IEEE int. conf. on Systems, Man, and Cybernetics, volume 1-8, pages 2602-2609. IEEE Xplore, 2007

A Base-line character recognition for Syriac-Aramaic

Elizabeth Tse and Josef Bigun, *fellow of IEEE*

***Abstract*— Serto is the cursive alphabet of Syriac-Aramaic, which is used by the largest corpus of documents in libraries in Aramaic. A lingua franca, and often a source language, Aramaic has influenced major Judaic, Christian and Islamic thoughts as well as the development of science. The script is cursive, e.g. Arabic, and consequently it has a hand-writing appearance compared to Latin. Serto, and Aramaic in practice, has not an automatic character recognition system, OCR. Most library documents are reproductions using printed characters. The readers would strongly benefit from having an OCR, as these reproductions are predominantly books, printed in the pre-computer era. We propose a segmentation-free OCR using linear symmetry features with an individual threshold for the tensors of the characters, and an ordered search sequence. It yields ~ 90 % correctly identified characters in the average. As a first recognition scheme for Serto, it represents a base-line OCR for Syriac-Aramaic.**

I. Introduction

There are many documents written in Aramaic, as this Semitic language is the language spoken and written for the longest uninterrupted period in time. This is partly due to the fact that during nearly 2 millennia starting from 8 BCE it served as a lingua franca in a large geographic region, from the Middle East to the Far East. Syriac is a dialect of Aramaic and is of particular interest as it contains a rich corpus of ancient religious documents important to Judaic, Christian and Islamic thoughts as well as scientific documents of ancient times.

Yet the Serto Script, which is the name of the cursive alphabet of Syriac and represents the largest corpus in libraries, currently does not have a character recognition system. Most of these documents are reproduced by printed letters from the pre-computer era. Consequently a printed Syriac-Aramaic character recognition system is desirable to make these documents computer searchable.

The approach proposed in this paper uses a segmentation-free process using linear symmetry with a threshold of correlation for each character, and an ordered sequence of characters to be searched for. The system, using the same font type and size for the training corpus and testing corpus, has yielded results of approximately 90 percent correctly identified characters for the overall system. As a first recognition scheme for Syriac-Aramaic, the suggested scheme represents a base-line reference system for this cursive alphabet which appears more like hand-writing than printed text when compared to Latin.

There as been no OCR system or scheme for the Serto script of Syriac. However, Clocksin [1] has studied Estrangelo which is another script of Syriac. The two scripts of the language (Serto and Estrangelo) differ significantly both in style and their intended use. The Estrangelo script came to be used more for titles and scripts in ancient monuments and documents whereas Serto is used for the bulk of the documents. Accordingly the overwhelming share of Syriac-Aramaic texts is in Serto.

Additionally, Syriac-Aramaic is spoken daily by a small, but yet still vibrant minority, approximated to 2 million globally. Thus, an OCR for Syriac-Aramaic is highly desirable, both for scholarly and social purposes that include preventing the knowledge about an ancient language from disappearing.

A summary of the OCR systems used by other languages than Aramaic is given in Section II.

II. Related OCR Systems

The two fields which share the most in common with Serto script are Arabic OCR systems and Cursive handwriting OCR systems. Both of these fields deal with cursive characters where characters are connected together and the form of the letter depends on the position where the letter occurs in a word. After describing these OCR systems, we introduce the segmentation-free approach to character recognition.

A. Arabic OCR systems

Arabic writing shares some features with Serto but the two scripts have their differences, which are significant even for untrained Latin readers. The similarities are that both scripts have cursive characters written from right to left.

OCR systems for Arabic continue to be an area of ongoing research. Currently there are two approaches to recognition: holistic and analytical.

The holistic approach looks at the whole word when doing the recognition; there is no attempt to try to break it down into characters for identification. These systems look at words by methods based on Dynamic Programming or Hidden Markov models. [2]

The analytical approach does not look at words as a whole when doing recognition, instead smaller units are used. [2]

Segmentation beyond the word and sub-word segmentation can be done using segmentation into primitives or segmentation into characters.

In most systems segmentation is the source of the majority of errors the system makes. [3]

Manuscript received July 14, 2007.

E. Tse is with Halmstad University, S-301 18 Halmstad, Sweden (e-mail: mi04elts@stud.hh.se)

J. Bigun is with IDE, Halmstad University, S-301 18 Halmstad, Sweden (corresponding author josef.bigun@ide.hh.se).

B. Cursive handwriting OCR systems

Research on OCR systems for cursive handwritten Latin text is very intensive. The study of segmentation is still ongoing with varying degrees of success in character segmentation. Some problems are the start and end points of characters, the thickness of the writing and the style of writing used [4].

The segmentation used currently is either word segmentation or over segmentation using some of the same techniques used in Arabic OCR systems.

C. Using a segmentation-free approach

The system proposed in this paper uses a segmentation-free approach because the Serto script has characters that are cursive with difficult to determine start and end points for characters. This is one difference between Serto and Arabic or the cursive handwritten form of Latin languages. Since segmentation in these languages is difficult and not easy as in scripts like printed Latin, removing the need for segmentation becomes an alternative way of dealing with the problem of segmentation, at least to obtain a quick baseline recognition scheme.

Premaratne [5] has used a segmentation-free approach of character recognition which uses Linear Symmetry (LS) in the recognition process, as we do in this study, but for the Sinhala script (used in Sri-Lanka) with approximately 95% recognition of characters. However, the same accuracy level cannot be expected for Serto, as Serto is cursive whereas Sinhala has isolated characters.

III. The Serto Script of Syriac

As the analysis below at the alphabet level shows, the recognition that needs to be done by an OCR when scanning Serto is more challenging than Latin because the character differences are very subtle and the characters are connected. Once the characters are displayed in their alphabetic form as in Fig. 1, the next step is to look at these characters as they appear in a more natural connecting way, [6].

Serto is written from right to left and from the top of the page to the bottom of the page. The letters are cursive and depending on where in the word or sub-word they are located take on the different shape. Each of the 22 characters have a different form if the character is an isolated character, a character at the end of the word, a character in the middle of a word or a character at the beginning of a word.

There are also some characters which can have another character attached on its right side but cannot have a character connected to it on its left side. These characters only have two forms, an isolated form and a form for a character at the end of the word. Any word which has one of these characters is separated into sub-words, having the character which can only have right connecting characters as the end of the sub-word and the next character in the word as the starting character of the following sub-word in the word.

The number of characters, 22, can shape wise (though not semantically) be expanded to 72 including the variations of each character as shown in Fig.1. Furthermore, there are some special characters used in specific cases not related to location. Adding these 6 special characters to the 72 main characters and their variations, gives a total of 78 shape wise unique characters, see also Section IV E.

Some characters look quite similar, with only small differences. The characters that presented problems at first were ones that were very similar in appearance like: *dālat* (ܕ,ܪ) and *rēš* (ܪ,ܢ); *taw* (ܬ,ܠ) in its form at the beginning of a word or sub-word and *ālap* (ܐ,ܠ) in its form at the end of a word or sub-word; *bēt* (ܒ,ܒ,ܒ) and *kāp* (ܟ,ܟ,ܟ,ܟ) in the form in the middle of a word or sub-word and at the beginning of a word or sub-word; *wāw* (ܘ,ܗ) and *qop* (ܩ,ܩ,ܩ,ܩ); *lāmad* (ܠ,ܠ,ܠ,ܠ) and $^{C}\bar{e}$ (ܥ,ܥ,ܥ,ܥ); and *ḥēt* (ܚ,ܚ,ܚ,ܚ) and *yod* (ܝ,ܝ,ܝ,ܝ).

The characters *dālat* (ܕ,ܪ) and *rēš* (ܪ,ܢ), presented some difficulty in recognizing them as there is only a difference of a dot. The dot is either above or below the main stroke of the character. Other than the location of the dot, both *dālat* (ܕ,ܪ) and *rēš* (ܪ,ܢ) appear the same.

The characters *taw* (ܬ) in its form at the beginning of a word or sub-word and *ālap* (ܠ) in its form at the end of a word or sub-word have similar structures. Both characters have a vertical line with a horizontal line connect to it at the right hand side of the vertical line. The distinction between these two is that for the *ālap* character, a small stroke or tail that comes from the vertical stroke does not finish at the point where it joins the horizontal stroke. There is an easier way to distinguish these two characters when they are in their natural form in a piece of text. The fact that *taw* in this form (ܬ) can only appear at the beginning of a word or sub-word and that *ālap* in this form (ܠ) can only appear at the end of a word or sub-word makes the characters easy to determine once they are examined in a real text.

The characters *bēt* (ܒ,ܒ,ܒ) and *kāp* (ܟ,ܟ,ܟ,ܟ) in the form in the middle of a word or sub-word and at the beginning of a word or sub-word are similar in that they are all characters having a horizontal stroke with a curved stroke starting at the right hand side and ending on the left hand side before connecting again with the horizontal stoke. In Latin these look like a backward c written in cursive form. The main difference between these two is the length and radius of the curved line. With *bēt* (ܒ,ܒ,ܒ) the curve has a large radius and length making it appear more of a smoother curve compared to the more compact curve of the *kāp* (ܟ,ܟ,ܟ,ܟ) in the form in the middle of a word or sub-word and at the beginning of a word or sub-word.

The characters *wāw* (ܘ,ܗ) and *qop* (ܩ,ܩ,ܩ,ܩ) are similar in that they are all very similar to the Latin o. The *wāw* (ܘ,ܗ) character at the beginning of a word is less likely to be confused as it only has no horizontal line at the base of the character. The *wāw* (ܘ,ܗ) at the end of word or sub-word is very similar to all *qop* (ܩ,ܩ,ܩ,ܩ) character variation in that they all have horizontal components at the bottom of the circle. The main way to distinguish these is that *wāw* (ܘ,ܗ) at the end of a word or sub-word only has a horizontal stroke to

the right hand side of the circle part of the character. The *qop* (ﻕ,ﻗ,ﻘ,ﻖ) characters all have a horizontal stroke to the left of the circle part of the character.

The characters *lāmad* (ﻝ,ﻟ,ﻠ,ﻞ) and $^{C}\bar{e}$ (ﺓ,ﺔ,ﺔ,ﺔ) are practically identical except for the length of the stroke in the north-west direction for $^{C}\bar{e}$ (ﺓ,ﺔ,ﺔ,ﺔ)is about half the size as it is for the *lāmad* (ﻝ,ﻟ,ﻠ,ﻞ) characters.

The characters *h̲ēt* (ﻣ,ﻤ,ﻤ,ﻢ) and *yod* (ﻱ,ﻴ,ﻴ,ﻲ) are very similar except that *h̲ēt* (ﻣ,ﻤ,ﻤ,ﻢ) has one more bump or small peak off the baseline of the character compared to the same variation of the *yod* (ﻱ,ﻴ,ﻴ,ﻲ) characters.

A. Understanding how the characters are connected

The Serto script has characters which overlap in the horizontal direction. The overlap happens for characters which are connected to one another where one character has a vertical stroke in the upper direction and the other in the lower direction, or when one character is close to the baseline of another character and the other character to which it is connected has a long vertical stroke, has an angle to it or a curve.

Figure 2 shows some characters from a text which has overlapping bounding boxes.

Both characters are easily identified but where does one character end and the next start? Is there a place on the baseline of where one ends and the other starts? The answer is no, there are not places where you can draw a vertical line and be able to separate the two characters with one being fully on one side of the line and the other being fully on the other side of the line. More sophisticated representation than vertical rectangles are needed, which adds to the implementation complexity.

B. Words and Sub-words

There are some characters that can only be connected to another character on the right hand side of the character. This is the reason why there are sub-words (within words) that actually do not represent complete words but are caused by the writing system. In Fig. 3, a word is separated into sub-words because of the characters which can only be connected on the right hand side. In Fig. 4, two words which appear next to each other are shown.

The spacing between words and sub-words is different as shown in Fig. 3 and Fig 4. There is a much large distance between two words then there are between 2 sub-words. The exact distance depends on the font used, but in general the distance between two words is double the distance between 2 sub-words.

IV. The Recognition Process

The recognition process proposed in this paper examines the correlation of the tensor of the character in the page to be recognized with each of the tensor templates of the characters through a filtering process. To do this each image neighborhood is filtered through the LS tensor yielding the LS tensor image. In the recognition process the scalar products between the tensor image and the tensor template of the character (being searched for) are computed for every pixel via complex correlation filtering. This is repeated for every character of the alphabet until all characters in the image are identified.

If the correlation which is calculated for the pixel being focused on meets or exceeds the threshold for correlation, the pixel is identified as the centre pixel of a possible character. The pixel column and row coordinates are then checked against the vector which stores the list of all identified characters. If there are no centre pixels of previously identified characters which occur inside the exclusion area of the possible character, the character is identified. The column and row value of the pixel, the pixel which was focused on, is added to the vector of identified character along with the ASCII value of tensor template of the character. If there are any previously recognized characters in the exclusion area of the possible character, then character is rejected as a (new) character.

Once a character is either accepted or rejected, the recognition process moves on to the next pixel to be recognized on the page upon which the OCR is being preformed and the correlation is calculated for this new pixel.

A. Linear Symmetry Tensor as Template

The LS Tensor of an image is constructed using four 1-D filters:

dx (Gaussian Kernel) . (1)
dy (= -dx^{T}) . (2)
gx (Gaussian Kernel) . (3)
gy (= gx ^{T}) . (4)

The two derivative convolutions dxf and dyf of the original image are constructed using the pairs dx, gy and dy, gx respectively.

dxf = convolution(gy, convolution(dx, Image) (5)
dyf = convolution(gx, convolution(dy, Image) (6)

The spatial averages of the complex quantity below represents the orientation information of the LS Tensor [7]. Thus we only use 2 components of the direction tensor (which has 3 components) and do so by representing it as a single, but complex valued scalar, LS, for every pixel, for efficient implementation of tensor scalar products.

$$\hat{LS} = (dxf + j * dyf)\text{^}2 \quad \text{where} \quad j = \sqrt{(-1)}$$

$$LS = convolution(Gx, convolution(Gy, \hat{LS})) \quad (7)$$

where Gx and Gy are gaussians that are larger than gx and gy.

B. The size of the local neighbourhood

The LS is constructed by averaging the infinitesimal direction, $\hat{LS}$, of the local neighbourhood as represented by Gx and Gy, an area around the pixel, for each of the pixel in the image.

C. The Frame for comparison

A frame is used to limit the area of the page (to be

recognized) by making an area around the pixel being examined the same size as the size of tensor template of the character being searched for.

D. The Correlation between character template and the image

The Correlation between the character Tensor template and the tensor image of the page which is being searched for characters actually implements a scalar product, and therefore, a complex conjugation is needed before multiplications. This is implemented as follows (with Absolute operation calculating the magnitude of the complex valued pixels):

Absolute(convolution(conjugate(LS Tensor of Character), LS Tensor of Image)) (8)

E. Adding special characters

In the Serto script there are a few special characters that are modified letters. In Fig. 5, we show these special characters. These modifications do not represent new characters.

For ease in the recognition process, however, these are viewed as distinct characters and each is given its own template.

The special character rēš (ﺭ,ﺮ) in its plural form was also the second group of characters our scheme turned out to have difficulties with in preliminary studies with. In the Serto script to denote the plural form, a dot is added to the top of a character. The letter *rēš* (ﺭ,ﺮ) has a dot already above the main part of the character, so a second one is added. This modification made by the written language convention makes it difficult to identify it as a rēš (ﺭ,ﺮ) letter. As shown in Fig 5, this modified letter is made a new character, giving both forms of this letter their own tensor template, the problem is limited; thus meaning there will be less likelihood in misidentifying this letter as dālat (ﺩ,ﺪ).

F. Finding the order of characters to be recognized by minimizing confusion

To find the order in which the characters should be identified, each character was studied in its isolated form as in Fig. 1 then recognized, to identify the other characters with which it could be confused. A matrix is made for each character showing all the characters that are also identified when searching for the original character.

By putting all the characters which can be mistaken for a different letter, before the character it is confused with, the confusion can be reduced.

The second step was to study the letters as they naturally occurred. We used pages of text for this purpose. Each page contained approximately 110 words. Repeating the process used for the isolated characters and by starting from the already partly ordered list derived from looking at the isolated characters, we could obtain a convergence of the order of characters to be recognized.

G. Correlation threshold

The threshold of correlation is a value used as an acceptance level. The character is searched for by examining every pixel currently for easy implementation, though it is possible to avoid searches in the space between text-lines. The scalar product is computed between the LS tensor of the frame and the corresponding tensor template for the character, by using complex correlation. The characters are accepted as possibly recognized if the correlation exceeds the threshold of a candidate character. Otherwise they are rejected.

H. The exclusion area around identified characters

The fact that characters are not segmented means the same character can be identified more than once as the same character or as a different character at neighboring pixels. Setting a box around the found character in which another character may not be identified can solve the problem of multiple identifications. However, the idea of using a box the same size as the template character is not something that would work well in this case because of overlapping characters as discussed in section III A.

Instead, defining an additional area around the character boxes turned out to be a better approach to avoid the overlapping bounding-boxes. As in Fig. 2, the end point and start point overlap causes even the bounding-boxes overlap.

In Fig. 2, the two boxes contain some of the pixels which belong to the other character. In setting a box around each character which is found pending on the bounding-box of the template, the second character, enclosed in the solid box, may not be identified because its box overlaps the dashed box and the solid box would be rejected as a character.

Using a box other than the bounding-box around the centre pixel, allows for more flexibility with little impact on recognition performance. As in Fig. 6, making a number of pixels above and below as well as to the right and to the left of the centre pixel of a character, allows for the same character not to be identified multiple times as the same character and allows for characters that have some overlapping column pixels.

The problem now becomes how large the box around the centre pixel should be. In the approach used in this paper, a standard vertical and horizontal distance around the centre pixel was chosen for all characters.

I. Testing the system

Once the training of the system was completed, 20 new pages of text were used to test the system. These are pages which have not been seen by the system before.

J. Parameters for the testing the system

The testing was performed by using the following parameters: all pages that are processed had a skew angle of less than 1 degree; all non-Serto script characters are removed manually; a single source (one book) is used for all the pages used in tests; all images are scanned in using 300dpi; all templates are made from characters which appear in the single source.

V. The Process to Prepare the System

The recognition process proposed uses a comparison between the correlation of each of the 72 characters (the 22 letters in the alphabet including all the variations) plus the 6 special characters templates and the image to be recognized.

Each of the character templates is searched for in the tensor image (of the page to be recognized) pixel by pixel. Using a frame of pixels of an area comparable in size to the template size, the correlation of the template character and the tensor image is calculated moving through the image (to be recognized) pixel by pixel.

A. Constructing the template

Each of the characters, the letters in all their variations and the 6 special characters need to have templates created.

1) Finding the characters to make the templates of

Here, the characters that the templates are made from are found out of the pages from the book on which the system will perform the character recognition.

2) The tensor template

An adjustable box is used around the character from which the tensor is made to determine the coordinates of the top left pixel's column and row, and the bottom right pixel's column and row of the box. All pixels inside the box will be used in the tensor template.

The process for computing a Tensor described in Section IV.A is used for the template character Tensor except instead of performing the calculation on the whole page, only the area in box described above is used.

B. Finding the initial threshold for the template

Once the tensor is constructed, a threshold is needed. The plot of tensor scalar products is used to find the threshold. The threshold found by examining the plot is a temporary threshold which will be modified when the characters template is studied further together with other character templates and when tests are made on the system.

C. The second step in determining the correlation threshold

Figure 1 is used to test each character to find a better threshold. Each character template is compared with the tensor representation of a page that has all the characters to be recognized. The threshold is changed until a limited number of characters are identified as the character being searched for. It might not be possible to limit the character identified to the one that is being searched for. If there is more than one character that is identified and changing the threshold does not help, then a list is made of the characters that are also confused with the character being searched. This list will be used later to determine the order of characters searched for by the system.

D. Compute the tensor for all the pages which OCR is to be preformed on

The system does not look at the pages as ordinary image files for the analysis. The pages must be in the Tensor representation for the image.

E. The scheme is run for all the characters

The above scheme is run through for all the characters and the results are analyzed. A confusion matrix is made to help analyze the results. For every letter, which has a minimum of 2 characters, the correctly identified characters are counted, the missed characters are counted and all the wrongly identified characters are recorded on the matrix. Once the matrix is finished, it is used to find out which characters need more modifications to the threshold and to the order in which the characters are searched. At the beginning the confusion matrix had entries for all the different characters not just the letters.

F. Determining the order of the characters to be searched for

The results from the confusion matrix are used for determining the order of characters to be searched for in the program. If it appears that one character is being missed when it is being searched for, but found when searched for another character, the missed character is then put before the character with which it is confused.

G. Adjusting the threshold of correlation

The third step in adjusting the threshold is done from the confusion matrix. If too many characters are missed in the test, lowering the threshold may help, or if too many other characters are also found when searching for the character, the threshold can be increased. Before a new full test is made, a smaller test is performed with the newer threshold to see if this change in threshold will improve the results.

H. Perform the test again

The test is preformed again with the changes to the threshold and the changes to the order of the characters to be found. A new matrix is made and the steps from Sections E through G, above, are repeated and the test is redone. This process is continued until the results are at a level that is no longer improved.

VI. Results

The Serto script OCR system proposed in this paper generated approximately 90-percent overall character recognition rate after a test was performed. The individual results for the characters were also high with a few problems achieving character recognition rates beyond 80 percent for certain characters.

The results for each letter and the 6 special characters varied from almost 100 percent recognition to as low as 59 percent. Letters such as *hēt* (ܗ), *tet* (ܛ) *yod* (ܝ), *ḥēt* (ܚ), *nun* (ܢ) and *qop* (ܩ) were found to be less than 80% correctly identified.

There are few characters with low recognition rates; these are characters with high similarity to other characters. Two of the lowest recognition rates were found in *qop* and *yod*.

The *qop* characters were often confused with the characters

hēt and *wāw*. The character *qop* as it appears in the middle and end of a word or sub−word was mistaken for *wāw* as it appears at the end of the word or sub−word.

The *wāw* character as it appears at the beginning of a word or sub−word was often misidentified as *hēt* as it appears at the beginning of a word or sub-word or *hēt* was misidentified as *wāw*.

The *yod* letter had the lowest recognition rate – approximately 59 percent. The character *yod* was often not identified as any character, was misidentified as *hēt* or was misidentified as part of a larger character.

There is also a problem with the character being missed because they are presumed to be part of a large character.

The *yod* character, which is a tiny modification of a straight line, is missed in the identification process because another character which has already been identified is close to the *yod* character. When a check to see if there are any identified characters in the exclusion area of the *yod* character, the simple test used rejects the *yod* character. This is illustrated in the Fig. 6.

These characters also get confused with the *hēt* characters. Two *yod* characters which appear next to each other in a word or sub-word can be misidentified as *hēt* rather then 2 instances of the *yod* character. This problem was also found in human recognition of individual letters.

Further work needs to be done to obtain better recognition rates for these 4 characters so they aren't misidentified as *hēt* or not identified as anything.

Some of the characters experience better recognition rates when they came before or after other characters. An example of these can be seen with character ܘ had to come before ܘ, ܠ and ܠ. Characters ܠ and ܠ had to be before ܡ, ܡ, ܣ, ܣ which had to be before characters ܫ and ܫ.

A solution to the false rejection of the character could be addressed by having each character have an exclusion area best suited to that character. Some larger characters which have the problem of being identified more than once as the same character would need a larger exclusion area and smaller characters like *yod* could have a smaller exclusion area. This has not been attempted due to the limited time-period for the current study, but would be a ripe area for future research.

VII. Conclusion

There are many historical documents which are written in Aramaic, and most of these are written in the form of the Serto script of Syriac. Currently there is no OCR system to put these documents into electronic form; documents are either scanned in as pictures into electronic format or someone takes the time to type the document on a word processor. If the document is turned into a picture, any information needed from the document must be found by reading the whole document. If the document is hand typed into a word processing program it takes a great deal of time. Neither of these methods allow for large amounts of such documents to be turned into an electronic format.

The system proposed in this paper is segmentation-free processing using linear symmetry with a threshold of correlation for each character, and ordered sequence of characters to be searched for; has given results of 90% correctly identified characters for the overall system.

The combination of using both ordered sequence of character and the appropriate threshold of correlation together yielded the best results. However because there is no segmentation done before the recognition process, a character on the original image can be identified as the same character more than once. Adding an exclusion area around the centre pixel around the character and checking to see if any previously identified character has a centre pixel within the exclusion area, helps avoid this problem. If there is previously identified character within the exclusion area, the character is simply not added to the list of recognized characters.

Further work is needed to make the system more flexible in terms of fonts and to speed up the execution of the system (e.g. not processing the inter-line spaces). Yet we believe that the approach proposed here gives a good insight on the challenges and provides for a baseline recognition system. For future research in the character recognition for Syriac-Aramaic, a baseline system would serve as a scheme to be improved upon and/or to measure progress against.

Acknowledgment

The Syriac text is set using the MELTHO fonts from Beth Mardutho: The Syriac Institute [www.BethMardutho.org].

References

[1] Clocksin, W.F. "Handwritten Syriac character recognition using order structure invariance", Proc. 17th International Conference on Pattern Recognition, ICPR04, IEEE Computer Society Press, 2004, pp. 562-565.

[2] Amin, A. "Off-line Arabic Character Recognition: The state of the Art", Pattern Recognition, Vol. 31, No.5, pp 517-530, 1998.4. al-Badr, B and Mahmoud, S, "Survey and bibliography of Arabic optical text recognition". Signal Processing 41, 1995, pp 49-77.

[3] AL-Badr, B. and Mahmoud, S.A. "Survey and bibliography of Arabic optical text recognition". Signal Processing 41 (1995), pp 49-77.

[4] Schlapbach,A and Bunke, H "Off-line Handwriting Identification Using HMM Based Recognizers", Proceedings of the 17th International Conference on Pattern Recognition (ICPR'04), IEEE Computer Society, 2004, pp. 1051-1055.

[5] Premaratne, H.L "A Printed Sinhala Script Recognition System". Thesis for the Degree of Licentiate of Engineering, Technical Report No. 440L, Chalmers University of Technology.

[6] Healey, J. F. "First Studies in Syriac", University of Birmingham, Birmingham, 1980

[7] Premaratne, H.L., and Bigun, J. "A segmentation-free approach to recognize printed Sinhala script using linear symmetry", Pattern Recognition, 37:2081-2089, 2004.

Fig. 1. The Serto Alphabet.

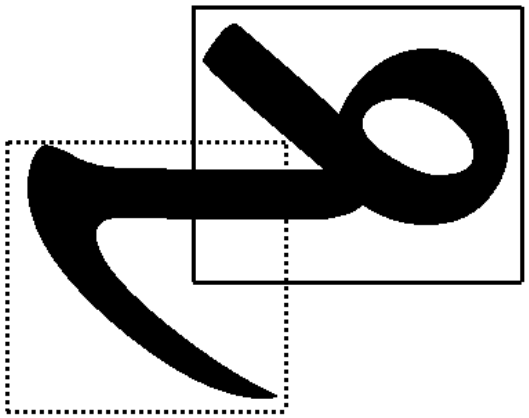

Fig. 2. Two characters that overlap.

Fig. 3. (Left) Word broken into 2 sub-words.

Fig. 4. (Right) Two words.

Fig. 5. Special modification of letters.

Fig. 6. Non-overlapping boxes.

Fig. 7. Misidentified as part of a larger character.