2

JPEG for Arabic Handwritten Character Recognition: Add a Dimension of Application

Abdurazzag Ali Aburas and Salem Ali Rehiel
International Islamic University Malaysia
Electrical and Computer Engineering
Malaysia

1. Introduction

The ultimate objective of any Optical Character Recognition (OCR) system is to simulate the human reading capabilities. That is why OCR systems are considered a branch of artificial intelligence and a branch of computer vision as well [Sunji Mori (1999)] Character recognition has received a lot of attention and success for Latin and Chinese based languages, but this is not the case for Arabic and Arabic-like languages such as Urdu, Persian, Jawi, Pishtu and others [Abdelmalek Z (2004)]. Researchers classify OCR problem into two domains. One deals with the image of the character after it is input to the system by, for instant, scanning in which is called Off-line recognition. The other has different input way, where the writer writes directly to the system using, for example, light pen as a tool of input. This is called On-line recognition.

The online problem is usually easier than the offline problem since more information is available [Liana M & Venu G (2006)]. These two domains (offline & online) can be further divided into two areas according to the character itself that is either handwritten or printed character. Roughly, the OCR system based on three main stages: preprocessing, feature extraction, and discrimination (called also, classifier, or recognition engine) Figure 1.1 depicts the block diagram of the typical OCR system. Traditional OCR systems are suffering from two main problems, one comes from features extraction stage and the other comes from classifier (recognition stage). Feature extraction stage is responsible for extracting features from the image and passing them as global or local information to the next stage in order to help the later taking decision and recognizing the character. Two challenges are faced; if feature extractor extracts many features in order to offer enough information for classifier, this means many computations as well as more complex algorithms are needed. Thus, long processor time will be consumed. On the other hand, if few features are extracted in order to speed up the process, insufficient information may be passed to classifier. The second main problem that classifier is responsible for, is that most of classifiers are based on Artificial Neural Networks (ANNs). However, to improve the intelligence of these ANNs, huge iterations, complex computations, and learning algorithms are needed, which also lead to consume the processor time. Therefore, if the recognition accuracy is improved, the consumed time will increase and vice versa.

Source: Advances in Robotics, Automation and Control, Book edited by: Jesús Arámburo and Antonio Ramírez Treviño,
ISBN 78-953-7619-16-9, pp. 472, October 2008, I-Tech, Vienna, Austria

Figure 1.1 The typical OCR block diagram

To tackle these problems, a new OCR construction is proposed in this chapter, where neither features extractor nor ANN is needed. The proposed construction relies on the image compression technique (JPEG). Taking advantages of the compressor, that it compresses the image by encoding only the main details and quantizes or truncates the remaining details (redundancy) to zero. Then generates a unique vector (code) corresponding to the entire image. This vector can be effectively used to recognize the character since it caries the main details of the character's image. The importance of the main details is that they are common amongst the same character which is written by different writers.
The principle is illustrated and the algorithm is designed and tested in the related sections. In the next section a brief review of the related work is described, Arabic character characteristics are illustrated in section 3. In section 4, an overview of JPEG compression is shown. In section 5, the concept and the proposed algorithm are presented. The experimental results are shown and discussed in section 6. The conclusion of the chapter is drawn in section 7.

2. Review of the related work

Beside the main goal of any OCR system which is simulating human's reading capability, the accuracy and time consuming are very important issues in this aspect. Based on the latest survey which is published in May 2006 by Liana M. and Venu G. [Liana(2006)] all covered papers have presented their proposals seeking high accuracy and less time. Each one has treated the issue from different angle of view. Their work can be classified into three main categories: preprocessing problems, features extraction problems, and recognition (discrimination) problems. For preprocessing stage, where the image is often converted to a more concise representation prior to recognition the most common methods are: A skeleton which is a one-pixel thick representation showing the centrelines of the character. Skeletonization is also called "thinning" facilitates shape classification and feature detection. Many researchers have used skeletonization in their proposed preprocessing stages like for instant [S. Mozaffari et al., (2005)], [S. Alma'adeed et al., (2002)], [S. Alma'adeed et al., (2004)]. Another method which is known as "contour" is the Freeman chain code of the character's border. In this method chain code stores the absolute position of the first pixel and the relative positions of successive pixels along the character's border. This methods has

been practiced by many like [R. Safabakhsh & P. Adibi (2005)], [L. Souici-Meslati & M. Sellami (2004)], [N. Farah et al., (2004)], however, skeletonization suffers from mislocalization of feature and ambiguities particular to each thinning algorithm whereas the contour approach avoids these problems since no shape information is lost. For the features extraction stage, where the main role is played and mostly the accuracy of recognition depends on the information passed from this stage to the classifier (recognizer). These information can be structural features such as loops, branch-points, endpoints, and dots or statistical which includes but is not limited to, pixel densities, histograms of chain code directions, moments, and Fourier descriptors. Because of the importance of this stage many approaches and techniques have been proposed. For instant, Clocksin in [W.F. Clocksin & P.P.J. Fernando (2003)] applied moment functions to image and polar transform image. Many like [A. Amin (2003)], [G. Olivier et al., (1996)], [I.S.I. Abuhaiba et al., (1998)], used loops, dots, curves, relative locations, height, sizes of parts of characters, loop positions and types, line positions and directions, and turning points. Where others used statistics from moments of horizontal and vertical projection like[H. Al-Yousefi & S.S. Udpa (1992)]. Histogram of slopes along contour is used by [M. Dehghan et al., (2001)]. Fourier descriptors have also been used in this stage by [R. Safabakhsh & P. Adibi (2005)]. Few used techniques such as wavelet or fractal like [Amir M. et al., (2002)], [Saeed M. et al., (2004 a)] and [Saeed M. et al., (2004 b)]. Artificial Neural Networks (ANNs) are the common seed of most if not all classifiers "recognition or discrimination stage". Many variations have been used in order to overcome the main disadvantage of ANNs which is time consuming. Sherif and Mostafa in [Sherif K. & Mostafa M (1996)], for example, presented a parallel design for backpropagation Neural Networks approach in order to accelerate the computation process. If we want to mention who did use ANNs in OCR, we may list all of them at least within our concern field in this chapter "off-line Arabic handwriting character recognition".

3. Arabic character characteristics

The Arabic alphabet contains basically 28 letters are written from write to left. Each letter can take from two to five different shapes, thus, roughly the alphabet set can expand to 84 different shapes according to the position of the letter (beginning, middle, end or isolated) as well as according to the style of writing (Nasekh, Roqa'a, Farisi and few others). This is one reason makes Arabic recognition complex. The second reason is the similarities among the different letters and the differences among the same latter. For example, letters Baa, Taa, and Thaa (number 2,3 & 4 respectively in Arabic Alphabet)are three different letters, but they have similar body shape, they only differ in number and position of dots (one, two or three dots below or above the body of the character). Also Jeem, Hhaa & Khaa (number 5, 6, & 7 respectively in Arabic Alphabet) differ only in one dot. On the other hand the differences among the same letter, for example, letter Haa (number 26 in the Arabic Alphabet) has three completely different shapes through its positions. Officially, there is a set of rules to write Arabic letters, but few follow. These rules may help OCR to extract features or segment text, such as so called base-line rule. This rule states that there are three lines. Each letter or group of letters has their lines where they should be lie.

4. JPEG compression background

The most important current standard for image compression is JPEG [W.B Pennebaker & J.L. Mitchell (1993)]. In the JPEG baseline coding system, which is based on the discrete

cosine transform (DCT) and is adequate for most compression applications, the input and output images are limited to 8 bits, while the quantized DCT coefficient values are restricted to 11 bits. The human vision system has some specific limitations, which JPEG takes advantage of to achieve high rates of compression.

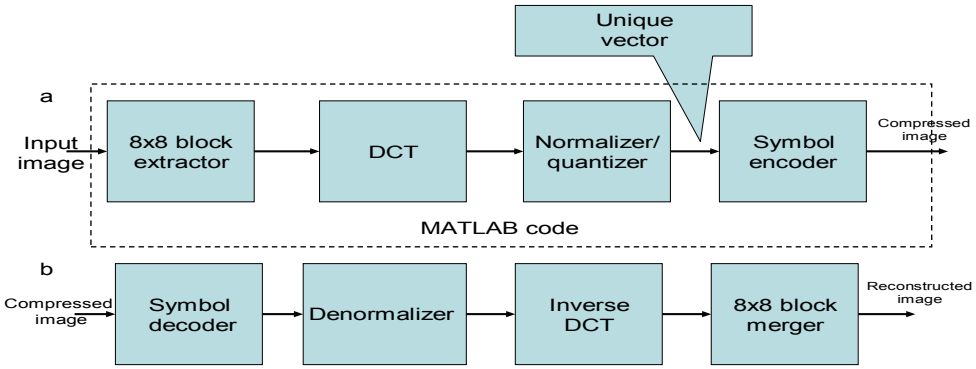

Figure 4.1 JPEG block diagram. (a) encoder and (b) decoder

As can be seen in the simplified block diagram of Figure 4.1, the compression itself is performed in four sequential steps: 8x8 sub-image extraction, DCT computation, quantization, and variable-length code assignment. Since we do not concern in this work about the reconstruction part, the only part of compression is used (dashed box) and the vector will be tapped immediately after quantization stage.

5. Proposed algorithm

The main concept of the proposed algorithm is based on the property that the JPEG compressed image is a vector which can uniquely represent the input image to be correctly reconstructed later at decompression stage. This property can be effectively used to recognize the character's image. A JPEG approximation MATLAB code is used to generate a vector of coefficients that represent the important information (or main details for the character's image). In the next sections the experimental work is illustrated

5.1 Experiment's tools and platform environment

The data consists of groups for the twenty eight Arabic alphabets, each group was written by 48 writers from different ages and educational backgrounds. Figure 5.1 shows one sample of one group "letter Sheen".

An 40x40 8-bit pixel colour image was used as input image. For experimental reason, the MATLAB version 7.2.0.232 (R2006a) has been used but in order to improve the speed of the proposed algorithm a prototype of C++ code might be used. The Intel Pentium 4 CPU, 3.00GHz clock and 960MB RAM of the computer, has been used

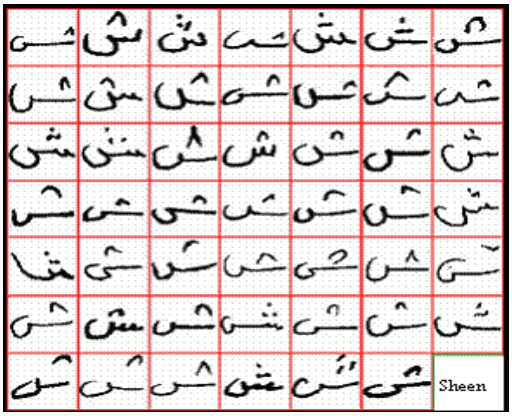

Figure 5.1 A group sample of letter Sheen

5.2 Proposed algorithm top-down flowchart

Figure 5.2 illustrates the sequence of the proposed algorithm's steps. After the character's image is scanned in the system the JPEG approximation MATLAB code will produce a vector. This vector is assumed to uniquely represent input image since it carries the important details of that image. Figure 5.3 shows a sample for letter Sheen and letter Ain. Then Euclidean distance between this vector and each vector in codebook will be measured. Finally, the minimum distance points to the corresponding character, and then the character is recognized.

5.3 Proposed system components

The two main components are the code of the compression stage (i.e., JPEG compressor in the flowchart shown in Figure 5.2) and the codebook

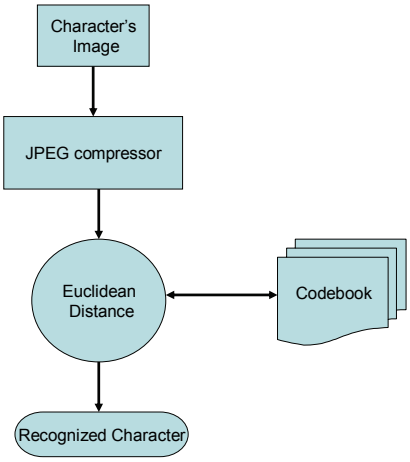

Figure 5.2 Flowchart diagram of the proposed algorithm

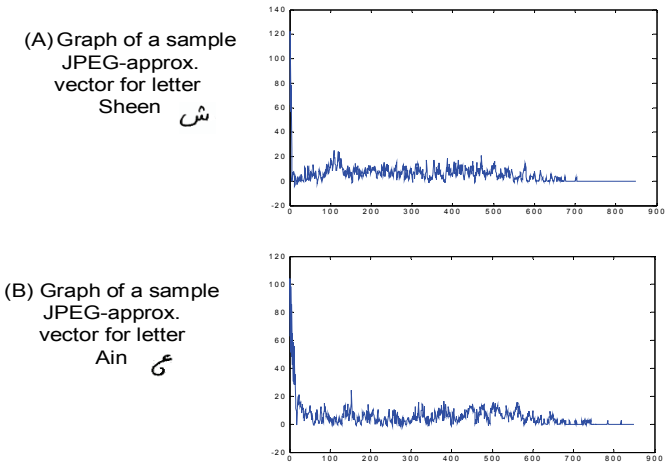

Figure 5.3 Sample from the produced JPEG-approximation vector; (A) for letter Sheen, (B) for letter Ain

5.3.1 Compressor MATLAB code

The code listed in List 5.1 is to compress an image (x) using a JPEG approximation based on 8x8 DCT transforms, coefficient quantization, and Huffman symbol coding. Input "Quality" determines the amount of information that is lost and compression achieved. Y is an encoding structure containing fields:
[y.size] is the size of the image (x)
[y.numblocks] is the number of the 8x8 encoded blocks
[y.quality] is the quality factor (as percent)
[y.huffman] is the Huffman encoding structure, as returned by mat2huff (see List 8.1)

```
% This function appeared in  [R.C. Gonzalez, R. E. Woods, & S.L.Eddins (2003)]
function y=im2jepg(x,quality)
global vctr % vctr is the vector obtained after the image is
        % JPEG compreseed and before Huffman takes place
error(nargchk(1,2,nargin));     % check input argument
if ndims(x) ~=2 | ~isreal(x)|~isnumeric(x)|~isa(x,'uint8')
   error('the input MUST BE a UINT8 image');
end
if nargin<2
   quality=1;              % Default value for quality
end
m=[ 16 11 10 16 24 40 51 61
   12 12 14 19 26 58 60 55
   14 13 16 24 40 57 69 56
   14 17 22 29 51 87 80 62
```

```
    18 22 37 56 68 109 103 77
    24 35 55 64 81 104 113 92
    49 64 78 87 103 121 120 101
    72 92 95 98 112 100 103 99] * quality;
order=[ 1 9 2 3 10 17 25 18 11 4 5 12 19 26 33 ...
      41 34 27 20 13 6 7 14 21 28 35 42 49 57 50 ...
      43 36 29 22 15 8 16 23 30 37 44 51 58 59 52 ...
      45 38 31 24 32 39 46 53 60 61 54 47 40 48 55 ...
      62 63 56 64];
  [xm,xn]=size(x)
  x=double(x)-128;
  t=dctmtx(8);
%Compute DCTs of 8x8 blocks and quantize the coefficients
  y=blkproc(x,[8 8], 'P1*x*P2',t,t');
  y=blkproc(y,[8 8],'round(x./P1)',m);
  y=im 2col(y,[8 8], 'distinct');  %Break 8x8 blocks into columns
  xb=size(y,2);
  y=y(order,:);
  eob=max(x(:))+1              %Create end-of-block symbol
  r=zeros(numel(y)+size(y,2),1);
  count=0;
  for j=1:xb
     i=max(find(y(:,j)));
     if isempty(i)
        i=0;
     end
     p=count+1;
     q=p+i;
     r(p:q)=[y(1:i,j);eob];
     count=count+i+1;
  end
  count
  r((count+1):end)=[];   % Delete unused portion of r
  y.size=uint16([xm xn]);
  y.numblocks= uint16(xb);
  y.quality = uint16(quality *100);
  y.huffman =mat2huff(r);
  vctr=r;  % This is the produced vector for the proposed system
```

List 5.1 MATLAB code for the compressor stage

In accordance to the block diagram of Figure 4.1 (A) "dashed box", function **im2jpeg** processes distinct 8x8 sections or blocks of input image x one block at a time (rather than the entire image at once). Two specialized block processing functions – **blkproc** and **im2col** – were used to simplify the computations (more information available @ MATLAB Mathworks® documentary). Function **blkproc** is used to facilitate both DCT computation and coefficient denormalization and quantization, while **im2col** is used to simplify the quantized coefficient reordering and zero run detection. The function **im2jpeg** listed in List 5.1 uses an alternate matrix formulation

$$\mathbf{T} = \mathbf{H}\,\mathbf{F}\,\boldsymbol{H} \tag{1}$$

Where **F** is an 8x8 block of image f(x,y), **H** is an 8x8 DCT transformation matrix generated by dctmtx(8) (more details for MATLAB functions can be found in MATLAB® Documentary), and T is the resulting DCT of **F**. note that the **H** is used to denote the transpose operation.
The statement **y = blkproc(x,[8 8], 'P1*x*P2',h,h')** computes the DCTs of image x in 8x8 blocks, using DCT transform matrix **h** and transpose **h'** as parameters **P1** and **P2** of the DCT matrix multiplication, P1*x*P2

5.3.2 Codebook building

The codebook can be built as following:

1. Apply the MATLAB code listed in List 5.1 to all available database (our database contains 1968 written characters) to get 1968 vectors
2. Group the 1968 vectors (vctr) according to their represented character. For instance, the group of letter Sheen has (in our database) 48 different Sheen that were written by 48 different writers, so it will have 48 vectors (vctr)
3. Average each group (MATLAB function "**mean**" was used in our experiments). By now you should get one vector (vctr) for each group. These are the codes (vctrs) located in the codebook.

6. Results and discussion

Once the codebook is installed the algorithm is ready for testing. Test should cover all available data (1968 tests) to ensure that the code book can handle wide range of expected character images and recognize them with minor errors. Two criteria may be used to examine the system, accuracy and rate of recognition. The accuracy can be measured as the percent of the correctly recognized characters to the tested characters. The rate of recognition (or the speed) is the time system takes from the image inputs until it is recognized. Since the process is a character's shape-independent, the taken times will be equal for all images. Adding a timer code can return the time spent by the system. In addition to JPEG approximation MATLAB code the system needs to calculate Euclidean distance and the minimum value (the shortest distance). The Euclidean distance (d) between two vectors X and Y can be defined as :

$$d = \sum \sqrt{(x - y)^2} \tag{2}$$

Whose MATLAB function is

$$\text{D=dist(x, y)} \tag{3}$$

Table 6.1 shows the accuracy recognitions' percentage for each character obtained by the proposed algorithm. The overall average is also shown. The system was able to recognize the characters during short time comparing with any existing system using ANN because it saves time taken by features extractor as well as it uses codebook (lookup table) instead of ANN. Misrecognition occurred to some letters is because of the nature of the handwritten character itself neither the algorithm. Thus, there is no rejection state, the character must be recognized in all situations. It is either correct recognized or incorrect recognized this confusing in recognition may occur even to humans. To verify this conclusion the algorithm was examined with printed characters and obtained better results.

	character		Accuracy %		character		Accuracy %
1	ا	Alif	90.8333	15	ض	Dthad	51.2500
2	ب	Baa	32.5000	16	ط	Ttaa	45.0000
3	ت	Taa	20.0000	17	ظ	Dthaa	38.7500
4	ث	Thaa	80.4167	18	ع	Ain	28.3333
5	ج	Jeem	86.6667	19	غ	Ghen	32.5000
6	ح	Hhaa	100.0000	20	ف	Faa	30.4167
7	خ	Khaa	70.0000	21	ق	Qaf	22.0833
8	د	Dal	36.6667	22	ك	Kaf	45.0000
9	ذ	Theal	28.3333	23	ل	Lam	55.4167
10	ر	Raa	23.1250	24	م	Meem	71.0417
11	ز	Zaay	28.3333	25	ن	Noon	42.9167
12	س	Seen	73.1250	26	هـ	Haa	78.3333
13	ش	Sheen	90.8333	27	و	Waw	73.4583
14	ص	Sad	62.7083	28	ى	Yaa	20.6875
Average Accuracy							52.0975

Table 6.1 The recognition accuracy

Moreover, to test algorithm's intelligence some experiments were done such as taking off the dot from letter Khaa خ or Jeem ج the algorithm recognized them as Hhaa ح, and from letter Dhad ض the algorithm recognized it as Sad ص. Also from Theal ذ and Zaay ز they were recognized as Dal د and Raa ر respectively. Furthermore, when a character was turned upside-down, the farthest Euclidian distance; instead of the shortest Euclidian distance, pointed to that character

7. Conclusion

In this chapter we proposed a new structure of off line OCR system which is not based on ANN, to avoid the time consuming problems. Moreover, it benefited from the JPEG image compression property that is high compression ratio which produces minimum compressed image size and every compressed image has a unique vector which helps to identify each character. By using this unique vector, the proposed system has recognized the input character after measuring the Euclidean distance between the vector and the vectors in the codebook, then the shortest distance pointed to the corresponding letter. In addition to the advantage of speed using codebook, it can be universal by means of character's nature (language, writing mode) as well as character's image size. We used 40x40 8-pexil color image as input image. The result was considerably high in terms of accuracy and recognition rate. We used JPEG approximation MATLAB code . Future work could be done using in advanced on line OCR system, also could be implemented by using fast programming languages such as C++ and it is highly recommended to be used for different Biometrics applications.

8. Appendix: MATLAB code

```
% This function appeared in [R.C. Gonzalez, R. E. Woods, & S.L.Eddins (2003)]
function y=mat2huff(x)
```

```
if ndims(x)~= 2|~isreal(x) | (~isnumeric(x) & ~islogical(x))
   error('X must be a 2-d real numeric or logical matrix');
end
% store the size of input x
y.size = uint32(size(x));
% find the range of x values and store its minimum value biased
% by +32768 as a UINT16
x=round(double(x));
xmin=min(x(:));
xmax=max(x(:));
pmin=double(int16(xmin));
pmin= uint16(pmin+32768); y.min=pmin;
%compute the input histogram between xmin and xmax with uint
%width bins, scale to UINT16, and store
x=x(:)';
h=histc(x,xmin:xmax);
if max(h) > 65535
   h=65535*h/max(h);
end
h=uint16(h); y.hist=h;
%code the input matrix and store the result.
map = huffman(double(h));      % Make a Huffman code map
hx=map(x(:)-xmin+1);          % Map image
hx=char(hx)';                 % Convert to char array
hx=hx(:)';
hx(hx==' ')=[];               % Remove the blanks
ysize=ceil(length(hx)/16);     % Compute encoded size
hx16=repmat('0',1,ysize*16);   % Pre-allocate modulo-16 vector
hx16(1:length(hx))=hx;         % Make hx modulo-16 in length
hx16=reshape(hx16,16,ysize);   % Reshape to 16-character words
hx16=hx16'-'0';                % Convert binary string to decimal
twos=pow2(15:-1:0);
y.code=uint16(sum(hx16.*twos(ones(ysize,1),:),2))';
```

List 8.1 MATLAB code (mat2huff)

9. Acknowledgements

All images of handwritten Arabic letters were provided by Prof. Zaki Kheder, University of Jordan. Authors would like also to thank Dr Rafael C. Gonzales for giving permission to use his MATLAB codes published in [R. C. Gonzalez et al (2003)]

10. References

A. Amin. (2003). Recognition of Hand-Printed Characters Based on Structural Description and Inductive Logic Programming" *Pattern Recognition Letters*, vol. 24, pp. 3187-3196.

Abdelmalek Z. (2004). ORAN: A Basis for Arabic OCR system, *Proceeding of 2004 International Symposium on Intelligent Multimedia, Video and Speech Processing*, Hong Kong, pp. 703-706.

Amir M., Karim F. & Abolfazl T. (2002). Feature Extraction with Wavelet Transform for Recognition of Isolated Handwritten Farsi/Arabic Characters and Numerals. *IEEE*, DSP, pp. 923- 926.

G. Olivier, H. Miled, K. Romeo, and Y. Lecourtier. (1996). Segmentation and Coding of Arabic Handwritten Words," *Proc. 13th Int'l Conf.Pattern Recognition*, vol. 3, pp. 264-268, 1996.

H. Al-Yousefi and S.S. Udpa. (1992). Recognition of Arabic Characters. *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol.14,pp.853-857.

I.S.I. Abuhaiba, M.J.J. Holt, and S. Datta. (1998). Recognition of Off-Line Cursive Handwriting. *Computer Vision and Image Understanding*, vol. 71, pp. 19-38.

L. Souici-Meslati and M. Sellami. (2004). A Hybrid Approach for Arabic Literal Amounts Recognition. *The Arabian J. Science and Eng.*,vol. 29, pp. 177-194, 2004.

Liana M & Venu G. (2006). Offline Arabic Handwriting Recognition: A Survey. IEEE, *Transactions On Pattern Analysis and Machine Intelligence*, vol. 28, No. 5, pp. 712-724.

M. Dehghan, K. Faez, M. Ahmadi, and M.Shridhar. (2001). Handwritten Farsi (Arabic) Word Recognition: A Holistic Approach Using Discrete HMM. *Pattern Recognition*, vol. 34, pp. 1057- 1065.

N. Farah, L. Souici, L. Farah, and M. Sellami. (2004). Arabic Words Recognition With Classifiers Combination: An Application to Literal Amounts. *Proc. Artificial Intelligence: Methodology, Systems and Applications*, pp. 420-429.

R. C. Gonzalez, R. E. Woods, S. L. Eddins. (2003). *Digital Image Processing Using MATLAB*. Prentice Hall; 1st edition, September 5, 2003

R. Safabakhsh and P. Adibi. (2005). Nastaaligh Handwritten Word Recognition Using a Continuous-Density variable-Duration HMM. *The Arabian J. Science and Eng.*, vol. 30, pp. 95-118.

S. Alma'adeed, C. Higgens, and D. Elliman. (2002). Recognition of Off-Line Handwritten Arabic Words Using Hidden Markov Model Approach. *Proc. 16th Int'l Conf. Pattern Recognition*, vol. 3, pp. 481-484.

S. Alma'adeed, C. Higgens, and D. Elliman. (2004). Off-Line Recognition of Handwritten Arabic Words Using Multiple Hidden Markov Models. *Knowledge-Based Systems*, vol. 17, pp. 75-79.

S. Mozaffari, K. Faez, and M. Ziaratban. (2005). Structural Decomposition and Statistical Description of Farsi/Arabic Handwritten Numeric Characters. *Proc. Int'lConf. Document Analysis and Recognition*,pp. 237- 241.

Saeed M., Karim F, & Hamidreza R. (2004 a). Feature Comparison between Fractal Codes and Wavelet Transform in Handwritten Alphanumeric Recognition Using SVM Classifier. IEEE, *Proceedings of the 17th International Conference on Pattern Recognition* (ICPR'04).

Saeed M., Karim F., and Hamidreza R. (2004 b). Recognition of Isolated Handwritten Farsi/Arabic Alphanumeric Using Fractal Codes. *IEEE, 0-7803-83387- 7/04*, pp. 104-108.

Sherif K. and Mostafa M. (1996). A Parallel Design and Implementation for Backpropagation Neural Network Using MIMD Architecture. *IEEE*, pp. 1361-1366.

Sunji Mori, H. Nishida & H. Yamada. (1999). *Optical Character Recognition*. John Wiley & Sons

W.B Pennebaker and J.L. Mitchell. (1993). *The JPEG Still Image Data Compression Standard*. New York: Van Nostrand Reinhold.

W.F. Clocksin and P.P.J. Fernando. (2003). Towards Automatic Transcription of Syriac Handwriting. *Proc.Int'l Conf. Image Analysis and Processing*, pp. 664- 669.

Advances in Robotics, Automation and Control

Edited by Jesus Aramburo and Antonio Ramirez Trevino

ISBN 978-953-7619-16-9
Hard cover, 472 pages
Publisher InTech
Published online 01, October, 2008
Published in print edition October, 2008

The book presents an excellent overview of the recent developments in the different areas of Robotics, Automation and Control. Through its 24 chapters, this book presents topics related to control and robot design; it also introduces new mathematical tools and techniques devoted to improve the system modeling and control. An important point is the use of rational agents and heuristic techniques to cope with the computational complexity required for controlling complex systems. Through this book, we also find navigation and vision algorithms, automatic handwritten comprehension and speech recognition systems that will be included in the next generation of productive systems developed by man.

How to reference

In order to correctly reference this scholarly work, feel free to copy and paste the following:

Abdurazzag Ali Aburas and Salem Ali Rehiel (2008). JPEG for Arabic Handwritten Character Recognition: Add a Dimension of Application, Advances in Robotics, Automation and Control, Jesus Aramburo and Antonio Ramirez Trevino (Ed.), ISBN: 978-953-7619-16-9, InTech, Available from: http://www.intechopen.com/books/advances_in_robotics_automation_and_control/jpeg_for_arabic_handwritten__character_recognition__add_a_dimension_of_application

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821