Handwriting Arabic Character Recognition Using Features Combination

Fitriyatul Qomariyah  
Faculty of Tarbiyah  
IAIN Madura  
Pamekasan, Indonesia  
fitriyatulqomariyah@iainmadura.ac.id

Fitri Utaminingrum  
Faculty of Computer Science  
Brawijaya University  
Malang, Indonesia  
f3_ningrum@ub.ac.id

Muchlas  
Faculty of Tarbiyah and Teaching Training  
Maulana Malik Ibrahim State Islamic University  
Malang, Indonesia  
acculaszuby@gmail.com

Article History  
Received Feb 20th, 2021  
Revised Oct 2th, 2021  
Accepted Oct 3th, 2021  
Published Oct, 2021

Abstract — The recognition of Arabic handwriting is a challenging problem to solve. The similarity among the fonts appears as a problem in the recognition processing. Various styles, shapes, and sizes which are personal and different across individuals make the Arabic handwriting recognition process even harder. In this paper, the data used are Arabic handwritten images with 101 sample characters, each of which is written by 15 different handwritten characters (total sample 101x15) with the same size (81x81 pixels). A well-chosen feature is crucial for making good recognition results. In this study, the researcher proposed a method of new features extraction to recognize Arabic handwriting. The features extraction was done by grabbing the value of similar features among various types of font writing, to be used as a new feature of the font. Then, City Block was used to compare the obtained feature to other features of the sample for classification. The average accuracy value obtained in this study was up to 82%.

Keywords — arabic character recognition; handwriting arabic character; OCR; features extraction; features combination
1 INTRODUCTION

Optical Character Recognition (OCR) is an electronic machine used to convert handwriting to text that is readable and editable using the computer[1]. The main function of Optical Character Recognition (OCR) is to resemble a human’s reading ability. Thus, the OCR system is a part of artificial intelligence which is also a branch of computer science. The recognition of Arabic handwriting using Optical Character Recognition has been a popular interest among researchers presently[2].

However, the recognition of Arabic handwriting is more difficult and more complex than recognition processing of other characters from other languages[3]. The shapes and sizes of every font in Arabic characters can be different among individuals which is the main challenge of this study. Besides, unlimited variations of styles in Arabic handwriting that influence the accuracy of the recognition processing[4], several methods have been deployed in the recognition of Arabic handwriting[5].

One of the methods is by using features extraction [6]. The selection of appropriate features influences the recognition processing especially, the recognition of Arabic handwriting. Various Arabic fonts that have intense similarity among each other and a variety of writing styles across individuals are complex problems in determining the best feature to be used for the recognition processing to get the most salutary result[7].

Several studies have been conducted regarding this issue. Lawgali and Bouridane have conducted a study to recognize Arabic handwriting using Discrete Cosine Transform (DCT) and Discrete Wavelet Transform (DWT) extraction features [8]. The accuracy of using DCT was 59.81% for DWT on a figure of 64x64. M.A. Abdullah and H.H Al-Fraidi et.al also administered a study to recognize Arabic handwriting [9]. In their study, the recognition process was conducted in several phases; extracting the shapes and features of the fonts. Using this method, the accuracy was 81%. Another study was also done by Sahlol and Suen who used integral projection (vertical and horizontal projection) and some other features with an accuracy score of 88% [10]. But this research is not good for the recognition of Arabic characters.

In this study, a new approach of utilizing the form of images method was proposed. This approach was constructed by integrating similar features of a character from a different person’s handwriting to make a new feature. The new feature was then used as a feature for certain characters. There were 101 different shapes of characters from 15 different handwriting of each of the characters. In summary, there were a total of 1,515 samples used in this study. A detailed explanation is appended below in Fig. 1.

![Diagram](image)

Figure 1. Design system

The details of the recognition process will be explained in the second part. Whilst, for the overcome and conclusion will be explained in the next part.
2 METHOD

Preprocessing is the initial process of improving an image to remove noise. As stated by Bahri and Maliki, preprocessing is a process to remove parts that are not needed in the input image for further processing [11]. Preprocessing in this study through several processes: convert the element of RGB (red, green, and blue) to grayscale, differentiate the foreground from the background using a threshold value, the change value 0 is turned to 1 and vise versa, thin out the image pixel and resize the image sizes to 81x81 pixels. Preprocessing is to facilitate researchers to identify each handwritten Arabic character.

2.1 Arabic Characters

The Arabic character is also called the Hijaiyyah letter which consists of 28 letters, (29 if hamzah is considered to be a separate letter). Alif, waw, and ya’ are used as long vowels (such as aa, ii, uu), Translation results as a diphthong or a double sound (such as ai, au), and also as a weak consonant.[12]

The Arabic alphabet is written from right to left. Each letter has several different forms according to the position of the letter in a word. There are 84 different shapes according to the position of the letter (beginning, middle, end, and independent).[10]

Additional components that are connected to the body of Arabic letters are crucial pieces of information such as additional components of dots [13]. Note that, same shapes with different additional components produce different letters, see Fig. 2.

2.2 RGB to Gray

The RGB image is an image with each pixel consisting of three colors (red, green, and blue)[14]. The RGB color model is a color model based on the concept of adding strong primary light, namely red, green and blue. In a room where there is absolutely no light, the room is completely dark. No lightwave signal is absorbed by our eyes or RGB (0, 0, 0)[15]. If we add red light to the room, then the room will change color to red. For instance, in RGB (255, 0, 0) all objects in the room will only be seen in red when our light is replaced with green or blue. As we know, the RGB is a coloring system for digital appearance and is widely used for computer monitors, videos, cellphone screens, etc. The RGB color system consists of 100% Red, 100% Green and 100% Blue which results in 100% white. There is no black in RGB[16].

A grayscale image is a monochrome image consisting of brightness information only [17]. RGB to Gray is a process to convert the element of RGB to grayscale, see Fig. 3. Luminance is a standard algorithm of image processing. It is exercised by MATLAB with "rgb2gray" function which is often used in computer vision. Luminance [18] is designed to match the human's perception of brightness through the weight combination of RGB by the following Equation 1

\[
G_{\text{Luminance}} = 0.2989 \times R + 0.5870 \times G + 0.1140 \times B \tag{1}
\]

In which R is the red element, G is the green element, and B is the blue element.

2.3 Otsu Thresholding

Thresholding is the simplest method of image segmentation. From the grayscale image, thresholding can be used to form a binary image. A binary image is a digital image that has only two possible values for each pixel. The two colors are black and white or 0 and 1.[19]

Otsu thresholding is a simple segmentation method in segmentation techniques, so it can be easier to divide homogeneous areas based on similarity criteria to identify objects. Otsu Thresholding is a method that differentiates the foreground from the background using a threshold value.
However, poor brightness and different background colors are the traits to determine the threshold value. For instance, it is assumed that \( C \) is an original image that has bright \((C_1)\) and dark\((C_2)\) sides, while \( P(s) \) is the probability of every side which can be formulated as follows this Equation 2 and Equation 3 [20]:

\[
P_1(s) = \sum_{i=0}^{s} P_i \text{ for } C_1[s > 0] \tag{2}
\]

\[
P_2(s) = \sum_{i=s+1}^{L-1} P_i = 1 - P_1(s) \text{ for } C_2[s + 1, L - 1] \tag{3}
\]

\( P_1(s) \) and \( P_2(s) \) are probabilities of the dark and the bright side of \( C_1 \) and \( C_2 \), whilst \( P_i \) is the normal histogram of the overall size of the image. To determine the threshold value \( k \), bimodal histogram, statistics stationery, hence, adaptive and uniform illumination area can be modified to obtain the threshold value using this Equation 4:

\[
\hat{k} = \frac{\sigma^2_B}{\sigma^2_G} \tag{4}
\]

In which \( \hat{k} \) is threshold value, \( \sigma^2B \) is the global variant of the overall picture, while \( \sigma^2G \) is the variant across classes. Fig. 4 describes the result of converting RGB image to binary image by using otsu.

![Figure 4](image)

**Figure 4.** RGB Image (a) and Binary image (b)

2.4. **Complement**

In complementary grayscale or color images, each pixel value is subtracted from the maximum pixel value supported by the class (or 1.0 for double-precision images). The difference in pixel values is used as the image output pixel value. In the output image, light areas become darker and dark areas become lighter. For color images, green becomes magenta, blue becomes yellow, red becomes cyan, and vice versa [21]. In this process, the pixel value 0 is turned to 1 and vice versa. See the example in Fig. 5 and Fig. 6.

![Figure 5](image)

**Figure 5.** Example of Complement

![Figure 6](image)

**Figure 6.** Binary Image (a) and Complement image (b)

2.5. **Thinning**

Thinning algorithm is an algorithm that slenderizes the pixel of an image[22]. Thinning is an operation of image processing that operates on a binary image by scraping parts of the object into a line that represents the framework of the object, it can be seen in Fig. 8. This algorithm works by doing reduplication of the pixel layer until there is no more pixel to delete[23]. This method performs pixel removal. Usually, the checking process is done twice, and if there is no more change, then the reduplication stops. Algorithm ZS is a thinning algorithm proposed by Zhang and Suen in 1984 that uses a 3x3 system as shown in Fig. 7 [24].

![Figure 7](image)
There are two sub-iterations in this algorithm: to remove the restraint pixel of South-East and the pixel point of North-West. The second is to remove the restraint pixel of North-West and the pixel point of South-East.

At first sub-iteration (odd iteration), $p_1$ is removed from the pattern if $p_1$ is following the conditions which can be written by Equation 5, Equation 6, Equation 7 and Equation 8:

$$2 \leq B(p_1) \leq 6$$  \hspace{1cm} (5)
$$A(p_1) = 1$$  \hspace{1cm} (6)
$$p_2 \times p_4 \times p_6 = 0$$  \hspace{1cm} (7)
$$p_4 \times p_6 \times p_8 = 0$$  \hspace{1cm} (8)

In the second sub-iteration (even iteration), $p_1$ is removed from the pattern if $p_1$ is following the conditions that can put in writing in Equation 9, Equation 10, Equation 11 and Equation 12:

$$2 \leq B(p_1) \leq 6$$  \hspace{1cm} (9)
$$A(p_1) = 1$$  \hspace{1cm} (10)
$$p_2 \times p_4 \times p_6 = 0$$  \hspace{1cm} (11)
$$p_4 \times p_6 \times p_8 = 0$$  \hspace{1cm} (12)

Where $A(p_1)$ is the number pairs of 0-1 (white-black) in transverse clockwise from 8 neighborhood in $(p_1)$ like $(p_2), (p_3), (p_4), ..., (p_8), (p_9)$. $B(p_1)$ is the number of which not zero in 8 neighbors in the following Equation 13:

$$B(p_1) = \sum_{i=2}^{9} P_i$$  \hspace{1cm} (13)

If no conditions are matched, $p_1$ is not removed from the foreground.

2.6. Resize Image

Image resizing means changing the size of the digital image in pixels. There are times when the size changes to be smaller or larger. When the size is small or large, it then gets to be done proportionally both on the length and width of the image. In this study, we resize the image sizes to 81x81 pixels. This process is done to equalize all the image sizes so that the feature extraction process can be effective. Fig. 9, Fig. 10, Fig. 11, and Fig. 12 are example image with different Arabic letters.

Arabic characters are quite different from other characters of other languages. In the recognition processing of Arabic handwriting in computer vision, similar features of some
characters are the traits to solve. In addition to this complex problem, another challenging trait is also triggered by various styles, shapes, and sizes of handwriting among individuals. Therefore, the selection of the feature is the key to the success of recognition processing. In this study, a new features extraction was proposed to be used in Arabic handwriting recognition. This study involved 101 samples of characters which each was written by 15 different handwriting characters (the total of the sample was 101x15) with the same image sizes of 81x81 pixels as shown in Table 1. Firstly, data of 15x30 Arabic handwriting images were coded and changed to binary codes (consisting of 0 and 1) where foreground means 1 and background means 0 as shown in Fig. 5 and Fig. 6. 15 of different writing styles of a character were used and analyzed. Then, features of the data were extracted by changing the matrix dimension, see Fig. 14 which illustrates pixel value from Fig. 13 in m x n, then converted to 1xn as shown in Fig. 15. After that, the next step was to find the same pixel among the nine different handwriting to be used as the new feature of the character as illustrated in Fig. 16 in which the pixel in pink shows a similar pixel value.

![Figure 13. Example of Baa](image1)

![Figure 14. Binary image of Baa](image2)

![Figure 15. One Dimension of Binary Images](image3)

![Figure 16. The Same Pixels Value of Baa One Dimension](image4)
After all the features from the samples of the previous processing were obtained, the next step was to compare each of the features with all the samples using City Block. City Block is an algorithm that is used to measure the gap or similarity between the two objects. The formulation of the City Block is presented as Equation 14 [25]:

$$D = \sum |F_i - \bar{F}_i|$$  \hspace{1cm} (14)

In which $D$ is the degree of distinction between two objects, $F_i$ is input object, and $\bar{F}_i$ is compared object/output.

3 RESULT AND DISCUSSION

a. Trial

This study involved 101 samples of characters which each was written by 15 different handwriting characters (the total of the sample was 101x15). The data used in this study is preprocessed first, the preprocessing carried out includes convert the element of RGB (red, green, and blue) to grayscale using equation (1), differs the foreground from the background using threshold value (2-4), changing the value 0 to 1 or vice versa using the complement algorithm, attenuation image pixels using a thinning algorithm (5-13), the last preprocessing resize image data of handwriting Arabic character with the same size of 81x81. The example of data used in this study is shown in Table 1.

Table 1. Handwriting Arabic Character Data

<table>
<thead>
<tr>
<th>No</th>
<th>Character</th>
<th>First</th>
<th>Middle</th>
<th>End</th>
<th>Isolated</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alif</td>
<td>-</td>
<td>ایا</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Baa</td>
<td>ببب</td>
<td>ببب</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>Ta</td>
<td>تتت</td>
<td>تتت</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>Tha</td>
<td>وفف</td>
<td>وفف</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>Gheem</td>
<td>ججج</td>
<td>ججج</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

12 Seen
13 Sheen
14 Sad
15 Dad
16 Tah

This article is distributed under the terms of the Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License. See for details: https://creativecommons.org/licenses/by-nc-nd/4.0/
First, perform preprocessing on all samples of Arabic handwriting. After preprocessing, the next process is features extraction and recognition using City_Block (14). The result of handwriting Arabic character is presented using the calculation accuracy as shown below.

\[
\text{System Performance} = \frac{\text{The Same Pixels of input and fitur}}{\text{All Pixels of input and fitur}} \times 100\% \quad (14)
\]

Table 2. The Experiment Result of Handwriting Arabic Character Data

<table>
<thead>
<tr>
<th>No</th>
<th>Character</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alif</td>
<td>85.51%</td>
</tr>
<tr>
<td>2</td>
<td>Baa</td>
<td>84.41%</td>
</tr>
<tr>
<td>3</td>
<td>Ta</td>
<td>80%</td>
</tr>
<tr>
<td>4</td>
<td>Tha</td>
<td>82%</td>
</tr>
<tr>
<td>5</td>
<td>Gheem</td>
<td>81.1%</td>
</tr>
<tr>
<td>6</td>
<td>Ha’a</td>
<td>82.30%</td>
</tr>
<tr>
<td>7</td>
<td>Kh’a</td>
<td>80%</td>
</tr>
<tr>
<td>8</td>
<td>Dal</td>
<td>82%</td>
</tr>
<tr>
<td>9</td>
<td>Dhal</td>
<td>81%</td>
</tr>
<tr>
<td>No.</td>
<td>Character</td>
<td>Accuracy</td>
</tr>
<tr>
<td>-----</td>
<td>-----------</td>
<td>----------</td>
</tr>
<tr>
<td>10</td>
<td>Ra</td>
<td>85%</td>
</tr>
<tr>
<td>11</td>
<td>Zeen</td>
<td>81%</td>
</tr>
<tr>
<td>12</td>
<td>Seen</td>
<td>85%</td>
</tr>
<tr>
<td>13</td>
<td>Sheen</td>
<td>80%</td>
</tr>
<tr>
<td>14</td>
<td>Sad</td>
<td>81%</td>
</tr>
<tr>
<td>15</td>
<td>Dad</td>
<td>80%</td>
</tr>
<tr>
<td>16</td>
<td>Tah</td>
<td>86.2%</td>
</tr>
<tr>
<td>17</td>
<td>Dhad</td>
<td>81.1%</td>
</tr>
<tr>
<td>18</td>
<td>Aen</td>
<td>83%</td>
</tr>
<tr>
<td>19</td>
<td>Ghen</td>
<td>81.3%</td>
</tr>
<tr>
<td>20</td>
<td>Fa</td>
<td>80.5%</td>
</tr>
<tr>
<td>21</td>
<td>Qaf</td>
<td>80%</td>
</tr>
<tr>
<td>22</td>
<td>Kaf</td>
<td>82.7%</td>
</tr>
<tr>
<td>23</td>
<td>Lam</td>
<td>83.1%</td>
</tr>
<tr>
<td>24</td>
<td>Meem</td>
<td>82.9%</td>
</tr>
<tr>
<td>25</td>
<td>Noon</td>
<td>81.5%</td>
</tr>
<tr>
<td>26</td>
<td>Ha</td>
<td>79%</td>
</tr>
<tr>
<td>27</td>
<td>Waw</td>
<td>86%</td>
</tr>
<tr>
<td>28</td>
<td>Ya</td>
<td>78%</td>
</tr>
</tbody>
</table>

From the results obtained, by using Features Combination, the letters Ya and Ha have the smallest accuracy values compared to other letters, which are about 78% and 79%. This is because the shape of the letters is almost similar to other letters, the level of inclination of the writing, the position of the points that are not the same from the 10 samples, different writing styles. In addition, the shape of the writing similarity, the slope, and the size of the writing greatly affects the level of accuracy, the detail of each letter can be seen in Fig. 17.

### 4 CONCLUSION

The results, based on the experiment can be concluded that the recognition of handwriting Arabic characters using features combination method is very good for recognizing handwriting Arabic characters. In this study, we used image sizes of 81x81 pixels, we got a fairly good accuracy value of up to 80%. The great quality of data will increase the accuracy. Accuracy is influenced by several factors, such as the degree of similarity, slope, and size.

### AUTHOR’S CONTRIBUTION

As the first author, Fitriyatul Qomariyah contributed technically and wrote to this research. She did the research with the supervision from the second author namely Fitri Utaminingrum for ideas and technical aspects. Last, the research got theoretically supervision from the third author with Muchlas.

### COMPETING INTERESTS

Comply with the publication ethics of this journal, Fitriyatul Qomariyah, Fitri Utaminingrum and Muchlas as the authors of this article declare that this article is free from conflict of interest (COI) or competing interest (CI).

### REFERENCES


