Urdu Optical Character Recognition Technique for Jameel Noori Nastaleeq Script

Engr. Reema Qaiser Khan[1], Engr. Wafa Qaiser Khan[2]

[1,2]*Computer & Software Engineering Department Bahria University Karachi Campus, Karachi, Pakistan*

[1]reema.qaiser@bimcs.edu.pk
[2]wafa.qaiser@bimcs.edu.pk

Abstract—Urdu OCR's have been an object of interest for many developers in the recent years. Active research is being done pertaining to Urdu OCR's, but because of the complexity associated with Urdu fonts; it still lacks perfection halting it from coming up to the surface. The main objective was to create a technique that could be applied to any of the existing Urdu fonts/scripts. In this paper, the authors have developed a technique which is capable of extracting the Urdu font "Jameel Noori Nastaleeq" from images and converts it into editable textual Unicodes. The approach comprises of pre-processing techniques, label connected components, feature extraction, and image comparison. The identified objects are saved as templates which are then compared to the white pixel position length database created by the authors in order to identify the templates which are then converted into Unicode.

***Keywords*— Urdu Optical Character Recognition, Pixels, Image Processing, Connected Component Labeling, Jameel Noori Nastaleeq Script, Unicode.**

I. INTRODUCTION

In order to digitize an entire novel, it could either be done by the protracted process of typing character by character or by using a high end scanner finishing the task in seconds. OCR is the technology that can be used for lengthy documents that can be made available electronically. This quick method is cost effective also. OCR is used in libraries to preserve holdings. OCR is the process where the computer recognizes written or printed text. This is done by analyzing the image, cropping out the preferred word and converting it into character codes such as ASCII/Unicode.

ASCII (American Standard Code for Information Interchange) is the numerical representation of a character for computers to understand [1]. 128 characters are encoded into 7 bit binary integers as illustrated in figure 1.

Similarly, Unicode which is a type of ASCII code but both the methods differ in ranges. ASCII defines 127 characters, which map to the numbers 0–126. Unicode defines (less than) 221 characters which; similarly, map to numbers 0–221 (though not all numbers are currently assigned and some are reserved).

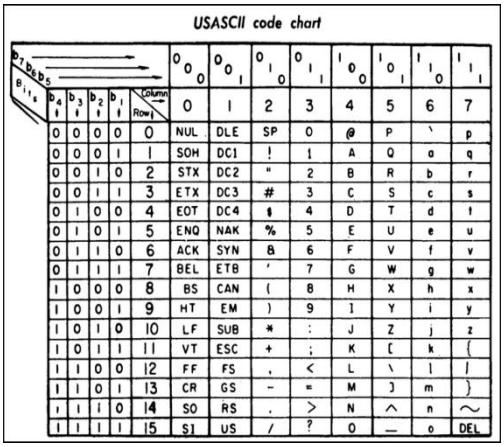

Fig. (1). ASCII chart from a 1972 printer manual. (b1 is the least significant bit) [1]

For every character Unicode provides a unique number as illustrated in figure 2 and 3. Unicode is a computing standard for encoding, representation of text expressed in most of the systems [2].

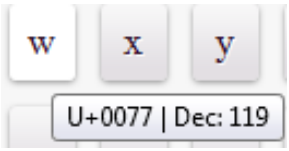

Fig. (2). Latin Small Letter W. where U+ represents the Unicode Number: U+0077 and Dec is value that is used for html or char [2].

0630	1584 ذ	1585 ر	1586 ز	1587 س	1588 ش	1589 ص
0640	1600 ـ	1601 ف	1602 ق	1603 ك	1604 ل	1605 م

Fig. (3). Table represents Urdu Unicode Characters

When Urdu language OCR is discussed, it is realized that this domain needs a lot of attention. The author proposes method to improve the preprocessing techniques for the nastaleeq script [3]. The frequently used font in Urdu documents is Jameel Noori Nastaleeq script. Focusing on one font size 72, author began research and implementations. After setting font size of every character to 72, findings and results were accumulated.

The work that has been done in this field is discussed by some authors. The authors of this paper discuss the complexity of Urdu font Jameel Noori Nastaleeq and also use a cross correlation approach for extracting Jameel Noori Nastaleeq [4]. Javed et al discussed about how difficult it is to edit and separate information present in hard form than soft form [5].

In Section II of this paper, the methodology has been discussed, in Section III, technique has been conversed in detail, in Section IV, the results and other observations has been argued in detail, and in Section V the future work in the domain of Urdu OCR has been put to light.

II. METHODOLGY

The authors have used MATLAB R2013a which provides a multi paradigm numerical computing environment [6]. The authors have built their technique using MATLAB targeting one font size i.e. 72 of the Urdu font Jameel Noori Nastaleeq script.

III. IMPLEMENTED TECHNIQUE

The Urdu OCR technique consists of the following parts:

A. Read Image

In the first step, an image is fed as an input in any of the mentioned file extension tiff, jpeg, or png. The image consists of the Urdu font Jameel Noori Nastaleeq. From this point and onwards, different image pre-processing techniques and methods on the image are applied to get editable Urdu text.

B. Image Pre-Processing Techniques

In this step, different image pre-processing techniques are required to be applied in order to improve the characteristics of the image and by doing so a more clarity in the image is sought; hence, assisting program to accurately identify different objects and later cropping them out. The process is started by converting three dimensional RGB image into grayscale. Once it is done, the image is then converted it into a two dimensional binary image as illustrated in figure 4 and 5. Once the image is converted into binary, it is then converted into negative as illustrated in figure 6. The reason is to convert black text into white.

Fig. (4). Original RGB Image.

Fig. (5). Binarized Image.

When negative is applied to the image; 0 pixels are converted into 1's, and all the 1 pixels are converted into 0's i.e. black pixels turn into white pixels and white pixels turn into black.

Fig. (6). Negative Image.

Next step is to find the edges in the image. The function used for finding the edges in an image is BW = edge (I) [7]. This function returns a binary image of the same size as the original image with the found edges. Here BW is the edge binary image and I is the original binary image. The edges found are denoted by 1's (white pixels) and those that are not edges are denoted by 0's (black pixels) as shown in figure 7.

Fig. (7). Edges found in the image.

Now dilation is applied to the image. It is used to grow or thicken objects in an image. The following set operation is used to define the binary dilation:

$$A \oplus B = \{z | (\beta)\ z \cap A \neq \emptyset\} \qquad \text{(eq. 1)}$$

Dilation of A by B is denoted as $A \oplus B$ is a set consisting of structuring element origin locations. At least some portion of A is overlapped by reflected translated B. β is the reflected structuring element which is a set of pixel locations. These

pixel locations are defined by z. ∅ is an empty set [8] as shown in figure 8 and 9.

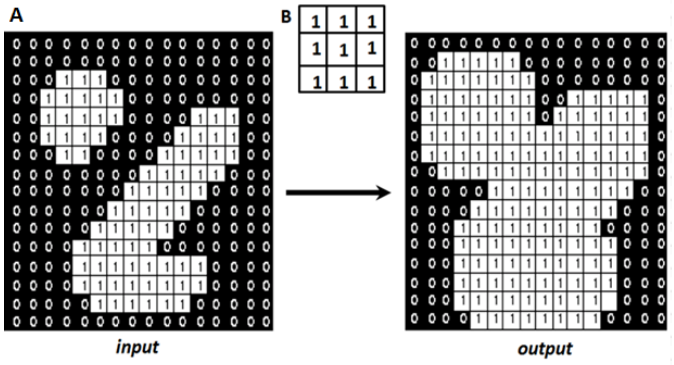

Fig. (8). Effect of dilation using a 3x3 structural element B.

Fig. (9). Dilated image.

Comparing Figure 7 and Figure 9, it is evident that the image has been dilated, thickening the outline .Incase if the image has broken links i.e. the outline of the objects are broken or incomplete, an additional pre-processing step is carried to fill holes. By using this technique, the broken links are re-connected as mentioned in figure 10.

Fig. (10). Region, broken links, and holes filled in image.

All these pre-processing steps are applied to extract the objects (connected white pixels make one object) from the image.

C. Labeling Components

The next step is to label the found objects in the 2D binary image. Using syntax [L,num]=bwlabel(BW,n), value of n can either be 4 or 8 which specifies 4 connected objects or 8 connected objects. It's 8 by default. Where num is the total number of objects found [8].

4-neighbors of pixel p is denoted by N4(p) and 8-neighbors of pixel q is denoted by N8(p). Two pixels p and q are said to be 4-adjacent if:

$$q \in N4(p) \qquad \text{(eq. 2)}$$

Similarly p and q are said to be 8-adjacent if: (Figure 11)

$$q \in N8(p) \qquad \text{(eq. 3)}$$

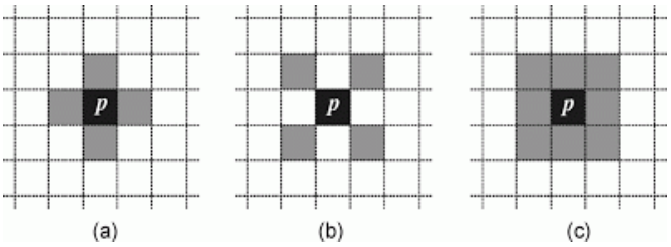

Fig. (11). (a) Pixel p and its 4-neighbors (b) diagonal neighborhood (c) 8- neighborhood.

Where L is a matrix containing values equals to or greater than 0 as illustrated in figure 12. Integer value 1 represents the first object found in the image; integer value 2 represents the second object found in the image and so on. L is of the same size as our binary image (BW).

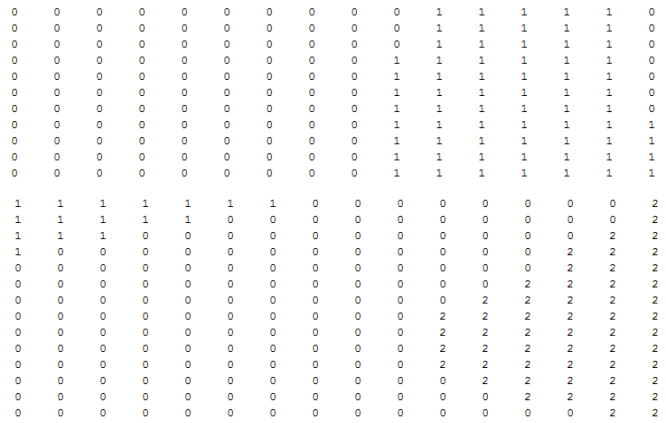

Fig. (12). A portion of the Matrix L representing the objects found. All the 1's represent the first object and all the 2's represent the second object.

The value of num is equal to 16 of the image used by authors and it is proved by bwboundaries [9] as shown in figure 13.

Fig. (13). Total objects found 16.

D. Drawing Boxes Around The Found Objects

Once the objects are identified, the program will draw boxes around them. Region-props is used to measure the image regions [10] as illustrated in figure 14.

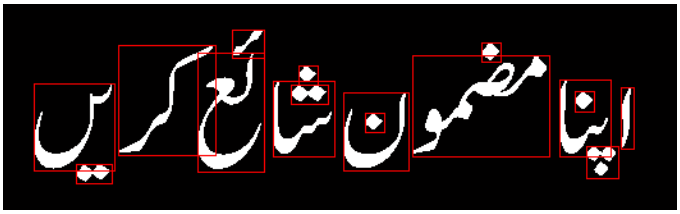

Fig. (14). Boxes drawn around the objects.

E. Crop out The Objects Found

All the boxes drawn around each object is then cropped out using imcrop. These cropped out images are all saved in a separate folder with image type .tif. It starts cropping from right to left as illustrated in figure 15.

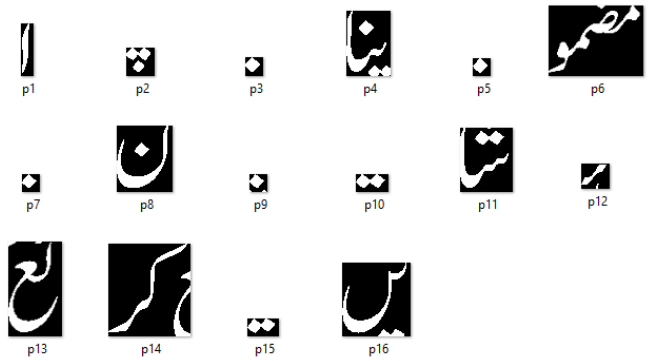

Fig. (15). Folder containing all the cropped images.

F. The Database

The database developed comprises of white pixel coordinates of each Urdu character. For example for the character Kee ے, its white pixel coordinates are calculated.

This is done by the following algorithm:

1. K=s (:,:,1); reads all row and column pixels present in the image, where s is the image containing the character.
2. Find all the white pixel coordinates and save them to the array [y x]
3. Array [y x] is saved in an array named position as position= [x y].

The white pixel coordinates for the word Kee ے are as follows:

Kee = [1 69; 1 70; 1 71; …….; 63 1; 63 2];

4. The total length of position is calculated using the value of length, the specified character is identified.

Kee position length = 914

Other characters are dealt likewise as illustrated in figure 16.

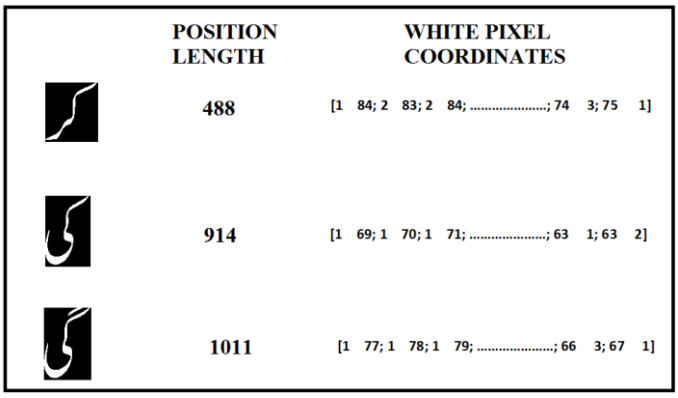

Fig. (16). Database consisting of Urdu characters, their position length and their white pixel coordinates.

G. Reading Image and Matching with the Database

Images are read from the folder in ascending order i.e. from the first image all the way to the last image. In order to match them with the database the following algorithm is applied:

1. K=I (:, :, 1); reads all row and column pixels present in the image. Where I is the image and K stores all those values.
2. Find all the white pixel coordinates and save them to the array [y x]
3. Array [y x] is saved in an array named position as position= [x y].
4. The position length is calculated; that length is matched with the database containing this data, if FOUND it is a MATCH.
5. For certain characters , if position(index)==value; here value is the x or y coordinates values that are stored in position of image

   I. If position index contains that certain value, it signifies that a match is found meaning that this position index and value corresponds to a character stored in the database that has the exact same value on that exact same index.

6. It was observed that when images of font 72 of different dimensions were fed, a single character gave different position length. In order to deal with this situation, a range that would justify the target word was defined.

When match is found, print that Urdu character in a Notepad txt file. This is achieved using the following algorithm and illustrated in figure 17:

1. Open and create Notepad txt file on a certain path.
2. Store Unicode char values of Urdu character in array str.
3. Using unicode2native, str was converted to the native character set of the machine which is the encoded string [11].
4. The encoded string which is of type uint8 is then printed to the opened Notepad txt file. Type uint8 is an unsigned 8 bit integer (range: 0 through 255 decimal).
5. Close the Notepad txt file.

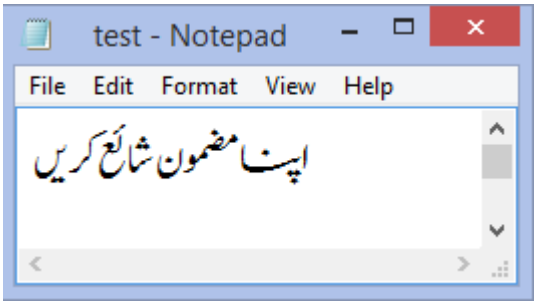

Fig. (17). Urdu characters printed on Notepad text file.

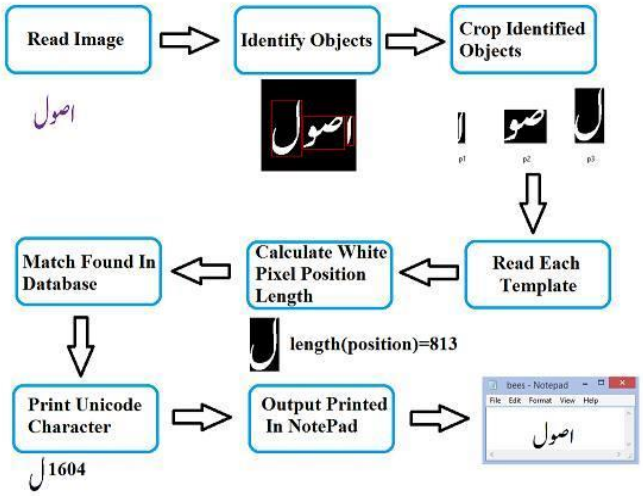

Fig. (18). Work flow of the Jameel Noori Nastaleeq Urdu OCR technique

The above figure 18 elaborates complete flow of the process.

IV. RESULTS AND DISCUSSIONS

During the execution of different Urdu Nastaleeq image templates, the following occurrences were observed:

A. Target Urdu character template with additional white pixels:

In some cases, when the target Urdu character was cropped out, additional white pixels of other neighboring characters were cropped along with. Due to this, the numbers of white pixels in the image were increased. The total number of white pixels represents each individual Urdu character. When neighboring white pixels were cropped out along with the target Urdu character, the white pixel count of that character increased, changing the target Urdu character's white pixel count. This leads to ambiguity as explained in figure 19.

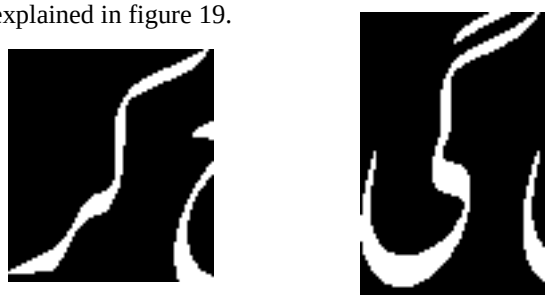

Fig. (19). Examples of two templates containing target character along with portion of neighboring character.

The technique surpasses this situation and is not affected by this intrusion. Since, each Urdu character coordinates were dealt in this paper, position length, and their indexes aforementioned in Section II part G. All the possible samples were identified that can occur along with their white pixels; then simply insert those values in a range so that they can be covered.

B. Similar Words Having Same Coordinates?

It was observed that similar words did not have the same white pixel coordinates as shown in Figure 20. The words Gee گیـ (a) and Kee کیـ (b) do not have the same white pixel coordinates though the highlighted part of Gee گیـ and the whole word Kee کیـ seem to be similar. Same is the case for noon ن and noon sakin ں as shown in Figure 21.

(a) (b)

Fig. (20). (a) The word Kee کیـ. (b) The word Gee گیـ

(a) (b)

Fig. (21). (a) The word Noon Sakin ں (b) The word Noon ن

C. Word Written In Different Font Size Have Different White Pixel Coordinates and White Pixel Positions:

The authors tested an Urdu word using different font sizes i.e. 10, 11, 12, 24, and 74 to see if they all had same white pixel coordinates and white pixel position length. It was concluded that the white pixel coordinates and position length differ. Therefore, if someone wants to store the word Noon ن, all Noons in different font sizes i.e. 8, 9, 10, 11, 12, 14, 16, 18, 20, 22, 24, 26, 28, 36, 48, and 72 must be stored. In order to make a generic Urdu OCR, each word's white pixel coordinates position length must be stored having different font sizes so that it is able to correctly identify words/objects as illustrated in figure 22.

FONT SIZE	NOON	WHITE PIXEL COORDINATES	POSITION LENGTH
10	ں	[1 5; 1 6; 2 6; ....; 4 2; 4 6; 5 5]	373
11	ں	[1 2; 1 6; 1 7; ....; 7 3; 7 4; 7 5]	502
12	ں	[2 3; 2 4; 2 5; ....; 9 2; 9 3; 9 5]	902
24	ں	[2 10; 2 11; 2 12; ...; 17 8; 17 9; 17 10]	1426
72	ں	[1 33; 1 34; 1 35; ....; 48 24; 48 25; 48 26]	4409

Fig. (22). The word Noon ن in different font sizes along with their white pixel coordinates and position lengths.

D. Final Outcome

The authors have applied their technique on different sentences of font size 72 Jameel Noori Nastaleeq. The result is as follows in figure 23:

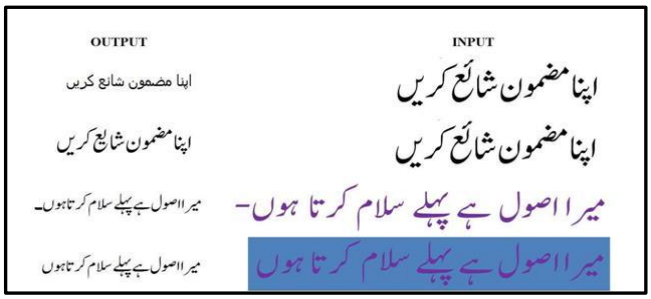

Fig. (23). The result of the applied technique. Input are the text existing in the images and their resultant output.

The images were successfully converted into Unicode Urdu font. It was also observed that it is very important to input the image in a fixed dimension or else the white pixel position lengths may differ. The resolution of the image plays a very important role. If the resolution is poor the white pixel position length may not match.

V. FUTURE WORK

The authors have worked with font size 72 of Urdu Jameel Noori Nastaleeq and now shall work on different font sizes. The author also plans to target other existing Urdu fonts. Gradually new techniques shall be uncovered in the near future.

VI. ACKNOWLEDGMENT

The authors would like to thank Allah S.W.T. and their beloved parents Engr. Qaiser Siraj Khan and Ghausia Qaiser Khan.

REFERENCES

[1] Wikipedia (2014), *ASCII* [Online]. Available: http://en.wikipedia.org/wiki/ASCII

[2] S. A. Sattar, "A technique for the design and implementation of an ocr for printed nastalique text," Ph.D. thesis, Dep. Comp. Sci. Info. Tech., N.E.D. Uni. Eng. Tech., Karachi, Pakistan, 2009.

[3] J. Tariq, U. Nauman and M. U. Naru, "Softconverter: A novel approach to construct OCR for printed Urdu isolated characters," In *Proceedings of 2nd International Conference on Computer Engineering and Technology (ICCET)*, 2010, vol. 3, pp: 495-498.

[4] UniCode Character Table (2014), *UniCode Character Table* [Online]. Available: http://unicode table.com/en/#control-character

[5] S. T. Javed and S. Hussain, "Improving Nastalique specific pre-recognition process for Urdu OCR," In *Proceedings of IEEE 13th International Multi topic Conference (INMIC)*, 2009, pp 1-6.

[6] Matlab 13a (2013), *MathWorks* [Online]. Available: http://www.mathworks.com/

[7] MathWorks, *Find edges in intensity image - MATLAB edge* [Online]. Available: http://www.mathworks.com/help/images/ref/edge.html

[8] R. C. Gonzalez, R. E. Woods, and S. L. Eddins, *Digital Image Processing Using MATLAB*, 5th Ed, Pearson, 2009.

[9] MathWorks, *bwboundaries* [Online]. Available: http://www.mathworks.com/help/images/ref/bwboundaries.html

[10] MathWorks, *Regionprops* [Online]. Available: http://www.mathworks.com/help/images/ref/regionprops.html

[11] MATLAB Function Reference (2014), *unicode2native* [Online], Available: http://matlab.izmiran.ru/help/techdoc/ref/unicode2native.html