arabic text. Arabic characters are classified according to the presence or absence of dots. These two classes are again classified according to the presence or absence of intersections and loops. This classification is useful in addressing the problems of skeletonization of the different groups.

The paper, also, addresses the problem of estimating suitable number of clusters to be used in skeletonization of Arabic characters as presented in [15]. Linear and standard methods for cluster number estimation are investigated. The analysis shows that the linear method gives, in general, better skeletons than the standard method as the skeletons are more accurate, have less number of clusters, do not suffer from extra tails, are not similar to the skeletons of other characters, and do not suffer from deflection. However, due to the use of approximate number of clusters the generated clusters may suffer from redundant loops. The paper introduces two algorithms for the elimination of redundant loops in the generated

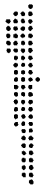
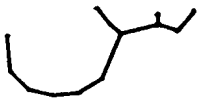
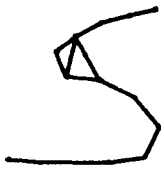
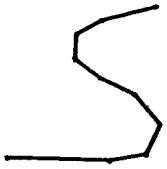
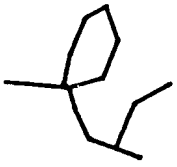
Char Name	Image	Cluster Numbers: Linear method, $K_1=1$	
		before refinement	after refinement
Alif (ا)		 8	 2
Seen (س)		 13	 13
Kaf (ك)		 14	 10
LamAlif (لا)		 16	 11
Ha (هـ)		 14	 14

Figure 15. Skeletonization of Some Arabic Characters Using the Linear Method of Cluster Number Estimation. The number under the skeleton is the cluster number.

skeletons. The experimental results confirms the applicability of the above techniques in producing adequate skeletons.

The combination of the cluster number estimation algorithm introduced in this paper, the CBSA skeletonization algorithm of [15], and the skeleton refinement algorithms of this paper result in a skeletonization algorithm that produces adequate skeletons without any *a priori* information about the suitable number of clusters. The skeletons generated using the proposed algorithms are essential in modeling and in Arabic optical character recognition techniques using this approach.

An Arabic character recognition algorithm that uses the skeletons of the presented algorithms is developed. The experimental results show encouraging results of the recognition rates and accuracy. The character recognition algorithm and its experimental results will be the subject of another paper.

ACKNOWLEDGEMENT

The authors would like to express their gratitude to the referees for their constructive criticism. The incorporation of their suggestions and comments significantly improved the clarity of the final manuscript.

REFERENCES

[1] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*. New York: Wiley, 1973.

[2] E. A. Patrick and L. Shen, "Interactive Use of Problem Knowledge for Clustering and Decision Making", *IEEE Transactions on Computers*, **C-20** (1971), p. 216.

[3] S. W. Zucker, "Region Growing: Childhood and Adolescence", *Computer Graphics and Image Processing*, **5** (1976), p. 382.

[4] F. R. D. Velasco, "A Method for the Analysis of Gaussian-like Clusters", *Pattern Recognition*, **12** (1980), p. 381.

[5] G. T. Toussaint, "The Relative Neighborhood Graph of a Finite Planar Set", *Pattern Recognition*, **12** (1980), p. 261.

[6] R. B. Urquhart, "Graph Theoretical Clustering Based on Limited Neighborhood Sets", *Pattern Recognition*, **15(3)** (1982), p. 173.

[7] N. Ahuja and M. Tuceryan, "Extraction of Early Perceptual Structure in Dot Patterns: Integration Region, Boundary, and Component Gestalt", *Computer Vision, Graphics, and Image Processing*, **48** (1989), p. 305.

[8] C. T. Zahn, "Graph-Theoretical Methods for Detecting and Describing Gestalt Clusters", *IEEE Transactions on Computers*, **C20** (1971), p. 68.

[9] N. Ahuja, "Dot Pattern Processing Using Voronoi Neighborhoods", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-4(3)** (1982), p. 336.

[10] R. Dubes and A. K. Jain, "Clustering Techniques: The User's Dilemma", *Pattern Recognition*, **8** (1976), p. 247.

[11] B. S. Everitt, *Cluster Analysis*. New York: Wiley, 1980.

[12] A. K. Jain and R. C. Dubes, *Algorithms for Clustering Data*. Englewood Cliffs: Prentice–Hall, 1988.

[13] P. H. Sneath and R. R. Sokal, *Numerical Taxonomy*. San Francisco: W. H. Freeman, 1973.

[14] I. Gath and A. B. Geva, "Unsupervised Optimal Fuzzy Clustering", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **PAMI-11(7)** (1989), p. 773.

[15] S. A. Mahmoud, I. S. I. Abuhaiba, and R. J. Green, "Skeletonization of Arabic Characters Using Clustering Based Skeletonization Algorithm (CBSA)", *Pattern Recognition*, **24(5)** (1991), p. 453.

[16] J. Bezdek and J. Dunn, "Optimal Fuzzy Partitions: A Heuristic for Estimating the Parameters in a Mixture of Normal Distributions", *IEEE Transactions on Computers*, **C-24** (1975), p. 835.

[17] J. Bezdek and P. Castelaz, "Prototype Classification and Feature Selection with Fuzzy Sets", *IEEE Trans. Syst. Man Cybernet*, **SMC-7(2)** (1977), p. 87.

Paper Received 6 November 1990; Revised 26 December 1990.