ABSTRACT

Optical Character Recognition (OCR) is a type of computer software designed to translate images of handwritten or typewritten text (usually captured by a scanner or a camera) into machine-editable text by recognizing characters at high speeds one at a time. OCR began as a field of research in pattern recognition, artificial intelligence and machine vision. It is becoming more and more important in the modern world according to economic reasons and business requirements. It helps humans ease their jobs and solve more complex problems by eliminating the time-consuming spent by human operators to re-type the documents and reduce error-prone processes.

The presence of any type of noise or a combination of them can severely degrade the performance of OCR system. Though, a number of preprocessing techniques are considered in the present work in order to improve the obtained accuracy of the recognized text. An OCR system for 3185 training samples and 13650 testing samples is presented for multi-font English texts. Experiments have shown that wavelet features produce better recognition rates 96% than DCT features 92%. An improvement overall recognition rates (about 3%) are obtained after classifying characters according to the proportion of Height to Width feature to produce 99% for wavelet and 95% for DCT.

Keywords: DCT, Feature Extraction, Optical Character Recognition (OCR), Pattern Recognition, Segmentation, Wavelet Transform.

1. INTRODUCTION

Since the start of the computing era, information has been represented digitally so that it can be processed by computers. Approximately, more than 200 million paper books are being published yearly. Paper books and documents were abundant and widely being published at that time; and hence, there was a need to convert them into digital format. OCR was invented to translate the traditional paper-based books into digital e-books(i.e., electronic files). It was estimated that over 2
Million e-books are available for download on the Internet. E-books require less storage space than paper books, they can also be replicated many times, shared online, and digitally processed easily, mainly searched, translated, edited, and annotated. OCR systems are not that perfect as they are erroneous and exhibit spelling errors in the recognized output text, especially when the images being scanned are of poor printing quality [1, 2].

OCR is one of the most fascinating and challenging areas of pattern recognition with various practical applications: Automated postal address reading, ZIP code reading, checks, payment slips, income tax forms, business forms, automatic car plate-number recognition, and it can be used as an aid for visually handicapped people when combined with speech synthesizer [1, 3, 4].

Automatic character recognition is a subfield of pattern recognition and can either be on-line or off-line. On-line recognition refers to those systems where the data to be recognized is input through a tablet digitizer, which acquires in real-time the position of the pen tip as the user writes. In contrast, off-line systems input the data from a document through an acquisition device, such as a scanner or a camera. Off-line character recognition is moreover divided into two categories: machine printed and handwritten [5].

The printed texts include all the printed materials such as: books, newspapers, magazines, and documents which are the outputs of typewriters, printers or plotters. OCR systems of machine-printed documents can be classified into [5, 6]:

- Mono-font OCR systems that deals with documents written with one specific font,
- Multi-font OCR systems that handles a subset of the existing fonts (recognition of more than one font),
- Omni-font OCR systems that allows the recognition of characters in any font.

Today many types of OCR software available in the markets like: Desktop OCR, Server OCR, Web OCR etc. Accuracy rate of any OCR tool varies from 71% to 98% [7].

2. PROBLEM DEFINITION

In modern society, we rely heavily on computers to process huge volumes of data. Related to this and for economic reasons or business requirements, there is a great demand for quickly converting the printed information in a document into an edited text in the computer. Often these data exist on paper and they have to be typed into the computer by human operators. Such time-consuming and error-prone processes have been lightened by the invention of OCR systems [1]. Unfortunately, OCR systems are still erroneous and inaccurate, especially when the source document is of low printing quality [2]. Therefore, accuracy of these systems can be dependent on text preprocessing and segmentation algorithms. Sometimes it is difficult to retrieve text from the image because of different size, style, orientation, complex background of image … etc which produce misspellings in the recognized text [2, 7].

OCR technology allows machine to recognize the text automatically in the same way as the combination of eye and mind of human body. In development of computerized OCR system, few problems can occur [7]:

- There is very little visible difference between some letters and digits for computers to understand. For example it might be difficult for the computer to differentiate between digit '0' and letter 'o' / 'O',
- It might be very difficult to extract text, which is embedded in very dark background or printed on other words or graphics.
3. AIM OF THE WORK

This paper aims to build an OCR Multi-font system which converts the printed English texts in a paper document (optical patterns exist in a digital image) into an edited text (its corresponding alphanumeric form or other symbols) in the computer in order to:

- eliminate the time-consuming spent by human operators to re-type the documents (the huge volumes of data),
- reduce the possible errors occurred in the typing process,
- save money (by cancelling the need of human typists),
- preserve space needed for paper books.

4. THE PROPOSED OCR SYSTEM

The following block diagram, shown in Figure (1), illustrates the proposed OCR system model.

![Block-Diagram of the Proposed OCR System Model](image)

Figure (1): Block-Diagram of the Proposed OCR System Model

The input scanned image text is passed through a sequence of preprocessing steps (noise removal, foreground/background separation, normalization, and binarization) prior to characters segmentation phase. Then, feature extraction methods (Discrete Cosine Transform (DCT) followed by Zigzag process, or Wavelet Transform (WT)) are applied to the segmented characters. The obtained feature-set is either stored in database as templates or references during the training phase when building database (DB) of characters features-set or is compared directly during the testing phase to those DB references in a pattern matching stage. Finally, decision rule is applied to produce recognition results beside the best matched characters.
4.1 The Input Scanned Image

The input documents used for training and testing are scanned and digitized by a page scanner at 300 dpi resolution connected to a computer system and saved in BMP format of 256 gray-levels. A number of scanned images that are used as inputs to our OCR system model are shown in Figure (2) below:

![Figure (2): 3–Samples of Scanned Images](image)

4.2 Document Image Preprocessing

Digital Images are generally corrupted by noise during the process of acquisition and transmission. This noise degrades the quality of digital image which produces several tiny dots scattered due to uneven gray-scale intensities which causes poor recognition rates. Consequently the performance of any system manipulating these images is also decreased. Therefore, removal of noise in document images corrupted by Gaussian and Impulse noises before OCR is important to guarantee better accuracy of characters recognition [1, 8].

Thus, image enhancement techniques adopted in this system model in the following sequence are employed prior to segmentation in order to simplify the process of characters segmentation [9, 10]:

a) **Noise Removal**: is applied to the scanned document images for two primary purposes: to eliminate the noise and to give an image a softer effect. The spatial convolution mask of Gaussian filter used for low-pass filtering is shown in Figure (3).

![Figure (3): Gauss core – weighted on distance (Gaussian filter)](image)

Gaussian filter smoothen the image to match the pixels nearby in a way that no point in the image differ from its surroundings to a greater extent. Image smoothing is accomplished in the spatial domain to remove errors, incorrect data and simplify the acquisition process of patterns [8, 10, 11].
b) **Foreground/ Background Separation:** is the process of separating the foreground regions (the area of interest containing the printed-text) in the image from the background regions (the useless area outside the borders of printed-text). The background regions generally exhibit a very low gray-scale variance value, whereas the foreground regions have a very high variance. Hence, a method based on variance thresholding can be used to perform the separation. Firstly, the image is divided into blocks and the gray-scale variance is calculated for each block in the image. If the variance is less than the global threshold, then the block is assigned to be a background region; otherwise, it is assigned to be part of the foreground. The gray-level variance for a block of size \( W \times W \) is defined as [9, 11]:

\[
V (k) = \frac{1}{W^2} \sum_{i=0}^{W-1} \sum_{j=0}^{W-1} \left( I(i, j) - M(k) \right)^2 \quad \ldots \ldots \quad (1)
\]

where \( V(k) \) is the variance for \( k^{th} \) block, \( I(i, j) \) is the gray-level value at pixel \((i, j)\), and \( M(k) \) is the mean gray-level value for the \( k^{th} \) block.

c) **Normalization:** is utilized to reduce the effect of non-uniform intensities and improving image quality by stretching its histogram. To be able to normalize an image the area which is to normalize within has to be known. Thus it is necessary to find the highest and the lowest pixel value of the current image. Every pixel is then evenly spread out along the scale by the following equation [10, 11]:

\[
N(i, j) = \frac{I(i, j) - I_{\text{min}}}{I_{\text{max}} - I_{\text{min}}} \times M \quad \ldots \ldots \quad (2)
\]

where \( I(i, j) \) is the gray-level value at pixel \((i, j)\), \( I_{\text{min}} \) is the smallest gray-level value found in the image, \( I_{\text{max}} \) is the largest gray-level value found in the image, \( M \) represents the new maximum gray-level value of the scale (i.e., \( M = 255 \)), and \( N(i, j) \) represent the normalized gray-level value at pixel \((i, j)\).

d) **Binarization:** is the process of turning a gray-scale image into a binary image (only two levels of interest 0 and 1) in order to improve the contrast, and consequently facilitates the feature extraction process. It is impossible to find a working global threshold value that can be used efficiently on every image because of the variations among the scanned images. Therefore algorithms to find the optimal value, based on *localized thresholds*, must be applied separately on each image to get a functional binarization. The image is partitioned into smaller blocks and threshold values are then calculated for each of these blocks. This enables adaptations that are not possible with global calculations. Localized thresholds demand a lot more calculations but mostly compensate it with better results [9, 11]. The local mean threshold for \( k^{th} \) block of size \( W \times W \) is computed below:

\[
\text{Local Mean} (k) = \frac{1}{W^2} \sum_{i=0}^{W-1} \sum_{j=0}^{W-1} \text{Block} (i, j) \quad \ldots \ldots \quad (3)
\]

where \( \text{Block} (i, j) \) is the gray-level value at pixel \((i, j)\). If the pixel value is lower than the threshold then the pixel is assigned to be part of the *Printed-text*; otherwise, it is assigned to be part of *Background*. 
Figure (4) illustrates the effect of applying the document image preprocessing techniques to a scanned document sample.

From Figure (4.c), it is noticeable that background regions are specified by black color, which is only to indicate the effect of this process despite the fact that in practical application the background is represented by white color.

**4.3 Characters Segmentation**

Segmentation is an important phase and the accuracy of any OCR heavily depends on it, where incorrect segmentation leads to reduction in recognition accuracy. Segmentation is the process that divides the whole document into smaller components, which include [6, 12]:

- Line,
- Word, and Character segmentation

The procedure adopted in this work for analyzing images to detect characters, as shown in Figure (5), is listed in the following sequence:

**Step1:** Perform "Row Scan" (Line segmentation) to find the number of lines and boundaries of each line in any input document image within which the detection can proceed.
Algorithm 1

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a)</td>
<td>Get image data from the processed image.</td>
</tr>
<tr>
<td>b)</td>
<td>Initialize lines boundaries (T: Top, B: Bottom) to -1.</td>
</tr>
<tr>
<td>c)</td>
<td>Perform row scan (from 1st to last row) for pixels value= 0 (i.e, black).</td>
</tr>
<tr>
<td>d)</td>
<td>Set the number of lines to 0.</td>
</tr>
<tr>
<td>e)</td>
<td>If black pixel is detected then register T as the top of the current line and move the pointer to the next row &amp; 1st column. Otherwise, continue to the next pixel (from left to right).</td>
</tr>
<tr>
<td>f)</td>
<td>If black pixel is found and T &lt;&gt; -1 then update B as the bottom of the current line and move the pointer to the next row &amp; 1st column.</td>
</tr>
<tr>
<td>g)</td>
<td>If no black pixel is found in the row and T &lt;&gt; -1 then increment the number of lines by 1.</td>
</tr>
<tr>
<td>h)</td>
<td>Start below the bottom of the last line found and repeat steps e) – g) to detect subsequent lines (Stop when the bottom of image is reached).</td>
</tr>
<tr>
<td>i)</td>
<td>Process dot &quot;.&quot; problem spacing found in (i and j) characters by merging lines below certain space threshold according to the font type &amp; size and decrement the number of lines by 1 (if such case occurred).</td>
</tr>
<tr>
<td>j)</td>
<td>Print out the number of lines detected and draw line boundaries on the processed image for each detected lines (for: T–1 and B+1).</td>
</tr>
</tbody>
</table>

Step 2: Perform "Column Scan" (orthogonally from 1st to last column) only for detected lines. Thus, detecting characters in an image does not necessarily involve scanning the whole image all over again.

Algorithm 2

<table>
<thead>
<tr>
<th>Step</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>a)</td>
<td>Initialize characters boundaries (L: Left, R: Right) to -1.</td>
</tr>
<tr>
<td>b)</td>
<td>Perform column scan (from 1st to last column) for pixels value= 0 (i.e, black).</td>
</tr>
<tr>
<td>c)</td>
<td>Set the number of characters to 0.</td>
</tr>
<tr>
<td>d)</td>
<td>If black pixel is detected then register L as the left of the current character and move the pointer to the next column &amp; 1st row. Otherwise, continue to the next pixel (from top to bottom).</td>
</tr>
<tr>
<td>e)</td>
<td>If black pixel is found and L &lt;&gt; -1 then update R as the right of the current character and move the pointer to the next column &amp; 1st row.</td>
</tr>
<tr>
<td>f)</td>
<td>If no black pixel is found in the column and L &lt;&gt; -1 then increment the number of characters by 1.</td>
</tr>
<tr>
<td>g)</td>
<td>Scan up to the right of the character found and repeat steps d) – f) to detect subsequent characters (Stop when the right-end of the last line is reached).</td>
</tr>
<tr>
<td>h)</td>
<td>Print out the number of characters detected and draw line boundaries on the image for each detected characters (for:L–1 andR+1).</td>
</tr>
</tbody>
</table>

Step 3: Perform "Row Scan" once more on the results obtained from the previous step in order to detect the actual character (top & bottom) boundaries.

Step 4: Bitmap images are created on the hard-disk for each segmented character relative to its boundaries; and its header information is generated from the original scanned image header besides updating dimensions.
Figure (5): Lines and Characters boundary detection for a scanned image sample

From the above figure, it is obvious that the detected lines-bound (top & bottom) stated by the "red-color" might not necessarily be the actual bounds for the characters in the same line because the characters have different heights. Hence, a confirmation of top and bottom boundaries for each character is needed as stated by the "green-color". The "blue-color" illustrates the detected characters-bound (left & right) for different characters widths.

4.4 Database Construction

In general, any OCR system depends on training samples as input data. In this work, database of (91) samples were collected from the data sets shown in Table (1) below for different font types ("Arial", "Calibri", "Courier New", "Lucida Sans" and "Times New Roman") and for different font sizes (8, 10, 12, 14, 16, 18, 20) to produce (3185) samples.

<table>
<thead>
<tr>
<th>No.</th>
<th>Data Sets</th>
<th>No. of Samples</th>
<th>Samples</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Digits</td>
<td>10</td>
<td>0 1 2 … 9</td>
</tr>
<tr>
<td>2</td>
<td>Capital English Letters</td>
<td>26</td>
<td>'A' 'B' 'C' … 'Z'</td>
</tr>
<tr>
<td>3</td>
<td>Small English Letters</td>
<td>26</td>
<td>'a' 'b' 'c' … 'z'</td>
</tr>
<tr>
<td>4</td>
<td>Some common ASCII Symbols</td>
<td>29</td>
<td>. , ; &quot; : [ ] - + * /= ( ) { } &lt;&gt; ! @ # $ % ^ &amp; ? \ _</td>
</tr>
</tbody>
</table>

4.5 Feature Extraction

Feature extraction is part of the data reduction process by forming a new “smaller” set of features from the original feature set of the patterns. This can be done by extracting some numerical
measurements from raw input patterns. Image features are of major importance in the isolation of regions of common property within an image[9, 10, 13]. In this work, two-sets of features were extracted from the segmented characters either by the use of Discrete Cosine Transform (DCT) or the spectral properties of Wavelet transform.

a) Discrete Cosine Transform (DCT)

DCT has become a standard method for many image processing & video compression algorithms. The two-dimensional DCT can be computed using the one-dimensional DCT horizontally (row-wise) and then vertically (column-wise) across the image because DCT is a function that separates the image into frequencies with large variance. The two-dimensional Forward DCT (2D FDCT) coefficients \( F(u, v) \) of \( M \times N \) block of image pixels \( f(x, y) \) are formulated as [11, 13, 14, 15]:

\[
F(u, v) = \frac{2}{\sqrt{MN}}C(u)C(v) \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} f(x, y) \cos \left( \frac{\pi(2x+1)u}{2N} \right) \cos \left( \frac{\pi(2y+1)v}{2M} \right)
\] .... (4)

\[
C(k) = \begin{cases} 
\sqrt{2} & \text{for } k = 0 \\
1 & \text{otherwise.}
\end{cases}
\] .... (5)

where \( C(k) \) is the normalization constant, \( u = 0, 1, \ldots, N - 1 \) and \( v = 0, 1, \ldots, M - 1 \).

The DCT coefficients \( D_{i,j} \) and \( i, j = 0, 1, \ldots, 7 \) of the corresponding image block of size 8x8, as example, are then ordered in a particular irregular sequence as shown in Figure (6). This irregular ordering of the coefficients is called Zig-zag ordering.

![Figure (6): Zig-zag ordering of DCT Coefficients](image)

The above sequence is broken into runs of nonzero (the early coefficients that contains the important "low-frequency" image information) and zero coefficients (the later coefficients in a block that contains the less-important "high-frequency" image information) [14, 15]. Therefore, the final DCT feature-set considered in this work is generated only from the number of significant (nonzero) coefficients denoted by \( N \) in Table (2) that starts from \( D_{0,0} \).

b) Wavelet Transform

The wavelet transform has been found very useful for the time-scale representation and has been widely used in signal processing and computer vision. The Wavelet transform is a multi-resolution technique that cut up data into different frequency components, and then analyzes each component with a resolution matched to its scale. The forward and inverse continuous wavelet
transform of \( x(t) \) "the signal to be analyzed" with respect to the basis function or wavelet \( \psi_{j,k}(t) \) at scale \( j \) (\( j>0 \)) and time delay \( k \) is written as follows \([16, 17, 18]\):

\[
\text{Forward CWT: } W(j,k) = \int x(t) \psi_{j,k}(t) \, dt \tag{6}
\]

\[
\text{Inverse CWT: } x(t) = \int \int W(j,k) \psi_{j,k}(t) \, dk \, dj \tag{7}
\]

where

\[
\psi_{j,k}(t) = \frac{1}{\sqrt{j}} \psi\left(\frac{t-k}{j}\right) \quad \text{and} \quad \psi(t) \text{is the mother wavelet} \tag{8}
\]

This multiresolution can also be obtained using filter banks, resulting in the Discrete Wavelet Transform (DWT) that are well suited to the digital computer because there are no derivatives or integrals, just multiplication and addition operations which correspond to the mathematical convolution operation. The procedure starts with passing the signal (sequence) through a half-band digital low-pass & high-pass filters. The DWT is computed by successive low and high pass filtering of the discrete time-domain signal \( X[n] \), as shown in Figure (7), and each result is down-sampled by two (\( \downarrow 2 \)) where the low-pass filter is denoted by \( G_0 \) while the high-pass filter is denoted by \( H_0 \). At each level, the high-pass filter produces detail information \( d[n] \) while the low-pass filter associated with scaling function produces coarse approximations \( a[n] \). The DWT of the original signal is then obtained by concatenating all the coefficients, \( a[n] \) and \( d[n] \), starting from the last level of decomposition \([10, 17, 18]\).

\[\text{Figure (7): Three Levels DWT Decomposition Tree}\]

After the conversion of input image from its lowest-level of pixel data in spatial domain \( I(x, y) \) into higher-level representation of wavelet coefficients \( W(x, y) \), a set of wavelet features (the energy of each band as stated in Eq.(9)) can be extracted by recursively decomposing sub-images in the low frequency channels as shown in Figure (8). The number of wavelet features for the 1st level is 4, and each progressing in wavelet level will correspond an increasing in features length by 3.

\[
\text{energy} = \frac{1}{M \times N} \sum_{i=1}^{M} \sum_{j=1}^{N} W(i,j)^2 \tag{9}
\]
Wavelet transform breaks an image down into four sub-sampled images. The results consist of one image that has been high pass filtered in both horizontal and vertical directions (HH), one that has been high pass filtered in the vertical and low pass filtered in the horizontal (LH), one that has been low passed in the vertical and high passed in the horizontal (HL), and one that has been low pass filtered in both directions (LL).

Numerous filters can be used to implement the wavelet transform. Daubechies (D4) wavelet is one of the most commonly used due to its efficiency. The Daubechies basis vectors are [10, 16, 18]:

\[
\text{Low - pass filter: } \frac{1}{4\sqrt{2}} [1 + \sqrt{3}, 3 + \sqrt{3}, 3 - \sqrt{3}, 1 - \sqrt{3}] \quad \ldots \quad (10)
\]

\[
\text{High - pass filter: } \frac{1}{4\sqrt{2}} [1 - \sqrt{3}, \sqrt{3} - 3, 3 + \sqrt{3}, -1 - \sqrt{3}] \quad \ldots \quad (11)
\]

4.6 Pattern Matching

The resulting test template, which is an N-dimensional feature vector, is compared against the stored reference templates to find the closest match. The process is to find which unknown class matches a predefined class or classes. For the OCR task, the unknown character is compared to all references in the database. This comparison can be done through Euclidean (E.D.) distance measure, shown below [9, 10]:

\[
E \cdot D. = \sqrt{\sum_{i=1}^{N} (a_i - b_i)^2} \quad \ldots \quad (12)
\]

where A and B are two vectors, such that A=[a_1 a_2 \ldots a_N] and B=[b_1 b_2 \ldots b_N].

In our approach the minimum distance classifier is used to measure the difference between the two patterns (feature vectors). This classifier assigns the unknown pattern to the nearest predefined pattern. The bigger distance between the two vectors, is the greater difference [9, 10].

5. EXPERIMENTAL RESULTS

OCR accuracy is defined as the ratio of correct recognized characters to the total number of characters (samples) tested, as shown by Eq. (13):
A number of experiments and test conditions were accomplished on (13650) samples to measure the performance of the proposed OCR system on various types of document images of different dimensions, font types and font sizes. As a result, the database size of training samples is computed as follows:

\[ \text{No. of Training Samples} = \text{No. of Samples} \times \text{No. of Fonts} \times \text{No. of Font Sizes} \tag{14} \]

A more appropriate comparison can be made if both DCT & Wavelet methods are measured under identical conditions. Based on the results shown in Table (2), one can deduce that all wavelet features produce better recognition rates 96% (from 92 – 98%) than DCT features 92% (from 87 – 95%). Different number of DCT coefficients (N) and wavelet decomposition levels values are examined according to the recognition rates. It is clearly indicated that two decomposition levels are most appropriate for wavelet feature vector construction, whereas 10-DCT coefficients is the suitable number of DCT features (N=10).

Table (2): OCR Accuracy for different Testing images using DCT & Wavelet features

<table>
<thead>
<tr>
<th>Test Image File</th>
<th>Font Type</th>
<th>No. of Chars</th>
<th>DCT N=5</th>
<th>DCT N=10</th>
<th>DCT N=15</th>
<th>Wavelet Transform Level1</th>
<th>Wavelet Transform Level2</th>
<th>Wavelet Transform Level3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test1.bmp</td>
<td>Arial</td>
<td>525</td>
<td>77.52</td>
<td>91.62</td>
<td>85.90</td>
<td>71.43</td>
<td>94.29</td>
<td>80.95</td>
</tr>
<tr>
<td>Test2.bmp</td>
<td>Arial</td>
<td>581</td>
<td>79.00</td>
<td>93.46</td>
<td>87.44</td>
<td>74.87</td>
<td>96.73</td>
<td>87.78</td>
</tr>
<tr>
<td>Test3.bmp</td>
<td>Arial</td>
<td>791</td>
<td>81.29</td>
<td>90.90</td>
<td>85.59</td>
<td>80.28</td>
<td>94.82</td>
<td>85.59</td>
</tr>
<tr>
<td>Test4.bmp</td>
<td>Arial</td>
<td>371</td>
<td>75.74</td>
<td>94.88</td>
<td>83.83</td>
<td>64.15</td>
<td>97.30</td>
<td>80.32</td>
</tr>
<tr>
<td>Test5.bmp</td>
<td>Arial</td>
<td>462</td>
<td>71.65</td>
<td>87.88</td>
<td>78.35</td>
<td>69.05</td>
<td>93.51</td>
<td>83.77</td>
</tr>
<tr>
<td>Test6.bmp</td>
<td>Arial</td>
<td>480</td>
<td>76.46</td>
<td>92.29</td>
<td>82.29</td>
<td>75.63</td>
<td>95.00</td>
<td>83.75</td>
</tr>
<tr>
<td>Test7.bmp</td>
<td>Arial</td>
<td>633</td>
<td>79.78</td>
<td>93.84</td>
<td>85.47</td>
<td>77.09</td>
<td>95.10</td>
<td>80.73</td>
</tr>
<tr>
<td>Test8.bmp</td>
<td>Arial</td>
<td>415</td>
<td>78.31</td>
<td>95.66</td>
<td>84.10</td>
<td>68.43</td>
<td>96.63</td>
<td>83.37</td>
</tr>
<tr>
<td>Test9.bmp</td>
<td>Arial</td>
<td>692</td>
<td>79.05</td>
<td>90.75</td>
<td>83.38</td>
<td>75.29</td>
<td>96.68</td>
<td>85.55</td>
</tr>
<tr>
<td>Test10.bmp</td>
<td>Arial</td>
<td>510</td>
<td>77.06</td>
<td>92.75</td>
<td>84.31</td>
<td>79.02</td>
<td>97.25</td>
<td>84.90</td>
</tr>
<tr>
<td>Test11.bmp</td>
<td>Arial</td>
<td>395</td>
<td>75.70</td>
<td>92.41</td>
<td>82.28</td>
<td>67.34</td>
<td>96.46</td>
<td>81.27</td>
</tr>
<tr>
<td>Test12.bmp</td>
<td>Arial</td>
<td>704</td>
<td>81.82</td>
<td>95.60</td>
<td>86.36</td>
<td>73.86</td>
<td>97.30</td>
<td>84.52</td>
</tr>
<tr>
<td>Test13.bmp</td>
<td>Courier</td>
<td>620</td>
<td>79.35</td>
<td>92.58</td>
<td>83.23</td>
<td>75.97</td>
<td>96.61</td>
<td>83.71</td>
</tr>
<tr>
<td>Test14.bmp</td>
<td>New</td>
<td>557</td>
<td>79.35</td>
<td>93.36</td>
<td>84.02</td>
<td>74.33</td>
<td>98.20</td>
<td>89.23</td>
</tr>
<tr>
<td>Test15.bmp</td>
<td>Courier</td>
<td>454</td>
<td>75.11</td>
<td>91.63</td>
<td>81.28</td>
<td>68.06</td>
<td>96.92</td>
<td>82.60</td>
</tr>
<tr>
<td>Test16.bmp</td>
<td>New</td>
<td>664</td>
<td>80.72</td>
<td>93.67</td>
<td>86.30</td>
<td>75.75</td>
<td>95.63</td>
<td>84.19</td>
</tr>
<tr>
<td>Test17.bmp</td>
<td>Lucida</td>
<td>325</td>
<td>72.92</td>
<td>94.46</td>
<td>82.15</td>
<td>68.92</td>
<td>92.62</td>
<td>79.38</td>
</tr>
<tr>
<td>Test18.bmp</td>
<td>New</td>
<td>653</td>
<td>78.56</td>
<td>92.04</td>
<td>82.54</td>
<td>76.11</td>
<td>94.49</td>
<td>82.24</td>
</tr>
<tr>
<td>Test19.bmp</td>
<td>Lucida</td>
<td>341</td>
<td>68.62</td>
<td>89.15</td>
<td>77.71</td>
<td>69.21</td>
<td>95.60</td>
<td>78.01</td>
</tr>
<tr>
<td>Test20.bmp</td>
<td>New</td>
<td>747</td>
<td>79.65</td>
<td>91.30</td>
<td>82.33</td>
<td>78.71</td>
<td>94.38</td>
<td>83.53</td>
</tr>
<tr>
<td>Test21.bmp</td>
<td>Times</td>
<td>720</td>
<td>76.11</td>
<td>90.28</td>
<td>79.44</td>
<td>77.64</td>
<td>96.81</td>
<td>85.00</td>
</tr>
<tr>
<td>Test22.bmp</td>
<td>New</td>
<td>373</td>
<td>71.05</td>
<td>87.13</td>
<td>80.43</td>
<td>72.12</td>
<td>98.93</td>
<td>86.33</td>
</tr>
<tr>
<td>Test23.bmp</td>
<td>Roman</td>
<td>475</td>
<td>73.68</td>
<td>89.05</td>
<td>79.79</td>
<td>68.63</td>
<td>95.37</td>
<td>80.63</td>
</tr>
<tr>
<td>Test24.bmp</td>
<td>Times</td>
<td>525</td>
<td>79.05</td>
<td>93.14</td>
<td>82.86</td>
<td>74.10</td>
<td>96.95</td>
<td>85.52</td>
</tr>
<tr>
<td>Test25.bmp</td>
<td>New</td>
<td>637</td>
<td>78.02</td>
<td>90.74</td>
<td>81.48</td>
<td>78.96</td>
<td>97.02</td>
<td>86.81</td>
</tr>
<tr>
<td>Total</td>
<td></td>
<td>13650</td>
<td>77.64</td>
<td>92.05</td>
<td>83.16</td>
<td>74.25</td>
<td>96.01</td>
<td>83.89</td>
</tr>
</tbody>
</table>
A good feature set has a major importance in the isolation of regions of common property within an image, and it should represent characteristic of a class that help distinguish it from other classes, as shown in Figure (9). The proportion of \( \frac{H}{W} \) was taken in our consideration for all text components (characters, digits, and symbols) for two reasons:

- it is invariant to different font sizes of the same font type,
- to speed-up the recognition process because the matching processes are limited only to the text components in the same class.

Consequently, an improvement overall recognition rates (about 3\%) are obtained after classifying characters according to the addition of a new discriminating feature (the proportion of Height to Width) that produce 99\% for wavelet and 95\% for DCT.

\[
\begin{align*}
\text{Courier New Font} & \\
\text{Size} = 20 & \\
\end{align*}
\]

\[
\begin{align*}
\text{Arial Font} & \\
\text{Size} = 20 & \\
\text{Size} = 16 & \\
\text{Size} = 10 & \\
\end{align*}
\]

\[
\frac{H_0}{W_0} < > \left( \frac{H_1}{W_1} = \frac{H_2}{W_2} = \frac{H_3}{W_3} \right)
\]

Figure (9): The Proportion of \( \frac{H}{W} \) Feature for the character sample 'E' in different Font types & sizes

6. CONCLUSIONS

A multi-fonts English texts OCR system for 3185 training samples and 13650 testing samples is presented that relies on DCT and wavelet features. Image enhancement techniques (noise removal, foreground/ background separation, normalization and binarization) are adopted in this work prior to segmentation in order to improve the recognition rates and simplify the process of characters segmentation.

It is found that wavelet method is appropriate for feature vector construction where all the recognition rates (96\%) outperform the DCT based recognition method (92\%). To enhance the recognition rates further and speeds up the recognition process, text-components (characters, digits, and symbols) are classified according to the proportion of \( \frac{H}{W} \) feature that produce 99\% accuracy for wavelet based method because it help distinguish a class from other classes which is invariant to different font sizes of the same font type.

REFERENCES


