

Segmentation and Recognition of Printed Arabic Characters

B. M. F. Bushofa and M. Spann.

School of Electronic and Electrical Engineering, The University of Birmingham

Abstract

Arabic characters differ significantly from other characters such as Latin and Chinese characters in that they are written cursively in both printed and handwritten forms and consist of 28 main characters. However most of their shapes change according to their position in the word. These shapes together with some other secondaries raise the number of classes to 120. Furthermore, some of these characters have the same shape but are distinguished by the presence of one, two or three dots above or below them.

In this paper words are first segmented into characters and secondaries are removed using novel algorithms. This reduced the classes to 32 out of 120 different classes. Information about these secondaries such as their number, position and type are recorded and used in the final recognition stage. Features of the skeletonised character are used for classification using a decision tree. A recognition rate of 97.23% is achieved.

1 Introduction

Machine recognition techniques for Arabic characters are still in their early stages primarily due to the fact that current techniques applied to Latin and Chinese characters are not directly applicable to Arabic. Arabic characters differ significantly from other characters, such as Latin and Chinese, in that they are written cursively from right to left in both printed and hand-written forms. Each character has two to four different forms, depending on its position in the word which increases the number of classes from 28 to 100, Fig.1(a). Furthermore, some characters have exactly the same shape but they are distinguished from each other by the addition of one, two or three dots that can be either above or below the character. In addition there are two supplementary characters that operate on vowels to create a stress (Hamza ء) and (Madda ~); the latter operates only on the character Alif, Fig.1(b). The character Lam-Alif is created as a combination of two characters, Lam and Alif. This new character together with the combination of Hamza and Madda increase the number of classes to 120 (Fig.1a & 1b)

Due to a lack of computing power no significant work was performed in the field of Arabic character recognition until the 1980's. Some of the early work was done by A. Amin *et al* [1,2] in which he proposed an algorithm to segment Arabic words using the vertical histogram of the word. The average value of this histogram is taken to be the separation point. A word based structural off-line recognition method was proposed by K. El-Gowely, *et al* [3]. In this method, segmentation is performed by vertical histogram calculation on the text from right to left on the baseline. Classification is based on features such as height, width and distance from the baseline. The reported rate of recognition is 94%. T. Elsheikh and R. Guindi [4] proposed a method to separate Arabic characters by calculating the minimum height of their contour which is

considered as the point at which the characters are connected. Al-Yousefi and Udpa [5] used an off-line statistical method, where words are segmented into characters using histogram techniques. Classification is obtained by applying a quadratic Bayesian classifier on the primary parts. El-Khaly and Sid-Ahmed [6] proposed a separation algorithm which calculates the minimum and maximum columns and lines of the word and its baseline. It then separates the characters at the intersection of the baseline with the minimum value of the sum of the black pixels at this column.

Unlike the previous recognition techniques discussed above, in this paper all the 120 possible classes of Arabic characters are considered. Furthermore, the number of classes are reduced by a proper selection of the segmentation position and by removing all the secondaries as the first step following the pre-processing stage. Features are extracted by taking direct measurements of the characters and are easier to separate than other techniques based on Fourier descriptors or moment invariants. Furthermore, these features consist of line segments and curves fitted with polygons using a newly developed method.

The technique can be summarized as follows. The input text is first smoothed and binarized. The characters are segmented and their secondaries are removed. The main strokes are skeltonized and their features are extracted. Classification is then carried out using these features and information about the secondaries.

Name	Alif	Ba	Ta	Tha	Jeem	Hha	Kha	Dal	Thal	Ra	Zay	Seen	Sheen	Sad	Name	Alif	Alif	Alif	Waow	Ya
Isolated	ا	ب	ت	ث	ج	ح	خ	د	ذ	ر	ز	س	ش	ص	Isolated	أ	إ	ؤ	ئ	
Start		بـ	تـ	ثـ	جـ	حـ	خـ					سـ	شـ	صـ	Start					نـ
Middle		بـ	تـ	ثـ	جـ	حـ	خـ					سـ	شـ	صـ	Middle					نـ
End	ا	بـ	ثـ	ثـ	جـ	حـ	خـ	دـ	ذـ	رـ	زـ	سـ	شـ	صـ	End	أ	أ	ؤ	ئ	

Name	Dhad	Tta	Za	Ain	Gain	Fa	Qaf	Kaf	Lam	Mem	Noon	Ha	Waow	Ya	Name	LamAlif	LamAlif	LamAlif	LamAlif
Isolated	ض	ط	ظ	ع	غ	ف	ق	ك	ل	م	ن	هـ	و	ي	Isolated	لا	لا	لا	لا
Start	ضـ	ظـ	ظـ	عـ	غـ	فـ	قـ	كـ	لـ	مـ	نـ	هـ		يـ	Start				
Middle	ضـ	ظـ	ظـ	عـ	غـ	فـ	قـ	كـ	لـ	مـ	نـ	هـ		يـ	Middle				
End	ضـ	عـ	عـ	عـ	غـ	فـ	قـ	كـ	لـ	مـ	نـ	هـ	و	ي	End	لا	لا	لا	لا

Fig.1. (a) Arabic alphabet in all its forms. (b) Supplementary characters (Hamza ء and Madda ~) and their position with respect to the main character (Alif, Waow and Ya).

2 Segmentation

Since Arabic writing is always cursive in both printed and hand-written forms, a segmentation procedure has to be developed to split these words into characters. Arabic words consist of one or more subwords. This is caused by some of the Arabic characters being connectable only from one side. Further, some of these characters cause an overlap between the subwords making separation by vertical scanning impossible. Most of the published work on the recognition of Arabic characters assume that the characters are already segmented. Other work has proposed algorithms to segment the Arabic characters. Since the connection of the Arabic characters occurs always at the baseline, (Fig. 2), most of these algorithms utilise this fact.

From Fig.1, one can easily see that those characters which have four shapes depending on their position in the word, in fact have only two shapes, one shape for the isolated case and the other when it is written at the beginning of the word, Fig.3(a). By a proper design of the segmentation algorithm it is possible to extract the right class.

One way of doing this is to make the algorithm search for occurrence of an angle formed by the joining of two characters as indicated by the arrows in Fig.3(b). This angle occurs at the baseline which is the horizontal profile consisting of the maximum number of black pixels.

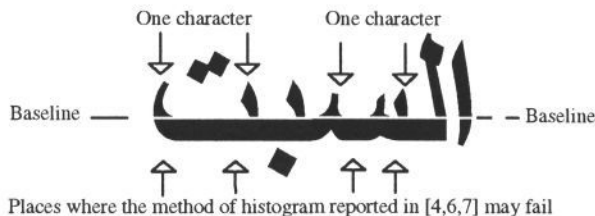


Fig.2. Example of an Arabic word (means Saturday). Arrows show places where algorithm [3,5] may fail by cutting in the middle of the characters ت and س.

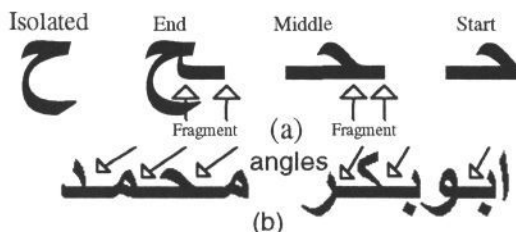


Fig.3. (a) Examples of characters which have close form when they are written isolated or in the end of the word and in the beginning or in the middle. (b) Example of two Arabic words. Arrows show angles formed by joining of the characters.

Once the baseline is found, the algorithm proceeds by scanning the image from right to left over this baseline using a 7x7 window and examining the neighbourhood of the central pixel. The central pixel is taken to be the candidate pixel for segmentation if its neighbourhood satisfies the 7x7 window of Fig.4 and according to the following steps :

x	x	x	y	0	0	0
1	x	x	y	0	0	0
x	1	x	0	0	0	0
x	1	1	p	0	0	0
1	1	1	1	1	1	1
x	x	x	x	x	x	x
x	x	x	x	x	x	x

1	1	1
1	p	0
1	0	0

Fig.4. A 7x7 and a 3x3 windows used to locate the point at which the two characters must be separated and to separate the character ح respectively.
(x: don't care. y: '1' for ح; '0' otherwise).

Step 1 When this condition is satisfied, the point p is regarded as the candidate point at which the two characters are split. To avoid having this condition inside the holes of some characters such as (ص and ط) the pixels above p are checked starting from p up to the maximum height of the line. If there is no black pixel then the pattern is divided at the point p. If there are some pixels then the image is further examined to see if the character on the left is Hha (ح). This is achieved by testing the first black pixel above p using a 3x3 window shown in Fig.4. If such a condition is satisfied then the fragment joining the character Hha is deleted.

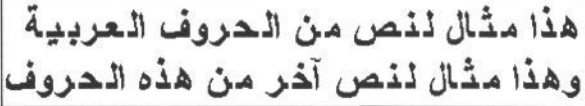
Step 2 If the condition is not satisfied the pixels marked 'y' in Fig.4 are examined. If both of them are black then the character on the left is Middle or End Ain (ع

جـ). If this condition is satisfied then the segmentation is carried out at the middle of the fragment joining this character with the character on the right.

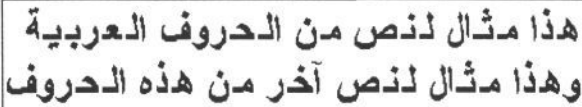
Step 3 The above steps are repeated until all the text lines are scanned.

Step 4 The contour following algorithm (described in section 3) is used to separate those characters which are not joined and might be overlapped.

Fig.5. shows an example of Arabic text after applying the histogram method reported by other authors [3,5] and this procedure. Although both algorithms succeed in segmenting the two lines of text, the algorithms of [3,5] generated more classes than our algorithm. This increases the processing time and leads to more misclassification.



(a)



(b)

Fig.5. (a) Example of Arabic text segmentation by the histogram method of [3,5].

(b) the same text after segmentation by the algorithm of this paper.

3 Removal of Secondaries of Arabic characters

The major obstacle to the recognition of Arabic characters is the presence of the dots and secondaries (Hamza ء and Mada ~). To avoid misclassification due to these secondaries and to reduce the number of classes to be processed these secondaries are separated before the characters are passed to the recognition algorithm. Information about these secondaries are passed directly to the recognition phase for classification.

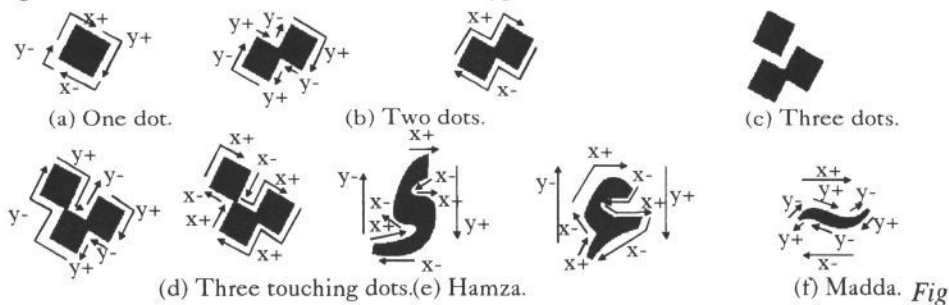
Unlike Al-Yousefi and Udpa algorithm [5] the algorithm of this paper can remove even those secondaries that fall inside the main stroke, (ج ظ ك) or that overlap the same line with main stroke, (ن ت ث). This technique employs a contour following algorithm which starts by finding the first black pixel which is the upper left most pixel in the bitmap of the character. It then moves clockwise around the present pixel from the previous pixel (initially set to a pixel immediately before the first pixel) until it meets another black pixel. This process is repeated until the first pixel is reached again and the coordinates of the border pixels are recorded. The segment (stroke) whose contour has been found is then removed from the map. The process continues until no more segments are left.

If only one segment is found then the character has no secondaries. The main segment is identified by comparing the size of each contour, which is the number of border pixels of each segment. This is carried out using the following steps :

Step 1 If two segments are found then if the size of the first segment is greater than the size of the second then the first segment is the main segment and the secondaries are considered to be below the main segment. Otherwise, the main segment is the second segment and the secondaries are above the main segment. A further test is then carried out to find the number of secondaries, (one, two or three dots or Hamza or Mada). This is necessary since dots are usually touching each other.

Step 2 If there are more than two segments then the size of the first segment is compared to the last segment. This is due to the fact that Arabic characters can have

secondaries either above or below but not both. Hence, either the first or the last segment is the main segment. After the main segment is identified the remaining segments are tested to find their number and type as follows :



6. Identification of secondaries by counting the number of changes of sign of their contour in both horizontal and vertical directions.

Case 1 If the main segment is the first then if the number of the remaining segments are two then there are two dots since Arabic characters cannot have more than two dots below and cannot have more than one Hamza or Madda in all cases, Fig.1.

Case 2 If the second segment is the last segment then if the number of remaining segments equals three then there are three dots. If the number of the other segments is one or two then the coordinates of the contour points previously recorded are tested to record the number of changes of sign in the horizontal and vertical directions as indicated by the arrows in Fig.6.

4 Thinning of the Arabic characters

Arabic characters can be thinned to a skeleton of width one pixel without distorting their shape. The selected algorithm of Wang and Zang [8] which uses the contour of the characters has the advantage of using the contour coordinates from the secondaries removal step, thus reducing the processing time. It was found to be suitable for thinning Arabic characters with some modifications to the conditions of pixel removal. The following modifications are necessary to gain adequate skeletons of some characters such as (Hha ح), (Lam-Alif ل), (Middle-Ha ه),

A pixel p is considered deletable if it satisfies the following conditions:

a) $1 < B(p) < 7$, where $B(p)$ is the number of non-zero pixels in the eight-neighbours of p . *This condition is modified to $1 < B(p) \leq 7$.*

b) $A(p) = 1$ or $C(p) = 1$, where $A(p)$ is the number of white-to-dark transitions when the neighbours of p are taking a clockwise walk around p and $C(p) = 1$ if p belongs to one of the following patterns:

x 0 0	0 0 x	x 0 0	0 0 x	x 1 0	0 1 x
1 p 0	0 p 1	1 p 0	0 p 1	0 p 1	1 p 0
0 1 x	x 1 0	0 1 x	x 1 0	0 0 x	x 0 0

This modified to:

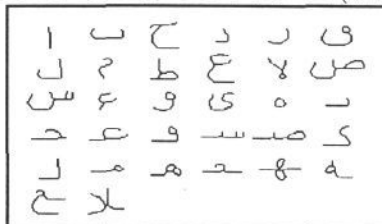
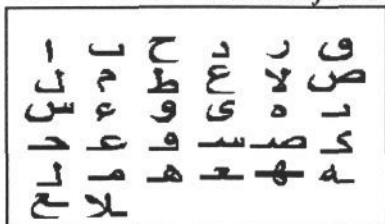


Fig.7. (a) The main set of Arabic characters. (b) Their skeletons.

The aim of these modifications is not only to improve the shape of the skeletons but also to make sure that the width of the skeleton is one pixel. This has the advantage of keeping the number of primitives per character to the minimum. Examples of the Arabic character set and their skeletons are given in Fig.7. (a) and (b) respectively.

5 Feature Extraction

5.1 Introduction

The features are extracted from the skeletonized characters by tracing each section of the skeleton from an end point to a junction point or vice versa. To obtain the junction and end points, each point in the skeleton is assigned a condition code according to the configuration of the pixels in the 3x3 window centred at the given point. The coding scheme is similar to [7] with some modifications.

A skeltonized character is represented by a binary matrix. For any pixel $p_{i,j}$ let δ be the sum of the 8 neighbouring pixels of p . The condition code $\chi_{i,j}$ of $p_{i,j}$ is assigned a value in the set $\{-9,-8,\dots,-1,1,2,\dots,9\}$ according to the following rules:

- (1) If $\delta \geq 3$, then $p_{i,j}$ is a junction point and $\chi_{i,j} = -9$
- (2) If $\delta = 1$, then $p_{i,j}$ is an end point and $\chi_{i,j}$ is assigned a value in $\{-8,-7,\dots,-1\}$ as shown in Fig.8(a). These indicate end points.
- (3) If $\delta=2$, then $\chi_{i,j}$ is assigned a value in $\{1,2,\dots,9\}$ as shown in Fig .8(b).

Rule 3 and Fig.8(b) are **different** from that in [7]. The positive condition codes are used to guide the tracing process which starts from the top-right most negative pixel. If a curve is traced it is then approximated by a polygon. Information about each line segment or polygon, such as start/end coordinates, length, direction (for the polygons), angle from the horizontal axis etc. are recorded.

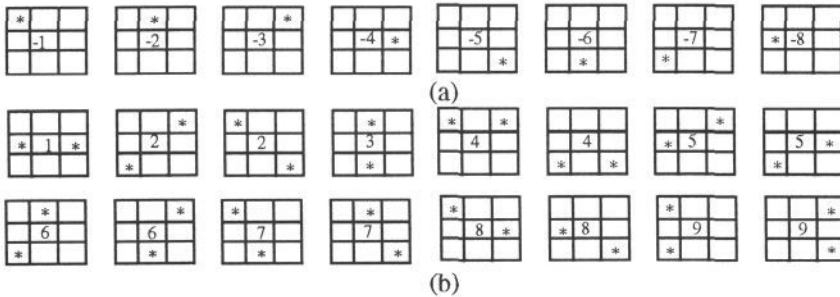


Fig.8. (a) Negative condition codes. (b) Positive condition codes.

5.2 Polygon approximation of curves

Most Arabic characters are made of curves. These curves are approximated by convex polygons that may be open or closed. The direction of open polygons is considered to be one of the following: east, northeast, north, northwest, west, southwest, south, or southeast. The closed polygon has no direction and the sum of angles between its sides is taken instead. The direction of the line segments is the angle it makes with the horizontal axis.

Fitting polygons to character curves is achieved by considering the changes of sign while the coordinates of its skeleton are traversed. Changes of sign either horizontally or vertically or both indicate that an end of an approximating line segment and a start of another has occurred. This process continues until the end of the curve is reached. The minimum length of the approximating line should be chosen in accordance with

the size of the character, otherwise a line of length of one pixel could result. This method is more robust and faster and simpler than the method reported in [9]. Fig.9, shows a flowchart of the algorithm together with an example of a fitted polygon to the curve of the character Ba \ominus using this algorithm.

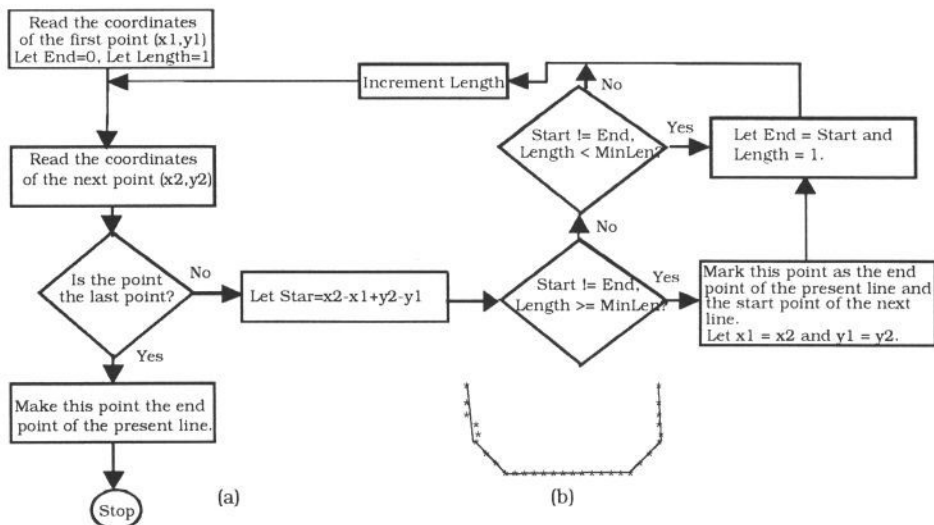


Fig.9. (a) Flowchart of the polygon fitting algorithm. (b) A polygon fitted to Ba \ominus

5.3 Convexity of the polygons

If the approximating polygon is not convex it will be decomposed into two or more convex polygons or line segments. The convexity of a polygon is determined by the sum of its interior angles, since it is well known that the sum of the interior angles of an n -sided convex closed polygon is $(n-2)\pi$ radians.

5.4 The Extracted features:

The extracted features of each primitive by the above mentioned procedures are as follows:

- (1) Type of primitive: this is either a line, a closed-polygon or an open-polygon.
- (2) Coordinates of the start and end points of the primitive.
- (3) Coordinates of the centre of the primitive if it is a line. Otherwise, the centre of the line joining the start and end points of the open-polygon.
- (4) Length of the primitive which is the distance between the start and end points for the line. For the polygon it is the sum of the length of each of its sides.
- (5) The direction: for lines this is the angle subtended by the horizontal axis and the actual line. For the polygons it is their direction as defined above.



Fig.10. Examples of features of some Arabic characters.

- (6) The starting/end point primitives: these are the primitives at the starting/end point of the primitive under consideration. These features are important since they describe the connectivity of the features.
 - (7) The number of sides of each primitive.
- Fig.10. shows an example of the primitives of some characters.

6 Recognition Process

6.1 The Decision Tree

In this stage all the characters are first divided into groups according to their number of primitives. Then for each group the types of its primitives (Line, Open polygon or Closed polygon) are used to divide it into subgroups. If a subgroup has characters that have similar types of primitives then they are further separated by the direction of their lines or open polygons. The number of sides of the polygon is also used to distinguish some characters. The final classification is then carried out using the information previously recorded about the secondaries. Fig.11 shows the decision tree.

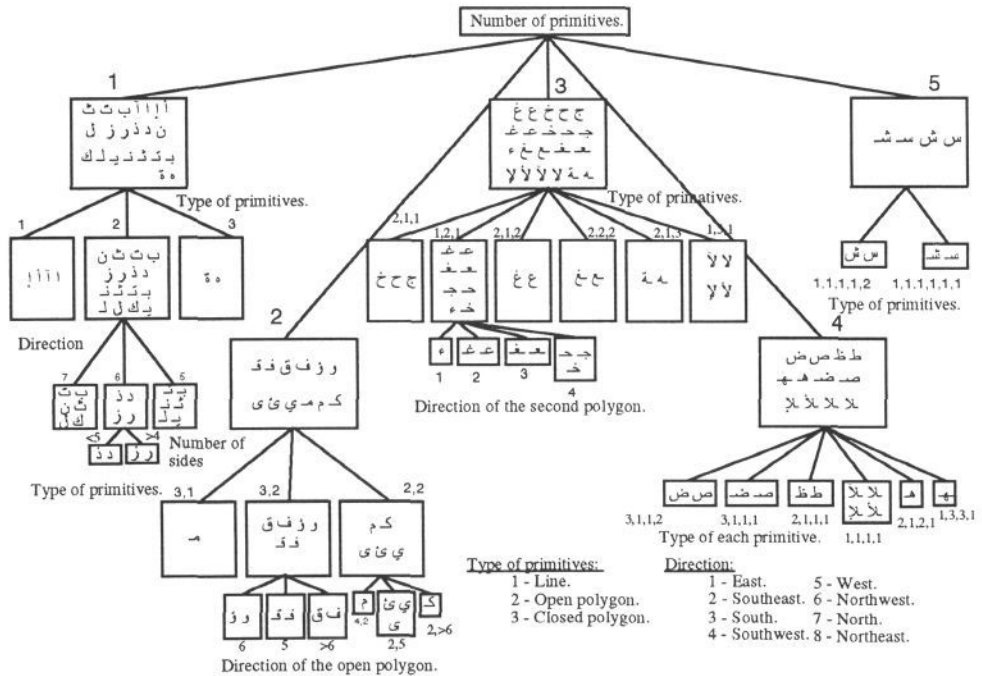


Fig. 11. Grouping of characters according to their number, type of primitives and direction of open polygons. Groups at the bottom are classified using the information about their secondaries.

6.2 Relaxation Matching

In this stage, when a character is rejected by the decision tree all of its features are packed in an array and are used to calculate the City Block distance between these features and features of each member of a set of predefined templates.

To minimise the time needed for the matching operation, the number of primitives are first compared to that of the template under consideration. If their difference is equal to zero or one, the distance between all of the features is calculated. Otherwise, the difference is set to a high number to ensure mismatch and the next template is

considered. The process is repeated until all the templates are visited. The reason that the difference can be zero or one is that sometimes when the size of the character grows beyond some certain level, the quantization error causes a local concavity which forces the original polygon to be divided into two polygons or one polygon and one line. This causes the character to be rejected by the decision tree.

Each feature in the character is matched to each feature in the template set. The absolute value of the difference due to each match is stored and all the differences are summed according to the City Block distance criterion. After all the distances between the character under consideration and each template are calculated, the minimum is picked. If this value is less than a certain threshold the character is classified according to the corresponding template and its secondary information, otherwise it is rejected.

7 Results and Discussion

7.1 Recognition using pre-segmented characters

Data of 4 and 15 different sizes, i.e. 1065 characters from each font are tested. This makes 60 samples from classes of 71 characters complete with their secondaries.

A total of 20 features for each of the character's primitives are extracted from its skeleton. Three of these features namely, type, direction and number of primitives together with the secondaries are used in the decision tree. When a character is not recognised by the decision tree all of its features are passed to the relaxation process to compare them with a set of standard features as summarised in Fig.12(a).

Font	Size	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	Tot	Rec. %
1	R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	96.43%
	M	2	2	2	-	3	1	3	3	2	2	3	3	3	3	4	38	
2	R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	98.59%
	M	3	2	-	-	2	-	-	-	-	-	1	1	-	3	3	15	
3	R	-	-	-	-	-	-	-	-	-	-	-	-	-	-	-	0	96.81%
	M	1	2	3	-	2	2	2	3	3	3	2	2	3	3	3	34	
4	R	-	-	-	-	-	-	-	-	-	1	-	-	-	-	-	1	97.09%
	M	1	2	2	2	2	-	2	2	3	1	2	2	3	3	3	30	

(a)

Font	Size	4	6	8	10	Tot	Rec. %
1	R	3	4	3	2	12	94.60%
	M	2	1	2	2	7	
2	R	4	5	3	4	16	93.75%
	M	2	1	1	2	6	

(b)

(R: Rejected; M: Misclassified).

Fig.12 (a). Recognition rates with respect of each font. Total recognition 97.23%. (b) Recognition rates of the segmented text. Total recognition 94.17%.

Most of the misclassifications occur between the characters Ra(ﺭ) and Dal(ﺩ). Although they look different in their complete form, their skeletons are closer. As the size of the characters shrinks the character Ra becomes more like Dal. When the size gets bigger the character Dal becomes more like Ra. This problem occurs also with characters Zay(ﺯ) and Thal(ﺙ) since they differ only by the presence of one dot from characters Ra(ﺭ) and Dal(ﺩ) respectively. Other misclassifications are due to the similarity between the Start_Ain(ﺀ) and the Line_Hamza(ء). As the latter is written immediately after the character Alif on the same line, its size is sometimes bigger than its usual size when written above or below the character Alif. Again they look different in their full form but their skeletons appear closer in shape. In most cases they are sorted by the direction of their fitted polygons. This direction is South or Southeast for the Start_Ain and East for the Line_Hamza. Misclassification occurs sometimes when the direction of the fitted polygon of the Line_Hamza becomes Southeast. This however, does not pose a big problem since this ambiguity can be resolved at the context level.

Line_Hamza occurs only after the character Alif and Start_Ain occurs at the beginning of the word or the subword.

7.2 Recognition of segmented characters

Three samples of text using two fonts, Fig.12(b) is passed to the segmentation procedure in four different sizes. Segmentation succeeds in most cases. However, it failed to segment some characters correctly due to the short distance between the two characters or one of the characters and its secondary in the presence of some noise introduced by the scanner. This resulted in a lower recognition rate of 94.17% over this set of samples. This however, can be resolved by feeding the rejection information back into the segmentation stage to resegment the word (or subword) at which the error occurred. This work is currently in progress.

8 Conclusion

Arabic characters are first segmented into characters using a newly developed algorithm. which segments the Arabic characters at the right position so that no extra classes are generated due to attached fragments. The secondaries of Arabic characters are also removed using a new algorithm. This reduces the classes of Arabic characters from 120 to 32 different classes.

The main strokes are skeletonized and their features are extracted. These features consist of line segments and polygons. The polygons are fitted to the curves using a newly developed method. Both the features and the information about the secondaries are used to classify the characters in two stages. In the first stage the decision tree is used to classify the characters using the minimum number of features. If any character is rejected in this stage it is passed to the second stage and compared with a set of predefined templates. Characters are classified in this stage according to a minimum distance criterion. A recognition rate of 97.23% over a set of 4260 samples consisting of 4 fonts and 15 sizes per font is achieved.

9 References

1. A. Amin, *et al*, "Handwritten Arabic character recognition by the IRAC system", *5th Int. Conf. Pattern Recognition, Miami, FL*, 1980, pp 729-731.
2. A. Amin, J. Mari, "Machine Recognition and correction of printed Arabic text", *IEEE Trans. Syst. Man Cybern. SMC*, **19**, 5, 1989, pp 1300-1306.
3. K. El-Gowely, O.El-Dessouki, A. Nazif. "Multi-phase Recognition of Multi-font Photocscript Arabic Text", *Proc. 10th Conf. on Pattern Recognition*, 1990, pp 700-702.
4. T. El-sheikh R. Guindi, "Computer Recognition of Arabic Cursive scripts", *Pattern Recognition*, **21**, 4, 1988, pp 293-302.
5. H. Al-Yousefi, S. Udpa, "Recognition of Arabic characters", *IEEE Trans. Pattern Analysis Machine Intell. PAMI*, **14**, 8, 1992, pp 853-857.
6. F. Elkhaly, M. A. Sid-Ahmed, "Machine Recognition of Optically Captured Machine Printed Arabic Text", *Pattern Recognition*, **23**, 11, 1990, pp 1207-1214.
7. L. Lam C. Y. Suen, "Structural Classification and Relaxation Matching of Totally Unconstrained Handwritten Zip-Codes ", *Pattern Recognition*, **21**, 1, 1988, pp 19-31.
8. P.S. P. Wang, Y. Y. Zhang, "A Fast and Flexible Thinning Algorithm", *IEEE Trans. Comput.*, **C**, **38**, 5, 1989, pp 741-754.
9. U. Ramer, "An Iterative Procedure for Polygonal Approximation of Plane Curves", *Comput. Graphics Image process.*, **1**, 1972, pp 244-256.