SACH'06

SUMMIT
ON
ARABIC AND
CHINESE HANDWRITING

Samuel Riggs IV Alumni Center
University of Maryland
College Park, MD 20742
September 27-28, 2006

Proceedings

SACH'06

The 2006 Summit on Arabic and Chinese Handwriting

Samuel Riggs IV Alumni Center
College Park, Maryland
September 27-28, 2006

Organized by: The Laboratory for Language and Media Processing
Institute for Advanced Computer Studies
University of Maryland
College Park, MD 20742

A limited number of additional copies of these proceedings are available for $60 from:
University of Maryland
Institute for Advanced Computer Studies
College Park, MD 20742
Phone (301) 405-6444
Fax: (301) 314-9115
Email: sach06@umiacs.umd.edu

Table of Contents

SESSION 4

SESSION 5

SESSION 6

SESSION 7

DEMONSTRATION AND POSTER ABSTRACTS

Message from the Organizers

Welcome to the inaugural Summit on Arabic and Chinese Handwriting.

The University of Maryland, along with government sponsors have invited leading researchers from industry, academia, and government agencies involved in document image analysis to a two-day Summit on Arabic and Chinese Handwriting Recognition (SACH'06). This event is intended to complement the bi-annual Symposium on Document Image Understanding Technology (SDIUT) meetings.

SACH'06 is sponsored in part by federal agencies funding document research and will serve as an overview of the state-of-the-art in handwriting recognition, highlighting Arabic and Chinese. The summit offers a forum for interaction with prominent researchers at the forefront of the scientific community and provides an opportunity for participants to help guide future directions in the field. Handwriting recognition remains the Holy Grail of document analysis and Arabic and Chinese scripts embrace many of the most significant challenges.

We are honored to have nine invited talks from six different countries representing innovative research in handwriting recognition around the world. These invited talks complement a series of technical talks and a government panel wrapping up the Summit. This proceeding is offered as a snapshot of the speakers' technical contributions, and encourages each of you to use the meeting to engage these experts in further discussions about their research.

As you know, meetings such as SACH would not be possible without the extra efforts of those working behind the scenes. We would like to thank Denise Best for her commitment to organizing the logistics of this event. We sincerely hope you enjoy your visit to Maryland and this Summit provides you with many opportunities to learn and collaborate.

Welcome!

David Doermann
Stefan Jaeger

Government Speaker

Dr. Joseph Olive
DARPA Information Processing Technology Office

Biographical Sketch

Dr. Olive has had over thirty years of experience in research and development at Bell Laboratory and 19 years experience in management. He has been the world leader in research of text-to-speech synthesis and has managed a world-class team in computer dialogue systems and human-computer communication. In his role as Director of speech research and CTO of Lucent's Business Unit Lucent Speech Solutions, he supervised the productization of Bell-Labs core speech technologies: Automatic Speech Recognition (ASR), Text-to-Speech Synthesis (TTS) and Speaker Verification (SV). He also led the dialogue research team to create a "next-generation" dialogue system for e-mail reading and navigation.

Dr. Olive graduated from the University of Chicago with a degree in Physics. While he was a graduate student, his research consisted of computational atomic physics requiring intensive use of computers for the computation of electrons distribution functions. He was also a member of the University of Chicago's computer center. Dr. Olive also earned an M.A. in music composition, a degree that he used to pursue a side career in writing music for small chamber groups, orchestras, computer music, and an opera for a computer, soprano and a small ensemble. After leaving the University of Chicago, Dr. Olive combined his interest in computation and music and began research in acoustics and signal processing.

Dr. Olive was a recipient of the National Endowment for the Arts grant in 1974 to write a computer opera. He was also the recipient of the Bell-Labs' Distinguished Member of Technical Staff award in 1984.

Technical Papers

Handwritten Chinese Character Recognition: Effects of Shape Normalization and Feature Extraction

Cheng-Lin Liu
National Laboratory of Pattern Recognition (NLPR)
Institute of Automation, Chinese Academy of Sciences (CASIA)
PO Box 2728, Beijing 100080, P.R. China
E-mail: liucl@nlpr.ia.ac.cn

Abstract

The field of handwritten Chinese character recognition (HCCR) has seen significant advances in the last two decades, owing to the effectiveness of many techniques, especially those for character shape normalization and feature extraction. This paper reviews the major methods of normalization and feature extraction, and evaluates their performance experimentally. The normalization methods include linear normalization, nonlinear normalization (NLN) based on line density equalization, moment normalization (MN), bi-moment normalization (BMN), modified centroid-boundary alignment (MCBA), and their pseudo-two-dimensional (pseudo 2D) extensions. As to feature extraction, we focus on some effective variations of direction features: chaincode feature, normalization-cooperated chaincode feature (NCCF), and gradient feature. We have compared the normalization methods previously, but in this study, will compare them with better implementation of features. As results, the current methods perform superiorly on handprinted characters, but are insufficient for unconstrained handwriting.

1 Introduction

Since the first work of printed Chinese character recognition (PCCR) was published in 1966 [1], many research efforts have been contributed, to both printed and handwritten Chinese character recognition (HCCR). Research on online HCCR was started as early as PCCR [2], whereas offline HCCR was started in late 1970s, and has attracted high attention from the 1980s [3]. Since then, many effective methods have been proposed to solve this problem, and the recognition performance has advanced significantly [4, 5]. This paper is mainly concerned with offline HCCR, but most methods of offline recognition are applicable to online recognition as well [6].

The approaches of HCCR can be roughly grouped into two categories: feature matching (statistical classification) and structure analysis. Based on feature vector representation of character patterns, feature matching approaches usually computed a simple distance measure (correlation matching), say, Euclidean or city block distance, between the test pattern and class prototypes. Currently, sophisticated classification techniques [7, 8, 9], including parametric and non-parametric statistical classifiers, neural networks, support vector machines (SVMs), etc., can yield higher recognition accuracies. Nevertheless, the selection and extraction of features remains an important issue. Structure analysis is an inverse process of character generation: to extract the constituent strokes and compute a structural distance measure between the test pattern and class models. Due to its resembling of human cognition and the potential of absorbing large deformation, this approach was pursued intensively in the 1980s and is still advancing [10]. However, due to the difficulty of stroke extraction and structural model building, it is not widely followed.

Statistical approaches have achieved great success in handprinted character recognition and are well commercialized due to some factors. First, feature extraction based on template matching and classification based on vector computation are easy to implement and computationally efficient. Second, effective shape normalization and feature extraction techniques, which improve the separability of patterns of different classes in feature space, have been proposed. Third, current machine learning methods enable classifier training with large set of samples for better discriminating shapes of different classes.

The methodology of Chinese character recognition has been largely affected by some important techniques: blurring [11], directional pattern match-

ing [12, 13, 14], nonlinear normalization [15, 16], modified quadratic discriminant function (MQDF) [17], etc. These techniques and their variations or improved versions, are still widely followed and adopted in most recognition systems. Blurring is actually a low-pass spatial filtering operation. It was proposed in the 1960s from the viewpoint of human vision, and is effective to blur the stroke displacement of characters of same class. Directional pattern matching, motivated from local receptive fields in vision, is the predecessor of current direction histogram features. Nonlinear normalization, which regulates stroke positions as well as image size, significantly outperforms the conventional linear normalization (resizing only). The MQDF is a nonlinear classifier suitable for high-dimensional features and large number of classes. Its variations include the pseudo Bayes classifier [18], the modified Mahalanobis distance [19], etc.

This paper reviews the major normalization and feature extraction methods and evaluate their performance in offline HCCR on large databases. The normalization methods include linear normalization (LN), nonlinear normalization (NLN) based on line density equalization [15, 16], moment normalization (MN) [20], bi-moment normalization (BMN) [21], modified centroid-boundary alignment (MCBA) [22], as well as the pseudo-two-dimensional (pseudo 2D) extensions of them [23, 24]. These methods have been evaluated previously [24], but in this study, they will be evaluated with better implementation of features.

Though many features have been proposed for character recognition, we focus on the class of direction histogram features, including chaincode direction feature, normalization-cooperated chaincode feature (NCCF) [25], and gradient direction feature. These features have yielded superior performance due to the sensitivity to stroke-direction variance and the insensitivity to stroke-width variance. The gradient direction feature was not paid enough attention until the success of gradient vector decomposition [26], following a decomposition scheme previously proposed in online character recognition [27]. Alternatively, the direction of gradient was quantized into a number of angular regions [28]. By NCCF, the chaincode direction is taken from the original image instead of the normalized image, but the directional elements are displaced in normalized planes according to normalized coordinates. An improved version of NCCF maps chaincodes into continuous line segments in normalized planes [29].

In the history, some extensions of direction feature, like the peripheral direction contributivity (PDC) [30] and the reciprocal feature field [31], have reported higher accuracy in HCCR when simple distance metric was used. These features, with very high dimensionality (over 1,000), are actually highly redundant. As background features, they are sensitive to noise and connecting strokes. Extending the line element of direction feature to higher-order feature detectors (e.g., [32, 33]) helps discriminate similar characters, but the dimensionality also increases rapidly. The Gabor filter, also motivated from vision research, promises feature extraction in character recognition [34], but is computationally expensive compared to chaincode and gradient features, and at best, perform comparably with the gradient feature [35].

We evaluate the character shape normalization and direction feature extraction methods on two databases of handwritten characters, ETL9B (Electrotechnical Laboratory, Japan) and CASIA (Institute of Automation, Chinese Academy of Sciences), with 3,036 classes and 3,755 classes, respectively. Two common classifiers, minimum distance classifier and modified quadratic discriminant function (MQDF), are used to evaluate the recognition accuracies.

The purpose of this study is twofold. First, the comparison of major normalization and feature extraction methods can provide guidelines for selecting methods in system development. Second, the results show to what degree of performance the state-of-the-art methods can achieve. We will show in experiments that the current methods can recognize handprinted characters accurately but perform inferiorly on unconstrained handwriting.

In the rest of this paper, we review major normalization methods in Section 2 and direction feature extraction methods in Section 3. Experimental results are presented in Section 4, and finally, concluding remarks are offered in Section 5.

2 Shape Normalization

Normalization is to regulate the size, position, and shape of character images, so as to reduce the shape variation between the images of same class. Denote the input image and the normalized image by $f(x, y)$ and $g(x', y')$, respectively, normalization is implemented by coordinate mapping

$$\begin{cases} x' &= x'(x, y), \\ y' &= y'(x, y). \end{cases} \tag{1}$$

Most normalization methods use 1D coordinate mapping:

$$\begin{cases} x' &= x'(x), \\ y' &= y'(x). \end{cases} \tag{2}$$

Under 1D normalization, the pixels at the same row/column in the input image are mapped to the same row/column in the normalized image, and hence, the shape restoration capability is limited.

Given coordinate mapping functions (1) or (2), the normalized image $g(x', y')$ is generated by pixel value and coordinate interpolation. In our implementation of 1D normalization, we map the coordinates forwardly from (binary) input image to normalized image, and use coordinate interpolation to generate the binary normalized image. For generating gray-scale normalized image, each pixel is viewed as a square of unit area. By coordinate mapping, the unit square of input image is mapped to a rectangle in the normalized plane, and each pixel (unit square) overlapping with the mapped rectangle is assigned a gray level proportional to the overlapping area [29].

In the case of 2D normalization, the mapped shape of a unit square onto the normalized plane is a quadrilateral [24]. To compute the overlapping areas of this quadrilateral with the pixels (unit squares) in the normalized plane, the quadrilateral is decomposed into trapezoids such that each trapezoid is within a row of unit squares. Each within-row trapezoid is further decomposed into trapezoids within a unit square. After generating the normalized gray-scale image, the binary normalized image is obtained by thresholding the gray-scale image (fixed threshold 0.5).

In our experiments, the normalized image plane is set to a square of edge length L, which is not necessarily fully occupied. To alleviate the distortion of elongated characters, we partially preserve the aspect ratio of the input image. By aspect ratio adaptive normalization (ARAN) [29, 36], the aspect ratio R_2 of normalized image is a continuous function of the aspect ratio R_1 of input image:

$$R_2 = \sqrt{\sin(\frac{\pi}{2} R_1)}. \tag{3}$$

R_1 is calculated by

$$R_1 = \begin{cases} W_1/H_1, & \text{if } W_1 < H_1 \\ H_1/W_1, & \text{otherwise} \end{cases} \tag{4}$$

where W_1 and H_1 are the width and height of the input image. The width W_2 and height H_2 of the normalized image are similarly related by the aspect ratio R_2. If the input image is vertically elongated, then in the normalized plane, the vertical dimension is filled (height L) and the horizontal dimension is centered and scaled according to the aspect ratio; otherwise, the horizontal dimension is filled (width L) and the vertical dimension is centered and scaled. ARAN is depicted in Fig. 1.

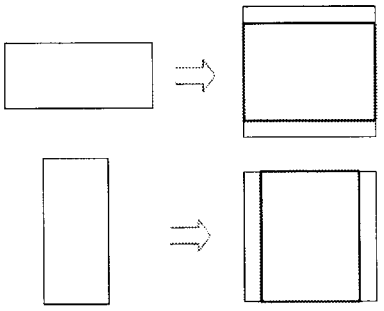


Figure 1: Aspect ratio adaptive normalization (ARAN). Rectangle with thick line: occupied area of normalized image.

The normalization methods depend on the coordinate mapping functions, defined by the 1D and pseudo 2D normalization methods as follows.

2.1 1D normalization methods

Given the sizes of input and normalized images, the coordinate mapping functions of linear normalization (LN) are simply

$$\begin{cases} x' &= \frac{W_2}{W_1} x, \\ y' &= \frac{H_2}{H_1} y. \end{cases} \tag{5}$$

Both the linear normalization and line-density-based nonlinear normalization (NLN) methods align the physical boundaries (ends of stroke projections) of input image to the boundaries of normalized image. The coordinate mapping of NLN is obtained by accumulating the normalized line density projections (line density equalization):

$$\begin{cases} x' &= W_2 \sum_{u=0}^{x} h_x(u), \\ y' &= H_2 \sum_{v=0}^{y} h_y(v), \end{cases} \tag{6}$$

where $h_x(x)$ and $h_y(y)$ are the normalized line density histograms of x axis and y axis, respectively, which are obtained by normalizing the projections of local line densities into unity sum:

$$\begin{cases} h_x(x) &= \frac{p_x(x)}{\sum_x p_x(x)} = \frac{\sum_y d_x(x,y)}{\sum_x \sum_y d_x(x,y)}, \\ h_y(y) &= \frac{p_y(y)}{\sum_y p_y(y)} = \frac{\sum_x d_y(x,y)}{\sum_x \sum_y d_y(x,y)}, \end{cases} \tag{7}$$

where $p_x(x)$ and $p_y(y)$ are the line density projections onto x axis and y axis, respectively, and $d_x(x, y)$ and $d_y(x, y)$ are local line density functions.

By Tsukumo and Tanaka [15], the local line densities d_x and d_y are taken as the reciprocal of horizontal/vertical run-length in background area, or a small constant in stroke area. While by Yamada et al. [16], d_x and d_y are calculated considering both background run-length and stroke run-length, and are unified to render $d_x(x,y) = d_y(x,y)$. The two methods perform comparably but the one of Tsukumo and Tanaka is computationally simpler [5]. By adjusting the density functions of marginal and stroke areas empirically in Tsukumo and Tanaka's method, we have achieved better performance than the method of Yamada et al. This improved version of NLN is taken in our experiments.

The 1D moment normalization (MN) method (a simplified version of Casey's method [20]) aligns the centroid of input image (x_c, y_c) to the geometric center of normalized image $(x'_c, y'_c) = (W_2/2, H_2/2)$, and re-bound the input image according to second-order 1D moments. Let the second-order moments be μ_{20} and μ_{02}, the width and height of input image are re-set to $\delta_x = 4\sqrt{\mu_{20}}$ and $\delta_y = 4\sqrt{\mu_{02}}$, respectively. The coordinate mapping functions are then given by

$$\begin{cases} x' &= \frac{W_2}{\delta_x}(x - x_c) + x'_c, \\ y' &= \frac{H_2}{\delta_y}(y - y_c) + y'_c. \end{cases} \tag{8}$$

The bi-moment normalization (BMN) method [21] aligns the centroid of input image as moment normalization does, but the width and height are treated asymmetric with respect to the centroid. To do this, the second-order moments are split into two parts by the centroid: μ_x^-, μ_x^+, μ_y^-, and μ_y^+. The boundaries of input image are re-set to $[x_c - 2\sqrt{\mu_x^-}, x_c + 2\sqrt{\mu_x^+}]$ and $[y_c - 2\sqrt{\mu_y^-}, y_c + 2\sqrt{\mu_y^+}]$. For the x axis, a quadratic function $u(x) = ax^2 + bx + c$ is used to align three points $(x_c - 2\sqrt{\mu_x^-}, x_c, x_c + 2\sqrt{\mu_x^+})$ to normalized coordinates $(0, 0.5, 1)$, and similarly, a quadratic function $v(y)$ is used for the y axis. Finally, the coordinate functions are

$$\begin{cases} x' &= W_2 u(x), \\ y' &= H_2 v(y). \end{cases} \tag{9}$$

The quadratic functions can also be used to align the physical boundaries and centroid, i.e., map $(0, x_c, W_1)$ and $(0, y_c, H_1)$ to $(0, 0.5, 1)$. We call this method centroid-boundary alignment (CBA). A modified CBA (MCBA) method [22] also adjusts the stroke density in central area by combining a sine function with the quadratic functions:

$$\begin{cases} x' &= W_2[u(x) + \eta_x \sin(2\pi u(x))], \\ y' &= H_2[v(y) + \eta_y \sin(2\pi v(y))]. \end{cases} \tag{10}$$

The amplitudes of sine waves, η_x and η_y, are estimated from the extent of the central area, which is defined by the centroids of partial images divided by the global centroid.

2.2 Pseudo 2D normalization methods

Horiuchi et al. proposed a pseudo 2D nonlinear normalization (P2DNLN) method by equalizing the line density functions of each row/column instead of the line density projections [23]. To control the degree of shape deformation, they blurred the line density functions such that the equalization of each row/column is dependent on its neighboring rows/columns. Though this method promises recognition, it is computationally expensive due to the row/column-wise line density blurring.

An efficient pseudo 2D normalization approach, called line density projection interpolation (LDPI), was proposed recently [24]. Instead of line density blurring and row/column-wise equalization, LDPI partitions the 2D line density map into soft strips. 1D coordinate functions are computed from the density projection of each strip and combined to a 2D function. Specifically, let the width and height of the input image be W_1 and H_1, the centroid be (x_c, y_c), we partition the horizontal line density map $d_x(x,y)$ into three horizontal strips:

$$d_x^i(x,y) = w^i(y) d_x(x,y), \quad i = 1,2,3. \tag{11}$$

$w^i(y)$ $(i = 1,2,3)$ are piecewise linear functions:

$$\begin{cases} w^1(y) = w_0 \frac{y_c - y}{y_c}, & y < y_c, \\ w^2(y) = 1 - w^1(y), & y < y_c, \\ w^2(y) = 1 - w^3(y), & y \geq y_c, \\ w^3(y) = w_0 \frac{y - y_c}{H_1 - y_c}, & y \geq y_c, \end{cases} \tag{12}$$

where w_0 controls the weight of the upper/lower part of line density map. A small value of w_0 renders the interpolated 2D coordinate function close to that of 1D normalization, while a large one may yield excessive deformation. The weight functions with $w_0 = 1$ are depicted in Fig. 2.

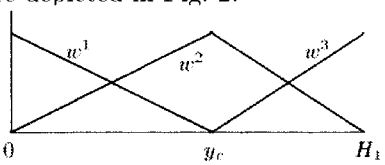


Figure 2: Weight functions for partitioning line density map into soft strips.

The horizontal density functions of three strips are

projected onto the x axis:

$$p_x^i(x) = \sum_y d_x^i(x,y), \quad i = 1,2,3. \tag{13}$$

The projections are then normalized to unity sum and accumulated to give 1D coordinate functions $x'^i(x)$, $i = 1,2,3$, which are combined to 2D coordinate function by interpolation:

$$x'(x,y) = \begin{cases} w^1(y)x'^1(x) + w^2(y)x'^2(x), & y < y_c, \\ w^3(y)x'^3(x) + w^2(y)x'^2(x), & y \geq y_c. \end{cases} \tag{14}$$

Similar to the partitioning of horizontal density, the vertical density map $d_y(x,y)$ is partitioned into three vertical strips using weight functions in x axis. The partitioned density functions $d_y^i(x,y)$, $i = 1,2,3$, are similarly equalized and interpolated to generate the 2D coordinate function $y'(x,y)$.

The strategy of LDPI is applied to extend other 1D normalization methods: MN, BMN, and MCBA. The extended versions are called pseudo 2D MN (P2DMN), pseudo 2D BMN (P2DBMN), and pseudo 2D CBA (P2DCBA), respectively. These methods do not rely on the computation of local line density map. Instead, they are directly based on the pixel intensity of character image. As the soft partitioning of line density map in LDPI, the input character image $f(x,y)$ is softly partitioned into three horizontal strips $f_x^i(x,y)$, $i = 1,2,3$, and three vertical strips $f_y^i(x,y)$, $i = 1,2,3$. The horizontal strips are projected onto the x axis:

$$p_x^i(x) = \sum_y f_x^i(x,y), \quad i = 1,2,3. \tag{15}$$

For P2DMN, the second order moment is computed from the projection of a strip:

$$\mu_{20}^i = \frac{\sum_x (x - x_c^i)^2 p_x^i(x)}{\sum_x p_x^i(x)}. \tag{16}$$

The width of this strip is re-set to $\delta_x^i = 4\sqrt{\mu_{20}^i}$, which is used to determine the scaling factor of 1D coordinate mapping:

$$x'^i(x) = \frac{W_2}{\delta_x^i}(x - x_c^i) + \frac{W_2}{2}. \tag{17}$$

And the 1D coordinate functions of vertical strips are computed from strip projections similarly.

For P2DBMN, the second order moment of a horizontal strip is split into two parts at the centroid of this strip: μ_{20}^{i-} and μ_{20}^{i+}. The bounds of this strip is re-set to $[x_c^i - 2\sqrt{\mu_{20}^{i-}}, x_c^i + 2\sqrt{\mu_{20}^{i+}}]$, which, together with the centroid x_c^i, are used to estimate the quadratic 1D coordinate mapping function $x'^i(x)$. The three 1D coordinate functions of vertical strips are computed similarly.

By P2DCBA, from the vertical projection of each horizontal strip $f_x^i(x,y)$ $(i = 1,2,3)$, the centroid coordinate x_c^i and two partial centroids x_{c1}^i and x_{c2}^i are computed to estimate the parameters of 1D coordinate mapping function $x'^i(x)$. Similarly, 1D coordinate functions $y'^i(y)$ $(i = 1,2,3)$ are estimated from the horizontal projections of vertical strips.

More details of pseudo 2D normalization can be found in [24]. Some examples of normalization using nine methods (LN, NLN, MN, BMN, MCBA, LDPI, P2DMN, P2DBMN, and P2DCBA) are shown in Fig. 3. We can see that whereas linear normalization (LN) keeps the original shape (only aspect ratio changed), NLN can effectively equalize the line intervals. The centroid-based normalization methods (MN, BMN, and MCBA) effectively regulate the overall shape (skewness of gravity, balance of inner/outer stroke density). The pseudo 2D methods make the stroke positions more uniform, especially, alleviate the imbalance of width/height and position of character parts.

3 Direction Feature Extraction

The implementation of direction feature is varying depending on the directional element decomposition, the sampling of feature values, the resolution of direction and feature plane, etc. Considering that the stroke segments of Chinese characters can be approximated into four orientations: horizontal, vertical, left-diagonal and right-diagonal, early works usually decomposed the stroke (or contour) segments into four orientations.

Feature extraction from stroke contour has been widely adopted because the contour length is nearly independent on stroke-width variation. The local direction of contour, encoded as a chaincode, actually has eight directions (Fig. 4). Decomposing the contour pixels into eight *directions* instead of four *orientations* (a pair of opposite directions merged into one orientation) was shown to significantly improve the recognition accuracy [26]. This is because separating the two sides of stroke edge can better discriminate parallel strokes. The direction of stroke edge can also be measured by the gradient of image intensity, which applies to gray-scale image as well binary image. The gradient feature has been applied to Chinese character recognition in 8-direction [37] and 12-direction [38].

Direction feature extraction is accomplished in

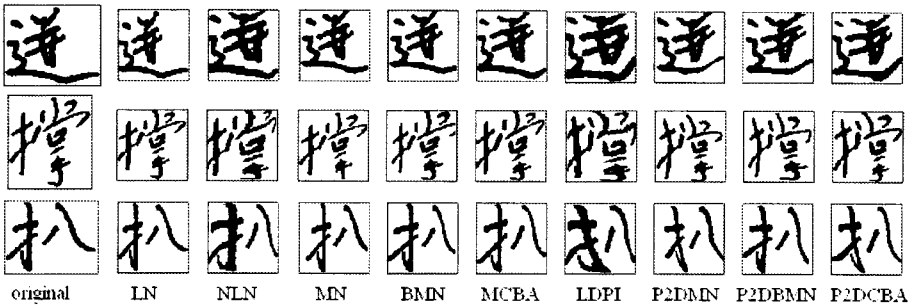


Figure 3: Character image normalization by nine methods. The leftmost image is original and the other eight are normalized ones.

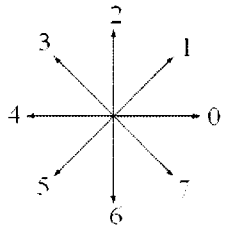


Figure 4: Eight directions of chaincodes.

three steps: image normalization, directional decomposition, and feature sampling. Conventionally, the contour/edge pixels of normalized image are assigned to a number of direction planes. The normalization-cooperated feature extraction (NCFE) strategy [25], instead, assigns the chaincodes of original image into direction planes. Though the normalized image is not generated by NCFE, the coordinates of pixels in original image are mapped to a standard plane, and the extracted feature is thus dependent on the normalization method.

Direction feature is also called direction histogram feature because at a pixel or a local region in normalized image, the strength values of N_d directions form a local histogram. Alternatively, we view the strength values of one direction as a directional image (direction plane).

In the following, we first describe the directional decomposition procedures for three types of direction features: chaicode direction feature, normalization-cooperated chaincode feature (NCCF), and gradient direction feature, and then address the sampling of direction planes.

3.1 Directional decomposition

Directional decomposition results in a number of direction planes (with same size as the normalize image), $f_i(x, y)$, $i = 1, \ldots, N_d$. We first describe the procedures for decomposing contour/gradient into eight directions, then extend to 12 directions and 16 directions.

In binary image, a contour pixel is a black point with at least one of its 4-connected neighbors being white. The 8-direction chaincodes of contour pixels can be decided by contour tracing, or more simply, by raster scan [39]. At a black pixel (x, y), denoting the values of 8-connected neighbors in counterclockwise as p_k, $0, 1, \ldots, 7$, with the east neighbor being p_0. For $k = 0, 2, 4, 6$, if $p_k = 0$, check p_{k+1}: if $p_{k+1} = 1$ (chaincode $k + 1$), $f_{k+1}(x, y)$ increases by 1; otherwise, if $p_{(k+2)\%8} = 1$ (chaincode $(k + 2)\%8$), $f_{(k+2)\%8}(x, y)$ increases by 1.

For NCCF, each chaincode in the original image is viewed as a line segment connecting two neighboring pixels, which is mapped to another line segment in a standard direction plane by coordinate mapping. In the direction plane, each pixel (unit square) crossed by the line segment in the main (x or y) direction is given a unit of direction contribution. To exploit the continuous nature of line segment, the strength of line direction falling in a pixel is proportional to the length of line segment falling in the unit square (continuous NCCF [29]). As in Fig. 5, where a line segment mapped from a chaincode covers four unit squares A, B, C and D. By discrete NCCF, the pixels A and C are assigned a direction unit, whereas by continuous NCCF, all the four pixels are assigned direction strengths proportional to the in-square line segment length.

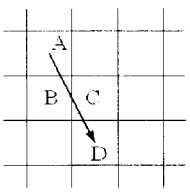


Figure 5: NCCF on continuous direction plane.

In gradient direction feature extraction, the gradi-

ent vector, computed on the normalized image using the Sobel operator, is decomposed into components in eight chaincode directions. The Sobel operator has two masks to compute the gradient components in two axes. The masks are shown in Fig. 6, and the gradient $\mathbf{g}(x,y) = [g_x, g_y]^T$ at location (x,y) is computed by

$$\begin{aligned} &g_x(x,y) \\ &= f(x+1,y-1) + 2f(x+1,y) + f(x+1,y+1) \\ &\quad -f(x-1,y-1) - 2f(x-1,y) + f(x-1,y+1), \\ &g_y(x,y) \\ &= f(x-1,y+1) + 2f(x,y+1) + f(x+1,y+1) \\ &\quad -f(x-1,y-1) - 2f(x,y-1) + f(x+1,y-1). \end{aligned} \tag{18}$$

-1	0	1
-2	0	2
-1	0	1

1	2	1
0	0	0
-1	-2	-1

Figure 6: Templates of Sobel gradient operator.

The gradient strength and direction can be computed from the vector $[g_x, g_y]^T$. The range of gradient direction can be partitioned into a number (say, 8 or 16) of regions and each region corresponds to a direction plane. More effectively, the gradient vector is decomposed into components in standard directions, following a strategy previously proposed in online character recognition [27]. In this scheme, if a gradient direction lies between two standard directions, the vector is decomposed into two components as shown in Fig. 7. The length of each component is assigned to the corresponding direction plane at the pixel (x,y).

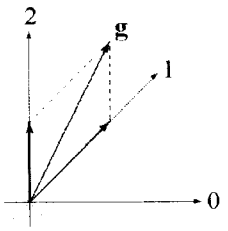


Figure 7: Decomposition of gradient vector.

Fig. 8 shows the direction planes of three decomposition schemes: NCCF, chaincodes of normalization image, and gradient of normalized image. The direction planes are arranged in the order of stroke orientation. We can see that the planes of chaincode directions (third row) are very similar to those of gradient directions. The planes of NCCF, describing the local directions of original image, show some difference. Comparing the original image and the normalized image, the orientation of the right-hand stroke, near left-diagonal orientation, deforms to near vertical. Consequently, the direction planes of left-diagonal orientation of NCCF are stronger than those of chaincodes and gradient, while the planes of vertical orientation of NCCF are weaker than those of chaincodes and gradient.

3.2 Extention to more directions

The extension of gradient decomposition into more than eight directions is straightforward: simply setting N_d standard directions with angle interval $360/N_d$ and typically, with one direction pointing to east, then decompose each gradient vector into two components in standard directions and assign the component lengths to corresponding direction planes. We set N_d to 12 and 16.

To decompose contour pixels into 16 directions follows the 16-direction extended chaincodes, which is defined by two consecutive chaincodes. In the weighted direction histogram feature of Kimura et al. [18], 16-direction chaincodes are down-sampled by weighted average to form 8-direction planes.

Again, we can determine the 16-direction chaincode of contour pixels by raster scan. At a contour pixel (x,y), when its 4-connected neighbor $p_k = 0$ and the counterclockwise successor $p_{k+1} = 1$ or $p_{(k+2)\%2} = 1$, search the neighbors clockwise from p_k until a $p_j = 1$ is found. The two contour pixels, p_{k+1} or $p_{(k+2)\%2}$ and p_j, form a 16-direction chaincode. For example, in Fig. 9, the center pixel has the east neighbor being 0, the north neighbor alone defining the 8-direction chaincode, and defining a 16-direction chaincode together with the southeast neighbor. The 16-direction chaincode can be indexed from a table of correspondence between the code and the difference of coordinates of two pixels forming the code, as shown in Fig. 10. Each contour pixel has a unique 16-direction code.

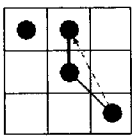


Figure 9: 16-direction chaincode formed from two 8-direction chaincodes.

For decomposing contour pixels into 12 directions, the difference of coordinates corresponding to a 16-direction chaincode is viewed as a vector (the dashed line in Fig. 9), which is decomposed into components in 12 standard directions as a gradient vector

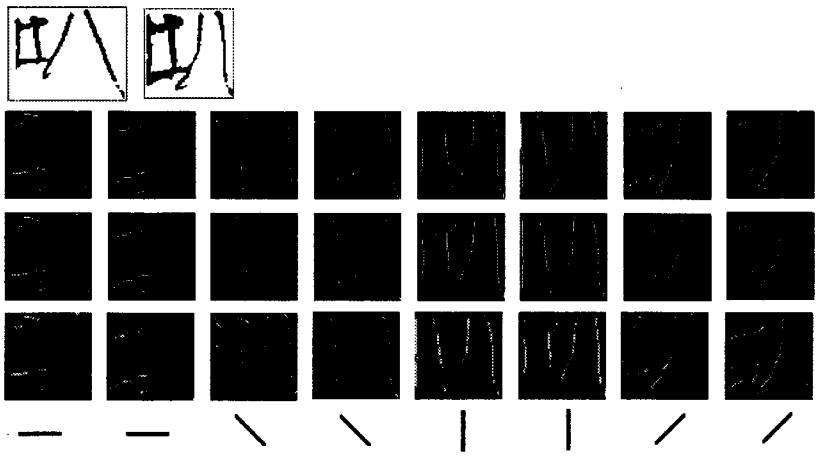


Figure 8: Original image and normalized image (top row), 8-direction planes of NCCF (second row), chaincode planes of normalized image (third row), and gradient direction planes (bottom row).

(-2,2) 6	(-1,2) 5	(0,2) 4	(1,2) 3	(2,2) 2
(-2,1) 7	(-1,1) 6	(0,1) 4	(1,1) 2	(2,1) 1
(-2,0) 8	(-1,0) 8	(0,0)	(1,0) 0	(2,0) 0
(-2,-1) 9	(-1,-1) 10	(0,-1) 12	(1,-1) 14	(2,-1) 15
(-2,-2) 10	(-1,-2) 11	(0,-2) 12	(1,-2) 13	(2,-2) 14

Figure 10: Difference of coordinates and the corresponding 16-direction chaincodes.

is done. In this sense, the 12-direction code of a contour pixel is not unique. For 12-direction chaincode feature extraction, a contour pixel is assigned to two direction planes, with strength proportional to the component length. For NCCF, the two corresponding direction planes are assigned strengths proportional to the overlapping length of the line segment mapped by coordinate functions, as in Fig. 5.

3.3 Blurring and sampling

Each direction plane, with same size as the normalized image, need to be reduced to extract feature values of moderate dimensionality. A simple way is to partition the direction plane into a number of block zones and take the total or average value of each zone as a feature value. Partition of variable-size zones was proposed to overcome the non-uniform distribution of stroke density [13], but is not necessary when nonlinear or pseudo 2D normalization is done. Overlapping blocks were taken to alleviate the effect of stroke-position variation on the boundary of blocks [40], yet a more effective way is to partition the plane into soft zones, which follow the principle of low-pass spatial filtering and sampling [39]. The blurring operation of Iijima [11] implies spatial filtering without down-sampling.

In implementation of blurring, the impulse response function (IRF) of spatial filter is approximated into a weighted window, which is also called a blurring mask. The IRF is often taken as a Gaussian function:

$$h(x,y) = \frac{1}{2\pi\sigma_x^2} \exp\Big(-\frac{x^2+y^2}{2\sigma_x^2}\Big). \qquad (19)$$

According to the Sampling Theorem, the variance parameter σ_x is related to the sampling frequency (the reciprocal of sampling interval). On truncating the band-width of Gaussian filter, an empirical formula was given in [39]:

$$\sigma_x = \frac{\sqrt{2}t_x}{\pi}, \qquad (20)$$

where t_x is the sampling interval. At a location (x_0, y_0) of image $f(x,y)$, the convolution gives a sampled feature value

$$F(x_0,y_0) = \sum_x \sum_y f(x,y) \cdot h(x-x_0, y-y_0). \qquad (21)$$

Fig. 11 show the blurred images (without down-sampling) of the direction planes in Fig. 8. By blurring, the sparse pixels in direction planes merge into strokes or blobs.

For ease of implementation, we partition a direction plane into a mesh of equal-size blocks and set the sampling points to the center of each block. Assume to extract $K \times K$ values from a plane, the size of plane is set to $Kt_x \times Kt_x$. From N_d direction

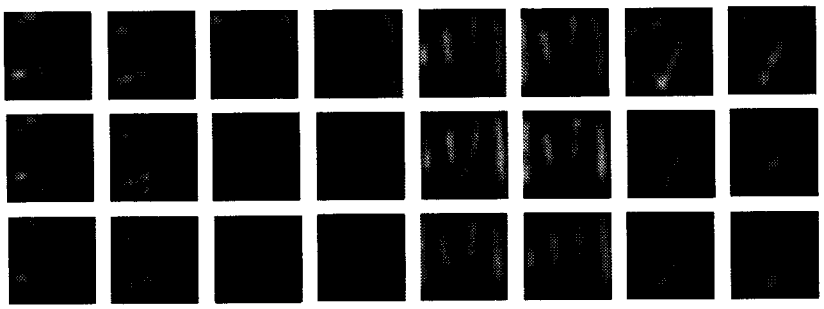


Figure 11: Blurred images (not down-sampled) of the direction planes in Fig. 8.

planes, the total number of extracted feature values is $N_d \times K^2$.

The extracted feature values are causal variables. Power transformation can make the density function of causal variables closer to Gaussian [7]. This is helpful to improve the classification performance of statistical classifiers. Power transformation is also called variable transformation [41] or Box-Cox transformation [42]. We transform each feature value with power 0.5 without attempt to optimize the transformation functions.

4 Performance Evaluation

We first compare the performance of the normalization and feature extraction methods on two large databases of handprinted characters (constrained writing), and then test the performance on a small set of unconstrained handwritten characters.

We use two classifiers for classification: the Euclidean distance to class mean (minimum distance classifier) and the MQDF [17]. For reducing the classifier complexity and improving classification accuracy, the feature vector is transformed to a lower dimensionality by Fisher linear discriminant analysis (FLDA) [7]. We set the reduced dimensionality to 160 for all feature types.

Denote the d-dimensional feature vector (after dimensionality reduction) by $\mathbf{x}$, the MQDF of class ω_i is computed by

$$\begin{aligned} g_2(\mathbf{x},\omega_i) &= \sum_{j=1}^{k} \frac{1}{\lambda_{ij}}[(\mathbf{x}-\mu_i)^T\phi_{ij}]^2 \\ &+\frac{1}{\delta_i}\left\{\|x-\mu_i\|^2 - \sum_{j=1}^{k}[(\mathbf{x}-\mu_i)^T\phi_{ij}]^2\right\} \\ &+\sum_{j=1}^{k}\log\lambda_{ij} + (d-k)\log\delta_i, \end{aligned} \tag{22}$$

where μ_i is the mean vector of class ω_i, λ_{ij} and ϕ_{ij}, $j = 1,\ldots,d$, are the eigenvalues and eigenvectors of the covariance matrix of class ω_i. The eigenvalues are sorted in non-ascending order and the eigenvectors are sorted accordingly. k denotes the number of principal axes, and the minor eigenvalues are replaced with a constant δ_i. We set a class-independent constant δ_i which is proportional to the average feature variance, with the multiplier selected by 5-fold holdout validation on the training data set.

In our experiments, k was set to 40. The classification of MQDF is speeded up by selecting 100 candidate classes using Euclidean distance. The MQDF is then computed on the candidate classes only. Candidate selection is further accelerated by clustering the class means into groups. The input feature vector is first compared to cluster centers, and then compared to the class means contained in a number of nearest clusters. We set the total number of clusters to 220 for the ETL9B database and 250 for the CASIA database.

MQDF is a promising classification method for HCCR. Even higher performance can be achieved by, e.g., discriminative learning of feature transformation and classifier parameters [37, 38]. The optimization of classifier, however, is not the concern of this paper.

4.1 Performance on handprinted characters

The normalization and feature extraction methods are evaluated on two databases of handprinted characters. The ETL9B database contains the character images of 3,036 classes (71 hiragana, and 2,965 Kanji characters in the JIS level-1 set), 200 samples per class. This database has been widely evaluated by the community [18, 40, 43]. The CASIA database, collected by the Institute of Automation, Chinese Academy of Sciences, in early 1990s, contains the handwritten images of 3,755 Chinese characters (the level-1 set in GB2312-80 standard), 300 samples per class. Some sample images of the CASIA database are shown in Fig. 12.

In the ETL9B database, we use the first 20 and last 20 samples of each class for testing, and the re-

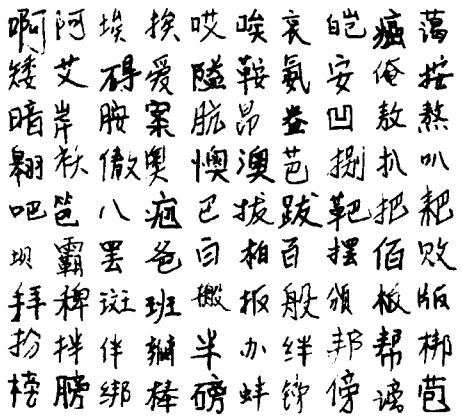


Figure 12: Test samples in the CASIA database.

maining samples for training classifiers. In the CASIA database, we use the first 250 samples of each class for training, and the remaining 50 samples per class for testing.

We first compare the performance of three direction features with varying direction resolutions with a common normalization method (Tsukumo and Tanaka's nonlinear normalization (NLN) method with our modifications). The direction resolution of features is set to 8, 12, and 16. For each direction resolution, three schemes of sampling mesh are tested. For 8-direction features, the mesh of sampling is set to 7×7 (M1), 8×8 (M2), and 9×9 (M3); for 12-direction, 6×6 (M1), 7×7 (M2), and 8×8 (M3); and for 16-direction, 5×5 (M1), 6×6 (M2), and 7×7 (M3). We control the size of normalized image (direction planes) to be around 64×64, and the dimensionality (before reduction) to be less than 800. The settings of sampling mesh are summarized in Table 1.

Table 1: Settings of sampling mesh for 8-direction, 12-direction, and 16-direction features.

Mesh	M1		M2		M3	
	zones	dim	zones	dim	zones	dim
8-dir	7×7	392	8×8	512	9×9	648
12-dir	6×6	432	7×7	588	8×8	768
16-dir	5×5	400	6×6	576	7×7	784

On classifier training and testing using different direction resolutions and sampling schemes, the error rates on the test set of ETL9B database are listed in Table 2, and the error rates on the test set of CASIA database are listed in Table 3. In the tables, the chaincode direction feature is denoted by **chn**, NCCF is denoted by **nccf**, and the gradient direction feature by **grd-g**. For chaincode feature extraction, the normalized binary image is smoothed using a connectivity-preserving smoothing algorithm [39]. Gradient feature is extracted from gray-scale normalized image.

Table 2: Error rates (%) of 8-direction, 12-direction, and 16-direction features on ETL9B database.

	Euclidean			MQDF		
chn	M1	M2	M3	M1	M2	M3
8-dir	2.91	2.94	3.02	1.08	1.05	1.09
12-dir	2.43	2.56	2.54	1.02	1.00	**1.97**
16-dir	2.52	**2.40**	2.52	1.20	1.00	1.00
nccf	M1	M2	M3	M1	M2	M3
8-dir	2.61	2.61	2.71	0.93	0.87	0.89
12-dir	2.05	2.06	2.13	0.82	**0.77**	0.78
16-dir	2.05	**2.04**	2.11	0.98	0.85	0.79
grd-g	M1	M2	M3	M1	M2	M3
8-dir	2.59	2.58	2.66	0.93	0.90	0.89
12-dir	2.27	2.30	2.31	0.94	0.86	0.86
16-dir	2.29	**2.19**	2.25	1.08	0.94	**0.85**

Table 3: Error rates (%) of 8-direction, 12-direction, and 16-direction features on CASIA database.

	Euclidean			MQDF		
chn	M1	M2	M3	M1	M2	M3
8-dir	5.95	6.07	6.22	2.45	2.37	2.37
12-dir	5.34	5.44	5.53	2.34	2.26	**2.21**
16-dir	5.44	**5.28**	5.37	2.72	2.35	2.29
nccf	M1	M2	M3	M1	M2	M3
8-dir	5.31	5.35	5.49	1.94	1.92	2.00
12-dir	4.44	4.48	4.55	1.86	1.75	**1.73**
16-dir	4.53	**4.42**	4.52	2.17	1.92	1.82
grd-g	M1	M2	M3	M1	M2	M3
8-dir	5.31	5.34	5.41	2.05	2.01	1.97
12-dir	4.94	4.90	4.98	2.09	2.00	**1.95**
16-dir	4.98	4.85	**4.80**	2.42	2.09	1.98

We can see that on either database, using either classifier (Euclidean or MQDF), the error rates of 12-direction and 16-direction features are mostly lower than those of 8-direction features. This indicates that increasing the resolution of direction decomposition is beneficial. The 16-direction feature, however, does not outperform the 12-direction feature. To select a sampling mesh, let us focus on the results of 12-direction features. We can see that by Euclidean distance classification, M1 (6×6) outperforms M2 (7×7) and M3 (8×8), whereas by MQDF, the error rates of M2 and M3 are lower than those of M1. Considering that M2 and M3 perform comparably while M2 has lower complexity, we take the sampling mesh M2 with 12-direction features for fol-

lowing experiments. The original dimensionality of direction features is now $12 \times 7 \times 7 = 588$.

Table 4: Error rates (%) of various normalization methods on ETL9B database.

	Euclidean			MQDF		
	chn	nccf	grd-g	chn	nccf	grd-g
LN	6.36	5.94	5.97	2.38	2.09	2.11
NLN	2.56	2.06	2.30	1.00	0.77	0.86
MN	2.35	2.07	2.12	0.95	0.83	0.82
BMN	2.33	2.04	2.09	0.92	0.81	0.80
MCBA	2.52	2.19	2.27	1.00	0.84	0.86
LDPI	2.08	1.65	1.90	**0.82**	**0.64**	**0.73**
2MN	2.05	1.65	1.84	0.86	0.69	0.74
2BMN	**1.97**	**1.60**	**1.78**	0.84	0.69	**0.73**
2CBA	2.13	1.81	1.93	0.86	0.72	0.77

Table 5: Error rates (%) of various normalization methods on CASIA database.

	Euclidean			MQDF		
	chn	nccf	grdg	chn	nccf	grd-g
LN	11.38	10.46	10.31	4.11	3.49	3.54
NLN	5.44	4.48	4.90	2.26	1.75	2.00
MN	5.61	4.89	4.90	2.50	2.04	2.06
BMN	5.30	4.54	4.56	2.35	1.93	1.92
MCBA	5.48	4.73	4.83	2.31	1.87	1.96
LDPI	**4.49**	**3.69**	4.21	**1.96**	**1.52**	**1.75**
2MN	4.99	3.97	4.33	2.24	1.70	1.91
2BMN	4.63	3.71	**4.07**	2.15	1.62	1.76
2CBA	4.75	3.95	4.25	2.11	1.68	1.78

On fixing the direction resolution (12-direction) and sampling mesh (7×7), we combine the three types of direction features with nine normalization methods. The weight parameter w_0 of pseudo 2D normalization was set to 0.75 for good recognition performance [24]. The error rates on the test sets of two databases are listed in Table 4 and Table 5, respectively. In the tables, the pseudo 2D normalization methods P2DMN, P2DBMN, and P2DCBA are denoted by **2MN**, **2BMN**, and **2CBA**, respectively, for saving space. Comparing the normalization methods, we can see that pseudo 2D methods are superior to 1D ones, and the linear normalization (LN) is inferior to other 1D normalization methods. To compare the three types of features, let us view the error rates of 1D normalization methods and those of pseudo 2D methods separately.

It can be seen in Table 4 and Table 5 that with 1D normalization, the NCCF and the gradient feature perform comparably, and both outperform the chaincode feature. Four normalization methods, namely, NLN, MN, BMN, and MCBA, perform comparably. With pseudo 2D normalization, the NCCF perform best, and the gradient feature outperforms the chaincode feature. Comparing the pseudo 2D normalization methods, LDPI and P2DBMN outperform P2DMN and P2DCBA (especially on the CASIA database). On both two databases, the best performance is yielded by the NCCF with LDPI normalization, and the NCCF with P2DBMN is competing.

The gradient feature performs comparably with NCCF with 1D normalization but inferiorly with pseudo 2D normalization. This is because pseudo 2D normalization, though equalize the stroke density better than 1D normalization, also deforms the stroke directions remarkably. While the gradient feature describes the deformed stroke directions, NCCF takes the stroke directions of original image.

4.2 Computation times

To compare the computational complexity of normalization methods, we profile the processing time in two sub-tasks: coordinate mapping and normalized image generation, and the latter is dichotomized into binary image and gray-scale image. Smoothing is involved in binary normalized image, but not for gray-scale image.

On the test samples of CASIA database, we counted the CPU times on Pentium-4-3GHz processor. The average times per sample are shown in Table 6. We can see that the processing time of coordinate mapping varies with the normalization method. The linear normalization (LN) is very fast. NLN is more expensive than other 1D methods, and LDPI is more expensive than other pseudo 2D methods. This is because NLN and LDPI involve line density computation. Nevertheless, all these normalization methods are not computationally expensive. Normalized image generation for pseudo 2D normalization methods is very time consuming because it involves quadrilateral decomposition. The processing time of binary normalized image includes a smoothing time of about 0.3ms.

The CPU time of feature extraction is almost independent of normalization method. It has two parts: directional decomposition and blurring. The average CPU times of three direction features are shown in Table 7. The processing time of blurring depends on the sparsity of direction planes (zero pixels are not considered). The direction planes of chaincode feature are sparse, and those of gra-

Table 6: Average CPU times (ms) of normalization on CASIA database. Binary normalized image involves smoothing.

	coordinate	binary	grayscale
LN	0.002	0.318	0.133
NLN	0.115	0.331	0.143
MN	0.017	0.321	0.126
BMN	0.024	0.332	0.135
MCBA	0.032	0.336	0.137
LDPI	0.266	1.512	1.282
P2DMN	0.143	1.514	1.236
P2DBMN	0.147	1.536	1.274
P2DCBA	0.156	1.542	1.261

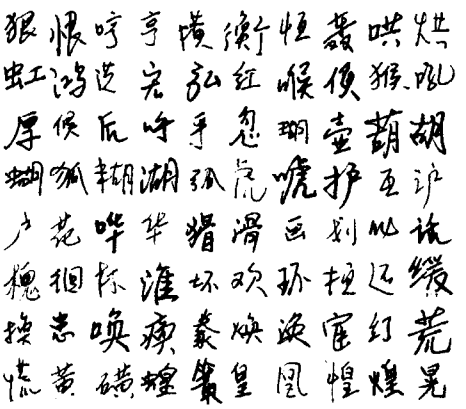


Figure 13: Samples of unconstrained handwritten characters.

dient feature are densest. The directional decomposition of NCCF is most time-consuming because it involves line segment decomposition, and gradient direction decomposition is more expensive than chaincode decomposition. Overall, NCCF is still the most computationally efficient because it saves the time of normalized image generation. The average CPU time of NCCF ranges from 1.21ms to 1.47ms, covering coordinate mapping and feature extraction. For reference, the average classification time of MQDF (with cluster-based candidate selection) for 3,755 classes is 3.63ms, which can be largely reduced by fine implementation, however.

Table 7: Average CPU times (ms) of feature extraction on CASIA database.

	direction	blurring
chn	0.121	0.439
nccf	0.458	0.752
grd-g	0.329	1.276

4.3 Performance on unconstrained handwriting

The error rates of the best methods on handprinted characters, say, 0.64% on ETL9B database and 1.52% on CASIA database, are fairly high. To test the performance on unconstrained handwritten characters, we use the classifiers trained with the training set of CASIA database to classify the samples in a small data set, which contains 20 samples for each of 3,755 classes, written by 20 writers without constraint. Some samples of the unconstrained set are shown in Fig. 13.

On the unconstrained set, we evaluate various normalization methods with two good features: NCCF and gradient direction feature. The error rates are listed in Table 8. We can see that as for handprinted characters, the pseudo 2D normalization methods yield lower error rates than their 1D counterparts on unconstrained characters as well. However, line-density-based nonlinear normalization methods, NLN and LDPI, are evidently inferior to the centroid-based methods. The P2DBMN method performs best, and the P2DCBA method is competitive. Comparing the two types of direction features, the gradient features outperforms NCCF with 1D normalization methods, but with pseudo 2D normalization, NCCF is superior. Overall, the error rates on unconstrained characters are still very high.

Table 8: Error rates (%) on unconstrained handwritten Chinese characters.

	Euclidean		MQDF	
	nccf	grdg	nccf	grd-g
LN	36.04	35.44	21.80	21.46
NLN	26.61	27.04	16.23	16.96
MN	25.63	25.18	16.40	16.09
BMN	25.15	24.53	15.91	15.64
MCBA	25.53	25.06	15.76	15.66
LDPI	24.93	25.60	15.12	15.73
P2DMN	23.48	24.38	14.84	15.37
P2DBMN	**22.75**	**23.35**	**14.27**	**14.64**
P2DCBA	23.80	23.78	14.73	14.79

Fig. 14 shows some misclassified samples by MQDF with the best normalization-feature combination (P2DBMN and NCCF). Most of the misclassified characters are similar in shape to the assigned class: some are inherently similar (top three rows in Fig. 14), and some others are similar due to cursive writing (fourth and fifth rows). Yet the characters

of the bottom row are not highly similar to their assigned classes.

Figure 14: Misclassified samples (each shown original image and normalized image) of unconstrained handwritten characters.

5 Concluding Remarks

We compared various shape normalization and selected feature extraction methods in offline handwritten Chinese character recognition. For direction feature extraction, our results show that 12-direction and 16-direction features outperform 8-direction feature, and the 12-direction feature has better tradeoff between accuracy and complexity. The comparison of normalization methods shows that pseudo 2D normalization methods outperform their 1D counterparts. On handprinted characters, a line-density-based method LDPI and a centroid-based method P2DBMN perform best. Comparing three types of direction features, the NCCF and the gradient feature outperform the chaincode feature, compete when 1D normalization is used, but with pseudo 2D normalization, the NCCF outperforms the gradient feature. Overall, the best normalization-feature combinations are LDPI and P2DBMN with NCCF.

We also tested the performance on a small set of unconstrained handwritten characters, with classifiers trained with handprinted samples. The error rates on unconstrained characters are very high, but the comparison of normalization and feature extraction methods reveals new insights. Though pseudo 2D normalization methods again outperform their 1D counterparts, the line-density-based LDPI method is inferior to the centroid-based methods, and the P2DBMN method performs best. The best normalization-feature combination on unconstrained characters is P2DBMN with NCCF.

Training classifiers with unconstrained handwritten samples will be able to improve the accuracy of unconstrained character recognition. To collect large database of unconstrained characters is an urgent task in the near future. To reduce the error rate to a fairly low level (say, 2% on isolated characters), however, simply using the current normalization and feature extraction methods and training current classifiers with larger sample set will not suffice. The methods of shape normalization, feature extraction, and classifier design need to be reconsidered for better recognizing cursively written characters. Training classifiers discriminatively can improve the accuracy of both handprinted and unconstrained character recognition.

Acknowledgements

This work is supported in part by the Hundred Talents Program of Chinese Academy of Sciences and the Natural Science Foundation of China (grant no.60543004).

References

[1] R. Casey and G. Nagy, Recognition of printed Chinese Characters, *IEEE Trans. Electronic Computers* **15(1)** (1966) 91-101.

[2] W. Stalling, Approaches to Chinese character recognition, *Pattern Recognition* **8** (1976) 87-98.

[3] S. Mori, K. Yamamoto, and M. Yasuda, Research on machine recognition of handprinted characters, *IEEE Trans. Pattern Anal. Mach. Intell.* **6(4)** (1984) 386-405.

[4] T.H. Hildebrandt and W. Liu, Optical recognition of Chinese characters: advances since 1980, *Pattern Recognition* **26(2)** (1993) 205-225.

[5] M. Umeda, Advances in recognition methods for handwritten Kanji characters, *IEICE Trans. Information and Systems* **E29(5)** (1996) 401-410.

[6] C.-L. Liu, S. Jaeger, and M. Nakagawa, Online recognition of Chinese characters: the state-of-the-art, *IEEE Trans. Pattern Anal. Mach. Intell.* **26(2)** (2004) 198-213.

[7] K. Fukunaga, *Introduction to Statistical Pattern Recognition*, 2nd edition (Academic Press, 1990).

[8] R.O. Duda, P.E. Hart, and D.G. Stork, *Pattern Classification*, 2nd edition (Wiley Interscience, 2001).

[9] A.K. Jain, R.P.W. Duin, and J. Mao, Statistical pattern recognition: a review, *IEEE Trans. Pattern Anal. Mach. Intell.* **22(1)** (2000) 4-37.

[10] I.-J. Kim and J.H. Kim, Statistical character structure modeling and its application to handwritten Chinese character recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* **25(11)** (2003) 1422-1436.

[11] T. Iijima, H. Genchi, and K. Mori, A theoretical study of the pattern identification by matching method, in *Proc. First USA-JAPAN Computer Conference*, Oct. 1972, 42-48.

[12] M. Yasuda and H. Fujisawa, An improvement of correlation method for character recognition, *Trans. IEICE Japan* **J62-D(3)** (1979) 217-224 (in Japanese).

[13] Y. Yamashita, K. Higuchi, Y. Yamada, and Y. Haga, Classification of handprinted Kanji characters by the structured segment matching method, *Pattern Recognition Letters* **1** (1983) 475-479.

[14] H. Fujisawa and C.-L. Liu, Directional pattern matching for character recognition revisited, in *Proc. 7th Int'l Conf. on Document Analysis and Recognition*, Edinburgh, Scotland, 2003, 794-798.

[15] J. Tsukumo and H. Tanaka, Classification of handprinted Chinese characters using nonlinear normalization and correlation methods, in *Proc. 9th Int'l Conf. on Pattern Recognition*, Rome, 1988, 168-171.

[16] H. Yamada, K. Yamamoto, and T. Saito, A nonlinear normalization method for hanprinted Kanji character recognition—line density equalization, *Pattern Recognition* **23(9)** (1990) 1023-1029.

[17] F. Kimura, K. Takashina, S. Tsuruoka, and Y. Miyake, Modified quadratic discriminant functions and the application to Chinese character recognition, *IEEE Trans. Pattern Anal. Mach. Intell.* **9(1)** (1997) 149-153.

[18] F. Kimura, T. Wakabayashi, S. Tsuruoka, and Y. Miyake, Improvement of handwritten Japanese character recognition using weighted direction code histogram, *Pattern Recognition* **30(8)** (1997) 1329-1337.

[19] N. Kato, M. Abe, Y. Nemoto, A handwritten character recognition system using modified Mahalanobis distance, *Trans. IEICE Japan* **J79-D-II(1)** (1996) 45-52.

[20] R.G. Casey, Moment normalization of handprinted character, *IBM J. Res. Develop.* **14** (1970) 548-557.

[21] C.-L. Liu, H. Sako, and H. Fujisawa, Handwritten Chinese character recognition: alternatives to nonlinear normalization, in *Proc. 7th Int'l Conf. on Document Analysis and Recognition*, Edinburgh, Scotland, 2003, 524-528.

[22] C.-L. Liu and K. Marukawa, Global shape normalization for handwritten Chinese character recognition: a new method, in *Proc. 9th Int'l Workshop on Frontiers of Handwriting Recognition*, Tokyo, Japan, 2004, 300-305.

[23] T. Horiuchi, R. Haruki, H. Yamada, and K. Yamamoto, Two-dimensional extension of nonlinear normalization method using line density for character recognition, in *Proc. 4th Int'l Conf. on Document Analysis and Recognition*, Ulm, Germany, 1997, 511-514.

[24] C.-L. Liu and K. Marukawa, Pseudo two-dimensional shape normalization methods for handwritten Chinese character recognition, *Pattern Recognition* **38(12)** (2005) 2242-2255.

[25] M. Hamanaka, K. Yamada, and J. Tsukumo, Normalization-cooperated feature extraction method for handprinted Kanji character recognition, in *Proc. 3rd Int'l Workshop on Frontiers of Handwriting Recognition*, Buffalo, NY, 1993, 343-348.

[26] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, Handwritten digit recognition: benchmarking of state-of-the-art techniques, *Pattern Recognition* **36(10)** (2003) 2271-2285.

[27] A. Kawamura, K. Yura, T. Hayama, Y. Hidai, T. Minamikawa, A. Tanaka, and S. Masuda, On-line recognition of freely handwritten Japanese characters using directional feature densities, in *Proc. 11th Int'l Conf. on Pattern Recognition*, The Hague, 1992, Vol.2, 183-186.

[28] G. Srikantan, S.W. Lam, and S.N. Srihari, Gradient-based contour encoder for character recognition, *Pattern Recognition* **29(7)** (1996) 1147-1160.

[29] C.-L. Liu, K. Nakashima, H. Sako, and H. Fujisawa, Handwritten digit recognition: investigation of normalization and feature extraction techniques, *Pattern Recognition* **37(2)** (2004) 265-279.

[30] N. Hagita, S. Naito, and I. Masuda, Handprinted Chinese characters recognition by peripheral direction contributivity feature, *Trans. IEICE Japan* **J66-D(10)** (1983) 1185-1192 (in Japanese).

[31] M. Yasuda, K. Yamamoto, H. Yamada, and T. Saito, An improved correlation method for handprinted Chinese character recognition in a reciprocal feature field, *Trans. IEICE Japan* **J68-D(3)** (1985) 353-360.

[32] L.-N. Teow and K.-F. Loe, Robust vision-based features and classification schemes for off-line handwritten digit recognition, *Pattern Recognition* **35(11)** (2002) 2355-2364.

[33] M. Shi, Y. Fujisawa, T. Wakabayashi, and F. Kimura, Handwritten numeral recognition using gradient and curvature of gray scale image, *Pattern Recognition* **35(10)** (2002) 2051-2059.

[34] X. Wang, X. Ding, and C. Liu, Gabor filter-base feature extraction for character recognition, *Pattern Recognition* **38(3)** (2005) 369-379.

[35] C.-L. Liu, M. Koga, and H. Fujisawa, Gabor feature extraction for character recognition: comparison with gradient feature, in *Proc. 8th Int'l Conf. on Document Analysis and Recognition*, Seoul, Korea, 2005, 121-125.

[36] C.-L. Liu, M. Koga, and H. Sako, H. Fujisawa, Aspect ratio adaptive normalization for handwritten character recognition, in *Advances in Multimodal Interfaces—ICMI2000*, T. Tan, Y. Shi, W. Gao (Eds.), LNCS Vol. 1948, Springer, 2000, 418-425.

[37] C.-L. Liu, High accuracy handwritten Chinese character recognition using quadratic classifiers with discriminative feature extraction, in *Proc. 18th Int'l Conf. on Pattern Recognition*, Hong Kong, 2006, Vol.2, 942-945.

[38] H. Liu and X. Ding, Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes, in *Proc. 8th Int'l Conf. on Document Analysis and Recognition*, Seoul, Korea, 2005, 19-23.

[39] C.-L. Liu, Y-J. Liu, and R-W. Dai, Preprocessing and statistical/structural feature extraction for handwritten numeral recognition, in *Progress of Handwriting Recognition*, A.C. Downton and S. Impedovo (Eds.), (World Scientific, 1997) 161-168.

[40] J. Guo, N. Sun, Y. Nemoto, M. Kimura, H. Echigo, and R. Sato, Recognition of handwritten characters using pattern transformation method with cosine function, *Trans. IEICE Japan* **J76-D-II(4)** (1993) 835-842 (in Japanese).

[41] T. Wakabayashi, S. Tsuruoka, F. Kimura, and Y. Miyake, On the size and variable transformation of feature vector for handwritten character recognition, *Trans. IEICE Japan* **J76-D-II(12)** (1993) 2495-2503 (in Japanese).

[42] R.V.D. Heiden, and F.C.A. Gren, The Box-Cox metric for nearest neighbor classification improvement, *Pattern Recognition* **30(2)** (1997) 273-279.

[43] N. Kato, M. Suzuki, S. Omachi, H. Aso, and Y. Nemoto, A handwritten character recognition system using directional element feature and asymmetric Mahalanobis distance, *IEEE Trans. Pattern Anal. Mach. Intell.* **21(3)** (1999) 258-262.

How to Deal with Uncertainty and Variability: Experience and Solutions

Hiromichi Fujisawa
Central Research Laboratory, Hitachi, Ltd.
Tokyo, Japan 185-8601
hiromichi.fujisawa.sb-at-hitachi.com

Abstract

Uncertainty and variability is one of the most important concepts sitting at the center of pattern recognition. It is especially so when patterns to be recognized are complex in nature and not controlled under any artificial constraints. Handwritten postal address recognition is one of such examples. This paper presents five principles of dealing with uncertainty and variability, and how to decompose the complex recognition task into manageable pieces of subtasks. When applicable, block diagrams will be shown to clarify the structure of various recognition components. This paper also presents how to implement those principles into real recognition engines. It will show that high accuracy and high robustness of a recognition system, which relate to uncertainty and robustness, respectively, can be attained only when comprehensive approaches are taken.

1 Introduction

Historically, character recognition was the first concrete subfield of pattern recognition, which was actually studied before 'pattern recognition' came into existence. For a long time, OCRs (Optical Character Readers) were reading machine-printed characters and handwritten characters in fixed, separate positions on specially designed forms. Today, OCRs are much more advanced than those days, and are capable of reading many varieties of documents including book pages, business forms, bank checks, mail letter surfaces, and so on.

The capabilities of OCRs have expanded in multiple dimensions. The first dimension is the script. Having started with Arabic numeral characters, the recognizable scripts have been expanded to Roman alphabets, Japanese syllabic letters, Kanji (Japanese Chinese characters), Chinese characters, Hangul characters, Indian scripts, Arabic scripts, and so on. The second dimension is the style of printing and writing. In the case of machine-printed characters, many different font faces in different sizes are now recognizable. As for handwritten characters, writers were first requested to 'hand-print' the numerals and upper-case alphabets in a restricted way. Today, advanced OCRs can recognize freely written scripts, however, in a limited context such as bank checks and mail pieces. These applications allow OCR algorithms to utilize contextual information to heighten the recognition accuracy to the industry strength.

Through these historical technology developments, *uncertainty and variability* have been the key issues in technical challenges. Uncertainty refers to the state of something we are not sure about. But, in the context of this paper, it is the state of a recognition engine or a recognition component being not certain about the input or about the output. Therefore, the issue consists of two parts. One is how to deal with the uncertainty in the input, and the other is how to reduce the uncertainty in the output, in other words, how to heighten the accuracy and reliability. These became more prominent when the recognition tasks became more complex.

Variability is another key issue as everybody agrees with. It is about variations, which exist in inputs. If there existed no variations, there would have been no problems in pattern recognition. Usually, with the more variations, the more recognition performance degrades, where we try to minimize the degradation. Therefore, the issue is about robustness, and it is about how to make the system more robust.

As a sequel, modern complex applications require the researchers and engineers to think more about uncertainty and variability. In response to this, this paper will present five design principles and implementation methodologies that lead to higher accuracy and higher robustness. The presentation includes examples from our experiences in developing a Japanese postal address recognition system. The technical challenges are mostly in handwritten address recognition.

The following chapters are organized as follows. In Chapter 2, uncertainty and variability in pattern recognition are discussed in some depth, and it will be shown they relate to accuracy and robustness, respectively. Chapter 3 will describe the problems in Japanese postal address recognition as an example of a complex recognition task, expecting the following chapters may be more understandable. Then, Chapter 4 will present five design principles to tackle the uncertainty and variability problems, followed by Chapter 5 presenting implementation methodologies.

2 Uncertainty and Variability

Uncertainty exists in the input and in the output in general. As mentioned earlier, a complex recognition task needs to be decomposed into smaller pieces of manageable tasks. This means that the whole recognition engine will consist of many components each of which solves smaller scale pattern recognition problems. It further means that, because each component has an

input and an output each of which holds uncertainty, composition of such components requires a careful design. It is a question of how to combine recognition components with each other when their inputs and outputs are uncertain. If they were simply connected in a cascade manner, the accuracy of the final output from 30 such components would be 74 percent even when each component's accuracy was 0.99. Usually, the number of decisions to be made is well over 30 to recognize one postal address, and to raise every component's accuracy above 0.99 is extremely difficult. Here comes the necessity of good design principles, which will be discussed in Chapter 4.

Variability, on the other hand, is only in the input. However, the variability in uncontrolled inputs such as in postal addresses is nonlinear, discontinuous, structural, and unpredictable. The problems of nonlinear, discontinuous, and structural variations require more than a single parametric method to solve them. Usually, it requires additional logics (or algorithms) and makes the whole recognition engine more complex. How to combine such new logics to other existing parts is again a technical issue. This is especially true when the variability in concern is not about a single character, but about a layout of the character lines of interest, and about their surrounding objects around the character lines of interest. Such spatial variations of printed/handwritten objects are difficult to handle. For example, Japanese mail surfaces may have many unrelated, intervening objects such as advertisement. Or, recipient addresses can be written in many different orientations. Finding a recipient address is generally difficult.

Unpredictability of variations, which is a kind of uncertainty as well, has a methodological implication. In other words, it means that capturing every (or most of the) variation(s) at a single step is almost impossible, and the same is true for coping with every problem stemming from a variation. Questions here are how to cycle the developmental steps, how to organize sample datasets, and how to prioritize the plurality of problems. These questions will be discussed in Chapter 5 with solutions.

Another discussion about the uncertainty and variability is their relationship to the recognition performance. Generally speaking, recognition performance is considered to have the following three facets:

- Accuracy
- Robustness
- Efficiency

Accuracy or reliability is a reflection of the certainty of the recognition output. Struggling with the uncertainty leads to a higher accuracy. Accuracy is usually measured as Nc/Na, where $Na = N - Nr$, Nc is the number of correctly recognized samples, Nr the number of rejected samples, and N the total number of samples. 'Read rate' is sometimes used as a measure of recognition performance as well. It is defined as Na/N. Error rate, Ne, is measured as either $Ne = (N - Nc - Nr)/N$ or $Ne = (Na - Nc)/Na$

Now, robustness is a concept, which is not much discussed in the context of recognition performance. It has not been quantitatively defined yet, but we have been thinking that robustness needs to be one of the performance facets. Robustness can be defined as a quantity that is inversely proportional to how much the read rate (or accuracy) degrades against variations. In this paper, instead of defining a single quantity, we try to see it as a read rate curve against variations. Figure 1 shows an example of such read rate profiles of a postal address recognition engine. The horizontal axis shows sample dataset numbers, where the sample datasets have been reordered according to each of these read rates, and the vertical axis show the read rates.

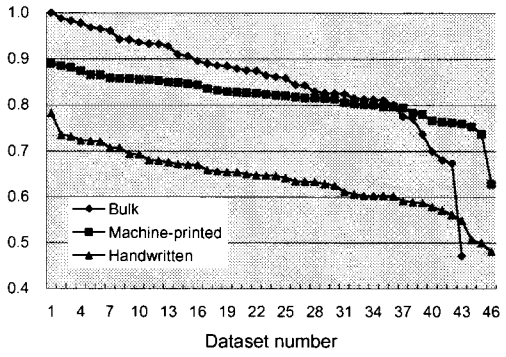


Figure 1: Read rate profiles.

In this setting, mailpiece image samples are captured from a mail sorting machine in separate multiple sessions, forming multiple datasets. Then, due to the varieties of capturing times, each dataset has different sample mixtures. Some datasets are easier to read, and others are more difficult than those, for example. But, these reflect the operational modes of the post offices.

We may see in this setting that the more flat the curve is, the more robust the recognition engine is. At this moment, we do not have a theoretical background for seeing the robustness characteristics in this way. However, it is important to note that each dot shows a read rate, which an operator (a customer) of the mail sorting machine could see on the control panel. Therefore, clearly, flattening of these curves is a reasonable target for the developers.

The third facet, efficiency, as a recognition performance, is also an important characteristic, representing recognition throughput per unit computation cost. Usually, the throughput performance is given as a specification. If the latency (recognition time) is not the concern, it is a matter of the amount of hardware resources and then, the cost of the system. We can get higher throughput by adding more hardware in parallel. But, in the case of mail sorting machines, the latency is important and therefore, it is not possible to depend on a bigger hardware only. The recognition algorithms are always constrained by the maximum latency time. In a complex pattern recognition problem, recognition is an optimum search in a huge tree of search space as discussed later so that the search control parameters need to be carefully optimized.

3 Japanese Postal Address Recognition

3.1 Machine for Reading Postal Addresses

Postal address recognition as an application has contributed to the technology advancement of handwritten character recognition. It is technically 'rich' and semantically 'rich.' In other words, it has presented many technical challenges, but the domain is limited to postal addresses, which is semantically well defined. The latter notion is important in that linguistic approaches can be introduced together with address knowledge to reduce the search spaces for the optimal interpretation.

The task of a mail sorting machine (Figure 2) is to pick up mail pieces, one by one, from the feeder deck in front, transport the mail pieces through an image scanner to one of the sort bins, recognize the recipient address in the scanned image, and spray barcodes representing the recognized address code. Recognition should be ended before the corresponding mail piece reaches the barcode printer and the switches that determine the path to the selected bin. A bigger version of the machine has about 400 sort bins. In the case of the machine shown in Figure 2, mail pieces run at the speed of 3.4m/s, and maximum latency time for recognition is 3.7s. The throughput is 30,000 to 50,000 mail pieces per hour depending on the operational mode. Approximately, average recognition time of 1.5s and 21 computers can attain the maximum throughput.

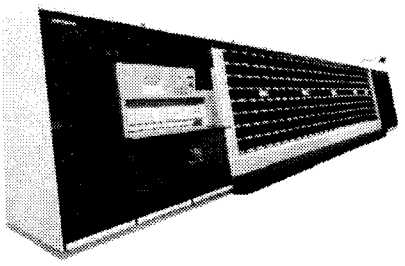


Figure 2: Mail Sorting Machine (Hitachi).

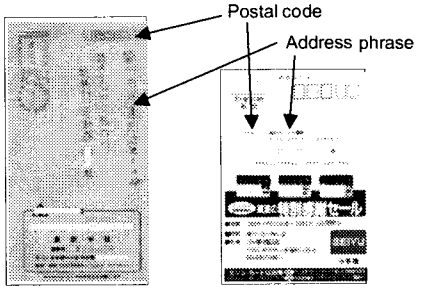


Figure 3: Mail sample images.
(Courtesy of the recipients)

3.2 Recognition task

The task of postal address recognition is to identify the character lines of a postal code and an address phrase, recognize, and interpret them. The output, i.e., the recognition result, is an address code that designates a delivery point out of 40 million address points in Japan. The address code is a variable length digit code, which sometimes includes alphabetic information also. Its first seven digits are equivalent to the postal code, which is written or printed in a separate field. With a few exceptions, a Kanji address phrase corresponds to this postal code. Therefore, a recognition engine can utilize this redundant information for higher recognition rate. The remaining digits are three field digits representing an address number. Sample images of Japanese mail pieces are shown in Figure 3.

The recognition task can be decomposed into four steps: (1) image scanning and preprocessing, (2) layout analysis to extract a postal code region and a recipient address block, (3) character string recognition and interpretation, and (4) post processing to verify the recognition result and do some extra recognition tasks. A block diagram for these recognition components are shown in Figure 4.

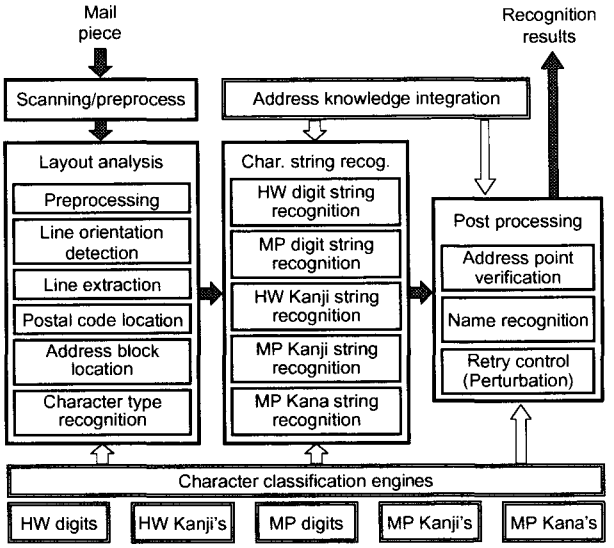


Figure 4: Recognition components.

As shown in the diagram, each step consists of subcomponents for solving more focused pattern recognition subproblems. Each subcomponent does some kind of decision making. Therefore, uncertainty and variability are carried over to these levels. For example, in layout analysis, the accuracy of line orientation determination from four possible cases of 0°, 90°, 180° and 270° is not 100 percent. This is true for other subcomponents as well, and care is necessary when they are connected in series. Concretely speaking, each subcomponent outputs multiple choices for the next step, when the decision is uncertain.

There are subcomponents that are attached in parallel. It is sometimes necessary to classify an input into one of the cases anticipated and to process these differently. For example, problems in recognizing handwritten character strings and machine-printed character strings are different so that we have decided to develop separate recognition engines though they are similar in their architectures. Character type recognition for this adaptive treatment is also prone to err; therefore similar means are applied. Namely, when the decision is uncertain, multiple hypotheses are generated here again.

3.3 Technical Difficulties

Technical difficulties can be understood in advance in only an abstract way. In order to improve recognition performance, we need to find difficulties in relation to the target recognition engine. This section describes *performance influencing factors* (PIFs) in general and Chapter 5 will describe how to deal with concrete difficulties.

We call such factors that influence recognition performance PIFs. These are variations with many aspects as listed in Table 1. Unfortunately, each of these factors usually requires adding additional logics to the baseline algorithms to ensure proper recognition. Some require new methods as well. Some are too costly to achieve. It is necessary to prioritize the problems in order of significance.

Table 1: Performance Influencing Factors.

Components	Performance Influencing Factors
Optical scanning	Low quality printing
	Address written in a dark ink on a dark colored envelope
	Reflecting plastic address window
Binarization and image coding	Low contrast image
	Non-uniform contrast
	Faint/dark printing
	Complex background texture
Address block location	Interfering background such as ads
	Character size variations
	Printing/writing orientations
	Mixture of different orientations
	Irregular address block location
	Interfering sender's address block
	Window shadow
	Non-square handwritten address block
	Irrelevant numbers in address block
Address line segmentation	Character size variation
	Overlapping handwritten character lines
	Touching handwritten character lines
	Mixture of different writing orientations
	Shadow of plastic window
	Skewed letter images
Character segmentation	Touching characters
	Broken characters
	Non-uniform handwritten character size
	Zero-character-spacing
	Underlines
Character recognition	Multiple scripts
	Low quality character image
	Writing style variations
	Peculiar writing variations
	Writing instrument variations
	Extremely small characters
	Mixture of writing and printing
Character string recognition / Address phrase interpretation	Address expression variations
	Abbreviated address expression
	Incomplete address expression
	Address hidden by window
	Address hidden by cancellation stamp
	Extra punctuation
	Wrong address
	Obsolete address

4 Five Design Principles

4.1 Hypothesis-Driven Principle

Resolving uncertainty in the input needs a hypothesis to start with. For example, to locate an address block, we need to have line candidates. To do so, we need to know the line orientation. This requires us to have line candidates. Here is a dilemma. To break the dilemma, we can generate hypotheses first, and process the input accordingly, and then test the result. In this example, we hypothesize horizontal orientation and extract horizontal line candidates; and then try vertical orientation. Then, we can evaluate (or compare) the two results to see which hypothesis was right. At the same time, we can get the result. We call this approach *Hypothesis-Driven* approach.

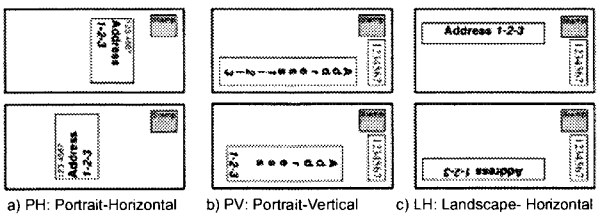
In reality, address block location is more complex. In the case of Japanese letter envelopes, there are six layout types for each of handwritten and machine-printed versions (Figure 5). In total, there are 12 types. To cope with this problem, the method we developed hypothesizes the following six types, and evaluates the resulting block candidates based on a Bayesian approach [1]. Note that character orientation is determined later by applying character recognition.

- P-PH: Printed portrait horizontal
- P-PV: Printed portrait vertical
- P-LH: Printed landscape horizontal
- H-PH: Handwritten portrait horizontal
- H-PV: Handwritten portrait vertical
- H-LH: Handwritten landscape horizontal

The evaluation of the hypotheses is made based on the confidence value, which is defined as the *a posteriori* probability of the corresponding hypothesis after observing evidence as in Eq. (1). The evidence for each hypothesis, H_k, is obtained as a feature vector $\boldsymbol{e}_k$. The features taken are (1) the averages of the height and width of character lines, (2) the variance of the height and width of character lines, (3) the area of an address block candidate, and (4) the position of the candidate. L in Eq. (1) is a likelihood ratio of hypothesis H_k to null hypothesis $\overline{H}_k$ and is computed as in Eq. (2) assuming the statistical independence in the features.

$$P(H_k \mid \boldsymbol{e}_k) = \frac{\frac{P(H_k)}{P(\overline{H}_k)} L(\boldsymbol{e}_k \mid H_k)}{1 + \frac{P(H_k)}{P(\overline{H}_k)} L(\boldsymbol{e}_k \mid H_k)} \tag{1}$$

$$L(\boldsymbol{e}_k \mid H_k) = \frac{P(\boldsymbol{e}_k \mid H_k)}{P(\boldsymbol{e}_k \mid \overline{H}_k)} \cong \Pi_{i=1}^{n} \frac{P(e_{ki} \mid H_k)}{P(e_{ki} \mid \overline{H}_k)} \tag{2}$$



a) PH: Portrait-Horizontal b) PV: Portrait-Vertical c) LH: Landscape- Horizontal

Figure 5: Six layout types of the address block.

A general scheme for this approach can be diagramed as in Figure 6. Hypothesis generation in general produces hypotheses by analyzing the input dynamically. However, in the case of address block location, the hypotheses are predetermined as described above so that not much is done. Each process, P_1 to P_n, generates (multiple) candidates of address blocks together with the corresponding feature vectors e_k. Hypothesis testing (the evaluation step) computes the aforementioned confidence values and orders the candidates according to those values.

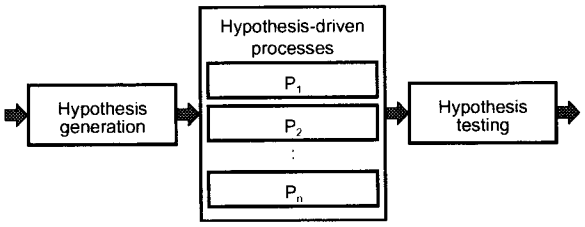


Figure 6: Hypothesis driven approach.

An illustration of the address block location process is given in Figure 7 for a simple case. First, based on the size of the input image, the type of the letter is determined as a postcard rather than an envelope in this example. Then, connected component analysis is applied to the binary image, and line candidates are extracted based on both the horizontal and vertical hypotheses. In the case of a postcard, existence of a message area is hypothesized, and if true, the message area, which is the lower half, is discarded. The upper half is then analyzed for an address block.

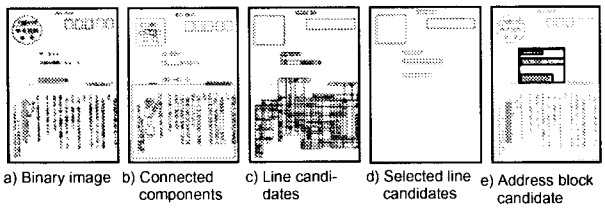


Figure 7: Address block location process.

4.2 Deferred Decision/Multiple Hypotheses Principle

The principle is: "*Don't decide when it is uncertain. Leave it to other experts. Give options when it is uncertain.*"

Usually, pattern recognition consists of sequential steps of sublevel pattern recognition problems, and it can be solved by using dynamic programming in the case of well defined problems [2]. However, in the case of Japanese postal address recognition, such subproblems are as many and varied as listed below, and require a somewhat more heuristic approach.

- Line orientation detection
- Character size (large/small) detection
- Character line formation and extraction
- Address block identification
- Character type (machine-printed/handwritten) identification
- Script (Kanji/Kana) identification
- Character orientation identification
- Character segmentation
- Character classification
- Word recognition
- Phrase interpretation
- Address number recognition
- Building/room number recognition
- Recipient name recognition
- Final decision making (accept/reject/retry)

As discussed before, these steps cannot be connected in a cascade manner transferring only the top choice of each step to the next. If the total number of decision makings were 30, the accuracy of 0.99 at each step could only produce the total accuracy of 0.74. However, though attaining the accuracy of 0.99 is extremely difficult, it is possible to attain the cumulative accuracy of 0.99 or higher if we take multiple candidates. Therefore, the Multiple Hypothesis Principle is to let the system propagate multiple hypotheses from component to component. This creates however a hierarchical tree of hypotheses to search for the optimum solution.

So, the next question is how to do the search. Among known methods of Hill Climbing Search, Best First Search, Beam Search and others [3], we use basically Hill Climbing Search (with backtracking), by which we can reach an 'optimum solution' in a shortest time. When the 'optimum solution' is rejected at the final decision step where it is verified against an address directory, the second best solution is searched. Then, this process is repeated while the recognition time affords continuation. An additional variation to Hill Climbing Search taken here is Beam Search at the later stages to boost the recognition accuracy. To use Beam search at the earlier stages is too costly.

Another important thing to note here is to carefully control the number of branches (candidates) so that the search time will remain within reasonable time bounds. Instead of setting a uniform fixed number for the entire search space, we make branching decisions adaptively by comparing the scores of branches with absolute and relative thresholds. Actually, as a special case, if the scores at a given stage are all below an absolute threshold, that node is abandoned and backtracked.

The effectiveness of this approach is shown in Figure 8, where AB stands for address block, MP machine-printed, and HW handwritten.

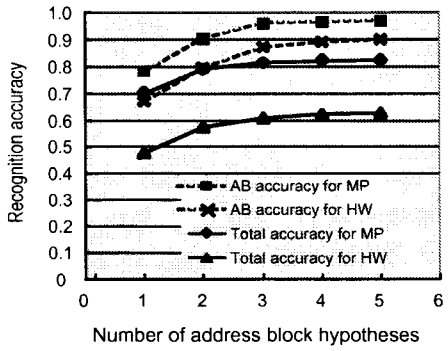


Figure 8: Relationship between the number of address block candidates and the read rate.

A general scheme of this approach is illustrated in Figure 9. The switches are controlled according to the optimum search strategy.

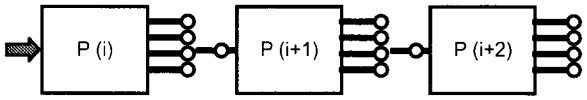


Figure 9: Multiple hypotheses approach.

4.3 Information Integration Principle

There are three types of *Information Integration* approaches known today.

The first type is integration of processes to solve several problems altogether, as a single bigger problem. It is to say "*Solve problems by experts from different fields as a team.*" The second type is combination of processes to solve one problem. It is saying "*Solve a problem by multiple same-field experts as a team.*" This last type is seeking more information to raise the certainty in the result. It is equivalent to saying "*Solve a problem by seeking more information for higher certainty.*"

4.3.1 Process Integration

Historically, segmentation-recognition integration is one of the instances of this approach. This technique is used to solve the problem of touching handwritten digit recognition [4]. The segmentation component performs hypothetical presegmentation and generates a network of segmentation hypotheses. Large connected components are considered touching digits, and each of them is separated into two at the hypothesized points. Uncertainty in touching itself and a touching position is handled in terms of multiple hypotheses.

Recognition-interpretation integration is anther two-process integration. If segmentation is reliable, character recognition and linguistic interpretation can be integrated into one problem-solving scheme. Handwritten Kanji recognition generates a lattice, which represents character classification candidates, and the lattice is then transformed into a finite state automaton [5]. The automaton can be searched for valid sequences of characters, i.e., words, effectively.

Most frequently used process integration is a three-process integration of segmentation, character classification and linguistic interpretation. The approach called lexicon-driven recognition, which belongs to this class of techniques, has been successfully used for handwritten check amount recognition and handwritten postal address recognition [6] [7] [8] [9].

In this approach, linguistic constraints, i.e., linguistic knowledge, are used to solve the segmentation uncertainty and the character classification uncertainty at the same time. Linguistic constraints limit the search space, and guide the search in terms of a language model such as a TRIE structure (Figure 10) or a Recursive Transition Network representing a context free grammar.

Then, the interpretation of the input is a search for a path in the segmentation hypothesis network that best matches one of the paths in the tree (TRIE) as shown in Figure 10, and vice versa. As the name 'lexicon-driven' suggests, a language model can guide the search. Hypothesizing a character class at a time, the edges from the current node of the segmentation-hypothesis network are evaluated against the hypothesized character class. If the matched edges give a score greater than a threshold, this character class and the corresponding segmentation edge(s) are kept in the search tree, which is a working memory to control the search process. Beam Search can be used as a search control strategy.

In this way, instead of applying character classification to every presegmented pattern in the network, character matching is only done between the hypothesized character class and the patterns on the edges. This efficient search process is quite attractive when we note that there are more than 4,000 Kanji character classes.

A successful application to Japanese handwritten address phrase recognition is reported in [9]. The method we have developed recognizes Kanji address phrases and generates a 7-digit code, which corresponds to a postal code. In an experiment on 3,589 actual mail pieces and a lexicon containing 111,349 address phrases, the recognition accuracy was 83.7% when error was 1.1%, by using Beam Search. The recognition time was about 100ms by using Pentium III/600MHz. This method has capabilities of noise elimination, touching character detection, touching character splitting, and partial matching.

This method is also attractive in that it is capable of searching only meaningful patterns (phrases) by ignoring noisy portions and irrelevant character strings. It is similar to 'word spotting' in speech recognition. Figure 11 shows a case where the second and third characters are overlaid with a cancellation stamp. In

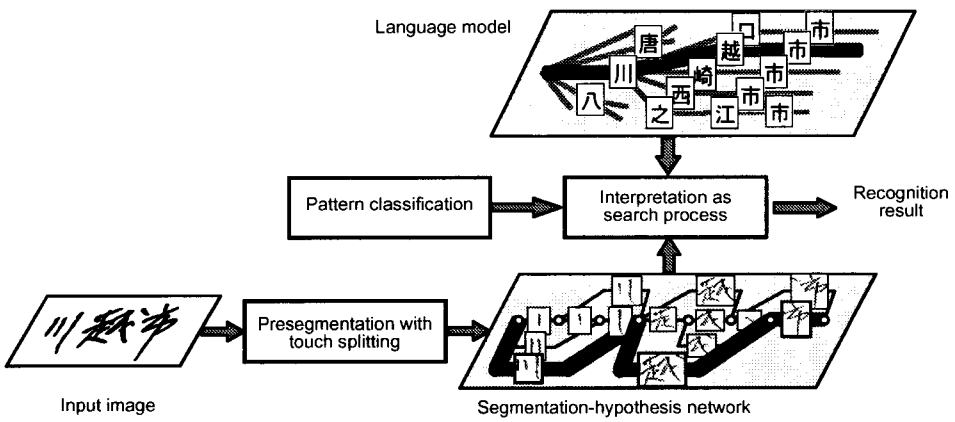


Figure 10: Segmentation-Recognition-Interpretation Integration

this case, however, the first three characters represent a prefecture name, which is redundant, fortunately. After the first five presegmented patterns are rejected in the search process, the matching process started with the fourth character (fifth positions from the left) correctly recognizes the city name (4th to 6th characters) and town name (7th to 8th characters). Of course, the language model should have been installed with the shortened phrases (without a prefecture name) in addition to the fully expanded phrases (with a prefecture name). In a reality, it is important to gather such variations before operation, and to include them into the model.

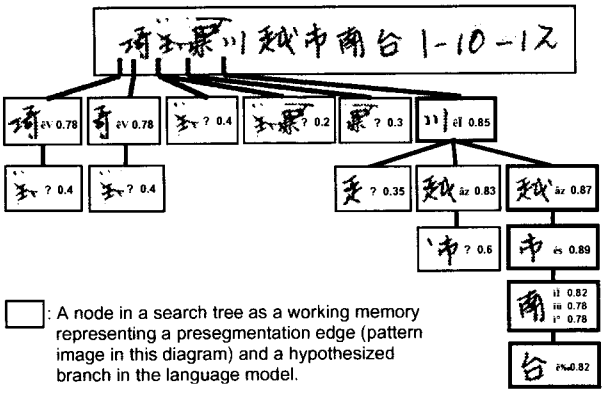


Figure 11: Adaptive search for an address phrase.

Regarding the language model, the above example uses a TRIE structure, but it has a limitation. When there are word alternatives, word omissions, and subphrase variations, a TRIE structure needs to have subtree repetitions, which sometimes requires more memory than is affordable. In such a case, we can use a recursive transition network model [10].

4.3.2 Combination-Based Integration

Classifier combination is the second type of Information Integration. It is based on the idea of applying multiple independent classifiers to the same input, and then, combining the results somehow for better accuracy [11] [12] [13]. Expectation is that classifiers are complementary in their characteristics, and therefore, combining those results can boost the recognition accuracy. Differences in the classifiers can be any combination of recognition schemes (statistical approaches, structural approaches, neural network approaches, etc.), features (structural features, mathematical transformation, Gabor filter, directional feature, etc.), training samples, and so on.

Known methods for combination are either at the abstract level, the rank level, or the measurement level depending on the kind of information used [11]. Abstract level combination is to use the top candidate of each classifier output to make a final decision. Majority voting and Dempster-Shafer approaches are known for implementation. This type of combination is applicable to most of the existing classifier engines because it only uses the class labels in the output. Rank level combination uses the ranked lists of candidates from all classifiers to re-rank the results. The third kind, measurement level combination, is to use additionally some kind of measurements as the classifier output to produce a more reliable classification result. Normalized confidence values, which are computed from the recognition score of each classifier (e.g., distance, probability, etc.), are used to obtain a total score.

4.3.3 Corroboration-Based Integration

Corroboration is the process of finding additional evidence for higher certainty. Looking for more input information to obtain the same result. One good example is bank check recognition, where a legal amount and a courtesy amount are recognized and combined to get a more reliable result [14]. Another example is postal address recognition. Reading both Japanese 7-digit postal codes and Kanji address phrases, which are almost equivalent to each other, can heighten the read rate and accuracy.

Still another example in postal address recognition is to recognize a recipient name (company and/or person's name) to reduce the uncertainty and to identify an address point. When address number recognition gives multiple candidate address points due to some ambiguity, or when room number recognition fails, we can still use the partial recognition result to squeeze out address point candidates. Then, by consulting a directory, we can list up the candidate recipient names. By having these candidate names, the recognition engine can recognize a recipient name in the mail piece image. As a matter of fact, recipient name recognition is necessary in such a case where the recognized address point is a place that has more than one residence or more than one company. This case is not corroboration, however.

The effectiveness of recipient name recognition is shown in Figure 12 for corroboration cases.

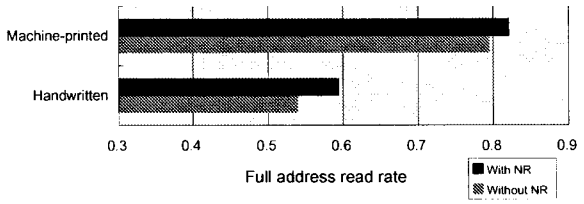


Figure 12: Effect of name recognition (NR)

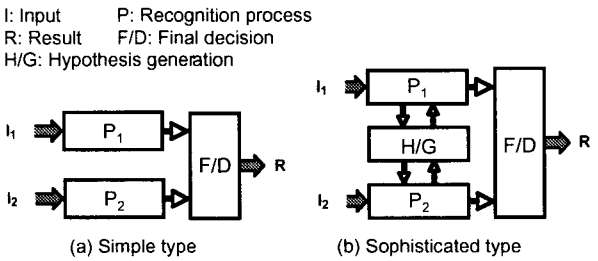


Figure 13: Corroboration schemes.

Corroboration-based information integration can be used in several ways (Figure 13). One is to reduce the error rate for higher reliability. Bank check recognition is of this kind. Another is to increase the read rate (accept rate) provided that the error rate does not

increase. This is the case for postal address recognition. In any case, the final choice of the two objectives is at the discretion of users, and depends on the optimization criteria given at a higher level.

4.4 Conflicting Solutions Principle

The conflicting solutions principle is to say, "*When a problem is difficult, try different approaches as well.*"

There are many image level problems in postal address recognition, which include problems of characters that touch each other, underlines that touch characters, window shadow noise, cancellation stamps covering address characters, and so on. All of these require special problem solving mechanisms. Conflicting Solutions Principle (which we originally named Multiple Solution Principle) is to provide more than one solution to solve the problem. Possibly, the solutions are complementary or drastically different from each other.

For example, for touching characters, there may be two different solutions. One is to attempt to separate a touching pattern [4]. The other is to design and train character classifiers so as to recognize a touching pattern as a whole; i.e., a holistic approach. Training of the classifier is simply applied against a dataset that additionally includes samples of frequently touching character pairs. Touching digit recognition can easily rely on this approach since the number of combinations is not so big when compared to the number of Kanji classes. The two solutions are then merged to give a more reliable recognition result.

The problem of window shadow noise and underlines (solid and dotted lines) are of interest to discuss here (see Figure 14 and Figure 15). These extra patterns sometimes interfere with recognition but at the same time they may help identify the location of address lines. Some of Japanese standard envelopes have preprinted underlines for address and recipient name fields. Therefore, they are a good clue to locate an address block. The same is true for window shadows if they form stable contours.

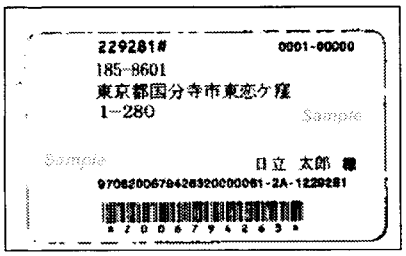


Figure 14: Window shadow noise.

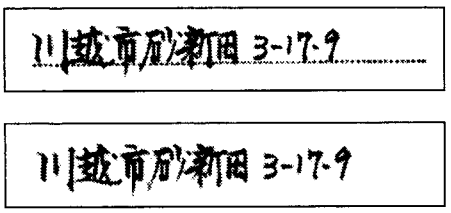


Figure 15: Dotted underscore.

So, the conflicting solutions here are on the one hand to attempt to detect such lines, and on the other to attempt to eliminate noisy lines and solid/dotted lines. Elimination of thin, broken, noisy components and recovering of a line from those are done by erosion (thinning) and dilation (thickening) processes, respectively. Separate line extraction algorithms extract stable solid/dotted lines.

These two different solutions implemented as recognition components generate two branches in the hypothesis tree, which was described in section 4.2. Then, based on the Hill Climbing strategy, one of the branches is followed, and if the final address recognition result stemming from this first branch is verified, recognition terminates. Otherwise, the second branch is followed. Generally speaking, instead of choosing one branch in this way, the results of these two branches can be merged as well.

4.5 Perturbation Principle

The perturbation principle is to say, "*When a problem is difficult, try to modify it slightly.*"

The perturbation principal applied to character recognition is not new actually. It has been used for a rather long time. In the 1980s, it was used in commercial OCRs in Japan when structural character recognition approaches were being used for handwritten digit recognition. If an input pattern could not be recognized due to topological differences, a slight change in images or parameters was induced for another cycle of recognition.

Recently, the perturbation approach has been studied more systematically [15]. Various transformations are applied to an input image to generate slightly 'perturbed' patterns, assuming that these new patterns still belong to the same class as the original one. The same recognition engine then recognizes these perturbed patterns. The recognition results are combined as in classifier combination. As for transformations, they can be morphological operations such as dilation and erosion, and/or geometrical transformations, such as rotation, slant (horizontal/vertical), perspective, and shrinking/expanding transformations.

The perturbation approach has been successfully applied in Japanese postal address recognition as well. We have applied a rotational transformation to a rejected input image, which is then given to the whole recognition process for a second cycle. In practice, to ensure bounds on recognition time, only one or two cycles of perturbation can be applied. However, we can obtain about 10 percentage points improvement on the average. It is interesting to note here that rejection often occurs in the middle of the recognition process, leaving more time to compute. When we did not force limitation to the recognition time, and applied several perturbation operations that include a rotational transformation, re-binarization, and reversing the decisions about orientation and character type, we could show that 53 percent of the rejected handwritten addresses could be correctly recognized with about 12 percent errors.

To widely use this approach, there are two practical issues; i.e., additional computational cost and additional chances of errors. As for the first issue, in considering the Moore's Law and the continuous computer performance improvements, the perturbation approach seems to be very promising to pursue further. To reduce the number of additional errors, a possible solution would be to find a consistent result in the many recognition results of different perturbations. The methods or approaches developed in the researches of classifier combination would be other options.

5 Implementation Methodologies

5.1 Cycles of Robustness Implementation

In developing pattern recognition systems, there need to be a higher-order consideration, or methodology, on how to raise the recognition performance in front of the unlimited real samples. It is almost impossible to design the whole recognition system in detail in advance and then build it. Unpredictable problems are always waiting for us. Therefore, the developmental process goes step-by-step, unfortunately, by evaluating the performance, analyzing the problems, and then solving them. It goes like a spiral [16].

Logically, the first step of robustness implementation is to identify performance influencing factors (PIFs). It is to know the enemies, or to know the technical difficulties. However, we can identify difficulties in reality only after we have a recognition engine or a simulator. Difficulties are the weak points of the methods, algorithms, and a recognition engine we have now. So, we do simulation experiments on a huge dataset of real samples. A field test of a prototype engine for more than several months or longer is preferable to collect a sufficient number of real samples covering possibly all difficulties. Some problems exist in seasonal changes in the incoming mail piece streams.

The more complex the recognition target is, the more complex the recognition engine becomes. It further means the more samples we need (see Table 1 again). The cycles of robustness implementation is as follows.

- Acquire (more) samples and groundtruthing
- Develop a better/additional method and algorithm
- Implement the method and algorithm
- Test and evaluate them
- Analyze errors and rejections
- Identify (additional) PIFs
- Make a plan of the next cycle including additional sample acquisition and approaches to attack the identified factors
- Repeat the above until the performance is satisfactory.

In these cycles, additional sample acquisition is one of the keys to make the whole process more efficient and successful. Samples that contain difficulties are good to form *acceleration datasets*. This discussion relates to *active learning* or *learning with queries*, which is one of the hot topics in machine learning [17]. In an actual project running these cycles, there are project management issues as well [18].

5.2 Sample Datasets

Again, the samples that cover the PIFs are the key for the success. Collection is a laborious task and requires repetitive processing. The width and depth of variations vary depending on the progress of the development and on the problems. Development may start with a small number of samples. Difficult problems require more samples.

Four kinds of datasets are required:

- Validation datasets
- Training datasets
- Test datasets
- Acceleration datasets

The validation datasets are for selecting an approach, architecture, and algorithms from their alternatives, while the training datasets are for tuning the parameters of recognition components. These datasets need to be devised so that each recognition component can use them effectively. For example, there should be datasets specially arranged for algorithms of address block location, address line segmentation, character string recognition, and so on. In other words, these datasets are the databases of intermediate data.

The test datasets, which should not be used for training, are for evaluating the recognition performance as a whole. It is recommended to keep the test samples in multiple datasets. It is not appropriate to mix the batches of collections into a huge dataset. Such discipline is not only convenient for experiments, but also essential for evaluating the profile of recognition performance (Figure 1). In the case of postal address recognition, mail pieces to be sorted have different characteristics depending on the operational time of the day, week, or month. For example, delivery sequence sorting is done in very early mornings, outward dispatch sorting in daytime, and bulk mail (business mail) sorting in the afternoon. The same is true for the seasons and the areas; namely, there are seasonal and locational changes as well. Therefore, sample images from different time zones can form separate batches of sample images. By doing so, each batch of samples may reveal the performance differences, and possibly the problems seen from the customers.

The acceleration datasets play a major role in the robustness implementation. They are the datasets of "live images" that have been rejected by the recognition engine under the test, where, by definition, *live images* are sample images captured during the system operation. By using these problematic image samples, the cost of groundtruthing, analyzing and identifying major performance influencing factors can be minimized. This strategy is parallel to that of active learning, where additional samples are acquired, and most informative samples are selected for additional classifier training [17].

5.3 Basic Tactics for Improvements

In building up the recognition engine, control of errors and rejections require attention. Although the error and rejection can be traded off against each other in general, we should note that it is very difficult to convert an

erroneous result into a correct one by a single improvement step. Therefore, the basic tactic is to exterminate errors first. To do so, we can set the thresholds high so that the errors turn into rejections. Then, we try to turn the rejections into correct results.

In the following, we discuss the tactics more concretely. To do so, we first classify the final recognition results into the following seven classes:

- C: Correct acceptance
- R1: Rejection due to competing candidates having close scores
- R2: Rejection due to the top candidate having a low score
- R3: Rejection due to an empty candidate list
- E1: Error due to the right candidate being lost in competition
- E2: Error due to the right candidate having a low score
- E3: Error due to the right answer not appearing in the candidate list

When we have E1 or R1, the absolute score values of the top choices are good enough, or at least within a permissible range, but there are competing (wrong) candidates. In this case, the plan is to change the relative threshold parameters to convert the E1 case into R1, and to train classifiers further to convert R1 into C. When we have E2 and R2, the plan is usually to re-train the classifiers because we have the right candidate with a low score. (A 'low score' means that the score is smaller than an absolute threshold.) However, when we have E3 and R3, there must be something wrong somewhere, possibly in the earlier stages, because no right candidate is given. Then, we need to seek the 'problems' and make a plan to convert E3 and R3 at least to R1 or R2. (A rejection by the directory verification is classified as R3.) The 'problems' are usually very diverse, which are coming from PIFs shown in Table 1. In either case of errors, the threshold values (absolute and relative) should be reviewed to see if we could turn them into rejections.

In the case of 'semantically rich domains' like postal address recognition, we are able to keep the errors low by consulting with a directory, and we can have many R2 and R3 rejections instead. To repeat, it is very important to turn errors into rejections by keeping the right candidate always in the hypotheses. If we succeed to place us in this situation, the improvement process may proceed in a positive, straightforward way. So, the tactic is to reduce the error rate, first, giving more rejections, and then raise the correct rate next.

5.4 Example Case

In this section, we would like to show a real example of robustness implementation. It is a case of development of a Japanese postal address recognition engine used in a Hitachi Mail Sorting Machine (Figure 2). The data shown here is from the simulation experiments in the laboratory.

Read rate profiles of handwritten postal address are shown in Figure 16 for four software versions, V1 through V4. They are from Oct. 1997, Nov. 1997, Mar. 1998, and Jun. 1999, respectively. The average and the standard deviation of the read rates for the same four versions are also plotted in Figure 17, where AV, SD, BK, MP, and HW stand for the average, the standard deviation, bulk mail (business mail), machine-printed mail, and handwritten mail, respectively. Robustness, when defined as the standard deviation of read rates against a number of sample datasets, as shown in these figures, has not been improved much, except for bulk mail (BK).

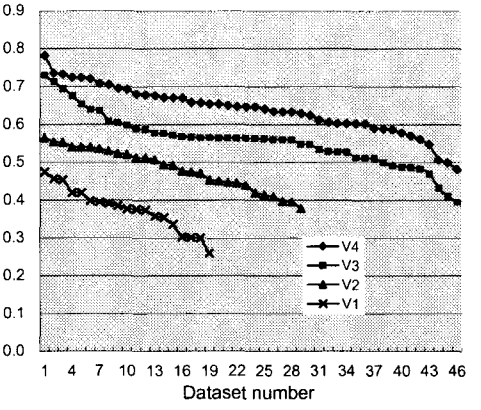


Figure 16: Read rate profiles for handwritten datasets.

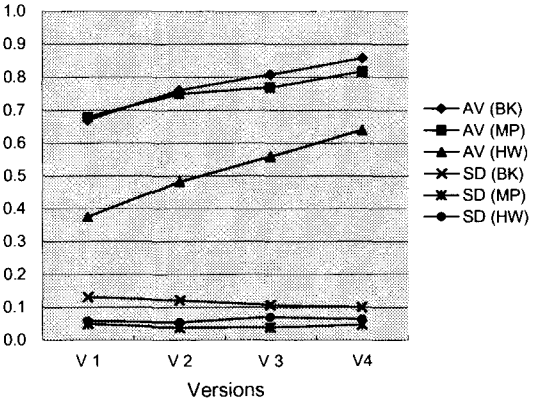


Figure 17: Read rate improvements.

6 Conclusion

This paper has discussed approaches to attacking uncertainty and variability, which are the main themes of pattern recognition. These relate to accuracy and robustness, respectively. We have identified and explained five design principles to tackle these common problems of pattern recognition. These principles are:

- Hypothesis-driven principle
- Deferred decision/Multiple hypotheses principle
- Information integration principle
- Conflicting solutions principle
- Perturbation principle

Among these, information integration has been studied intensively in this research community in general. It includes three approaches: (1) Process integration, (2) Combination-based integration, and (3) Corroboration-based integration. These principles have been explained by relying on the examples from our experiences in the

development of a Japanese postal address recognition engine.

One of the objectives of this paper has been to show how to combine recognition components (and subcomponents) to make the whole. As discussed, each component receives an input, which has uncertainty in it, and produces an output, which also has uncertainty. Simple combination does not produce satisfactory recognition results. The above principles help solve this combination problem.

Variability in input patterns directs us to turn to classifiers, for example, but it also directs us at a higher level to think about the developmental process of a recognition system. It is not a simple process to develop a recognition system with sufficient recognition performance. This paper has also discussed higher level issue.

In concluding, comprehensive approaches are mandatory to build an industry strength system. There is no single prescription for a high accuracy and high robustness.

Reference

[1] T. Kagehiro, M. Koga, H. Sako, and H. Rule," *Proc. ICPR2004*, Vol. 2, Aug. 2004, pp. 582-585.

[2] K. S. Fu, Y.T. Chien, and G.P. Cardillo, "A Dynamic Programming Approach to Sequential Pattern Recognition," IEEE Trans. Electronic Computers, Vol. EC16, Mar. 1967, pp. 313-326.

[3] P. H. Winston, "Artificial Intelligence," Addison-Wesley Publishing Company, Apr. 1979, pp. 89-105.

[4] H. Fujisawa, Y. Nakano, and K. Kurino, "Segmentation Methods for Character Recognition: From Segmentation to Document Structure Analysis," *Proc. IEEE*, Vol. 80, No. 7, 1992, pp. 1079-1092.

[5]K. Marukawa, M. Koga, Y. Shima, and H. Fujisawa, "An Error Correction Algorithm for Handwritten Chinese Character Address Recognition," *Proc. 1st ICDAR*, Saint-Malo, France, Sep. 1991, pp. 916-924.

[6] F. Kimura, M. Sridhar, and Z. Chen, "Improvements of Lexicon-Directed Algorithm for Recognition of Unconstrained Hand-Written Words," *Proc. 2nd ICDAR*, Tsukuba, Japan, Oct. 1993, pp. 18-22.

[7] C. H. Chen, "Lexicon-Driven Word Recognition," *Proc. 3rd ICDAR*, Montreal, Canada, Aug. 1995, pp. 919-922.

[8] M. Koga, R. Mine, H. Sako, and H. Fujisawa, "Lexical Search Approach for Character-String Recognition," *Document Analysis Systems: Theory and Practice*, S.-W. Lee and Y. Nakano eds., Springer, 1999, pp. 115-129.

[9] C.-L. Liu, M. Koga and H. Fujisawa, "Lexicon-driven Segmentation and Recognition of Handwritten Character Strings for Japanese Address Reading," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 24, No. 11, Nov. 2002, pp. 425-1437.

[10] H. Ikeda, N. Furukawa, M. Koga, H. Sako, and H. Fujisawa, "A Context-Free Grammar-Based Language Model for Document Understanding," *Proc. DAS2000*, Rio de Janeiro, Brasil, Dec. 2000, pp. 135-146.

[11] C. Y. Suen, C. Nadal, T. A. Mai, R. Legault, and L. Lam, "Recognition of Totally Unconstrained Handwritten Numerals Based on the Concept of Multiple Experts," *Proc. 1st IWFHR*, Montreal, Canada, 1990, pp. 131-143.

[12] L. Xu, A. Krzyzak, and C. Y. Suen, "Methods of Combining Multiple Classifiers and Their Applications to Handwriting Recognition," *IEEE Trans. Systems, Man and Cybernetics*, Vol. 22, No. 3, 1992, pp. 418-435.

[13] T. K. Ho, J. J. Hull, and S. N. Srihari, "Decision Combination in Multiple Classifier Systems," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 16, No. 1, Jan. 1994, pp. 66-75.

[14] G. F. Houle, D. B. Aragon, R. W. Smith, M. Shridhar, and F. Kimura, "A Multi-Layered Corroboration-Based Check Reader," *Proc. IAPR Workshop on Document Analysis Systems*, Malvern, USA, Oct. 1996, pp. 495-546.

[15] T. M. Ha and H. Bunke, "Off-Line, Handwritten Numeral Recognition by Perturbation," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 19, No. 5, May 1997, pp. 535-539.

[16] H. Tang, E. Augustin, C. Y. Suen, O. Baret, M. Cheriet, "Spiral Recognition Methodology and Its Application for Recognition of Chinese Bank Checks," *Proc. 9th IWFHR*, Kokubunji, Japan, Oct. 2004, pp. 263-268.

[17] R. O. Duda, P. E. Hart, and D. G. Stork, "Pattern Classification," Second Edition, John Wiley & Sons, 2001 (p. 480).

[18] H. Fujisawa and H. Sako, "Balance between Optimistic Planning and Pessimistic Planning in a Mission Critical Project," *Proc. IEMC2003*, Albany, NY, Nov. 2003, pp. 605-609.

An Efficient Candidate Set Size Reduction Method for Coarse-Classifier of Chinese Handwriting Recognition

Feng-Jun Guo , Li-Xin Zhen, Yong Ge

Motorola Labs, China Research Center, 38F, CITIC Square 1168, Nanjing Rd. W., Shanhai, P.R.C,
Feng-JunGuo@email.mot.com

Yun Zhang

Electronic Engineering Department, Shanghai Jiaotong Univ., 800 Dongchuan Rd., Shanghai, 200240, P.R.C
eyunzhang@sjtu.edu.cn

Abstract

In this paper, we introduce an efficient clustering based coarse-classifier for Chinese handwriting recognition system to accelerate the recognition procedure. We define a candidate-cluster-number for each character. The defined number indicates the within-class diversity of a character in the feature space. Based on the candidate-cluster-number of each character, we use a candidate-refining module to reduce the size of the candidate set of coarse-classifier. Experiments show that the method effectively reduces the output set size of coarse-classifier while keeping the same coverage probability of candidate set. The method has low computation-complexity.

1 Introduction

For Chinese character recognition system, a coarse classifier is used to prune the search candidates of fine classifier based on a complex feature. It accelerates the recognition procedure of a system. The coarse classifier technique has been widely used in many practical systems. T.S Lin etc presented a coarse-classifier using structure-based feature [6]. S.R. Lay etc. introduced a radical-based coarse-classifier method [7]. Y.Y. Tang provided an overlap clustering based method and proposed a group classifier concept [8]. Y. Yang provided a coarse-classifier method using pivots [11].

A typical coarse classifier method is clustering based method. In the method, all characters are grouped into a few clusters. Each cluster covers an amount of characters, which is denoted as member-character of its own cluster. When a sample is inputted into a recognition system, the coarse-classifier selects the first n nearest clusters of the input as candidate-cluster[11]. System sets member-characters of these clusters as candidate set for the following fine-classifier.

In clustering based Chinese coarse-classifier, in order to achieve a same predefined correct coverage-rate, different character need different number of candidate-clusters. The reason is that within-class diversity of each character is different.

In the method of this paper, for each character, we calculate a candidate-cluster-number. The number is served as a measure of within-class diversity of feature space of a character. Based on such numbers of each character, we reduce the size of candidate set of the coarse-classifier. The experiments section of this paper shows that the proposed method effectively reduces the size of the coarse-classifier. The computation-consumption of the method is very low.

The paper is organized as follows. Section 2 introduces the workflow of conventional clustering-based coarse-classifier in detail. In section 3, we describe our candidate-set size reduction method. In section 4, we provide the experimental result. In section 5, conclusions are stated.

2 Clustering-based Coarse-classifier

Clustering method is a fundamental technique of pattern recognition [1, 2, 3, 4, 5, 9]. In [10], T.Kanungo provided the latest achievement of K-means cluster method.

Figure 1 shows the workflow of a clustering based coarse-classifier. In the training procedure, firstly, system calculates one or several templates (cluster center of feature vectors) for each character. Based on these templates, it creates a small number of clusters. Then each template is allocated into its nearest cluster. So a cluster covers the templates of several characters. We name a cluster as owner-cluster of its covered characters, and denote its covered characters as the element-character of its nearest cluster.

When a sample is inputted, the coarse-classifier calculates the similarity between its feature and each

cluster's center. Then these similarities are sorted in descending order. Finally, the first n nearest clusters is selected as the candidate-clusters of the input, and the element-characters of these n clusters make up the candidate set of coarse-classifier.

In the sorted clustering list, we denote $o_{clust}(i)$ the order-number of the i^{th} cluster for the input sample. Let call $o_{clust}(i)$ the similarity-order of the i^{th} cluster. The term will be used in next section

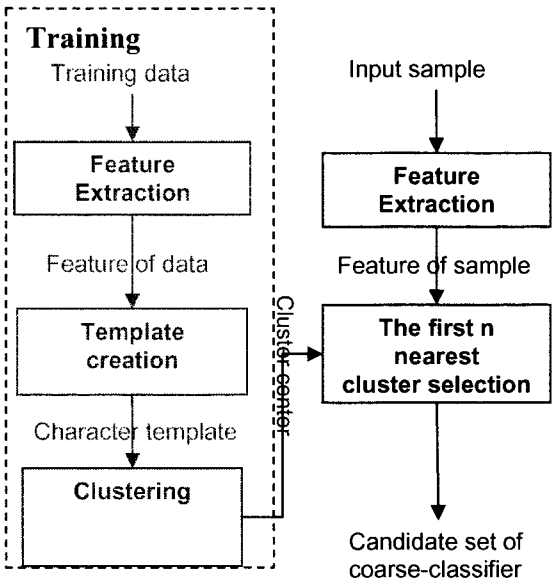


Figure 1: Workflow of clustering-based coarse-classifier

3 Candidate set size reduction method

In this section, we describe how to reduce the size of candidate set by implementing feature distribution property of characters.

3.1 Feature distribution property of element-character of cluster

Let hit_r be a hit-rate of a coarse-classifier. It is a probability that the candidate set of the coarse-classifier contains the ground-truth of the input samples. Eq. 1 shows definition of the hit-rate.

$$hit_r = C_{cr} / C_{sum} \quad (1)$$

where, C_{cr} is the counts of samples that are correctly covered by candidate set of coarse-classifier, and C_{sum} is the total number of samples in a test set.

As mentioned in section 2, in the conventional clustering based coarse-classifier method, system selects a uniform number of candidate-clusters to construct a candidate set. In fact, in order to get a predefined hit-rate, different character needs different number of candidate-clusters. Some characters can get very high hit rate by selecting a very small number of nearest clusters. This is because within-class diversity of each character is different.

Figure 2 provides an intuitive description for within-class diversity of different characters. In figure 2, the dash-line ellipses show the boundary of feature space of character 1, 2 and 3. The black dots are the template of each character. Solid-line ellipse and circle represent the boundary of each cluster. The black cross ('+') is the center of each cluster. In this figure, the character 1 is an element-character of cluster A. Its feature space has small within-class diversity. The features of all samples of character 1 are very close to the center of the cluster A. So for character 1, we just need to select the first 1 nearest cluster as candidate-cluster, and the hit-rate of the coarse-classifier is 100%. But for character 3, it is an element-character of cluster C. Its feature space has big within-class diversity. For the sample in left-down boundary of character 3, cluster C is the 3rd nearest cluster of it. So in order to achieve 100% hit-rate, character 3 should select the first 3 nearest clusters as candidate-clusters of the coarse-classifier.

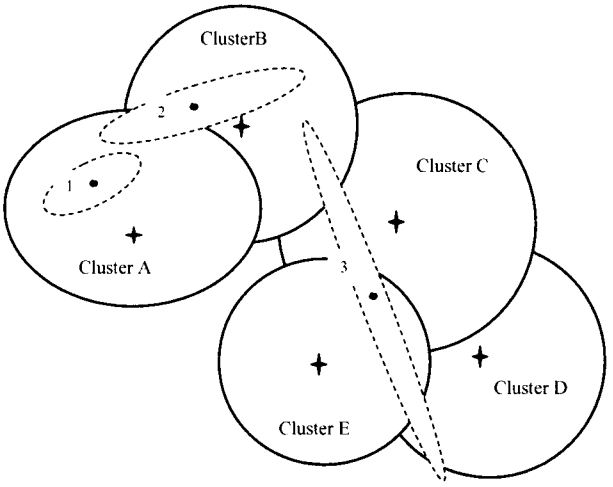


Figure 2: Cluster and class feature distribution

Based on figure 2, we draw following hypothesis. If the within-class diversity of the feature of a character is small, the character just needs a small number of candidate-clusters to get a high hit-rate. Otherwise, it needs a larger number of candidate-clusters to get similar hit-rate.

3.2 Candidate-set refining method using feature distribution property

In a large size training set, let p_{hit} be a threshold of the hit-rat for each character, and suppose that the value of p_{hit} is very close to 100%. As mentioned in 3.1, in order to achieve the hit-rate p_{hit}, each character needs different number of candidate-clusters

in the coarse-classifier. Let $ord_{\max}(P)$ denote the number of candidate-clusters for a character P, which guarantees that the hit-rate of character P is equal to p_{hit}. We call $ord_{\max}(P)$ the candidate-cluster-number of the character P.

Then in the data structure of a cluster of coarse-classifier, we store not only the element-character index of the cluster, but also the candidate-cluster-number of each character. System uses the candidate-cluster-number of each character to refine candidate set of the coarse-classifier. The idea is as follows. Suppose an input is a sample from character P, and the owner-cluster of P is the i^{th} cluster. Let $o_{clust}(i)$ be the similarity-order of the i^{th} cluster. Because the value of p_{hit} is close to 100%, for training samples of the character P, there are a very high possibility that eq.2 holds.

$$o_{cluster}(i) <= ord_{\max}(P) \quad (2)$$

Let n be the average number of candidate-clusters of the training set. When a sample is inputted, suppose the i^{th} cluster is one of the first n nearest clusters of the sample, and character P is the element-character of the i^{th} cluster. If $o_{clust}(i)$ is larger than $ord_{\max}(P)$, the possibility that the input is a sample of character P is very low. Then we remove character P from candidate set of coarse-classifier. This is a basic method that we refine the candidate set.

Figure 3 shows the workflow of the candidate-set refining method. When a handwriting sample is inputted into the recognizer, the coarse-classifier firstly extracts the first n nearest clusters, where n is the average number of candidate-clusters. For each element-character of the n nearest clusters, we compare its $ord_{\max}$ with the o_{clust} of its owner-cluster. If the o_{clust} of its owner-cluster is larger than its $ord_{\max}$, we remove the element-character from the candidate set. Finally, we can reduce the size of candidate set of coarse-classifier in this way by a candidate-set refining model.

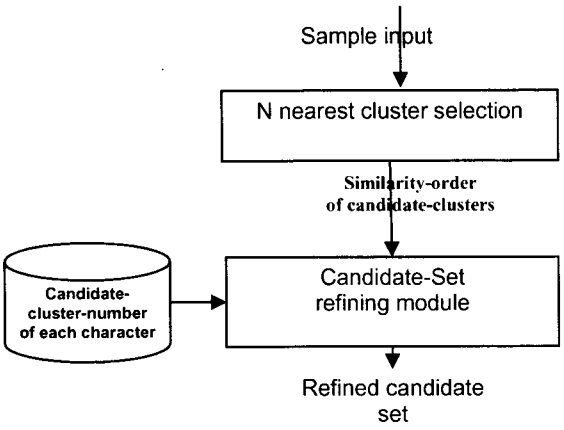


Figure 3: Workflow of the refining method of candidate set

When the size of a training set is large, the value of cluster-candidate-number based on the training set can be implemented to an independent test set. Experiments show that the proposed method has good generalization property.

4 Experiments

The database used in our experiments contains 3755 daily-used handwritten Chinese characters. We separate these samples into a training set and a test set. In the training set there are 1,169,209 samples. Each class has more than 300 samples written by different people. In the test set, each character has 50 samples. In our experiment, the feature vector of coarse classifier is a compressed stroke-direction feature with 80 dimensions. The clustering method is based on LBG algorithm [9].

Firstly, we show the results for training data. Then we provide the result for a test data. The result for the independent test set shows the generalization property of the method.

For the clustering based coarse-classifier, clustering is base on the templates of each character. For each character, the number of templates can be set as one or several. The number of templates of each character has an effect on the size of candidate-set. For the results of test data, we discuss the results of single-template based clustering and multi-template based clustering respectively. Furthermore, we discuss how to select the value of p_{hit}.

4.1 Result for training set

Table 1 shows the results of train set. We list the results that the coarse-classifier applies different total number of clusters. Here, the value of p_{hit} is set as 0.996. Let num_{sum} and num_{can} be the total number of clusters and the number of candidate-cluster used in coarse-classifier respectively. In tables of this section, the original method stands for the method without candidate-set refining module. Experimental data in table 1 shows that, for different total number

of clusters, the method in this paper reduces the size of candidate-set while keeping the hit-rate.

Table 1: Results of training set: original method and the method in this paper

Method	num_{sum}	Hit Rate	num_{can}	Candidate set size
Original method	64	99.10%	9	670
	128	99.03%	12	458
	256	99.06%	15	282
	512	99.05%	17	181
Method in this paper	64	99.09%	12	525
	128	99.04%	15	339
	256	99.06%	19	225
	512	99.05%	22	154

4.2 Result for test set

In order to evaluate the generalization property of the method, we test the proposed method using an independent test set. For all experiments data in section 4.2, the value of p_{hit} is set as 0.996.

4.2.1 Clustering based on single-template of each character

In the experiments in table 2, the cluster of coarse-classifier is based on single-template of each character. The data in table 2 shows, for the single-template case, the proposed method reduce the size of candidate-set for the test set too.

Table 2: Results of test set: clustering base on single-template of each character

Method	num_{sum}	Hit Rate	num_{can}	Candidate set size
Original method	64	99.14%	9	639
	128	99.04%	12	421
	256	99.04%	15	262
	512	99.05%	18	159
Method in this paper	64	99.14%	12	542
	128	99.04%	15	359
	256	99.04%	19	230
	512	99.05%	23	147

4.2.2 Clustering based on multi-template of each character

In our experiment, for the multi-template case, if several templates of one character fall into the candidate set of coarse-classifier, the size of candidate set just add 1 when we calculate the size of the candidate-set. Table 3 and 4 use the rule to calculate the size of candidate set.

Table 3 and table 4 give the comparison of the results of the original and proposed method for multi-template case. For the experiments of table 3, when we conduct clustering, each character has 2 templates. For table 4, the template-number of each character is 4. Data in tables 3 and 4 show that, in multi-template case, the proposed method reduce the size of candidate-set too.

Table 3: Comparison the result that each character has 2 templates

Method	num_{sum}	Hit Rate	num_{can}	Candidate set size
Original method	128	99.09%	10	382
	256	99.09%	12	232
Method in the paper	128	99.09%	13	340
	256	99.09%	17	218

Table 4: Comparison the result that each character has 4 templates

Method	num_{sum}	Hit Rate	num_{can}	Candidate set size
Original method	128	99.10%	8	357
	256	99.01%	9	204
Method in the paper	128	99.09%	9	319
	256	99.02%	10	189

From the data of table 2, 3 and 4, we can find that the increasing of the total cluster number weakens the advantage of the proposed method. But in the coarse-classifier, the number of cluster should not be very large. Otherwise, the coarse-classifier will take much computation time. So the proposed method in the paper is very useful to construct an efficient coarse-classifier.

4.3 Selection of the hit-rate threshold when calculating candidate-cluster-number

In the method of this paper, the value of p_{hit} is very important for the final result. It affects the value of candidate-cluster number of each character. Figure 4 shows the hit-rates of test set for different values of p_{hit} in training procedure. For all experiments of section 4.3, the number of total cluster of coarse-classifier is 128.

The cursives in figure 4 show that the value of p_{hit} is the upper bound of the hit-rate for the proposed method. If a recognition system requires a high hit-rate, p_{hit} should be set at a high value. Even when p_{hit} is set as 100%, for the test set, the hit-rate of the proposed method has some degrade compared with one of the original method. It means that the sample of training data does not cover all possible boundary-case of feature space. So a large training set is very important for the proposed method.

Figure 5 shows the relationship between the hit-rate and the candidate-size for different values of p_{hit} in training procedure. The figure5 shows, smaller p_{hit} reduce more element-characters from the candidate-set. When the system uses a small p_{hit}, it can't achieve a very high hit-rate because the value of p_{hit} is the upper bound of its hit-rate. So the selection of p_{hit} is a trade-off between the upper bound of the hit-rate and the amount of size reduction of the candidate-set.

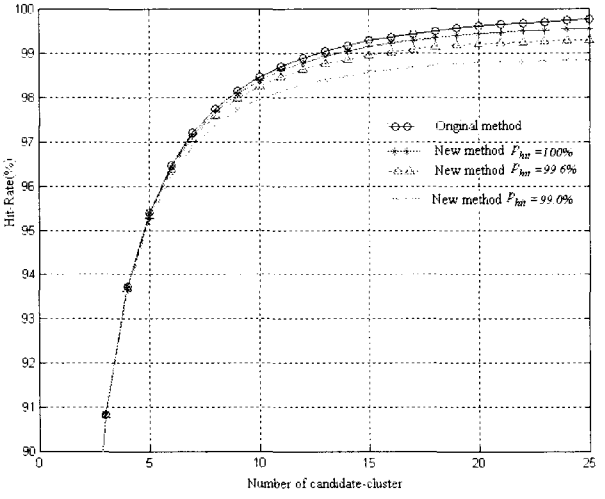


Figure 4: The upper bound of hit-rate for different p_{hit}

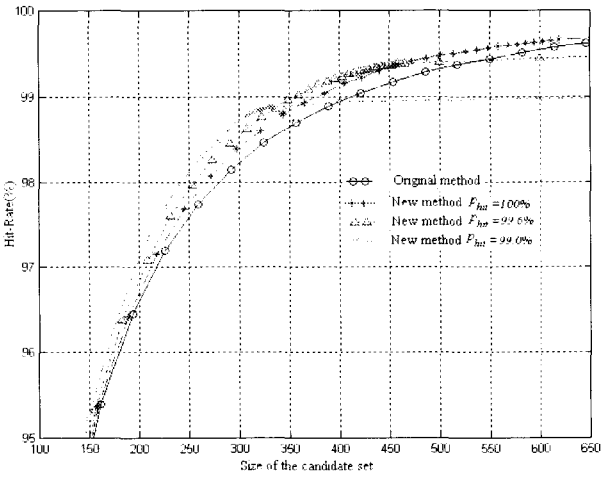


Figure 5: The relation between hit-rate and the size of candidate-set for different p_{hit}

5 Conclusion

In this paper, we provide an efficient candidate set size reduction method of the coarse-classifier. The method implements the feature distribution information of each character. It defines a candidate-cluster-number of each element-character. Using the number, we refine the selected candidate-clusters to reduce the size of candidate set. Experiments show the method has good generalization property. Compared with conventional methods, it adds very little computation consumption and ROM storage size.

References

[1] R. O. Duda, P. E Hart, D. G. Stork, Pattern Classification. John Wiley & Sons, New York, 2001.

[2] R.S. Bradley and U. Fayyad, "Refining Initial Points for K-means Clustering", Proc. 15th Int'l Conf. Machine Learning, Madison, Wisconsin, USA, 1999, pp. 91-99.

[3] S. P. Lloyd, "Least Squares Quantization in PCM", IEEE Trans. Information Theory, vol 28, 1982, pp.129-137.

[4] J. MacQueen, "Some Methods for Classification and Analysis of Multivariate Observations", Proc. Fifth Berkeley Symp. Math. Statistics and Probability, vol 1 Berkeley , California, USA, 1967, pp.281-296.

[5] E. Forgey, Cluster Analysis of Multivariate Data: Efficiency vs. Interpretability of Classification", Biometrics, vol 21, 1965, pp. 768.

[6] T.Z. Lin and K.C. Fan, "Coarse Classification of On-Line Chinese Characters via Structure Feature-Based

[7] S.R. Lay, C.H. Lee, N.J. Cheng, C.C Tseng, etc, " On-Line Chinese Character Recognition with

Effective Candidate Radical and Candidate Character Selections", Pattern Recognition, Vol 29, No.10.1996 pp 1647-1659.

[8] Y.Y Tang, L.T Tu, J. Liu, S.W Lee etc , "Offline Recognition of Chinese Handwriting by Multifeature and Multilevel Classification", IEEE Trans. PAMI, vol 21, No.3, 1999, pp.258-262

[9] Y.Linde, A. Buzo, and R. M. Gray, "An algorithm for vector quantizer design", IEEE Trans. On Comm., COM-28(1), 1980, pp.84-95

[10] T.Kanungo, D.M. Mount, N.S. Netanyahu, etc, "An Efficient k-Means Clustering Algorithm: Analysis and Implementation", IEEE Trans. PAMI, vol 24, No.7, 2002, pp.881-982

[11] Yiping Yang, O. Velek, M. Nakagawa, "Accelerating Large Character Set Recognition using Pivots", Proc. 7th ICDAR, vol. 4C, Edinburgh, Scotland, 2003, pp.262–267

Recent results of On-line Japanese handwriting recognition and its applications

Masaki Nakagawa Junko Tokuno Bilan Zhu Hideto Oda Akihito Kitadai
Tokyo University of Agriculture and Technology
Naka-cho 2-24-16, Koganei, Tokyo, 184-8588, Japan

Abstract

This paper discusses on-line handwriting recognition of Japanese characters, mixture of ideographic characters (Kanji) of Chinese origin and phonetic characters made from them. Most of Kanji character patterns are composed of multiple subpatterns called radicals which are shared among many (sometimes hundreds of) Kanji character patterns. This is common is Oriental languages of Chinese origin, i.e., Chinese, Korean and Japanese. It is also common that each has thousands of characters. Due to these characteristics, structured character pattern representation (SCPR) composed of subpatterns is effective in terms of the size reduction of a prototype dictionary (a set of prototype patterns) and the robustness to deformation of common subpatterns. In this paper, we show a prototype learning algorithm and HMM-based recognition for SCPR. Then, we combine the SCPR-based on-line recognizer with a compact off-line recognizer employing quadratic discriminant functions. Moreover, we also discuss on-line handwritten Japanese text recognition and propose character orientation free and line direction free handwritten text recognition and segmentation. Finally, as applications of on-line handwritten Japanese text recognition, we show segmentation of mixed objects of text, formulas, tables and line-drawings and handwritten text search.

1 Introduction

As PDAs, tablet PCs, and other pen-based or paper-based systems such as Anoto pen [1], e-pen [2] spread, the demand for improving on-line handwriting recognition and liberating it from writing constraint is still increasing. In on-line handwriting recognition, both temporal information of pen tip movement and spatial shape information are available, so that it is able to yield higher recognition accuracy than off-line handwriting recognition. Moreover, on-line handwriting recognition provides good interaction and adaptation capability because the writer can respond to the recognition result to correct misrecognition and rejection.

The research of on-line handwriting recognition started in the 1960s and has been receiving intensive interest from the 1980s. The comprehensive survey before 1990s is made in [3][4]. As recent survey papers, Plamondon et al. mainly reviewed the status of western on-line handwriting recognition [5] while Liu et al. and Jaeger et al. reviewed that of on-line Chinese and Japanese handwriting recognition [6][7]. In this paper, we mainly discuss on-line Japanese handwriting recognition including our recent results.

The Japanese character set consists of various characters: numerals, symbols, Hiragana, Katakana and, Kanji characters of Chinese origin. Hiragata and Katakana are phonetic characters. The former consists of 83 characters and the latter consists of 86 characters. On the other hand, Kanji characters are idiographic characters. Two classes are defined for the purpose of computer processing: JIS first level and JIS second level (JIS stands for Japanese Industrial Standard). While the JIS 1st level set contains 2,965 characters which are common characters and necessary for reading newspaper, the JIS 2nd level set contains 3,390 characters which are less common characters and special characters for naming.

Most of Kanji character patterns are composed of multiple subpatterns called radicals which are shared among many (sometimes hundreds of) Kanji character patterns. This is common in Oriental languages of Chinese origin, i.e., Chinese, Korean and Japanese. Among Kanji character patterns, some patterns are pretty simple and consist of a single radical while other patterns are complex and consist of multiple radicals.

In the field of pattern recognition, large volumes of sample patterns are as important as recognition methods. We then spent four years to compile two databases of on-line Japanese handwritten character patterns named "TUAT Nakagawa Lab. HANDS-kuchibue_d-97-06" (hereafter Kuchibue_d) [8] and "TUAT Nakagawa Lab. HANDS-nakayosi_t-98-09" (hereafter Nakayosi_t) [9]. Kuchibue_d stores 11,962 character patterns from each of 120 people (1,435,440 patterns) and Nakayosi_t stores about 10,403 patterns from 163 people each (1,695,689 patterns), thus they store more than 3

million patterns in total. About 50 institutions including more than 10 groups from abroad are using our databases so that we will base our experiments on these databases.

The large number of Japanese character categories affects the classification techniques. In case of western handwriting recognition, Hidden Markov models (HMM) are successfully applied. However, they are not common in Japanese handwriting recognition because they require huge amount of training data for each character. Therefore, DP-matching is the core of many on-line Japanese handwriting recognizers with several modifications proposed [10]-[12]. These large categories also affect the size of the dictionary (a set of prototype patterns) and the recognition speed, so that it has been difficult to use a powerful on-line recognizer or combined on-line/off-line recognizer which is effective to improve the recognition accuracy in a small computer.

The demand to remove writing constraint for on-line handwriting recognition is getting higher and higher since people can write more freely on enlarged surfaces of tablet PCs, electronic whiteboards and paper-based handwriting environments. However, segmentation and recognition of on-line handwritten Japanese text is a challenging work, since the variation of character size is large and people write text horizontally, vertically or even slantwise.

Against these problems, we present structured character pattern representation (SCPR)-based on-line handwriting recognition which has significant effect for the Japanese character set of the large category size in Section 2. Section 3 describes the combination of the SCPR-based on-line recognizer with a compact off-line recognizer. Section 4 presents on-line handwriting Japanese text recognition method liberated from constraints on line direction and character orientation. Section 5 describes some applications. Section 6 draws conclusion.

2 Character Representation

In this section, we describe Japanese Kanji character pattern representation which strongly related to the method of handwriting recognition.

In the representation for input patterns, the sequence of feature points or line segments in time series are commonly used in on-line Japanese handwriting recognition [10]-[17] The stroke order and the stroke directions of the input pattern are kept in these sequences. Some recognition methods employ the off-line features (e.g. directional features, loop of strokes) extracted from the images of the input patterns [18]-[20]. Employing these representations instead of the raw data of the input pattern (sampling points by digital pen) reduces the data size and noises.

Early researches attempted to extract subpatterns in an input character pattern and recognize the input pattern as the composite of subpatterns, but this did not succeed since subpattern extraction was very difficult. Instead, many systems employ the pattern structure composed of subpatterns in prototype representation and expand it to a sequence of feature points or line segments when matched with the input pattern. Shape variations or stroke order variations are registered as multiple alternatives into a subpattern so that they are shared among character patterns which include the subpattern in their shapes.

We will see this in more detail in the following sections.

2.1 Structured Character Pattern Representation (SCPR)

Japanese Kanji characters are mostly composed of multiple subpatterns called radicals. SCPR is to represent a character pattern as a composite of basic subpatterns (primitive that they are not further decomposed) and structural information on how to combine them (Figure 1) [12].

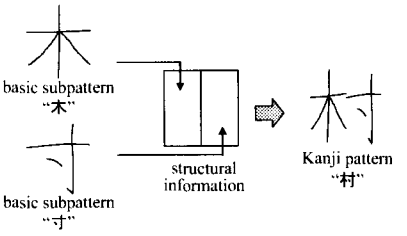


Figure 1: SCPR.

Here, we present two on-line handwriting recognition systems. One is based on the prototype learning algorithm (PLA) and linear-time elastic matching (LTM) [12][13] and the other system is based on HMM [21]. We call the former system "Sys_LTM" and the latter system "Sys_HMM". Both of them employ the SCPR dictionary in which prototypes of basic subpatterns (BSs) are shared among character categories as shown in Figure 2.

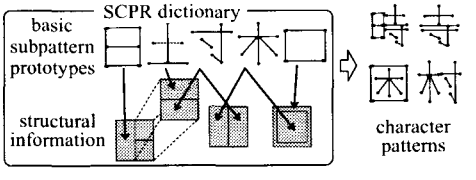


Figure 2: SCPR dictionary.

In the SCPR dictionary of Sys_LTM, all the BS prototypes as well as the character pattern prototypes are represented by a square shape with .128 x 128 resolution, and each of them is a sequence of feature points in a time series. When they are included in

prototypes of larger subpattern prototypes or character pattern prototypes, their sizes are reduced to bounding boxes in structural information through linear mapping (Figure 3). We call a result of the linear mapping a "mapped BS prototype", even if the mapping is sometimes identical (with no deformation). Hereafter, we refer to this as an MBS prototype.

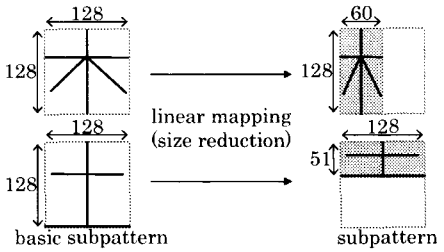


Figure 3: Linear mapping.

The SCPR dictionary of Sys_HMM has similar structural information and links to component SCPR-based HMMs in place of BS prototypes.

SCPR provides advantages to the size reduction of the dictionary (a set of prototype patterns) and the robustness against deformation of common radicals.

2.2 Prototype Learning Algorithm (PLA)

No matter what classification method is employed, prototypes greatly influence the performance of classifiers. PLA is a method to better approximate discrimination boundaries between different categories in a feature space [22]-[25]. Liu et al. have shown the advantages of PLA in off-line handwritten Kanji character recognition [26].

For on-line handwriting recognition systems whose prototypes are the sequences of feature points (e.g. Sys_LTM), we have proposed a PLA to improve the BS prototypes by moving their feature points [27]. The base learning method of our PLA is the generalized learning vector quantization (GLVQ) [25]. GLVQ updates the genuine prototype (the closest prototype in the correct class) and the rival prototype (the closest one in different classes) using learning patterns.

2.2.1 Recognition by LTM and PLA

Our PLA uses the correspondences between feature points that are the results of LTM in the process of recognition. In Figure 4, the dash lines show the correspondences. Although general elastic matching methods commonly generate one-to-many or many-to-one correspondences between the feature points, our method of Sys_LTM generates only one-to-one correspondences by discarding the uncertain correspondences.

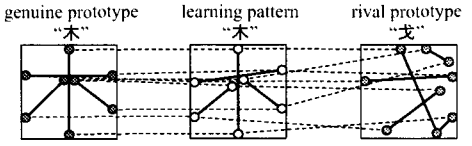


Figure 4: Correspondences between feature points.

2.2.2 SCPR-based PLA

To improve the prototypes in the SCPR dictionary, we should consider that each prototype matched with the learning pattern is a composite of MBS prototypes. Figure 5 shows the process of our PLA to improve the feature point v in the BS prototype. Each $u(v)$ is a feature point of the MBS prototype mapped from the feature point v, and each p_l is a feature point in the learning pattern corresponding to $u(v)$. S is the bounding box size of each MBS prototype, and $G(p_l - u(v), S)$ is the function to normalize the displacement $p_l - u(v)$ by the bounding box size. Every displacement between $u(v)$ and p_l is measured and reflected in the feature point v using $G(p_l - u(v), S)$.

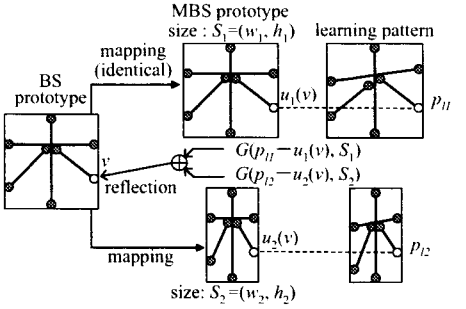


Figure 5: Displacement reflection.

We transform the formulae of GLVQ into (1) and (2) to update the feature points of the BS prototypes.

$$\begin{cases} x_i' = x_i + 4\alpha(t)l_k(1-l_k)\dfrac{G(d_i,S_i)G(x_l-x_i,w_i)}{\{G(d_i,S_i)+G(d_j+S_j)\}^2} \\ y_i' = y_i + 4\alpha(t)l_k(1-l_k)\dfrac{G(d_i,S_i)G(y_l-y_i,h_i)}{\{G(d_i,S_i)+G(d_j+S_j)\}^2} \end{cases} \quad (1)$$

$$\begin{cases} x_j' = x_j - 4\alpha(t)l_k(1-l_k)\dfrac{G(d_i,S_i)G(x_l-x_j,w_j)}{\{G(d_i,S_i)+G(d_j+S_j)\}^2} \\ y_j' = y_j - 4\alpha(t)l_k(1-l_k)\dfrac{G(d_i,S_i)G(y_l-y_j,h_j)}{\{G(d_i,S_i)+G(d_j+S_j)\}^2} \end{cases} \quad (2)$$

In the above formulae (1) and (2), $u(v_i)=(x_{ui}, y_{ui})$ in the genuine prototype is mapped from $v_i=(x_i, y_i)$ of one BS prototype, while $u(v_j)=(x_{uj}, y_{uj})$ in the rival prototype is mapped from $v_j=(x_j, y_j)$ of the other BS prototypes. The feature point $p_l=(x_l, y_l)$ in the learning pattern corresponds to $u(v_i)$ and $u(v_j)$. The term $\alpha(t)$ denotes the learning rate. The other parameters l_k, μ_k, d_i, and d_j are defined as follows:

$$l_k = l_k(\mu_k) = \frac{1}{1+e^{-\mu_k}} \quad (3), \qquad \mu_k = \frac{G(d_i,S_i) - G(d_j,S_j)}{G(d_i,S_i) + G(d_j,S_j)} \quad (4)$$

$$d_i = \|p_l - p_i\| \quad (5), \qquad d_i = \|p_i - p_i\| \quad (6)$$

We obtained the distribution of the feature points in learning patterns corresponding to each feature point in MBS prototypes since the size of the distribution (the average distance from the center of the distribution to every $l(v)$ in the distribution) shows to what degree each feature point in the MBS prototype can move (Figure 6).

To describe the size of distribution, we employed the average distance from the center of the distribution to every $l(v)$ in the distribution (Figure 7).

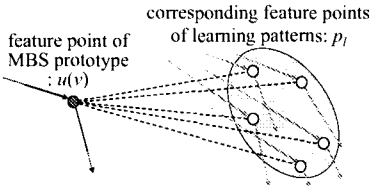


Figure 6: Distribution of feature points.

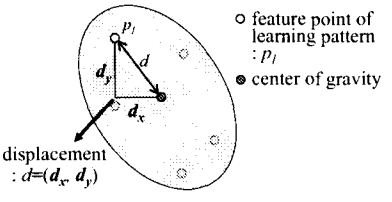


Figure 7: Distance between the center of gravity and feature points.

We employed our database: Nakayosi_t to obtain the distribution. Through the experiment using the database, we estimated the relations between the bounding box size of MBS prototype S=(w, h) and the distribution size D for vertical and horizontal directions as follows.

$$D_w(w) = 0.0846\ w + 1.7 \quad (7)$$

$$D_h(h) = 0.0539\ h + 3.5 \quad (8)$$

The normalization formulae using the relation are as follows.

$$G_w(x_1 - x_2, w) = (x_1 - x_2)\{D_w(128)/D_w(w)\} \quad (9)$$

$$G_h(y_1 - y_2, h) = (y_1 - y_2)\{D_h(128)/D_h(h)\} \quad (10)$$

2.2.3 Evaluation for SCPR-based PLA

In the experiment to evaluate our PLA, we improved the SCPR dictionary of Sys_LTM. As the set of learning patterns, we employed all the character patterns of the JIS 1st level set in Nakayosi_t. We employed another database Kuchibue_d for evaluation. In the first step of learning, we generated the averaged BS prototypes by learning patterns. Then, we performed our PLA. Before learning, the recognition rate of Sys_LTM with the dictionary was about 84.4% for the data set of evaluation. After learning, the rate became about 89.1%.

2.2.4 SCPR-based PLA for Off-line Recognition

Now, we describe SCPR and the learning method for off-line recognition. In general off-line recognition methods, directional features with four-directional or eight-directional quantization are extracted from a character pattern divided into an array of cells [20][28]. Therefore, a character (as same as a prototype) is represented by a matrix of directional features. In Figure 8, $f_v(i, j)$ is the set of directional features extracted from the cell of the i-th row and j-th column in the character pattern.

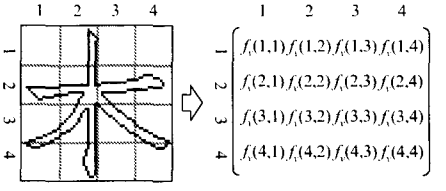


Figure 8: Matrix of the directional features extracted from the character pattern.

When the character C is composed from a set of subpatterns, we can show it as follows:

$$C = \sum_{i=1}^{Ns} A_i S_i \quad (11)$$

(Ns: number of subpatterns, A_i: linear mapping to make a subpattern, F_i: matrix of f_v to describe basic subattern)

Since subpatterns don't overlap each other in Japanese characters,

$$F_i = A_i^{-1} C \quad (12)$$

The directional features in a basic subpattern are transformed through linear mapping. Therefore, when a character pattern is learned and reflected back to a subpattern, the directional features are again mapped through inverse mapping with the directions modified. To enable mapping and inverse mapping that may change the directions of segments, the directional features must be extracted and represented finely enough.

For off-line character recognition, the dimensions of features are often reduced by the K-L transformation:

$$y = \psi' x \quad (13)$$

In this case, the reduced set of features is difficult to decompose into subpatterns. However, if the reduction is small and ψ is chosen to minimize the mean square error, the original features can be approximated by

$x \approx \psi y$ (14)

This enables decomposition into subpatterns and reflection back to their features. Then we can realize structural learning even for the common off-line character recognition methods.

2.3 SCPR-based HMM

The HMM has been successfully applied not only to Western handwriting recognition [29][30] but also to on-line handwriting recognition of Chinese [31], Japanese [15][32]-[34] and Hangul characters in Korea [35] owing to its promising ability to model deformations of strokes and variations of sampled feature points. In case of alphanumeric on-line handwriting recognition, "character HMM" has been widely employed, where each whole character pattern in the alphabet is modeled typically by one HMM and all words are represented by a sequence of character HMMs. We need to prepare only some dozens of HMMs at most. Whereas, there are thousands of characters in Oriental characters of Chinese origin, so that the character HMM leads to an infeasible character recognition system requiring huge amount of memory and training data. To tackle this problem, the SCPR-based HMM [31][34] has been proposed.

2.3.1 Recognition by SCPR-based HMM

The SCPR-based on-line handwriting recognition system (Sys_HMM) basically consists of a feature extraction module, a SCPR dictionary mentioned in Section 2.1, SCPR-based HMMs, and a decoder as shown in Figure 9. Note that off-stroke information (vector from the pen-up to the next pen-down) is not necessary when we model pen-coordinate features.

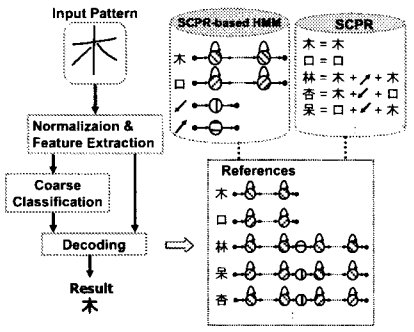


Figure 9: Recognition by SCPR-based HMM.

In the SCPR-based HMMs, the decoder generates HMMs for each character pattern by connecting one or more than one SCPR-based HMMs according to the SCPR dictionary and calculates the probability that an input pattern is produced from the HMMs by the Viterbi algorithm. By doing so, we can handle a large number of character patterns with a small number of HMMs.

2.3.2 Modeling of Pen-coordinate Features by SCPR-based HMM

In the SCPR-based HMMs, the pen-direction feature extracted from consecutive pen-tip positions has been almost always employed [31][34]. In contrast, the pen-coordinate feature has not been employed, though it is no less important than the pen-direction feature. This is due to that the pen-coordinate information is more subject to change when subpatterns are composed into each character pattern.

We then proposed SCPR-based HMMs which model both the pen-direction feature and pen-coordinate feature. The basic idea of our approach is based on the linear mapping and its inverse mapping presented in Section 2.1. As mentioned there, the BS prototypes are reduced to bounding boxes in structural information through a linear mapping when they are included in larger subpatterns or character patterns (Figure 10). In contrast, we apply the inverse of the above mapping when we estimate SCPR-based HMMs. A simple idea is to enlarge the size of the bounding box of a mapped basic subpattern in a learning pattern to the normalization size (128×128 resolution in our system). By applying the inverse mapping, we can exclude character dependency of each subpattern (difference in size and position when it appears in different character patterns) to model pen-coordinate features of the subpattern by SCPR-based HMMs. Based on the idea, we propose adaptation and estimation of SCPR-based HMM parameters as described below.

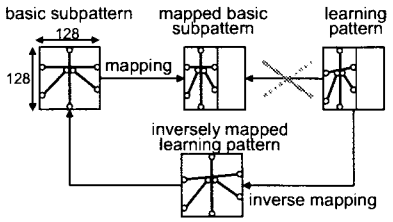


Figure 10: Mapping and inverse mapping.

(1) Adaptation of SCPR-based HMM Parameters

When a SCPR-based HMM is incorporated into a character pattern by mapping it into the bounding box in structural information, the parameters for the pen-coordinate feature ($\boldsymbol{o}_t=(x_t,y_t)$, where t denotes sampling time) are mapped to different values depending on which character pattern and where it is incorporated into.

Note that the observation probability distribution of each SCPR-based HMM is represented by a single Gaussian distribution given by

$$b_i(o_t) = \frac{\exp(-\frac{1}{2}(o_t - \mu_i)'\Sigma_i^{-1}(o_t - \mu_i))}{\sqrt{(2\pi)^n|\Sigma_i|}} \quad (15)$$

with the mean vector μ, the covariance matrix Σ. Here, let $\vec{\mu}_i = (\mu_{ix}, \mu_{iy})$ be the mean vector of the Gaussian distribution at a state q_i of a SCPR-based HMM, an adapted mean vector is given by

$$\hat{\mu}_{ix} = \mu_{ix} \times \frac{w}{128} + sp_x \quad (16), \quad \hat{\mu}_{iy} = \mu_{iy} \times \frac{h}{128} + sp_y \quad (17)$$

where μ_{ix} and μ_{iy} denote the mean vectors of the pen-coordinate feature x and y, and (sp_x, sp_y) denotes the top-left corner and $<w,h>$ denotes $<width, height>$ of the bounding box.

Moreover, we assume that the diagonal covariance matrix $\Sigma_i=(\sigma^2_{ixx}, \sigma^2_{iyy})$ of the Gaussian distribution at each state S_i of a SCPR-based HMM is correlated to the bounding box size of a subpattern and adapt Σ_i to each character pattern according to the correlations. From our analysis based on the database Kuchibue_d, we convert the diagonal covariance matrix: $\Sigma_i=(\sigma^2_{ixx}, \sigma^2_{iyy})$ to $\hat{\Sigma}_i = (\hat{\sigma}^2_{ixx}, \hat{\sigma}^2_{iyy})$ as follows:

$$\hat{\sigma}_{ixx} = \begin{cases} \sigma_{ixx} - 0.085 \times (128 - w) & (\sigma_{ixx} \geq 9.4707) \\ 0.085w + 0.005 & (\sigma_{ixx} < 9.4707) \end{cases} \quad (18)$$

$$\hat{\sigma}_{iyy} = \begin{cases} \sigma_{iyy} - 0.080 \times (128 - h) & (\sigma_{iyy} \geq 8.99614) \\ 0.080h + 0.007 & (\sigma_{iyy} < 8.99614) \end{cases} \quad (19)$$

(2) Estimation of SCPR-based HMMs

Handwriting usually has noises due to hand vibration etc., so that the inverse mapping may magnify these noises and reflect them into the subpattern. We then employ the displacement normalization mentioned in Section 2.2.2, instead of the inverse mapping for each subpattern to estimate SCPR-based HMMs.

We extract every occurrence of a subpattern from all the training character patterns by the Viterbi algorithm. Then, based on the equations: (7) and (8), we convert the pen-coordinate features of each occurrence to those for the normalization size as follows:

$$x_t' = \mu_{ix} + (x_t - \hat{\mu}_{ix}) \times \frac{D_w(128)}{D_w(w)} \quad (20)$$

$$y_t' = \mu_{iy} + (y_t - \hat{\mu}_{iy}) \times \frac{D_h(128)}{D_h(h)} \quad (21)$$

where (x_t, y_t) and (x_t', y_t') are feature points of a pre-normalized subpattern and a normalized subpattern, respectively. Moreover, $\vec{\hat{\mu}}_i = (\hat{\mu}_{ix}, \hat{\mu}_{iy})$ is the mean vector of a state q_i where a feature point (x_t, y_t) is observed.

After converting pen-coordinate features of all the occurrences of a subpattern to those of the normalization size, we update parameters for each state by taking the average of those converted pen-coordinate features.

2.3.3 Evaluation for SCPR-based HMMs

We made experiments to compare the conventional character HMMs and the proposed SCPR-based HMMs with respect to the amount of training patterns to model deformation of strokes. In the experiments, we used only Kanji categories in the JIS 1st level set in Kuchibue_d. Patterns from 60 writers were used for training, and those from the remaining 60 writers were used for test. Figure 11 shows the recognition rates when varying the amount of training patterns. Note that there are 5,632 character patterns per writer.

As the result, the SCPR-based HMMs achieved better recognition performance with a smaller amount of training patterns than the character HMMs. This is because each subpattern appears in many character patterns so that each SCPR-based HMM is trained by a larger number of training patterns than the number of training character patterns This result also shows that there is no big difference between the recognition rate of the SCPR-based HMMs and the character HMMs in case that there is an enough amount of training data.

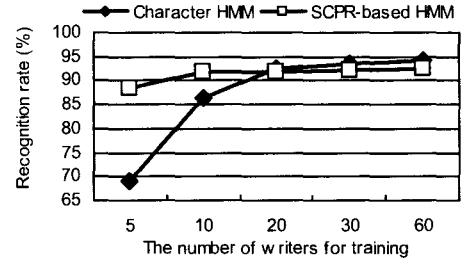


Figure 11: Comparison of character HMM and SCPR-based HMM.

3 Classifier Combination

This section describes the combined recognizer composed of the SCPR-based on-line recognizer and the off-line recognizer.

A on-line handwritten pattern is easily converted to an off-line pattern by discarding temporal information, so that we can apply the off-line method. Although the on-line method is more robust against stroke catenation, running strokes and deformation of character patterns compared to the off-line method, the off-line method is free from stroke order variation and robust to duplicated strokes when people write two or more times. Therefore, by combining the on-line method with the off-line method, the recognition accuracy is improved since they can compensate their disadvantages. Several attempts have been made to combine the on-line method with the off-line method [36][37]. In Japanese character recognition, Tanaka et al. showed the initial first attempt to combine on-line and off-line classifiers [38] while Okamoto et al. showed the combination in the feature level, i.e., added on-line features to off-line features in off-line recognition scheme [20]. It seems that classifier combination is more flexible than feature

combination since we can employ the most suitable classification method for each set of features.

In this paper, we show succeeding research after [38] to improve recognition accuracy while increasing recognition speed and reducing memory size. The memory requirements for the off-line prototype dictionary are significantly larger than that for the on-line prototype dictionary. Especially, the dictionary size depends on the number of categories, so that the combined recognizer for Japanese characters is difficult to use in a small computer such as PDA. We then propose a compact combined recognizer composed of the SCPR-based on-line recognizer and an off-line recognizer whose prototype dictionary size is significantly small.

3.1 Combination Process

In order to combine the on-line recognition and the off-line recognition, a given on-line character pattern is converted to a bitmap image, then on-line recognition and off-line recognition are processed in parallel. We employ Sys_LTM mentioned in Section 2.1 as the on-line recognition method and Modified Quadratic Discriminant Function (MQDF2) [39] as the off-line recognition method.

3.1.1 Combination Rule

There exist various possibilities to combine outputs from multiple classifiers. Kittler et al. present many combination schemes, such product rule, sum rule, min rule, max rule, median rule and majority voting [40]. We employ the sum-rule in which the total score of a combined classifier is the addition of all classifiers. The sum rule is denoted as follows:

$$Assign \quad X \rightarrow C_j \quad if$$
$$\sum_{i=1}^{R} P(C_j \mid fv_i) = \max_{k=1}^{N_{cc}} \sum_{i=1}^{R} P(C_k \mid fv_i) \qquad (22)$$

where N_{cc} denotes the number of character categories, R denotes the number of classifiers, fv denotes a feature vector extracted from the input pattern X, C_k denotes prototype and $P(C_k \mid fv_i)$ denotes the probability that C_k occurs when fv_i is given.

3.1.2 Evaluation Score Normalization

Recognition results produced by each recognizer are pairs of a candidate character and an evaluation score which represents similarity or distance. However, each recognizer outputs a different type of evaluation score. The evaluation score of our on-line recognizer shows similarity. The higher the score is, the more likely the candidate is. On the other hand, by our off-line recognizer the lower the score is, the higher the likelihood is. To combine these recognizers, we apply our likelihood normalization approach [41][42].

3.2 Small Off-line Prototype Dictionary

MQDF2 for the off-line recognition is given as:

$$g_2(\mathbf{x}, \omega_i) = \sum_{j=1}^{m} \frac{1}{\lambda_{ij}} [\varphi_{ij}^T (\mathbf{x} - \boldsymbol{\mu}_i)]^2 \sum_{j=1}^{k} + \frac{1}{\delta} \{ \|\mathbf{x} - \boldsymbol{\mu}_i\|^2 \qquad (23)$$
$$- \sum_{j=1}^{m} [\varphi_{ij}^T (\mathbf{x} - \boldsymbol{\mu}_i)]^2 \} + \sum_{j=1}^{m} \log \lambda_{ij} + (n-m) \log \delta$$

where μ is the mean vector, φ is the eigen vector, λ is eigen value, and δ is a modified eigen vector, n is the number of dimension and m is the number of λ.

While the SCPR dictionary of our on-line recognizer is only 150KB, the prototype dictionary of the off-line recognizer is about 90MB. To reduce the total size of memory, we propose a small prototype dictionary for the off-line recognizer by reducing parameters for MQDF2.

The size of the off-line prototype dictionary: Si is calculated from the size of each parameter $\{s_\mu, s_\varphi, s_\lambda, s_\delta\}$ where s_x denotes the size for the parameter x as follows.

$$Si = N_{cc} \times \{ n \times (s_\mu + m \times s_\varphi) + m \times s_\lambda + s_\delta \} \qquad (24)$$

In this study, we make two extreme sizes of dictionaries. One is 9.7MB whose n is 100 and m is 10. The other is 91.8MB whose n is 256 and m is 40. Each parameter $\{s_\mu, s_\varphi, s_\lambda, s_\delta\}$ requires 16 bits in both the dictionaries. Since the size of SCPR dictionary in the on-line recognizer is 150KB, the dictionary size of the combined recognizer is almost the same as that of the off-line recognizer. Hereafter, we call the combined recognizer which employs the 9.7MB dictionary "Sys_9.7MB", and the 91.8MB dictionary "Sys_91.8MB".

3.3 Evaluation of the Combined Recognizer

We trained the on-line recognizer using Nakayosi_t, and the off-line recognizer using ETL9B [43] written by 200 participants, each composed of 3,036 character patterns, JEITA-HP [44] written by 580 participants, each composed of 3,306 character patterns, NTT-AT [45] written by 51 participants, each composed of 1,237 character patterns, and Nakayosi_t. We also used Nakayosi_t for normalizing evaluation scores.

The recognition rates are shown in Table 1. These rates show that there is no big difference between the Sys_9.7MB and the Sys_91.8MB though the correct recognition accuracy of the off-line recognizer with 91.8MB dictionary is higher than that of the off-line recognizer with Sys_9.7MB by 3.7 point.

Table 1: Recognition rates of the combined recognizer (%).

On-line	Off-line		Combined	
	9.7MB	91.8MB	9.7MB	91.8MB
87.2	83.2	86.9	91.4	92.2

We also compare the processing time of each recognizer on a Pentium IV 3.06 GHz processor with 512MB RAM (Table 2). The Sys_9.7MB outperforms the Sys_91.8MB.

Table 2: Average processing time per character (ms).

On-line	Off-line		Combined	
	9.7MB	91.8MB	9.7MB	91.8MB
3.32	6.6	18.5	10.6	22.5

3.4 Combined Recognizer with Context Postprocessing

We also show the recognition accuracy of our combined recognizer with context postprocessing. In this experiment, a character bi-gram model is employed as a simple stochastic language model. Given a sequence of character patterns $\boldsymbol{X}=X_1X_2...X_i...X_N$, The problem is to find the character string $\boldsymbol{C}=C_1C_2...C_i...C_N$ to maximize the probability $P(\boldsymbol{C} \mid \boldsymbol{X})$. Using the Bayes rule:

$$P(\boldsymbol{C} \mid \boldsymbol{X}) = \frac{P(\boldsymbol{C}) \cdot P(\boldsymbol{X} \mid \boldsymbol{C})}{P(\boldsymbol{X})} \qquad (25)$$

The term $P(\boldsymbol{X} \mid \boldsymbol{C})$ shows the probability that $\boldsymbol{C}$ is written as $\boldsymbol{X}$. The term $P(\boldsymbol{C})$ shows the context likelihood.

Following the bi-gram model, the probability $P(\boldsymbol{C})$ is given as:

$$P(C) = P(C_1)\prod_{i=1}^{N-1} P(C_{i+1} \mid C_i) \qquad (26)$$

where N denotes the number of character patterns in a text string and C_i denotes each character pattern. The uni-gram probability $\boldsymbol{P}(C_1)$ is assumed to be independent from characters. In our study, the character bi-gram language model was trained with the ASAHI newspaper text corpus 'CD-HIASK'93'.

The recognition results are shown in Table 3. These results show that the correct recognition rate of Sys_9.7MB is equal to that of Sys_91.8MB and is raized up to 98.6%.

Table 3: Recognition rates of the combined recognizer employing bi-gram model (%).

On-line	Off-line		Combined	
	9.7MB	91.8MB	9.7MB	91.8MB
91.2	97.0	97.7	98.6	98.6

4 On-line Handwritten Japanese Text Recognition

According to the increasing size of writing surface of pen input devices, demand for on-line handwritten text recognition is getting higher. Due to the difference between Japanese and western languages and handwriting, handwritten recognition differs naturally. In this section, we describe some problems of handwritten Japanese text recognition and approaches to tackle those problems.

4.1 Problems of On-line Handwritten Japanese Text Recognition

Generally, on large writing surfaces Oriental languages of Chinese origin are often written horizontally, vertically or even slantingly in a mixed way.

Most of the previous publications and systems have been assuming only horizontal lines of text [46][47] while we have been trying to relinquish any writing constraint from on-line text input. We proposed a method to recognize mixtures of horizontal, vertical and slanting lines of text with assuming normal character orientation [48]. Then, we attempted handwriting recognition even with characters rotated like handwritings often made on whiteboards [49].

As mentioned before, Japanese text includes various sizes of character patterns ranging from so-called "half-width" characters like numbers and symbols, Kana characters and Kanji characters of only one radical in the middle to those consisting of multiple radicals. Moreover, handwriting even magnifies the size variations as shown in Figure 12. Some characters may be several times longer and/or wider than others.

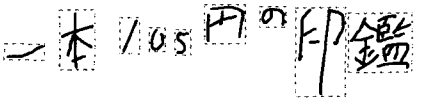


Figure 12: An example of Japanese handwritten text.

There are many characters in the Japanese character set that can be divided into multiple character patterns. For example, the patterns shown in Figure 13(a) can be read as either C_1, a character in itself, or as the two consecutive characters C_2C_3. Which of the two is correct is determined by the characters (or strings) proceeding and/or following it. In the example of Figure 13(b), the character C_4 follows, which causes the pattern of Figure 13(a) to be read as C_1. In Figure 13(c), on the other hand, the characters C_5C_6 follow, which causes the pattern to be read as two characters C_2C_3. This example shows that the position of character segmentation can be different even for the same handwritten pattern depending on the context and it is therefore difficult to segment characters deterministically on the basis of geometrical features alone.

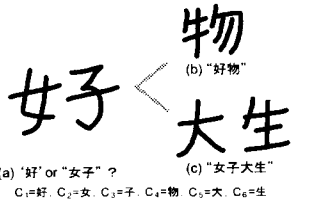


Figure 13: An example of segmentation ambiguity.

In the next section, we present an enhanced method to recognize on-line handwriting of arbitrary line directions and character orientations as well as their mixtures.

4.2 Flow of Processing

Here, we first define some terminologies. Character orientation is used to specify the direction of a character from its top to bottom while line direction is used to designate the writing direction of a sequence of characters until it changes (Figure 14). A text line is a piece of text separated by new-line or large space and it is further divided into text line elements at the changing points of line direction. Each text line element has its line direction (Figure 15). The line direction and the character orientation are independent.

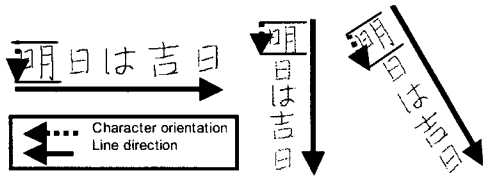


Figure 14: Line direction and character orientation.

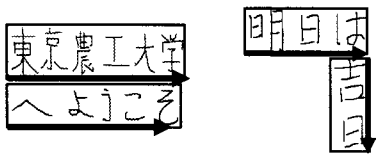


Figure 15: Text line element and line direction.

In the following subsections, we explain the procedure of our on-line recognition system of handwritten Japanese text, which is free from character orientation, line direction and any writing format constraint.

4.2.1 Separation of Handwriting into Text Line Elements

First, we estimate the average character size of from all the strokes written on a tablet by measuring the length of the longer side of the bounding box for each stroke, sorting the lengths from all the strokes and taking the average of the larger 1/3 of them. The estimated average character size is used to decide the threshold for separating written text into text line elements.

Next, we separate freely written text into text lines by a large off-stroke from a previous line to a new line. Then, we separate each text line into text line elements by the changing points of line direction.

In order to detect changing points of line direction, we employ a recursive procedure similar to that to detect corner points [50]. Among a series of coordinates of the centers for the bounding boxes of strokes forming a handwritten text line, it finds the most distant point (MDP) from the straight line connecting the starting point and ending point of the series of coordinates, and if the distance is larger than the threshold then apply the same procedure to the straight line from the starting point to the MDP and that from the MDP to the ending point with the result of detecting multiple points of directional change as shown in Figure 16 where B is the beginning point and E is the ending point. Thus a text line segmented by large space is further segmented into text line elements having different line directions.

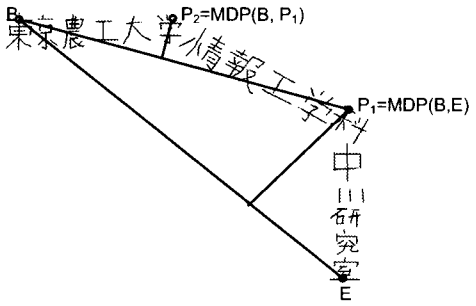


Figure 16: Detection of directional changing points.

Here, it is worth noting that points detected might be within character patterns rather than between characters as the P_1 and P_2 of Figure 16. We will treat the problem in the Section 4.2.4 and determine the best segmentation points while recognizing handwritten text.

4.2.2 Estimation and Normalization of Character Orientation

When Japanese characters are written, principal pen movement within real strokes is the same as the character orientation or $\pi/2$ counter clockwise to it. This is because Japanese characters, especially Kanji characters, are composed of downward and rightward strokes. Because of this, if we take the histogram of displacement direction of pen-tip coordinates, we will see two peaks as shown in Figure 17.

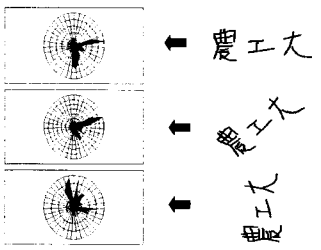


Figure 17: Two peaks in pen movement direction.

Therefore, let us assume the intensity of the histogram at the angle θ as $f(\theta)$. Then, take the θ that

makes the max value of $f(\theta)\times f(\theta+\pi/2)$ as the character orientation for each text line element. In order to make the peak detection more robust, we take convolution of $f(\theta)$ and a Gauss function to blur the peak so that it works for slanting characters that have rightward strokes with slightly upward inclination.

After the character orientation is estimated, the text line element can be normalized by rotating it until their character orientation become downward.

4.2.3 Hypothetical Segmentation for Each Text Line Element

We hypothetically segment a text line element after the character orientation normalization into character patterns using geometric features.

Hypothetical segmentation depends also on character orientation and line direction. After character orientation is normalized, it depends on line direction of a text line element as shown in Figure 18. Note that segmentation hypothesis is often made within character pattern and it is different even for the same character pattern depending on the line direction. The quantization can be finer but the 4-directional quantization shown in Figure 18 is adequate and effective to prevent a text line element from being segmented excessively. When the line direction is classified into, say, downward or upward (rightward or leftward), a considerable gap projected on the vertical axis (the horizontal axis) or a long off-stroke to the quantized line direction is employed as a candidate for segmentation but strokes or off-strokes to the opposite direction are used to merge its crossing strokes with the result that hypotheses on segmentation can be decreased, which is then effective to speed up the text recognition and to increase the recognition rate.

After the quantization of line direction, we extract multi-dimensional features such as distance and overlap between adjacent strokes from each off-stroke and apply the SVM to the extracted features to produces segmentation point candidates [51]. Since the character size may vary among text line elements, we estimate the character size again for every text line element. Then, we normalize the extracted multi–dimensional features by the re-estimated character size for each text line element. Each off-stroke is classified into segmentation point, non-segmentation point and undecided point according to the features. A segmentation point should be between two characters while a non-segmentation point is within a character pattern. An undecided point is a point where segmentation or non-segmentation judgment cannot be made. A segmentation unit bounded by two adjacent segmentation points is assumed as a character pattern. An undecided point is treated as two ways of a segmentation point or a non-segmentation point. When it is treated as a segmentation point, it is used to extract a segmentation unit.

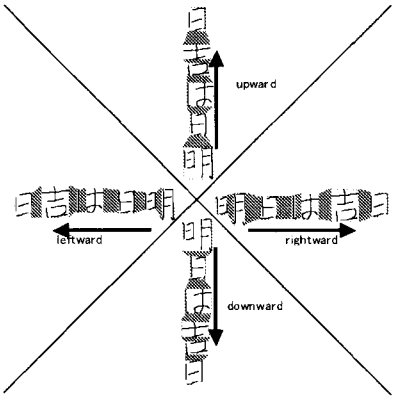


Figure 18: Quantization of line direction.

4.2.4 Construction of Candidate Lattice and Segmentation and Recognition

A candidate lattice is constructed for each text line where each arc denotes segmentation point and each node denotes a character recognition candidate produced by character recognition for each segmentation unit as shown in Figure 19.

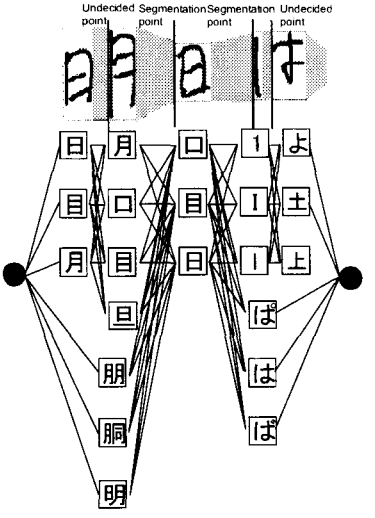


Figure 19: Candidate lattice.

However, for the strokes between the immediately preceding segmentation point and the immediately succeeding segmentation point of the changing points of line direction as them of '学' and '科' at the P_1 and P_2 shown in Figure 16, we treat them as two character orientations of the preceding text line element and the succeeding text line element to construct the candidate lattice.

Scores are associated to each arc or node following the stochastic model evaluating the likelihood composed of character pattern structure, character segmentation, character recognition and context.

The Viterbi search is made into the candidate lattice for a handwritten text line and the best segmentation and recognition is determined.

4.2.5 Model of Free Format Recognition

We made a model and recently formalized it for on-line handwritten Japanese text recognition free from line-direction constraint and writing format constraint such as character writing boxes or ruled lines [42][52]. The model evaluates the likelihood composed of character segmentation, character recognition, character pattern structure and context. The likelihood of character pattern structure considers the plausible height, width and inner gaps within a character pattern that appear in Chinese characters composed of multiple subpatterns.

The problem is to find the character string $\boldsymbol{C}=C_1C_2...C_i...C_N$ to maximize the likelihood $L(\boldsymbol{C}|\boldsymbol{X})$ that a handwritten text line pattern $\boldsymbol{X}$ is recognized as the character string $\boldsymbol{C}$. After several steps of approximations and modifications, we get the following formula:

$$L(C \mid X) = \log P(C_1) + \sum_{i=1}^{N-1} \log P(C_{i+1} \mid C_i) + \sum_{i=1}^{N} \left(\log P(X_i \mid St_i, C_i) + \log P(St_i / \overline{C} \mid C_i) \right) + \sum_{i=1}^{N-1} \left(\log P(gap_i / \overline{C} \mid C_i, C_{i+1}) \right) \quad (27)$$

Where,

N	:	number of characters in C.
$P(C_{i+1} \mid C_i)$	:	probability that a character C_{i+1} follows C_i (bi-gram probability).
$P(X_i \mid St_i, C_i)$	:	probability that a character C_i is written in a structure St_i and represented by the stroke sequence X_i.
$\overline{C}$	:	average size of the character sequence $\boldsymbol{C}$.
$P(St_i / \overline{C} \mid C_i)$	:	probability that a character C_i is written in a structure St_i.
$P(gap_i / \overline{C} \mid C_i, C_{i+1})$	:	probability that an outer gap : gap_i appears between C_i and C_{i+1}.

In the right-hand side of the above equation, the second term considers context likelihood in terms of bi-gram, the third term is related to character recognition likelihood, the forth term and fifth term evaluates character pattern structure likelihood and outer gap likelihood, respectively.

5 Application

In this section, we show two types of on-line handwriting applications: one is segmentation and recognition of mixed text, formulas, tables and line-drawings, the other is handwritten text search.

5.1 Segmentation and Recognition of Mixed Objects

As a writing area of pen input devices becomes large, users can easily write text, mathematical formulas and figures on the screen. It is one of the most important benefits of pen interfaces that people can write these objects by a single pen without switching the device, mode, software or whatever else and without any writing restriction such as grids or boxes. However, it requires the difficult task to separate on-line handwritten patterns into Japanese text, figures and mathematical formulas. We tried this problem very early [53]. Recently, we take a probabilistic approach for this problem and employ stroke features, stroke crossings and stroke densities. Moreover, we partially apply the approach of segmentation by recognition. Although the current recognizer for formulas is not a true recognizer, we have achieved about 81% correct segmentation for all the strokes in Kondate_t which is our newly prepared database of mixed patterns [54]. Figure 20 shows an example of separating mixed objects. Our approach is generally better but less effective in distinguishing figures from other components.

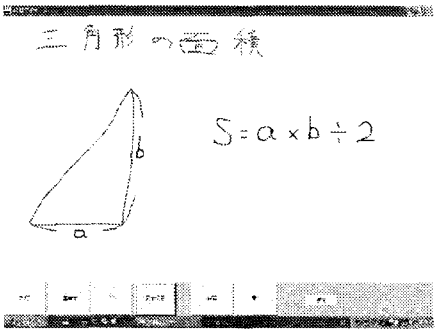


Figure 20: Separation of handwriting into text, formula and line drawing.

5.2 On-line Handwritten Text Search

As various pen input devices become popular, on-line handwritten text will be accumulated. Without a search method, however, accumulated on-line handwritten text will not be utilized effectively. Search of on-line handwritten text by employing pattern matching method without character recognition was reported in [55]. Lopresti et al. proposed stroke search method "Script Search Algorithm", which searches through a long handwritten text pattern and find approximate patterns of a pattern given as a keyword [56].

On the other hand, we have proposed a method for writing-box-free on-line Japanese handwritten text search which is based on the on-line Japanese handwritten text recognition as mentioned in Section 4 [57]. It searches for a target keyword in the candidate

lattice composed of candidate segmentations and candidate characters as shown in Figure 19 which has been generated beforehand by the background process of on-line handwriting recognition. Figure 21 shows an example of on-line handwritten text search. As the performance of recognizer is improved, the performance of search will be upgraded.

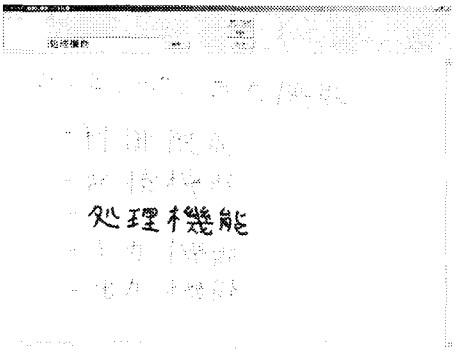


Figure 21: Searching a keyword from free format handwritten text.

6 Conclusion

We presented on-line handwriting recognition of Japanese characters. Since most of Kanji character patterns are composed of multiple subpatterns called radicals and these subpatterns are shared among many character patterns, we have employed structured character pattern representation (SCPR). SCPR is effective in terms of the size reduction of the prototype dictionary and the robustness to deformation of common subpatterns. Then, we described a prototype learning algorithm and HMM-based recognition for SCPR. We combined the SCPR-based on-line recognizer with a compact off-line recognizer. Moreover, we also presented on-line handwritten Japanese text recognition free from character orientation and line direction constraints. Finally, we showed some applications of on-line handwritten Japanese text recognition.

Acknowledgements

This research is being partially supported by Grant-in-Aid for Scientific Research under the contract number (B) 17300031 and the MEXT fund for Promoting Research on Symbiotic Information Technology.

References

[1] http://www.anoto.com

[2] http://www.e-pen.com

[3] C. C. Tappert et al., "The State of the Art in On-Line Handwriting Recognition," *IEEE, Trans. PAMI*, **12**, 8, pp.787-808, 1990.

[4] M. Nakagawa, "Non-Keyboard Input of Japanese Text—On-Line Recognition of Handwritten Characters as the Most Hopeful Approach," *J. Information Processing*, **13**, 1 pp.15-34, 1990.

[5] R. Plamondon et al., "On-Line and Off-Line Handwriting Recognition: a Comprehensive Survey," *IEEE Trans. PAMI*, **22**, 1, pp.63-82, 2000.

[6] S. Jaeger et al., "The state of the art in Japanese on-line handwriting recognition compared to techniques in western handwriting recognition," *IJDAR*, **6**, 2, pp.75-88, 2003.

[7] C-L. Liu et al., "On-Line Recognition of Chinese Characters: the State of the Art," *IEEE Trans. PAMI*, **26**, 2, pp.198-213, 2004.

[8] M. Nakagawa et al., "On-line handwritten character pattern database sampled in a sequence of sentences without any writing instructions," *Proc. 4th ICDAR*, 1, pp.376-381, 1997.

[9] M. Nakagawa et al., "Collection of on-line handwritten Japanese character pattern databases and their analysis," *IJDAR*, **7**, 1, pp.69-81. 2004.

[10] J. Shin et al., "Stroke Correspondence Search Method for Stroke-Order and Stroke-Number Free On-Line Character Recognition – Multilayer Cube Search (in Japanese)," *IEICE Trans.* **J82-D-II**, 2, pp.230-239, 1999.

[11] M. Kobayashi et al., "RAV(Reparameterized Angular Variations) Algorithms for Online Handwriting Recognition," *IJDAR*, **3**, 3, pp.181-191, 2001.

[12] M. Nakagawa et al., "Robust and highly customizable recognition of on-line handwritten Japanese characters," *Proc. of 13th ICPR*, **3**, pp.269-273, 1996.

[13] M. Nakagawa et al., "A Linear-Time Elastic Matching for Stroke Number Free Recognition of On-Line Handwritten Characters," *Proc. 4th IWFHR*, pp.48-56, 1994.

[14] T. Yokota et al., "An On-line Cuneiform Modeled Handwritten Japanese Character Recognition Method Free from Both the Number and Order of Character Strokes (in Japanese)," *Trans. IPSJ*, **44**, 3, pp.980-990, 2003.

[15] K. Takahashi et al., "A Fast HMM Algorithm for On-Line Handwritten Character Recognition," *Proc.4th ICDAR*, pp.369-375, 1997.

[16] K. Ishigaki et al., "A Top-down On-line Handwritten Character Recognition Method via the Denotation of Variation," *Proc. of ICCPCOL*, pp. 141-145, 1998.

[17] U. Rammer, "An Iterative Procedure for the Polygonal Approximation of Plane Closed Curves," *CGIP*, **1**, pp. 244-256, 1972.

[18] A. Kawamura et al., "On-line recognition of freely handwritten Japanese characters using directional feature densities," *Proc. 11th ICPR*, **2**, pp.183-186, 1992.

[19] M. Hamanaka et al., "On-line Japanese character recognition experiments by an off-line method," *Proc.*

2nd ICDAR, pp.204-207, 1993.

[20] M. Okamoto et al., "On-line Handwritten Character Representation using Directional Features and Direction-Change Features (in Japanese)," *Journal of IEE Japan*, **119**, 3, pp. 358-366, 1999.

[21] J. Tokuno et al., "Pen-Coordinate Information Modeling by SCPR-based HMM for On-line Japanese Handwriting Recognition," *Proc. 18th ICPR*, **3**, 2006.

[22] T. Kohonen, "Improved versions of learning vector quantization," Proc. IJCNNI, 1, pp.545-550, 1990.

[23] S. Geva et al., "Adaptive nearest neighbor pattern recognition," *IEEE Trans. NN*, **2**, 2, pp.318-322, 1990.

[24] B-H. Juang et al., "Discriminative learning for minimization error classification," *IEEE Trans. SP* 40, 12, pp.3043-3054, 1992.

[25] A. Sato et al., "A formulation of learning vector quantization using a new misclassification measure," *Proc. 14th ICPR*, pp 322-325, 1998.

[26] C-L. Liu et al., "Evaluation of prototype learning algorithms for nearest-neighbor classifier in application to handwritten character recognition," *PR*, 34, pp.601-615, 2001.

[27] A. Kitadai et al., "A Learning Algorithm for Structured Character Pattern Representation Used in On-line Recognitions of Handwritten Japanese Characters," *Proc. 8th IWFHR*, pp 163.168, 2002.

[28] C-L. Liu et al., "Preprocessing and Statistical /Structural Feature Extraction for Handwritten Numeral Recognition Process of Handwriting Recognition," *World Scientific*, pp.161-168, 1997.

[29] J. Hu et al., "HMM based on-line handwriting recognition," *IEEE Trans. PAMI*, **18**, 10, pp.1039-1045, 1996.

[30] T. Starner et al., "On-Line Cursive Handwriting Recognition Using Speech Recognition Methods," *Proc. ICASSP*, 5, pp.125-128, 1994.

[31] H. J. Kim et al., "On-line recognition of handwritten Chinese characters based on hidden Markov models," *Pattern Recognition*, **30**, 9, pp.1489-1499, 1997.

[32] H. Ito et al., "An on-line handwritten character recognition method based on hidden Markov model (in Japanese)," *Technical Report of IEICE, PRMU97-85*, pp.95-100, 1997.

[33] D. Okumura et al., "An HMM implementation for on-line handwriting recognition based on pen-coordinate feature and pen-direction feature," *Proc. 8th ICDAR*, pp.26-30, 2005.

[34] M. Nakai et al., "Sub-stroke approach to HMM-based on-line Kanji handwriting recognition," *Proc. 6th ICDAR*, pp.491-495, 2001.

[35] S-J. Cho et al., "Bayesian Network Modeling of Hangul Characters for On-line Handwriting Recognition," *Proc. 7th ICDAR*, pp.207-211, 2003.

[36] S. Manke et al., "Combining bitmaps with dynamic writing information for on-line handwriting recognition," *Proc. 13th ICPR*, pp.596-598, 1994.

[37] A. Vinciarelli et al., "Combining Online and Offline Handwriting Recognition," *Proc. 7th ICDAR*, pp.844-848, 2003.

[38] H. Tanaka et al., "Hybrid Pen-Input Character Recognition System Based on Integration of Online-Offline Recognition," *Proc. 5th ICDAR*, pp.209-212, 1999.

[39] F. Kimura et al., "Modified Quadratic Discriminant Functions and the Application to Chinese Character Recognition," *IEEE Trans. PAMI*, **9**, 1, pp.149-153, 1987.

[40] J. Kittler et al., "On combining classifiers," *IEEE Trans. PAMI*, **20**, 3, pp.222-239, 1998.

[41] O. Velek et al., "A New Warping Technique for Normalizing Likelihood of Multiple Classifiers and its Effectiveness in Combined On-Line/Off-Line Japanese Character Recognition," *Proc. 8th IWFHR*, pp.177-182, 2002.

[42] M. Nakagawa et al., "A Model of On-line Handwritten Japanese Text Recognition Free from Line Direction and Writing Format Constraints," *IEICE Trans.* **E88-D**, 8, pp.1815-1822, 2005.

[43] T. Saito et al., "On the Data Base ETL9 of Handprinted Characters in JIS Chinese Characters and Its Analysis," *Trans. IEICE*, **J68-D**, 4, pp.757-764, 1985.

[44] T. Kawatani, "Handwritten Kanji recognition with determinant normalized quadratic discriminant function," *Proc. 15th ICPR*, **2**, pp.343-346, 2000.

[45] http://www.ntt-at.com/products_e/jwords/

[46] T. Fukushima et al., "On-line Writing-box-free Recognition of Handwritten Japanese Text Considering Character Size Variations," *Proc. of 15th ICPR*, **2**, pp.359-363, 2000.

[47] S. Senda et al., "A Maximum-Likelihood Approach to Segmentation-Based Recognition of Unconstrained Handwriting Text," *Proc. of 6th ICDAR*, pp.184-188, 2001.

[48] Y. Inamura et al., "An On-line Writing-box-free and Writing-direction Free Recognition System for Handwritten Japanese Text, (in Japanese)," *Technical Report of IEICE, PRMU100-37*, pp.17-24, 2000.

[49] M. Nakagawa et al., "On-line Handwritten Japanese Text Recognition Free from Constrains on Line Direction and Character Orientation," *Proc. 7th ICDAR*, pp.519-523, 2003.

[50] S. Senda et al., "Box-free online character recognition integrating confidence values of segmentation, recognition and language processing (in Japanese)," *Technical Report of IEICE, PRMU98-138*, 1998.

[51] B. Zhu et al., "Segmentation of On-Line Handwritten Japanese Text Using SVM for Improving Text Recognition," *Proc. 7th Int'l Workshop on DAS*, pp. 208-219, 2006.

[52] M. Nakagawa et al., "A Formalization of On-line Handwritten Japanese Text Recognition free from Line Direction Constraint," *Proc. 17th ICPR*, 2004.

[53] K. Machii et al., "On-line Text/Drawings Segmentation of Handwritten Patterns," *Proc.2nd*

ICDAR, pp.710-713, 1993.

[54] K. Mochida et al., "Separating Figures, Mathematical Formulas and Japanese Text from Free Handwriting in Mixed On-Line Documents," *IJPRAI*, **18**, 7, pp.1173-1187, 2004.

[55] S. Senda et al., "MemoPad: Software with functions of Box-free Japanese Character Recognition and Handwritten Query Search (in Japanese)," *Technical Report of IEICE, PRMU99-75*, pp.85-90, 1999.

[56] D. Lopresti et al., "On the searchability of electronic ink," *Proc. of 4th IWFHR*, pp.156-165, 1994.

[57] H. Oda et al., "A search method for on-line handwritten text employing writing-box-free handwriting recognition," *Proc. 9th IWFHR*, pp.545-550, 2004.

Segmentation-driven offline handwritten Chinese and Arabic script recognition

Xiaoqing Ding and Hailong Liu
Dept. of Electronic Engineering, Tsinghua University,
State Key Laboratory of Intelligent Technology and Systems
100084, Beijing, China
dxq,lhl@ocrserv.ee.tsinghua.edu.cn

Abstract

The market of handwriting recognition applications is increasing rapidly due to continuous advancement in OCR technology. This paper mainly summarizes our recent efforts on offline handwritten Chinese script recognition using a segmentation-driven approach. Two essential problems are addressed in the paper, namely isolated character recognition and establishment of the probabilistic segmentation model. To improve the isolated character recognition accuracy, we propose a heteroscedastic linear discriminant analysis algorithm to extract more discrimination information from original character features, and implement a minimum classification error learning scheme to optimize classifier parameters. In the segmentation stage, information from three different sources, namely geometrical layout, character recognition confidence, and semantic model are integrated together into a probabilistic framework to give the best script interpretation. Recognition experiments on postal addresses and bank checks have demonstrated the effectiveness of our proposed algorithms. Over 80% correct address recognition rate is achieved on about 1,000 real handwritten Chinese mails, and the recognition reliability of bank checks is also largely improved after combining the extra courtesy recognition result with the legal amount recognition result. Some of our preliminary research on Arabic script recognition is also discussed in the paper.

1. Introduction

Research on handwritten script recognition has received more and more attentions in recent years, since it meets with the demands from a wide range of commercial applications, for example automatic postal address reading, bank check processing, recognition of handwritten contents in forms, etc. There are different ways to categorize handwritten script recognition. Depending on how the handwriting is acquired and converted to digital form, the research field can be distinguished as online and offline recognition. In the online case, dynamic time information captured from the writing device is utilized to increase the recognition accuracy, while in the offline case, such information is unavailable and the recognition accuracy is usually much lower. According to the language, the scripts to be recognized can be specified as Roman, Asian, Arabic etc, which can be very different in recognition strategy according to their respective characteristic.

In this paper, we mainly focus our attentions on the problem of offline handwritten Chinese script recognition. Compared with Roman script recognition [1] [2] [3] [4], there has been relatively less research work done in this area, most of the published paper concerns with segmentation problems [5] [6] [7] [8] [9] [10], post-processing [11], and specific applications [12] [13] [14] [15] [16].

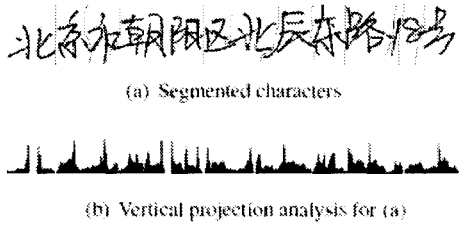
The Chinese handwritten script recognition is a very challenging problem due to the following reasons:

1) There exists great variety in the styles of handwritten scripts.

2) Accurate segmentation is very difficult under many situations. The gaps between characters are often small, sometimes adjacent characters even touch or overlap with each other. (Fig. 1)

3) Misclassifications on characters often occur, especially when the script is cursive.

4) The unique feature of Chinese characters poses extra difficulties: The pattern class number is extremely large (several thousands); many Chinese characters have very complex structures; there also exist many similar characters.



(a) Segmented characters

(b) Vertical projection analysis for (a)

Fig. 1. Typical handwritten Chinese script with touching and overlapping

Generally speaking, script recognition approaches can be divided into two categories, namely holistic approaches and segmentation-driven approaches. The former ones are more often used in western script recognition, where words appear as basic units, and treated as single entities and recognized as a whole. The size of lexicon (legal vocabulary) that can be handled by holistic recognition is usually limited. When the lexicon size is too large, or there exist no extra gaps between words e.g. in the case of Chinese scripts, the segmentation-driven approaches are more favorable. In this situation, scripts are first segmented into a sequence of isolated units, either characters or parts of characters (radicals or graphemes), and then these units are recognized separately to give the final script recognition result. In our work, the segmentation-driven approach is adopted.

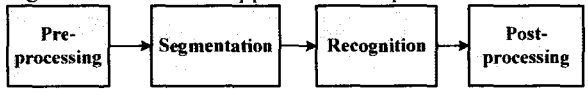


Fig. 2. Sequential script recognition procedure

A sequential script recognition scheme is illustrated in Fig.2. After a script image is acquired, some necessary pre-processing steps such as binarization, noise removal and slant correction are carried out first. Then the script image are segmented into individual characters and recognized by isolated character recognition engine one by one. In the end of the procedure, context based post-processing is used to correct the errors that occur in the recognition step. The deficiency of this sequential segmentation-then-recognition structure is that errors will be accumulated in each step. Incorrect segmentation result will lead to recognition error, and could not be corrected even by post-processing.

To overcome this deficiency, a global optimization scheme is proposed to replace the sequential structure in our script recognition system. After an over-segmentation procedure, a probabilistic segmentation framework is established, which takes account of information from geometrical layout, isolated character recognition results and contextual constrains. Multiple paths corresponding to different segmentation and recognition results are evaluated, and the best one is accepted as the final script interpretation. The flowchart of our script recognition system is illustrated in Fig.3. In this system two problems are essential: the design of a high performance isolated character recognizer, and the probabilistic segmentation model.

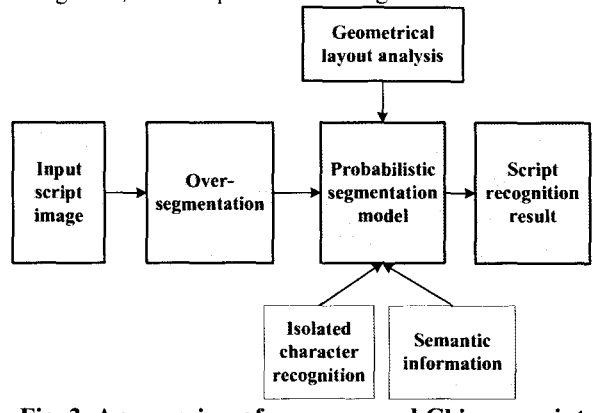


Fig. 3. An overview of our proposed Chinese script recognition system

The remainder of the paper is organized as follows: in Section 2 we discuss the algorithms for isolated character recognition; Section 3 deals with the over-segmentation procedure; in Section 4 we establish the probabilistic segmentation framework; experimental results on mail address and bank check recognition are then followed in Section 5. In Section 6 we present our work on Arabic script recognition. Finally, conclusions and future work are summarized in Section 7.

2. Isolated handwritten character recognition

Isolated character recognition plays a fundamental and crucial role in the whole script recognition system. In the past decades, huge amount of character recognition algorithms including preprocessing, feature extraction and classification have been reported [17] [18]. In our recognition engine, line-density nonlinear normalization proposed by Yamada [19] is first carried out on character images to reduce the stroke deformation. After that, high resolution gradient feature is extracted [20] [21], and modified quadratic discriminant function (MQDF) [22] is used as the classifier. To further enhance the discriminative ability of the recognition engine, we proposed a novel heteroscedastic linear discriminant analysis (HLDA) method to gain more discriminative information from the original feature vector [23], and apply a minimum classification error (MCE) learning scheme [21] to adjust the MQDF parameters estimated by the maximum likelihood method.

2.1. Heteroscedastic linear discriminant analysis

As we know, dimensionality reduction or specifically feature extraction is very important in pattern recognition. It is necessary not only for reducing computational cost, but also for dealing with the "curse of dimensionality" when the number of training samples is relatively small compared with feature dimensionality. From the viewpoint of classification, feature extraction should preserve as much class-separability as possible in the lower dimensional space. Of all the supervised feature extraction techniques, linear discriminant analysis (LDA) probably is the most frequently used one, which aims at maximizing the between-class scatter $\boldsymbol{S}_B$ while minimizing the within-class scatter $\boldsymbol{S}_W$. Although LDA is efficient and simple to implement, it also has its own deficiencies. Since LDA implicitly assumes that all the pattern classes are Gaussian distributed and have equal covariance, it can extract discriminative information only from the difference of class mean, while totally ignoring the discriminative information lying in the difference of class covariance. Therefore when the practical feature data is heteroscedastic, LDA cannot perform in an optimal way. Following Loog's work [24] [25], we have proposed a heteroscedastic linear discriminant analysis (HLDA) algorithm to generalize LDA to heteroscedastic feature data [23].

For simplicity, we first assume the within-class matrix equals to the identity matrix, i.e. $\boldsymbol{S}_W=\boldsymbol{I}$. The between-class scatter matrix $\boldsymbol{S}_B$ can be decomposed as

$$\boldsymbol{S}_B = \sum_{i=1}^{C-1}\sum_{j=i+1}^{C} p_i p_j \boldsymbol{S}_{Eij}. \tag{1}$$

In which

$$\boldsymbol{S}_{Eij} = (\boldsymbol{m}_i - \boldsymbol{m}_j)(\boldsymbol{m}_i - \boldsymbol{m}_j)^T \tag{2}$$

C is the pattern class number, p_i and $\boldsymbol{m}_i$ are the prior probability and mean vector of class i, respectively. S_{Eij} is called as the Directed Distance Matrix (DDM) associated with class i and j, since it not only gives the distance between two classes by its eigenvalues, but also implies the directions in which the distance could be found by its eigenvectors. Under the homoscedastic Gaussian assumption of LDA, the corresponding $\boldsymbol{S}_{Eij}$ has only one nonzero eigenvalue, which equals to the Euclidean distance

$$d_{Eij} = (\boldsymbol{m}_i - \boldsymbol{m}_j)^T(\boldsymbol{m}_i - \boldsymbol{m}_j). \tag{3}$$

$\boldsymbol{S}_{Eij}$ and d_{Eij} is related by $tr(S_{Eij})=d_{Eij}$.

To find the optimal transformation matrix Φ, Fisher criterion is used, which can also be represented by coupling the pairwise-class seperability criterions,

$$\begin{aligned}\boldsymbol{J}_F(\Phi) &= tr[(\Phi^T\boldsymbol{S}_W\Phi)^{-1}(\Phi^T\boldsymbol{S}_B\Phi)] \\ &= \sum_{i=1}^{C-1}\sum_{j=i+1}^{C} tr[(\Phi^T\Phi)^{-1}(\Phi^T\boldsymbol{S}_{Eij}\Phi)].\end{aligned} \tag{4}$$

Under the homoscedastic assumption, Euclidean distance is all we need to measure the dissimilarity between two Gaussian distributions. If we want to further consider the heteroscedastic situation, Chernoff distance should be used to replace Euclidean distance:

$$d_{Cij} = (\boldsymbol{m}_i - \boldsymbol{m}_j)\Sigma_{ij}^{-1}(\boldsymbol{m}_i - \boldsymbol{m}_j) + \frac{1}{\alpha(1-\alpha)}\log\frac{|\Sigma_{ij}|}{|\Sigma_i|^{\alpha}|\Sigma_j|^{1-\alpha}} \tag{5}$$

Where Σ_i and Σ_j are the covariance matrixes of class i and j, α is a constant determined by $\alpha = p_i/(p_i + p_j)$, and $\Sigma_{ij} = p_i\Sigma_i + p_j\Sigma_j$. The DDM corresponding to Chernoff distance can be derived as

$$\begin{aligned}\boldsymbol{S}_{Cij} &= \Sigma_{ij}^{-1/2}(\boldsymbol{m}_i - \boldsymbol{m}_j)(\boldsymbol{m}_i - \boldsymbol{m}_j)^T\Sigma_{ij}^{-1/2} \\ &+ \frac{1}{\alpha(1-\alpha)}\left(\log\Sigma_{ij} - \alpha\log\Sigma_i - (1-\alpha)\log\Sigma_j\right),\end{aligned} \tag{6}$$

since $tr(\boldsymbol{S}_{Cij})=d_{Cij}$. Replace $\boldsymbol{S}_{Eij}$ in (4) with $\boldsymbol{S}_{Cij}$, we can get the Chernoff criterion

$$\boldsymbol{J}_C(\Phi) = \sum_{i=1}^{C-1}\sum_{j=i+1}^{C} p_i p_j tr[(\Phi^T\Phi)^{-1}(\Phi^T\boldsymbol{S}_{Cij}\Phi)] \tag{7}$$

When the number of pattern classes is too large, e.g. in handwritten Chinese character recognition, the pairwise-class calculation scheme can be computationally too expensive. Also, the logarithm items in (6) are computationally unstable. Therefore we may discard the logarithm item and use the global mean vector $\boldsymbol{m}_0$ to avoid pairwise-class calculation, thus (7) is simplified as

$$\boldsymbol{J}_M(\Phi) = \sum_{i=1}^{C} p_i tr[(\Phi^T\Phi)^{-1}(\Phi^T\boldsymbol{S}_{Mi}\Phi)] \tag{8}$$

Where

$$\boldsymbol{S}_{Mi} = \Sigma_{i0}^{-1/2}(\boldsymbol{m}_i - \boldsymbol{m}_0)(\boldsymbol{m}_i - \boldsymbol{m}_0)^T\Sigma_{i0}^{-1/2} \tag{9}$$

And

$$\Sigma_{i0} = p_i\Sigma_i + \frac{1}{C}\boldsymbol{S}_W = p_i\Sigma_i + \frac{1}{C}\boldsymbol{I} \tag{10}$$

Notice that $\boldsymbol{S}_{Mi}$ corresponds to Mahalanobis distance $d_{Mi} = (\boldsymbol{m}_i - \boldsymbol{m}_0)^T\Sigma_{i0}^{-1}(\boldsymbol{m}_i - \boldsymbol{m}_0)$ since $tr(\boldsymbol{S}_{Mi})= d_{Mi}$. $\boldsymbol{J}_M(\Phi)$ is the new Mahalanobis criterion that we propose, and the heteroscedastic LDA algorithm based on this criterion is called M-HLDA.

The above discussion is proceeded under the assumption that $\boldsymbol{S}_W=\boldsymbol{I}$. If not, we can always first perform a whiten transformation $\boldsymbol{y}=\boldsymbol{S}_w^{-1/2}\boldsymbol{x}$ in the original feature space. After getting the optimal transformation Φ in the whiten feature space with the Mahalanobis criterion, we can transform back to the original feature space by an inverse whiten

transformation $\boldsymbol{x}=S_w^{1/2}\boldsymbol{y}$. The Mahalanobis criterion in the original feature space can be represented as

$$\boldsymbol{J}_M(\Phi)=\sum_{i=1}^{C} p_i tr[(\Phi^T \boldsymbol{S}_W^{1/2}\boldsymbol{S}_W^{1/2}\Phi)^{-1}(\Phi^T \boldsymbol{S}_W^{1/2}\boldsymbol{S}'_{Mi}\boldsymbol{S}_W^{1/2}\Phi)] \quad (11)$$

Where $\boldsymbol{S}'_{Mi}$ is the DDM matrix in the whiten feature space. The M-HLDA solution finally comes down to finding the eigenvectors corresponding to the largest d eigenvalues of matrix $\sum_{i=1}^{C} p_i \boldsymbol{S}_W^{-1}\times(\boldsymbol{S}_W^{1/2}\boldsymbol{S}'_{Mi}\boldsymbol{S}_W^{1/2})$. In this way, the simplicity of LDA solution is still retained, only a generalized eigenvalue problem needs to be solved when calculating the optimal M-HLDA feature transformation matrix.

We have applied the proposed M-HLDA feature extraction algorithm on the HCL2000 handwritten Chinese character database [20]. A 10% drop on the misclassification rate over conventional LDA was achieved.

2.2. Discriminative training on parameters of Modified quadratic discriminant function

In the compressed d-dimensional feature space, the MQDF distance for an unknown pattern $\boldsymbol{x}$ can be derived under Gaussian distribution assumption:

$$h_i(\boldsymbol{x})=\frac{1}{\sigma^2}\{\|\boldsymbol{x}-\boldsymbol{\mu}_i\|^2-\sum_{j=1}^{k}(1-\frac{\sigma^2}{\lambda_{ij}})[\boldsymbol{\varphi}_{ij}^T(\boldsymbol{x}-\boldsymbol{\mu}_i)]^2\} \quad (12)$$
$$+\sum_{j=1}^{k}\log\lambda_{ij}+(d-k)\log\sigma^2 \qquad i=1,2...C$$

And the decision rule is

$$C(x)=\arg\min_i h_i(x) \quad (13)$$

Where $\boldsymbol{\mu}_i$ and $\boldsymbol{\Sigma}_i$ denote the mean and covariance of class i, λ_{ij} and $\boldsymbol{\varphi}_{ij}$ denote the j-th eigenvalue (in descending order) and the corresponding eigenvector of $\boldsymbol{\Sigma}_i$, $k(k<d)$ is the number of dominant principal axes, and σ^2 is a constant to compensate for the estimation error on small eigenvalues caused by the lack of training samples.

The parameters of MQDF classifier are estimated by the maximum likelihood (ML) method. As we know, ML estimation aims at fitting the assumed probabilistic model, it is not optimum in terms of classification accuracy. Therefore, in this paper we apply a minimum classification error (MCE) training scheme to adjust the MQDF parameters. MCE training algorithm is originally proposed by Katagiri and Juang [26] [27], and later introduced in character recognition field to couple with different classifier forms [28][29][30].

To apply MCE training, we first reform the representation of MQDF distance as

$$h_i(\boldsymbol{x})=\sum_{j=1}^{d}\left(e^{-\tau_{ij}}\boldsymbol{\varphi}_{ij}^T\boldsymbol{x}-m_{ij}\right)^2+2\sum_{j=1}^{d}\tau_{ij}+H \quad (14)$$

Where H is a discrimination irrelevant constant to ensure $h(x)>0$. Meanwhile some necessary parameter transformations are implemented as

$$\tau_{ij}=\begin{cases}(\log\lambda_{ij})/2 & j\le k\\ \log\sigma & j>k\end{cases},\quad m_{ij}=e^{\tau_{ij}}\boldsymbol{\varphi}_{ij}^T\boldsymbol{\mu}_i \quad (15)$$

Then the MQDF parameters can be denoted as $\boldsymbol{\Theta}=\{\boldsymbol{\theta}_i\}=\{m_{ij},\tau_{ij},\boldsymbol{\varphi}_{ij}\}\quad i=1,2...C, j=1,2...d$

Assume that $\boldsymbol{x}_n$ is a training pattern from class m, the misclassification measure on $\boldsymbol{x}_n$ is defined by

$$d(\boldsymbol{x}_n,\boldsymbol{\Theta})=h_m(\boldsymbol{x}_n,\boldsymbol{\theta}_m)-\bar{h}_m(\boldsymbol{x}_n,\boldsymbol{\Theta},\eta) \quad (16)$$

The latter term on right side of (17) is a collective representation of all the rival classes

$$\bar{h}_m(\boldsymbol{x}_n,\boldsymbol{\Theta},\eta)=\left(\frac{1}{C-1}\sum_{i=1,i\ne m}^{C}h_i(\boldsymbol{x}_n,\boldsymbol{\theta}_i)^{-\eta}\right)^{-\frac{1}{\eta}}\ (\eta>0) \quad (17)$$

Embedding $d(\boldsymbol{x}_n, \boldsymbol{\Theta})$ into a sigmoid function, we get a continuous loss function $s(x_n, \Theta)$ with respect to Θ.

$$s(\boldsymbol{x}_n,\boldsymbol{\Theta})=\frac{1}{1+e^{-\gamma d(\boldsymbol{x}_n,\boldsymbol{\Theta})}}\quad(\gamma>0) \quad (18)$$

When η and γ approach infinity, $s(x_n, \Theta)$ becomes the simple 0-1 loss. The empirical loss on the whole training set $X=\{\boldsymbol{x}_1, \boldsymbol{x}_2,\ldots \boldsymbol{x}_N\}$ is the summarization of the individual loss

$$S(\boldsymbol{X},\boldsymbol{\Theta})=\sum_{n=1}^{N}s(\boldsymbol{x}_n,\boldsymbol{\Theta}) \quad (19)$$

$S(\boldsymbol{X},\boldsymbol{\Theta})$ is then minimized using a generalized probability descent (GPD) algorithm[23]. The ML estimation is taken as the initial $\boldsymbol{\Theta}$, and gradually better estimation can be obtained by an iterative scheme

$$\boldsymbol{\Theta}_{t+1}=\boldsymbol{\Theta}_t-\varepsilon_t\nabla s(\boldsymbol{x}_t,\boldsymbol{\Theta}_t) \quad (20)$$

According to derivation rules,

$$\frac{\partial s(\boldsymbol{x}_n,\boldsymbol{\Theta})}{\partial\boldsymbol{\theta}_i}=\frac{\partial s(\boldsymbol{x}_n,\boldsymbol{\Theta})}{\partial d(\boldsymbol{x}_n,\boldsymbol{\Theta})}\frac{\partial d(\boldsymbol{x}_n,\boldsymbol{\Theta})}{\partial h_i(\boldsymbol{x}_n,\boldsymbol{\theta}_i)}\frac{\partial h_i(\boldsymbol{x}_n,\boldsymbol{\theta}_i)}{\partial\boldsymbol{\theta}_i} \quad (21)$$

The three terms on right side of (21) are calculated as (22), (23), (24), respectively.

$$\frac{\partial s(\boldsymbol{x},\boldsymbol{\Theta})}{\partial d(\boldsymbol{x},\boldsymbol{\Theta})}=\gamma s(\boldsymbol{x}_n,\boldsymbol{\Theta})\left(1-s(\boldsymbol{x}_n,\boldsymbol{\Theta})\right) \quad (22)$$

$$\frac{\partial d(\boldsymbol{x}_n,\boldsymbol{\Theta})}{\partial h_i(\boldsymbol{x}_n,\boldsymbol{\theta}_i)}=\begin{cases}\frac{1}{C-1}\left(\frac{\bar{h}_m(\boldsymbol{x}_n,\boldsymbol{\Theta},\eta)}{h_i(\boldsymbol{x}_n,\boldsymbol{\theta}_i)}\right)^{\eta+1} & i\ne m\\ -1 & i=m\end{cases} \quad (23)$$

$$\frac{\partial h_i(\boldsymbol{x}_n,\boldsymbol{\theta}_i)}{\partial m_{ij}}=-2\left(e^{-\tau_{ij}}\boldsymbol{\varphi}_{ij}^T\boldsymbol{x}-m_{ij}\right)$$
$$\frac{\partial h_i(\boldsymbol{x}_n,\boldsymbol{\theta}_i)}{\partial\tau_{ij}}=-2e^{-\tau_{ij}}\boldsymbol{\varphi}_{ij}^T\boldsymbol{x}\left(e^{-\tau_{ij}}\boldsymbol{\varphi}_{ij}^T\boldsymbol{x}-m_{ij}\right)+2 \quad (24)$$

Theoretically the eigenvectors φ_{ij} can also be optimized by GPD algorithm, but it will take extra efforts to retain the orthogonal and normal constraints on them, therefore only m_{ij} and τ_{ij} are updated in our training procedure, and φ_{ij} is left unchanged. The learning rate ε_t in (20) is set as $\varepsilon_t = \varepsilon_0(1 - t/N)$, the initial learning rate ε_0, as well as the parameters η and γ in the classification loss function are all determined by experiments.

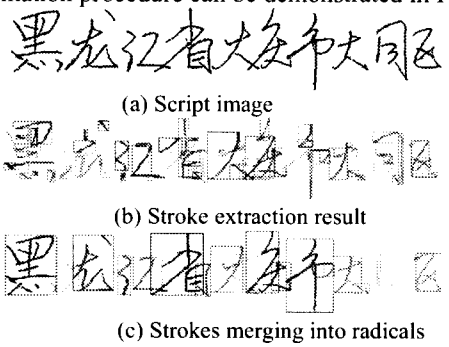
By using MCE training and other discrimination schemes, we have achieved very good performance on Chinese character recognition [18]. On ETL9B database, a 99.33% correct rate is achieved on the test set, on HCL2000 database a 99.56% correct rate is achieved on the test set.

3. Over-segmentation

Since it is extremely difficult to segment scripts into isolated characters without any prior knowledge, we adopt an over-segmentation then merging strategy in our script recognition system.

The goal of over-segmentation is to cut script image into a radical sequence. Each extracted radical should belong to only one character, thus correct segmentation can be realized by selecting a correct merging path of radicals. Ordinary segmentation algorithms, including projection histogram analysis and connected component analysis cannot deal with the case when adjacent characters touch with each other. Therefore, we attempt to extract the elemental unit in the Chinese characters i.e. stroke. Black pixel run-length analysis is implemented on the script image, run-lengths that are close and approximately in the same direction are tracked and form stroke. More details of the stroke extraction algorithm can be found in [31] [32].

Each stroke is contained within a rectangle called as bounding box. According to the extent of overlap between adjacent stroke bounding boxes, we can iteratively merge them into radicals. The whole over-segmentation procedure can be demonstrated in Fig. 4.



(a) Script image

(b) Stroke extraction result

(c) Strokes merging into radicals

Fig. 4. Over-segmentation procedure

4. Probabilistic framework for segmentation model

After the radical sequence is obtained from the script image, each possible radical merging path then corresponds to a segmentation result and thus produces a specific character image sequence. The segmentation is then converted to an optimal merging path selection problem. Obviously the geometrical layout have an important impact on deciding which path should be selected, and the isolated character recognition result, as well as semantic information also influence the segmentation results, therefore in our work, these information are all integrated in a probabilistic model.

Let $\boldsymbol{E}$ denote a radical sequence $(e_1, e_2,\ldots,e_N)$ given by over-segmentation, $\boldsymbol{S}$ denote a path which indicates merging $(e_1, e_2,\ldots,e_N)$ into a character image sequence $(I_1, I_2,\ldots,I_M)$ $(M \leqslant N)$, and $\boldsymbol{O}$ denote a interpretation that $(I_1, I_2,\ldots,I_M)$ is recognized as $(c_1, c_2,\ldots,c_M)$.

$$\begin{aligned} \boldsymbol{E} &= (e_1, e_2, \ldots, e_N) \\ \boldsymbol{S} &= (I_1, I_2, \ldots, I_M) \\ \boldsymbol{O} &= (c_1, c_2, \ldots, c_M). \end{aligned} \tag{25}$$

Then what we should do after over-segmentation is to find the optimal segmentation and interpretation $(\boldsymbol{S}^*, \boldsymbol{O}^*)$ given the radical serial $\boldsymbol{E}$, according to a maximum a posterior (MAP) criterion.

$$(\boldsymbol{S}^*, \boldsymbol{O}^*) = \arg\max_{S,O} P\{\boldsymbol{S}, \boldsymbol{O} \mid \boldsymbol{E}\} \tag{26}$$

Following the Bayesian formula,

$$\begin{aligned} P\{\boldsymbol{S}, \boldsymbol{O} \mid \boldsymbol{E}\} &= P\{\boldsymbol{S} \mid \boldsymbol{E}\} P\{\boldsymbol{O} \mid \boldsymbol{E}, \boldsymbol{S}\} \\ &= P\{\boldsymbol{S} \mid \boldsymbol{E}\} P\{\boldsymbol{O} \mid \boldsymbol{S}\} \end{aligned} \tag{27}$$

In the above equation, the item $P(\boldsymbol{S}|\boldsymbol{E})$ can be seen as a geometrical layout confidence, while the item $P(\boldsymbol{O}|\boldsymbol{S})$ can be seen as a string recognition confidence. We then discuss the representation of these two confidences in detail respectively.

4.1. Geometrical layout confidence

The geometrical layout confidence measures the probability of a certain merging way of radical series purely by geometrical shape analysis. This confidence is not associated to the character recognition results and semantic information.

In our system, the following three kinds of geometrical feature are evaluated (Fig. 5)

1) Distances between the centers of adjacent character images, which are denoted by d_i, i=1,2,...M-1

2) Widths of each character image, which are denoted by w_i, i=1,2,...,M

3) Ratios of height and width of each character image, we denote them as $r_i=h_i/w$, i=1,2,...,M

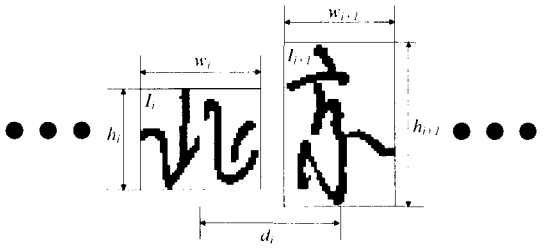


Fig. 5. Geometrical features extracted from the character image sequence

Other geometrical features can also be considered [12], while in our work we only use the above three features.

Treat these features as random variable and assume they come from Gaussian *i.i.d.* distribution. The mean and covariance of the distribution can thus be estimated on a large training set by maximum likelihood method. Since the writing styles of scripts vary significantly from case to case, the estimated parameter will not be stable, even after certain kinds of normalization are used. So we directly compute mean and variance inside a particular script image, thus the result will be adaptive to the writing style to a certain extent.

The joint probabilities of distance, width and ratio variables are calculated as

$$P(d_1,d_2,...d_{M-1}) = \prod_{i=1}^{M-1} \frac{1}{\sqrt{2\pi}\sigma_d} \exp\{-\frac{(d_i-\mu_d)^2}{2\sigma_d^2}\} \quad (28)$$

$$P(w_1,w_2,...w_M) = \prod_{i=1}^{M} \frac{1}{\sqrt{2\pi}\sigma_w} \exp\{-\frac{(w_i-\mu_w)^2}{2\sigma_w^2}\} \quad (29)$$

$$P(r_1,r_2,...r_M) = \prod_{i=1}^{M} \frac{1}{\sqrt{2\pi}\sigma_r} \exp\{-\frac{(r_i-\mu_r)^2}{2\sigma_r^2}\} \quad (30)$$

Thus, the geometrical layout confidence of the radical merging path $\boldsymbol{S}$ could be defined as

$$\begin{aligned} P(\boldsymbol{S} \mid \boldsymbol{E}) &= P(I_1,I_2,...I_M \mid e_1,e_2,...e_N) \\ &\propto P(d_1,d_2,...d_{M-1})P(w_1,w_2,...w_M)P(r_1,r_2,...r_M) \end{aligned} \quad (31)$$

4.2. Character string recognition confidence

Once a segmentation path $\boldsymbol{S}$ is decided, radicals can be merged into character images accordingly, and recognized by the isolated character classifier. For each input character image, the classifier outputs K candidates, together with the corresponding distances in an increasing order. Since misclassification often happens, the first candidate is not assured to be correct, thus the following work is to select a best character string interpretation from the candidate sets.

In the unconstrained or open vocabulary situation, N-gram model is often used, which has been introduced to natural language processing for a long time, and widely used in speech recognition and OCR post-processing [11].

According to Bayesian rule,

$$\begin{aligned} P(\boldsymbol{O} \mid \boldsymbol{S}) &= P(c_1,c_2,...c_M \mid I_1,I_2,...I_M) \\ &= \frac{P(I_1,I_2,...I_M \mid c_1,c_2,...c_M)P(c_1,c_2,...c_M)}{P(I_1,I_2,...I_M)} \end{aligned} \quad (32)$$

If we only considers the transition probability between two adjacent characters, N-gram model becomes Bi-gram, thus we have

$$P(c_1,c_2,...c_M) = P(c_1) \times \prod_{i=1}^{M-1} P(c_{i+1} \mid c_i) \quad (33)$$

Since the recognition results for different character images are independent, we can derive

$$\begin{aligned} P(I_1,I_2,...I_M \mid c_1,c_2,...c_M) &= \prod_{i=1}^{M} P(I_i \mid c_i) \\ &= \prod_{i=1}^{M} \frac{P(I_i)}{P(c_i)} \prod_{i=1}^{M} P(c_i \mid I_i) \end{aligned} \quad (34)$$

$P(c_i)$ is a prior probability, which can be assumed equal for each class, $P(I_i)$ can also be treated as a constant under fixed segmentation path.

$$\begin{aligned} P(\boldsymbol{O} \mid \boldsymbol{S}) &= P(c_1,c_2,...c_M \mid I_1,I_2,...I_M) \\ &= \frac{P(I_1,I_2,...I_M \mid c_1,c_2,...c_M)P(c_1,c_2,...c_M)}{P(I_1,I_2,...I_M)} \end{aligned} \quad (35)$$

$P(c_i|I_i)$ is the confidence for recognizing image I_i as c_i, it can be calculated by the output distances of the isolated character recognizer [33].

$$P(c_i \mid I_i) = \frac{\exp(-d_{k(c_i)}/\theta)}{\sum_{k=1}^{K} \exp(-d_k/\theta)} \quad (36)$$

Where d_k is the character recognition distance of the k^{th} candidate and $k(c_i)$ denotes the candidate position of c_i in the classifier outputs. θ is an estimated variance.

4.3. Optimization method

Taking a logarithm operation on (25)

$$\begin{aligned} (\boldsymbol{S}^*,\boldsymbol{O}^*) &= \arg\max_{S,O} \log P\{\boldsymbol{S},\boldsymbol{O} \mid \boldsymbol{E}\} \\ &= \arg\max_{S,O} \{\log P\{\boldsymbol{S} \mid \boldsymbol{E}\} + \log P\{\boldsymbol{O} \mid \boldsymbol{S}\}\} \end{aligned} \quad (37)$$

Optimizing the objective function directly could be computational too expensive, so we exploit a three-step approximate optimization scheme in our work.

1) In the first step, only the first part of the objective function i.e. the geometrical layout confidence $\log P(\boldsymbol{S}|\boldsymbol{E})$ is maximized, using a dynamic programming algorithm, we can get L candidate segmentation paths $\boldsymbol{S}_1$, $\boldsymbol{S}_2$...$\boldsymbol{S}_L$ (L=100~200) which have the largest L layout confidences $\log P(\boldsymbol{S}_l|\boldsymbol{E})$, l=1,2,...L. Other segmentation paths are considered unlikely and eliminated.

2) In the second step, the recognition confidence $\log P(\boldsymbol{O}|\boldsymbol{S})$ is maximized given a fixed segmentation path $\boldsymbol{S}_l$.

$$\boldsymbol{O}_l^* = \arg\max_{o} \log P(\boldsymbol{O} \mid \boldsymbol{S}_l), l = 1,2...L \quad (38)$$

A HMM model is used to formulize this procedure. Character images I_1, I_2,...I_M correspond to the observations in the HMM model, while the inherent classes c_1, c_2 ,...,c_M correspond to the states in the HMM model. The initial probabilistic distribution of states $\pi=\{\pi_i\}, 1\le i\le C$, and the transition probabilities between adjacent states $A^1=\{a^1_{ij}\}$, $1\le i,j\le C$, can both be estimated from the training data. The conditional probability matrix $\boldsymbol{B}=\{P(I/c)\}$, which represents the occurrence of observation image I given the state c can be calculated from the outputs of the isolated character recognizer.

Neglecting the subscript l in (38), the optimal class interpretation $(c_1^*,c_2^*,\ldots,c_M^*)$ given a character image sequence $(I_1,I_2,\ldots I_M)$ can be found by a Viterbi searching algorithm.

$$\begin{aligned}(c_1^*,c_2^*,\ldots c_M^*) &= \arg\max_{c_1,c_2,\ldots c_M} P(c_1,c_2,\ldots c_M \mid I_1,I_2,\ldots I_M) \\ &= \arg\max_{c_1,c_2,\ldots c_M} P(c_1)\times\prod_{i=1}^{M-1} P(c_{i+1}\mid c_i)\prod_{i=1}^{M} P(c_i \mid I_i)\end{aligned} \quad (39)$$

3) Finally in the third step, the two confidences are summed up, and the optimal segmentation path as well as the class interpretation $(\boldsymbol{S}^*,\boldsymbol{O}^*)$ is obtained by

$$(\boldsymbol{S}^*,\boldsymbol{O}^*) = \arg\max_{1\le l\le L}\left\{\log P(\boldsymbol{S}_l \mid \boldsymbol{E}) + \log P(\boldsymbol{O}_l^* \mid \boldsymbol{S}_l)\right\} \quad (40)$$

5. Applications

The handwritten Chinese script recognition algorithm discussed above can be used in a wide range of applications. In this paper we investigate on two typical applications, namely postal address reading and legal amount recognition on bank check.

5.1 Postal address reading

Automatic mail sorting machines have been used in China for years, however these machines can only recognize zip-codes on the envelope, the recognition rate is not satisfactory and zip-codes could not provide enough information about the details of the address. To provide higher grade of postal automation and better delivery service, the Chinese address scripts on the envelopes should also be well recognized.

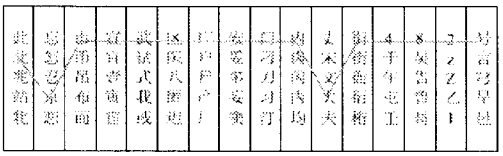
We extracted address scripts on 946 real envelopes from Beijing postal office and tested our recognition algorithm on them. The Bi-gram model is trained using a database (lexicon) containing over 100,000 Beijing address items. As is shown in Table 1, the total address recognition accuracy on character level is 87.2%, errors come from two sources, isolated character recognition error(3.1%) and segmentation error (9.7%). For address interpretation, script recognition is not the last step. The lexicon is employed again to match the recognition result, which can further reduce possible errors. In our experiment, the matching accuracies of various types of address are all over 80% [35]. Further efforts aiming at better utilizing lexicon in segmentation and recognition procedure are ongoing [40].

Table 1. Postal address recognition performance

Address item samples	946
Total character number	12,891
Segmentation error rate	9.7%
Isolated character recognition error rate	3.1%
Address recognition rate (character level)	87.2%
Address recognition rate (lexicon matching)	>80%

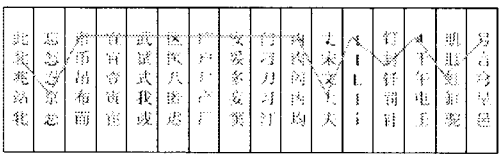
Fig. 6 illustrates the selection of segmentation path as well as character recognition candidates. Since the result represented by Fig. 6-a and 6-b corresponds to a higher probability than all the other alternatives, for example the one represented by Fig. 6-c and 6-d, it is accepted as the final address segmentation and interpretation result.

(a) Character image series merged according to merging path1



(b) Candidate and optimal recognition results of path1

(c) Character image series merged according to merging path2



(d) Candidate and optimal recognition results of path2

Fig. 6. Example of two different segmentation path and candidate selection results

Some incorrect segmentation and recognition postal address samples are shown in Fig.7. The numerals in address scripts can often cause trouble in segmentation, since the geometrical features of them are quite inconsistent with the surrounding Chinese characters. This problem will be further investigated in future work.

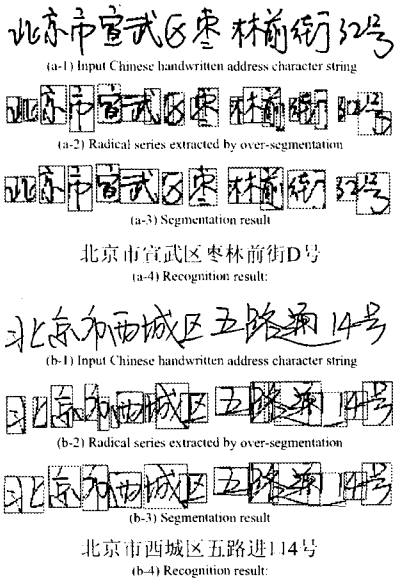
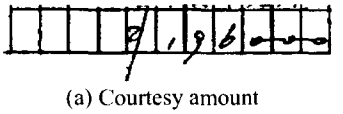


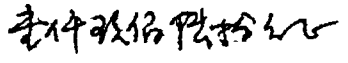
Fig. 7. Some error results of address recognition

5.2 Legal amount reading on bank check

On a bank check image, there usually exist two kinds of amount: courtesy amount and legal amount (Fig. 8). A good check reading system should be able to utilize the redundancy between them. Our aim is to recognize legal amounts using the proposed script recognition algorithm, and cross-validate the results with the recognition results of courtesy amounts, thus the check recognition errors could be efficiently reduced.



(a) Courtesy amount



(b) Legal amount

Fig. 8. Example of courtesy amount and legal amount on Chinese bank check

Compared with postal address reading, in the check reading application we only need to deal with a much smaller lexicon, which is favorable for recognition. The Chinese characters appearing in legal amounts can be categorized into three types: numerals (e.g. '壹'), units (e.g. '万') and termination tags ('正' or '整'). The transition between these characters should satisfy certain constraints: for example, the amount must start with a numerical character, and end with a termination tag; unit characters in one amount item must appear in the descend order, etc. The major transition rules between characters in legal amounts can be illustrated by Fig. 9. These constraints can actually be helpful in the learning of the Bi-gram model, since we are not able to acquire a large amount of check samples for training use.

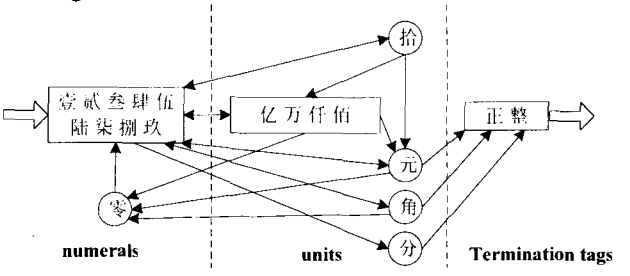


Fig. 9. Transition rules for characters in legal amount

Fig.10 is an illustrative example of the candidate selection procedure by HMM model. The correct legal amount should be '壹万陆仟元正' while the first candidate sequence output by the isolated character recognizer is '壹万陆伍元正', the recognition error is corrected after using the semantic information.

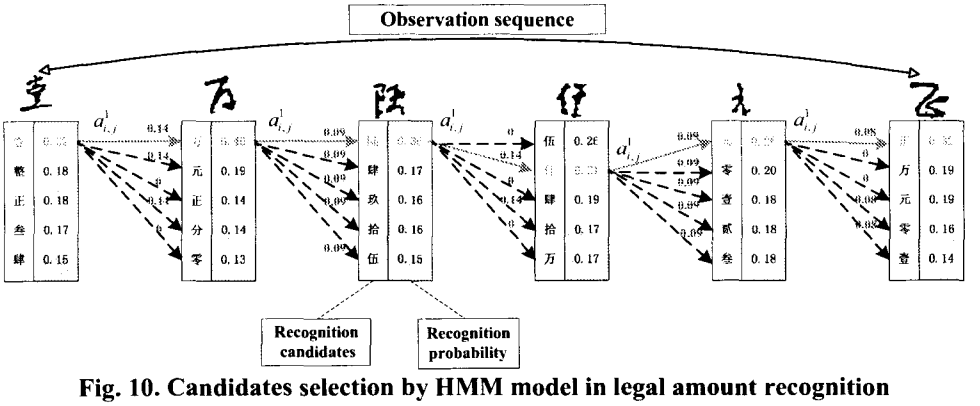


Fig. 10. Candidates selection by HMM model in legal amount recognition

If the probability of the final selected path is lower than a pre-defined threshold, we may consider the result unreliable and reject it. Usually rejection happens when the scanning quality of check image is too low, or the handwriting on the check is too cursive. (Fig. 11)

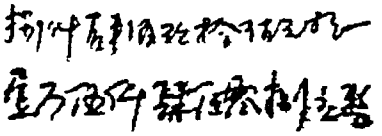


Fig. 11. Examples of rejected legal amount

We collected 2,053 real bank check image samples for our recognition experiments, all of which contain both courtesy amount and legal amount. 1,000 of the images are used as training samples, we use them to train the isolated character recognizer and establish the semantic model. The other 1,053 sample are used for testing.

Three check amount recognition algorithms are compared, namely a courtesy amount recognition (CAR) method, a legal amount recognition (LAR) method which is proposed in this paper, and a fusion algorithm utilizing both CAR and LAR results. The performance of the three algorithms is listed in Table 2. It can be observed that after combing extra legal amount recognition result in CAR, substantially lower rejection rate and higher recognition rate are achieved with an approximately equal error rate, the whole recognition reliability of check amount is largely improved.

Table 2. Check recognition performance comparison of different algorithms

Algorithm	Recognition Rate	Rejection Rate	Error Rate
CAR	45.59%	54.13%	0.28%
LAR	29.06%	70.75%	0.19%
Fusion	66.10%	33.62%	0.28%

6. Arabic script recognition

Due to the increasing interaction between western world and Arabic world, research on Arabic OCR has received more and more attentions in recent years. Since Arabic script has its unique characteristics, it is hard to directly implement the same recognition scheme as in Chinese script recognition. Several most prominent features that are closely relevant to Arabic script recognition are listed as follows:

1) There are 28 basic characters in Arabic alphabets, most of which have four different forms depending on their positions in a word.
2) Arabic script is very cursive. Characters in a word are usually connected along a baseline, which brings crucial challenges in designing the segmentation algorithm (Fig. 11)
3) Many Arabic characters have sub (secondary) parts, which are positioned either above or below the main parts of characters (Fig. 11). The relative positions between them also vary a lot. These 'parts' are also called grapheme in Arabic script.

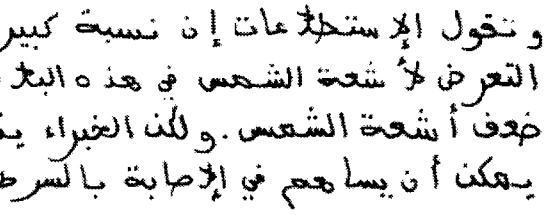


Fig. 11. Typical Arabic script samples

Taking these characteristics into consideration, in the recognition procedure of isolated Arabic character, we actually attempt to recognize graphemes instead of the integer characters. The main part of basic Arabic characters consists of 16 forms: ط, ع, ا, ح, د, ر, س, ى, و, ه, ن, م, ل, ك, ق, ف; the upper subparts consist of 4 forms: ◦ (1-dot), ◦◦ (2-dot), ◦◦◦ (3-dot) and ◦◦◦◦ (hazma); and the lower subparts have 2 forms: ◦ and ◦◦ . If the class label of an Arabic character image I is c, and the main part I^c, upper subpart I^u and lower subpart I^d can be labeled by c^c, c^u and c^d respectively, thus $I=(I^c, I^u, I^d)$, $c=(c^c, c^u, c^d)$. Notice in many cases the sub parts can be a null image, i.e. $c^u=\varnothing$ or $c^d=\varnothing$.

To get the correct segmentation, we must find the corresponding associations between the sub and main parts. Most of the past work did not investigate on this problem, instead they assume that the correct cutting columns on main parts also cut sub parts correctly, and attribute the sub parts to their nearest main parts [36][37]. However, this simple strategy may fail when dealing with the script as showed in Fig. 12. To solve this problem, we proposed a novel 3-queque segmentation model in [38], which considers all the possible combination of sub parts and main parts, and generate the candidate segmentation solutions.

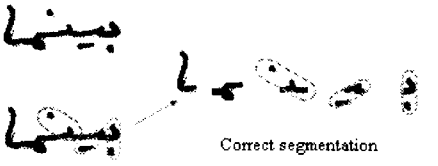


Fig. 12. Arabic script example that sub parts can not be simply associated to its nearest main part

As illustrated in Fig. 13, we first find the main connected component in the script image (13-a), then the baseline can be extracted on it by Hough transform (13-b). After that, a contour analysis is carried on the Arab script, and a series of candidate cutting points are found if they satisfy one of the following descriptions:

1) Local minima points on top contour (13-c),
2) Points on top (bottom) contour whose distance to the contour on the other side is smaller than a preset threshold (e.g. average stroke width) (13-d),
3) Intersection points of top contour and baseline (13-e).

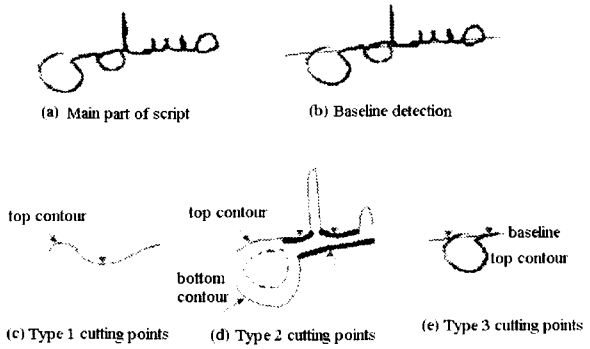


Fig. 13. Candidate cutting point detection on Arabic script

Two adjacent candidate cutting points should at least have some distance between them, therefore some redundant candidate cutting points are filtered out.

According to these candidate cutting points, the main part of Arabic script can be over-segmented into a sequence of graphemes, as illustrated in Fig. 14-b, if we further consider the sequence of the upper parts and lower parts, the whole script can be represented by a 3-queque grapheme sequence (Fig. 14-c). Denote the sequences of main parts, upper and lower sub arts as $(e_1,e_2,\ldots e_{Nc})$, $(e_1^u, e_2^u,\ldots e_{Nu}^u)$ and $(e_1^d, e_2^d,\ldots e_{Nd}^d)$, respectively, all in the left-to-right order. N_c, N_u and N_d are the numbers of graphemes in these three queues.

Assume the 3-quque grapheme sequence should be merged into a M character image sequence $(I_1,I_2,\ldots I_M)$ (Fig. 14-c), the corresponding segmentation path can be visualized in a 3-D space (Fig. 14-e). $X=\{x_0, x_1,\ldots x_M\}$ is a sequence of 3-dimensonal integer vectors which indicate the cutting position, the start position $x_0=\{0,0,0\}$ and the end position $x_M=\{N_c, N_u, N_d\}$.

The geometrical layout confidence calculation for Arabic script is very similar to the case of Chinese script recognition in Section 4.1. On another hand, the semantic information is not used in the probabilistic segmentation model for Arabic script due to the lack of training samples.

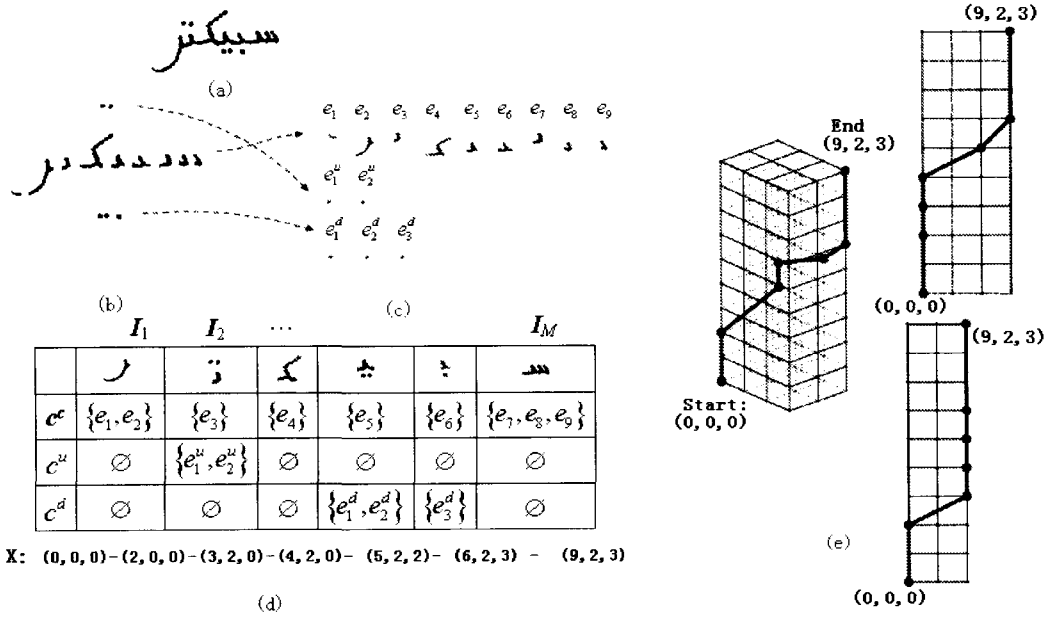


Fig. 14. Example of the cutting mechanism of the 3-queue segmentation model
(a)Arabic script (b) Over-segmentation result (c) 3-queue grapheme sequence
(d) Characters merged by the graphemes (e) Segmentation path displayed in a 3-D space

Given a character image sequence $(I_1,I_2,...I_M)$, the optimal class interpretation $(c_1^*,c_2^*,...,c_M^*)$ can be obtained by

$$(c_1^*,c_2^*,...c_M^*) = \arg\max_{c_1,c_2,...c_M} P(c_1,c_2,...c_M \mid I_1,I_2,...I_M)$$

$$= \arg\max_{c_1,c_2,...c_M} \prod_{i=1}^{M} P(c_i \mid I_i) \quad (41)$$

$$= \arg\max_{(c_i^c,c_i^u,c_i^d),1..M} \prod_{i=1}^{M} I(c_i \mid c_i^c,c_i^u,c_i^d) P(c_i^c \mid I_i^c) P(c_i^u \mid I_i^u) P(c_i^d \mid I_i^d)$$

In (41), items $P(c_i^c|I_i^c)$, $P(c_i^u|I_i^u)$ and $P(c_i^d|I_i^d)$ can be calculated from the distance outputs of the isolated grapheme recognizer, while the item $I(c_i|c_i^c,c_i^u,c_i^d)$ is a variable with value {0,1} reflecting logical constraint. If graphemes c_i^c, c_i^u, c_i^d can actually be combined into a legal Arabic character c_i, we set $I(c_i|c_i^c,c_i^u,c_i^d)=1$, otherwise $I(c_i|c_i^c,c_i^u,c_i^d)=0$. Since there are 16, 5 and 3 kinds of grapheme for main part, upper part and lower part respectively, the number of possible combination cases is $16\times5\times3=240$. However, there are only 28 valid Arabic alphabets, this indicates that logical constraint actually plays an important role in selecting the optimal segmentation path.

The Arabic script recognition algorithm is evaluated on a handwritten Arabic text database containing 20,000 characters. Half of the samples are manually segmented and labeled to train the isolated grapheme recognizer, and the other half is left for testing. The testing samples are divided into five subsets S_1, S_2,...,S_5 according to different writing styles, as illustrated in Fig.15. Each subset contains about 2000 characters. On the test set, we compare the recognition performance of our recognition algorithm with another method presented in [39], which simply associates the sub parts to the nearest main part in segmentation. From Table 3, it can be seen that the average recognition rate on the test set is 59.2% for our algorithm, while using the method in [39] only a 41.3% recognition rate is got. This result indicates the virtue of the proposed 3-queque segmentation model.

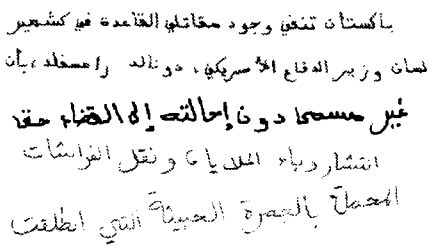


Fig. 15. Arabic script samples of different writing styles

Table 3. Performance comparison on Arabic script recognition accuracy(%)

	S_1	S_2	S_3	S_4	S_5	Ave
Proposed method	69.0	59.4	57.7	54.9	54.9	59.2
Method in [40]	43.1	44.1	46.5	34.6	38.2	41.3

7. Conclusions and future work

In this paper, a segmentation-driven offline handwritten script recognition framework is presented. Feature extraction algorithm based on heteroscedastic linear discriminant analysis is proposed, and classifier optimization method by minimum classification error learning is also presented to improve isolated character recognition accuracy. To get the optimal segmentation and interpretation of script, a probabilistic framework is established to integrate information from three sources: geometrical layout, character recognition confidence and semantic model. Experimental results on multiple applications, including Chinese postal address reading, bank check recognition and Arabic text reading have demonstrated the effectiveness of our algorithms.

Lots of work still remains to be done to further improve the handwritten script recognition accuracy. How to efficiently utilize the domain knowledge in specific applications is one important concern. Arabic script recognition accuracy is still low and needs major improvements for practical use. Due to the lack of Arabic training samples, we has not been able to use semantic information in our Arabic script recognition algorithm, which should be further investigated in the future.

8. Acknowledgements

The authors would like to thank Qiang Fu, Yan Jiang, Pingping Xiu and Junxia Gu for their valuable contributions to this paper. Financial support by National Natural Science Foundation of China (project 60472002) is gratefully acknowledged. The postal address reading project is also funded by Siemens AG under contract number 20030829 - 24022SI202.

9. References

[1] A.W. Senior and A.J. Robinson. An off-line cursive handwriting recognition system. IEEE Trans. PAMI, 20(3): 309-321, 1998

[2] N. Arica and V.F.T. Yarman. An overview of character recognition focused on off-line handwriting. IEEE Trans. on Systems, Man, and Cybernetics—Part C: Applications and Reviews, 31(2): 216-233, 2001

[3] Koerich, AL, R. Sabourin, C.Y. Suen, Large vocabulary off-line handwriting recognition: A survey. Pattern Analysis and Applications, 6(2): 97-121, 2003

[4] H. Bunke, Recognition of cursive Roman handwriting - past, present and future, In Proc. of 7th International Conference on Document Analysis and Recognition, pp. 448–459, 2003

[5] R.G. Casey and E. Lecolinet, A survey of methods and strategies in character segmentation, IEEE Trans. PAMI, 18(7): 690-706, 1996

[6] Y. Lu and M. Shridhar. Character segmentation in handwritten words - An overview, Pattern Recognition, 29(1):77-96, 1996

[7] L.Y. Tseng and R.C. Chen, Segmenting handwritten Chinese characters based on heuristic merging of stroke bounding boxes and dynamic programming, Pattern Recognition Letters, 19(8): 963-973, 1998

[8] Y.H. Tseng and H.J. Lee. Recognition-based handwritten Chinese character segmentation using a probabilistic viterbi algorithm. Pattern Recognition Letters, 20(8): 791–806, 1999

[9] Jiang Gao, Xiaoqing Ding and Youshou Wu, A segmentation algorithm for handwritten Chinese character recognition. In Proc. of 5th International Conference on Document Analysis and Recognition, 1999, pp. 633-636

[10] Shuyan Zhao, Zheru Chi, Pengfei Shi and Hong Yan. Two-stage segmentation of unconstrained handwritten Chinese characters. Pattern Recognition, 36(1): 145-156, 2003

[11] Yuanxiang Li, Xiaoqing Ding, Chew Lim Tan and Changsong Liu. Contextual post-processing based on the confusion matrix in offline handwritten Chinese script recognition. Pattern Recognition, 37(9): 1901-1912, 2004

[12] Junliang Xue and Xiaoqing Ding. Location and interpretation of destination addresses on handwritten Chinese envelopes. Pattern Recognition Letters, 22(6): 639-656, 2001

[13] M.L. Yu, P.C.K. Kwok, C.H. Leung, et al. Segmentation and recognition of Chinese bank check amounts. International Journal on Document Analysis and Recognition, 3(4): 207-217, 2001

[14] Cheng-Lin Liu., M. Koga and H. Fujisawa. Lexicon-driven segmentation and recognition of handwritten character strings for Japanese address reading. IEEE Trans. PAMI, 24(11): 1425-1437, 2002

[15] Yue Lu, Chew Lim Tan, Pengfei Shi and Kehua Zhang. Segmentation of handwritten Chinese characters from destination addresses of mail pieces. International Journal of Pattern Recognition and Artificial Intelligence, 16(1): 85-96, 2002

[16] Hanshen Tang, E. Augustin, C.Y. Suen et al. Recognition of unconstrained legal amounts handwritten on Chinese bank check. In Proc. of 17th International Conference on Pattern Recognition. 2004, pp. 610-613

[17] R. Plamandon and S.N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. IEEE Trans. PAMI. 22(1): 63-84, 2000

[18] C.Y. Suen, S. Mori, S.H. Kim and C.H. Leung, Analysis and recognition of Asian scripts - the state of the art, In Proc. of 7th International Conference on Document Analysis and Recognition, pp. 866-878, 2003

[19] H. Yamada, K. Yamamoto and T. Saito. A nonlinear normalization method for handprinted Kanji character recognition – line density equalization. Pattern Recognition, 23(9): 1023-1029, 1990

[20] Cheng-Lin Liu, K. Nakashima, H. Sako and H. Fujisawa, Handwritten digit recognition: investigation of normalization and feature extraction techniques, Pattern Recognition, 37(2): 265-279, 2004

[21] Hailong Liu and Xiaoqing Ding. Handwritten character recognition using gradient feature and quadratic classifier with multiple discrimination schemes. In Proc. of 8th International Conference on Document Analysis and Recognition, 2005, pp. 19-25

[22] F. Kimura, K. Takashina, S. Tsuruoka and Y. Miyake, Modified quadratic discriminant functions and its application to Chinese character recognition, IEEE Trans. on PAMI, 9(1): 149-153, 1987

[23] Hailong Liu and Xiaoqing Ding. Improve handwritten character recognition performance by Heteroscedastic linear discriminant analysis. In Proc. of 18th International Conference on Pattern Recognition. Vol. 1, pp. 880-883, 2006.

[24] M. Loog, R.P.W. Duin and R. Haeb-Umbach. Multiclass linear dimension reduction by weighted pairwise fisher criteria. IEEE Trans. on PAMI. 23(7): 762–766, 2001.

[25] M. Loog and R.P.W. Duin. Linear dimensionality reduction via a heteroscedastic extension of LDA: the Chernoff criterion. IEEE Trans. on PAMI. 26(6): 732–739, 2004

[26] B.H. Juang and S. Katagiri, Discriminative learning for minimum error classification, IEEE Trans. on Signal Processing, 40(12): 3043-3054, 1992

[27] S. Katagiri, B.H.Juang and C.H. Lee. Pattern recognition using a family of design algorithms based upon the generalized probability descent method. Proceedings of the IEEE, 86(11): 2345-2373, 1998

[28] H. Watanabe and S. Katagiri. Subspace method for minimum error pattern recognition, IEICE Trans. on Information and System, E80-D(12): 1095-1104, 1997

[29] Rui Zhang, Xiaoqing Ding and Jiayong Zhang, Offline handwritten character recognition based on discriminative training of orthogonal Gaussian mixture model, In Proc. of 6th International Conference on Document Analysis and Recognition, 2001, pp. 221-225

[30] Cheng-Lin Liu, H. Sako and H. Fujisawa. Discriminative learning quadratic discriminant function for handwriting recognition, IEEE Trans. on Neural Networks, 15(2): 430-444, 2004.

[31] Tseng Lin Yu, Chuang C T. An efficient knowledge based stoke extraction method for multi-font Chinese characters. Pattern Recognition. 25(12):1445-1458, 1992

[32] Rong Wang, Xiaoqing Ding and Changsong Liu. Handwritten Chinese address segmentation and recognition based on merging strokes. Journal of Tsinghua Univ (Sci & Tech), 44(4): 498-502, 2004

[33] Cheng-Lin Liu and M. Nakagawa. Precise candidate selection for large character set recognition by confidence evaluation. IEEE Trans. PAMI. 22(6): 636-642, 2000

[34] Qiang Fu, Xiaoqing Ding, Changsong Liu, Yan Jiang and Zheng Ren. A Hiddern Markov Model based segmentation and recognition algorithm for Chinese handwritten address character strings. In Proc. of 8th International Conference on Document Analysis and Recognition, 2005, pp. 590-594

[35] Yan Jiang, Xiaoqing Ding, Qiang Fu and Zheng Ren. Application of Bi-gram driven Chinese handwritten character segmentation for an address reading system. In 7th International Workshop on Document Analysis Systems, 2006, pp. 220-231

[36] C. Olivier, H. Miled, et al. Segmentation and coding of Arabic handwritten words. In Proc. of 13th International Conference on Pattern Recognition. 1996, pp. 264-268

[37] A. Cheung, M. Bennamoun and N.W. Bergmann. An Arabic optical character recognition system using recognition-based segmentation. Pattern Recognition, 34(2): 215-233, 2001

[38] Pingping Xiu, Liangrui Peng, Xiaoqing Ding and Hua Wang. Offline handwritten Arabic character segmentation with probabilistic model. In Proc. of 7th International Workshop on Document Analysis Systems, pp. 402-412, 2006

[39] Jin, J., et al., Printed Arabic document recognition system. Vision Geometry XIII. Edited by Latecki, Longin J. Mount, David M.; Wu, Angela Y. Proceedings of the SPIE, 2004. 5676: p. 48-55.

[40] Yan jiang, Xiaoqing Ding and Zheng Ren. Substring alignment method for lexicon based handwritten Chinese string recognition and its application to address line recognition. In Proc. of 18th International Conference on Pattern Recognition. Vol. 2, pp. 683-686, 2006.

Application of Pictographic Recognition Technology for Spotting Handwritten Chinese Words

Donald T Gantz, PhD **John J Miller, PhD**
George Mason University
4400 University Drive
Fairfax Virginia 22030
(703) 993-1511
dgantz@gmu.edu jmiller@gmu.edu

Mark A Walch
The Gannon Technologies Group
1000 North Payne Street
Alexandria, Virginia 22314
(703) 373-1962
mwalch@gannontech.com

Abstract

Pictographic Recognition Technology is a Graph-Theory–based methodology for the detection and extraction of specified words or groups of words from both handwritten and machine printed document collections. This technique converts written and printed forms into mathematical graphs and draws upon key properties of these graphs such as topology and geometric features to locate graphs that respond to specified search terms. In its initial implementation, Pictographic Search Technology has functioned by identifying individual characters in strings of handwritten script—both English and Arabic.

Handwritten Chinese presents the opportunity to exploit a key capability of Pictographic Recognition: detection of embedded forms. These embedded forms reflect graphical structures that are consistent for particular Chinese words—similar to the concept of "radicals". This paper focuses on current research toward using Pictographic Recognition techniques to spot handwritten Chinese words.

1 Introduction

Pictographic Recognition offers a means for spotting handwritten Chinese words within large document collections. The mathematical foundation for Pictographic Recognition is Graph Theory and the technique entails the use of graph-based pattern matching techniques to detect the "graphical signature" of objects contained in images of documents. The objects can be single characters, words, groups of words or characters, signatures and marks, symbols and virtually any other graphical form. Every written object contains a graphical signature constructed from graphs. These graphs may either constitute the full written object or be embedded within it. Pictographic Recognition permits these graphs, both full and embedded, to be automatically isolated, classified and linked to known graphs representing alphabetic characters, groups and parts of character components, parts of signatures and other forms. And, in the case of Chinese, these graphs can represent major components within the written word that form the basis for word spotting and recognition. To date, Pictographic Recognition has been implemented as a Prototype Application that performs word searches based on pictographic signatures. Current languages where Pictographic Recognition has been implemented include machine printed, hand printed and cursive English as well as machine printed and handwritten Arabic.

Chinese words typically consist of very complex graphical forms containing two or three times the number of edges and vertices that can be found in other languages such as English or Arabic. However, their strong pictographic roots suggest Pictographic Recognition should prove to be a useful tool for word identification.

Research to-date indicates that the complex graphical structure offers an advantage in identifying machine-printed Chinese words where the printed form offers such a level of consistency that graph complexity can be used as an identifier. That is, some printed Chinese forms can be recognized or spotted purely through the topology of their "whole word" graphs. However, since handwritten forms exhibit a much higher degree of variability than machine-printed forms, the complexity of Chinese poses an extremely daunting challenge both to computer-based recognition and even to automated techniques for spotting words of interest.

The authors believe spotting words of interest in Chinese writings often follows a model akin to "signature identification". That is, a person's signature is complex and almost never written exactly the same way twice. However, there are similar elements within each written signature that enable the

signature to be identified. Similarly, the Chinese word written by the same individual will be different—sometimes significantly different—with each new writing. And, the same word written by multiple individuals will exhibit even more differences. However, as is also true of signatures, different writings of the same Chinese word will retain a kernel of key graphical information that permits recognition. This graphical information is what Pictographic Recognition can detect. It must be emphasized that no single kernel can be expected to be consistent for all forms of the same word. However, it is believed a manageable set of kernels can be identified that can provide broad coverage across variant word forms sufficient to support word spotting—and ultimately support recognition. The current research focuses on isolating and identifying kernels of graphical information embedded in Chinese words. The objective is to build sets of these embedded forms that can be used for recognition-based word spotting.

Because of their pictographic roots, graph based forms are the basis upon which the Chinese language is predicated. For purposes of Pictographic Recognition, these basic building blocks of Chinese writing, the "radicals", are simply embedded graphs. It is believed that sets of embedded graphs can be extracted from Chinese writing and that the topology and features of these graphs will contribute to identification of the word in which they were embedded. These embedded graphs may include actual radicals plus other graphical forms that behave like radicals. The objective of the present research is to find forms that can be automatically extracted from multiple writings from different authors. Once isolated, these forms must be able to identify reliably the words from which they were extracted while minimizing confusion with other words.

Pictographic Recognition shares common roots with Optical Character Recognition ("OCR"), but it is a fundamentally different approach. Pictographic Recognition looks for the "Pictographic Signature" of words in their native form. In this context, Pictographic Recognition is somewhat similar to Optical Word Recognition ("OWR") but differs significantly from OWR in its ability to evaluate the actual characters and character groupings that comprise the unknown words. The greatest distinction of Pictographic Recognition from OCR and OWR processes is the ability to look both at the full form of a character or word as well as the ability to "drill down" into embedded graphical forms.

The discussion that follows will focus on the ability of Pictographic Recognition to identify Chinese words through identification of embedded graphical forms. At the heart of this recognition process is the ability to encode both the features and topology of a Chinese word into a form that can be used for rapid identification. The technique herein described is strongly rooted in an algorithmic approach, based on Graph Theory, which treats handwriting and printed text as mathematical graphs. The distinction that applies to Chinese writing is that the principles successfully applied to other languages with linear character relationships will now be extended to "two-dimensional" pictographic structures.

2 Conceptual Framework

Graph Theory is a branch of Mathematics that focuses on representing relationships as line diagrams containing nodal points and the linkages among these points. In graph terminology, the nodes are referenced as "vertices" and the links as "edges". Graphs are an effective way to represent written language since graphs can be created directly from the pen strokes used to compose letters and words. Words, characters and numerals take their form as graphs written on paper assuming distinctive shapes representing the letters of the alphabet as well as numerals and punctuation. The edges and vertices of these written graphs connect and cross and are straight or curved. Graphs accurately capture the essence of written language since they contain both the topological structure and geometric information sufficient to replicate complete written forms. Graphs are the foundation of written language and, therefore, a highly efficient and accurate tool within which to conduct searches, irrespective of languages as well as handwritten or machine generated formats.

Graphs contain all the information extracted from writing condensed into a concise mathematical format that is highly computable. Within graphs, this information takes two forms. The first form is the graph's topology that can be seen in the connectivity among the major graph components. The way pen strokes are crossed and connected is the framework for graph topology. Topology is the structure of the graph. The second form of information contained in graphs is the geometry of the graph. Geometry is expressed in terms of distances, angles and characteristics of graph components. The depth or shallowness of a curve, the distance between line crossings or the sharpness of angles are all examples of graph geometry. Geometry characterizes the shape of the graph. Collectively, topology and geometry account for the structure and shape of graphs. The topology and geometry of graphs are also quite computable. That is, they can be expressed as data to support computer-based processes that can be performed on graphs such as indexing and searching. The foundation of Pictographic Recognition is an algorithm that describes graph topology as a numeric

code. Any two graphs with the same structure will generate the same code. Any two graphs generating the same code are said to be isomorphic.

Although the details of how this code is constructed are beyond the scope of this paper, the method for generating the code involves reordering a graph's adjacency matrix based on connectivity of the vertices within the graph. This method will create identical adjacency matrices from graphs that are isomorphic. The vertex with the highest level of connectivity will always be placed first in the new graph ordering. This vertex is referenced as the "Prime Vertex". The Prime Vertex is important because it provides the basis for vertex ordering during key construction. It also provides a readily identifiable reference point within the graph that can be used as the origin for distance and angular measurements. The reference point afforded by the Prime Vertex provides a foundation for encoding the shape of a graph which is discussed later in this paper.

Linked to the graph topology and its attendant numeric code is graph geometry. Graph geometry can also be expressed as a feature vector used to compare graphs. A feature vector is a multi-dimensional expression containing the multitude of measurements that can be extracted from graphs. The topology of graphs can be coupled with feature vectors as a combined data structure. The result is a structure that is very computable while offering an effective way to use graphs as surrogates for characters and words contained in the images of documents.

The computational engine for Pictographic Recognition is an automated method for identifying and matching isomorphic graphs and storing these graphs in a database. Graphs are isomorphic when they are structurally identical—with the same number of edges and vertices connected in the same way—although they may appear to be different. Two graphs are considered isomorphic when there exists a one-to-one correspondence between their internal structures. That is, they have the same number of edges and vertices connected to form exactly the same topology.

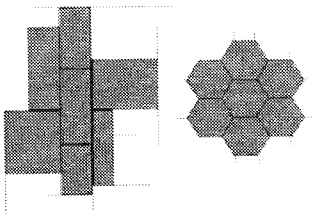


Figure 1: Illustration of two Isomorphic Graphs with different features.

Figure 1 illustrates this point. Although the graphs appear to be quite different, they are structurally identical: isomorphic. They appear different because their geometry is different. The topology is the same, but the angles and distances are different.

Figure 2 illustrates the concept of Isomorphic Graphs as it applies to characters from handwritten English. In this figure, each row shows three instances of the letter "a" written as the same isomorphic graph class. The numbers on the left hand side of the figure are the code names derived by the Pictographic Recognition process for each class of graph. These codes will always be the same for graphs having the same topology—isomorphic graphs. The class labeled "2;192" consists of graphs built from a loop and one edge. Class "4;112" contains graphs with three edges only connected at a central vertex. These typically characterize the letter "u" and the letter "a" written to be open at the top. Class "4;98.0.64" encompasses characters with a leading edge, an enclosed area—in graph terminology this is called a "face" -- and a trailing edge.

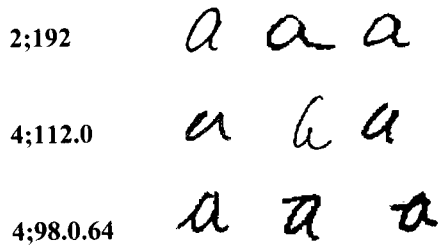


Figure 2: Sample letters "a" for three different graph isomorphic classes.

Figure 3 provides another English language example showing how graph isomorphic classes transcend individual letters. This figure presents letters "a" and "e" for isomorphic graphs classes coded "4;112.0" and 4;98.0.64".

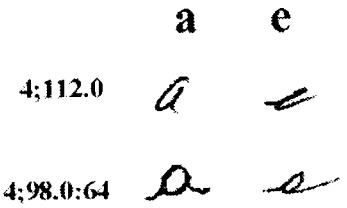


Figure 3: Example of letters "a" and "e" sharing the same isomorphic graph.

In English and Arabic implementations, Pictographic Recognition focuses on locating and identifying individual characters within handwritten words (machine printed matter can be considered to be

"perfect handwriting"). In these cases embedded character forms are encountered in a linear fashion. The discussion in this paper looks at Pictographic Recognition applied in a different manner—as a means for identifying embedded forms that span 2-dimensional shapes where there is both consistency and variability among embedded forms.

The premises underlying mining embedded forms can be summarized through the following points.

- Every line figure can be represented by a graph.
- Every graph can be identified by a unique code that represents its topology.
- Graphs fully capture the topology of written forms as well as all their embedding.
- The full topologies and all embeddings can be represented by codes related to graph isomorphism.
- Geometric feature information can also be stored in graphs.
- These geometric features can also be encoded and "bundled" with the topology information.
- Topology coupled with geometric features creates a very powerful means for classifying graph shapes.
- These classified graphs can be used to identify the written forms from which they were extracted.

3 Embedded Forms

The key to Chinese word spotting is the ability to isolate and to recognize graphs embedded within the graphs that constitute the actual word. The concept of embedded graphs contained in written forms is shown in Figure 4 which illustrates the letters "I", "L" and "F" embedded in the letter "E".

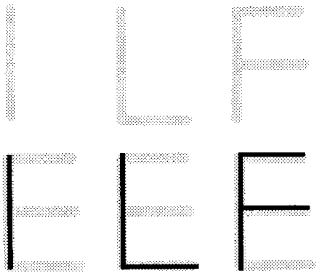


Figure 4: Examples of characters embedded in the letter "E".

Given a particular graph and a set of graphs of interest, Pictographic Recognition can isolate all embeddings that match the graphs of interest. Figure 5 shows a handwritten Chinese character and three potential embedded forms contained within the character. These are only a subset of the numerous embeddings that can be detected within these words.

Figure 5: Chinese character and three sample embedded forms.

Three factors relate to embedded form detection:

1. Graph Isomorphism
2. Shape Description
3. Embedding Percentage

The Graph Isomorphism represents the specific topology of the embedded graph as described by the unique code identifier provided through Pictographic Recognition. The Shape Description is an encoded means for articulating the geometry of the graph. This paper presents an angle-based encoding method that articulates the form of the graph as a series of directions from the Prime Vertex. And, the Embedding Percentage shows what proportion of the overall word graph the embedded graph occupies. Embedding percentage is relevant in two distinct cases: modeling and testing. In modeling, the embedding percentage determines what exemplars are best suited for reference purposes. In testing, the embedding percentage determines what proportion of an object must match a reference exemplar to determine a match does indeed exist.

These three parameters provide a means of defining and exploiting embedded graphs for recognizing handwritten Chinese.

4 A Chinese Primer

Unlike most of the world's scripts, which evolved towards syllabaries or alphabets, the Chinese script draws on its pictographic and ideographic origins to form picto-phonetic means of representing words and components of words.

Most of the world's scripts fall along a continuum, with purely pictographic means of representing a language on one end and purely phonetic means on the other. Many of the world's older scripts, such as Cuneiform, Egyptian hieroglyphics, and Chinese words, are picto-phonetic. This means that their characters represent both sound and meaning to different degrees.

Chinese is unique among the world's major scripts in use today in that it has several purely pictographic characters. This means that the character is a picture of the word it represents. For example, the character for the Chinese word *ren* "person", 人 depicts the legs and body of a person. The character for the word *shan* "mountain", 山 depicts the outlines of a mountain. It is important to remember that pictographic characters are not always immediately apparent. For example, the Chinese word *shui* "water", 水 is a picture of three streams flowing together.

Anyone who has played charades or "Pictionary" knows that a purely pictographic means of representing words does not go very far, since certain concepts are too abstract. Pictographically, concepts such as "good," "big," and "middle" can only be represented using pictures which in some way demonstrate the concept. For example, the word for "big", 大 *da* is a picture of a man 人 holding his arms out, describing something that is big. Interestingly, combining the words for "wife", 女 *nü*, and "children", 子 *zi* creates a combined pictograph for "good", 好 *hao*. Other characters are more abstract. The character for "middle", 中 *zhong* ,is a line drawn through an indeterminate object which in machine printed form looks like a square or rectangle.

These methods can produce several hundred characters. However, they are not sufficient to represent the vocabulary of an entire language. The example of charades is helpful here – in the same way that someone would act out a word that the target word sounds like, and then act out this word's meaning, most Chinese words combine two characters, one representing its meaning and the other giving a clue as to how to pronounce it.

The character representing its meaning, or the radical, will be one of 214 characters which represent broad semantic categories. For example, the radical from the word for "water", 水 *shui* will have to do with liquid, water, and fluidity. The character for "eye", 目 *mu* will have to do with sight, or vision. As a radical, *shui* becomes three dots on the left of the character. Using this radical, we can derive the characters 洲 *zhou* "island", 況 *kuang* "cold water", 油 *you* "oil", and 游 *you* "swim".

It also must be remembered that the radical and phonetic components do not necessarily occur on the left and right sides of the character, respectively. In some cases, the alignment can be vertical, as in 宙 *zhou* "universe", with the radical for a roof.

5 Radicals and Pseudo-Radicals

Since radicals are the basic building blocks of written Chinese, they are graphical forms that can be expected to hold some level of consistency across different Chinese words. Within the contexts of characters-as-graphs, Radicals can be viewed as embedded forms within the graph of a complete Chinese work. Since Pictographic Recognition is capable of detecting graphs embedded in other graphs, it is very consistent with the concept of Radicals in written Chinese.

The present study focuses on the concept of embedded graphs as "radicals" as a substitute for actual semantic values in written Chinese. That is, it is possible to identify automatically embedded graph forms that transcend numerous instances of the same written Chinese character. These forms are embedded graphs that perform the function of a radical and may or may not be the true radical from the Chinese word. The premise is that Pictographic Recognition can detect certain embedded forms inside Chinese words that a computer can use for recognition in a manner similar to the way a native Chinese speaker would "decode" the radicals and other stroke information into the actual word meaning.

These "pseudo-radicals" are empirically extracted graph forms that can be found in numerous instances of the same Chinese character. It must be stressed that handwritten Chinese, like other handwritten forms, exhibits a great deal of variability in topology and geometry so that it should not be expected that a single pseudo-radical (or any embedded graphical form) will encompass the majority of instances of any particular character. However, with enough training samples it is postulated that a set of common consistent graph forms can be identified.

6 Methodology

The methodology for evaluating the concept of pseudo-radicals involves isolating embedded graph forms and classifying these forms. A simple classification method can be used involving the previously discussed "Prime Vertex" from the isomorphic graph key generation process and Feature Angles (FAs) from this vertex to (1) all other vertices in the graph and (2) to certain points on each edge within the graph. The Feature Angles were expressed as three digit degree values from "000" to "359" and concatenated into a numeric string. The order of the directions was based on the ordering of the vertices used to build the isomorphic key in the Pictographic Recognition process. The string of Feature Angles serves as a very simple compact "shape code" in the

spirit of "chain codes" that are commonly used for shape definition.

Figure 6 shows the Chinese word "big" (大 *da* structured as a graph with the Prime Vertex in the center with Feature Angle pointers emanating as lines toward other vertices. This basic form can be translated into a shape code based on the composite directions.

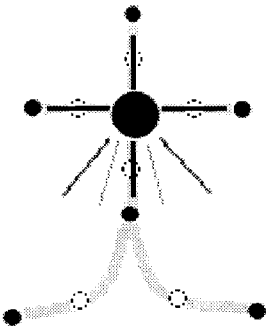


Figure 6: Chinese word "big" showing Prime Vertex and directions to other vertices (black dots) and interior points (white dots).

For purposes of illustration and omitting detailed discussion of the actual methodology for producing the order of the vertices, the directions from the Prime Vertex in Figure 6 would be the following (Our system uses 000 to represent due east, 090 to represent due south, and so forth.): 090 (due south), 000 (due east), 270 (due north), 180 (due west), 115 (southwest), and 065 (southeast) for the vertices and 090, 000, 270, 180, 100, 080 for the interior points, producing the set of angles: 090, 000, 270, 180, 115, 065, 090, 000, 270, 180, 100, 080.

Figure 7: Examples of two different characters with same graph isomorphism distinguished by Feature Angles.

To help the reader understand the value of the interior point angle, consider the two characters in Figure 7 above. The character on the left is a *ge* while the character on the right is a *ren*. The angles for the *ge* are 048, 078, 083, 115, 081 beginning with the vertex at the top of the character. The angles for the *ren* are 041, 162, 255, 045, 208 beginning with the vertex at the center of the character. (The reader may wish to verify how the angles are calculated.) Both characters are represented by the same isomorphism, namely 4:64.128, which represents "two lines". However, the angles to the interior points clearly distinguish between the *ge* with the bent line and the *ren* with straight lines. (In fact, the interior point for a bent line is placed at the point of maximum bending.) Using the angles to the interior points proves to be a powerful tool in the recognition of these characters.

7 Analysis of Chinese Sample Writings

Given the inherent complexity in handwritten Chinese, the Authors focused initially on the more simple character forms under the premise that identification of the complex forms will entail identifying a preponderance of embedded simple forms. To obtain a mixture of simple and complex forms, the "20 most common Chinese words" were selected for data collection.

The Authors collected handwriting samples from approximately 300 writers. Every writer was a native Chinese speaker. Each writer was given a printed form showing the 20 most common Chinese words. Each writer was asked to write the 20 words 3 times yielding 60 individual exemplars per writer. The Authors' initial findings, reported in this paper, focused on the six most simple character forms within this "top 20" selection. The remainder of this paper focuses on the findings obtained from this set. The six selected characters are *bu*, *da*, *ge*, *le*, *ren*, and *shang*. These characters are illustrated in Figure 8 below. Figure 8 contains a printed version, a handwritten exemplar similar to the printed version, and several other handwritten forms of the character. This figure shows the degree of variability and complexity which must be confronted when attempting recognition of Chinese words.

Numerous versions of these six characters were prepared for analysis by performing the following steps:

1. Scanning the handwritten documents into images.
2. Labeling all the words within the images to establish "ground truth".
3. Converting the images into graphs.
4. Computing all embedded versions of these graphs.
5. Calculating the isomorphic graph keys for each full graph as well as each embedded graph.
6. Calculating the Feature Angles for each full graph plus all embedded graphs.
7. Creating a database of graph information.

Name of Character	Printed Version	Handwritten Exemplar	Alternative Handwritten Forms		
bu	不				
da	大				
ge	个				
le	了				
ren	人				
shang	上				

Figure 8: Selection of six commonly used Chinese words showing variations in written forms.

Before the testing methology is fully described, two more points should be considered. The first is that the most reasonable exemplar for a particular written character may not be the embedding which covers 100% of the pixels of the character. This idea is represented in Figure 9. This figure contains a *ge* character. The 100% character in the left part of the figure contains what might be characterized as an "ink smear" which results in an isomorphism which is unnecessarily complex (8;112.24.0.0.32, an "H" with a line). The character on the right is the 94% embedding which results in a simpler isomorphism more in keeping with many other *ge*s (6;112.0.64, a "T" with a line). This example shows that taking subgraphs with embedding percents not equal to 100% can yield a more appropriate characterization of a writing.

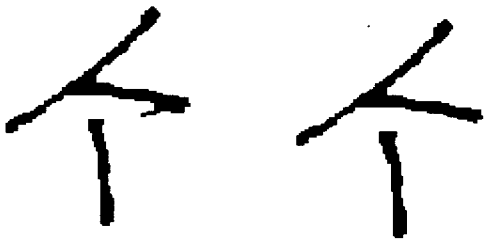


Figure 9: A 100% (left) and 94% (right) embedding of a *ge*.

Another concept which is useful is the idea of an augmented isomorphism. When the lines of a writing contain extreme bends (such as an "S" shape, the augmented isomorphism adds a "soft vertex" in the character, so that more of the shape of the writing can be captured by the Feature Angles. An example of this is given in Figure 10. This figure shows the character *le* with original vertices in gray and the "soft vertex" in a white box containing an "x". The presence of the soft vertex enables the Feature Angles to more closely represent the extreme bending of the line.

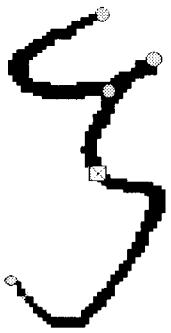


Figure 10: A *le* showing original vertices and augmented soft vertex.

The conjecture underlying the Authors' analysis is that it is possible to identify empirically a set of embedded graph-forms that will perform the function of "radicals" in Chinese word recognition. The methods presented in this study represent a first step in a process of building a "bullet proof" reference collection of forms in support initially of word spotting and ultimately of word recognition. The first steps toward building such a reference set are discussed as follows.

The six selected Chinese words are all different in their structure, but they have multiple common embedded graphs of the same isomorphic class. The key to distilling embedded graphs that behave as "pseudo-radicals" is to find graphs that are unique in their geometry and encompass significant portions of the overall Chinese word graph. The Authors' selection of simple words was intended to test the concept of embedded pseudo-radicals while controlling for other issues of complexity occurring in Chinese words. Next steps will involve extending these concepts to more complex character forms.

The selected characters principally exhibited six common embedded isomorphic forms. These forms, with descriptive captions, are shown in Figure 11 as follows:

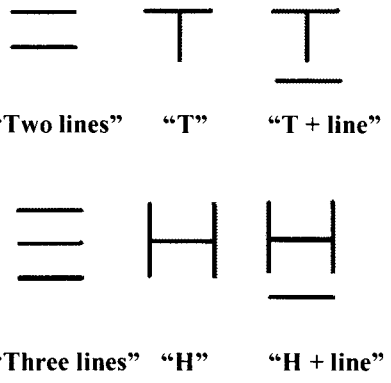


Figure 11: Six most common unique embedded graphs in six selected Chinese words.

Each of these graph types, extracted automatically, represents ones that occur both within a particular word and across different words at high embedding levels. It is the Authors' belief that a similar set of graphs can be extracted for more complex Chinese words but these graphs will be observed at a lower embedding level.

The analysis entailed identifying all embedded forms within the six selected Chinese words. The percentage of embedding ranged from 100% (the full graph) to 60%. That is, the full graphs were decomposed into all graph forms encompassing 60% or greater of the pixel content of the graph. Each full graph as well as its embedded forms was classified in two ways: (1) *Topology*--through the isomorphic graph keys and (2) *Geometry*--through the Feature Angles. These two parameters represent a very effective and compact way to represent graphical forms at varying levels of complexity.

A study was conducted to determine whether using the augmented isomorphisms and Feature Angle measurements would enable an automated system to identify characters for use in word spotting. The data for a particular character were divided into a training set and a test set in proportion approximately two-thirds in the training set and one-third in the test set.

The training set was then used to determine a set of augmented isomorphisms and Feature Angles which describe the character under study. The test set was then compared to this set of augmented isomorphisms and Feature Angles to see if any matches were found. In addition *all* the writings for the other five characters were compared to the set of augmented isomorphisms and Feature Angles to see if any matches were found for these characters. Effectiveness was measured in two dimensions. The first is "recall" which represents the percentage of all the characters being sought in the test set which were actually identified. The second is "precision" and is the percent of all the characters which were matched by the process which were actually the character being sought. High values of both these measures are desirable. This entire process was replicated for different divisions into training and test data and for various combinations of three parameters which govern the process.

The three parameters are the following: The first is the tolerable deviation in an angle which could still be considered a match. This parameter is called delta. Larger values of delta make it easier to find a match and might be predicted to increase recall but possibly decrease precision. The second parameter is the lower limit for embedding level for a sub-graph in the training data to be eligible for the exemplar pool. The third parameter is the lower limit for embedding level for a sub-graph in the test data to be used as a test comparison. Larger values of these latter two parameters make it harder to find a match and might be expected to increase the precision and possibly to decrease the recall.

For word matching, recall and precision are not equal. High values of recall are more important, since the point of word matching is to save work for a human reader. Thus it is important to catch almost all the instances of the character sought. The human reader can use context and other cues to sort out the true instances from the false. Increased precision makes the task of the human reader easier, but 100% precision is not required for word matching to be effective.

The results for a particular experiment for two characters are given in Figures 12a through 12d. These figures give the mean results for three replicates for each combination of parameters. The values of delta used in this experiment were 12, 13, 14, and 15 degrees. The values of the model embedding cut were 87, 89, 91, 93, and 95 percent. The values of the test embedding cut were 81, 83, 85, and 87 percent. All combinations were run so the graphs in the figures are based on 240 runs each.

Figure 12a shows that the mean recall is over 80% for all combinations of the parameters. A characteristic "saw-tooth" pattern is evident. This pattern, which is present in Figure 12c as well, shows that increasing delta increases recall (other things being equal) while increasing the cut percentages decreases recall. The saw-tooth comes from the fact that resetting the test cut to 81% from 87% increases recall, but to a value somewhat smaller than that for the smaller value of the modeling cut. Figure 12c shows the same general pattern, but with greater decreases in recall associated with increases in the modeling cut.

Figure 12b shows a severe saw-tooth pattern with precision generally decreasing with increasing delta and increasing with increasing values of the cut parameters. However, there is a large decrease in precision when the test embedding cut is changed back to 81% for the next higher level of the modeling cut. The saw-tooth pattern does not hold in Figure 12d. There is a general increase in precision with increased values of the cut parameters and decreased precision with increased delta.

To get a feel for what kind of performance could be expected from a "tuned" version of this process the following set of calculations was conducted. For each of five characters tested (*bu*, *ge*, *le*, *ren*, and *shang*), a search was conducted for the set of parameters which gave the highest value of precision for a minimum value of 92% for the recall. These results are presented in Table 1. These results show that it is hardest to achieve high precision for *ge* and *bu*, while much easier for *shang*, *le*, or *ren*.

Table 1: Best precision achievable for recall at least 92% for five characters

Character	Best Achievable Precision
Bu	45.7%
Ge	37.3%
Le	74.9%
Ren	78.0%
Shang	65.3%

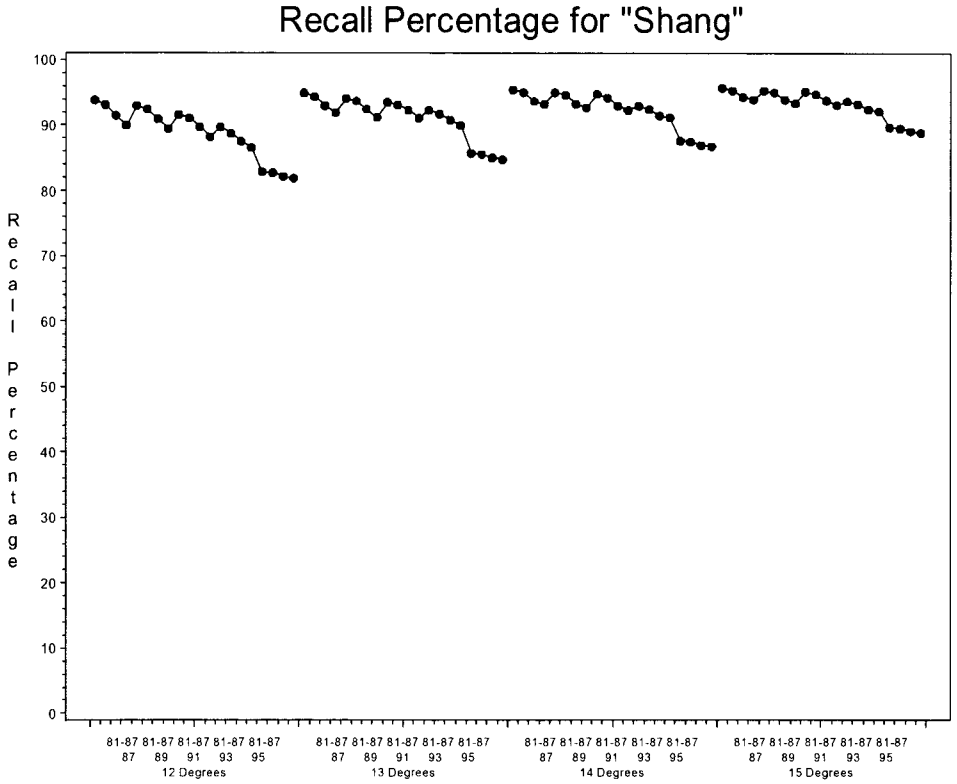


Figure 12a: Mean percentage recall for study of *shang*.

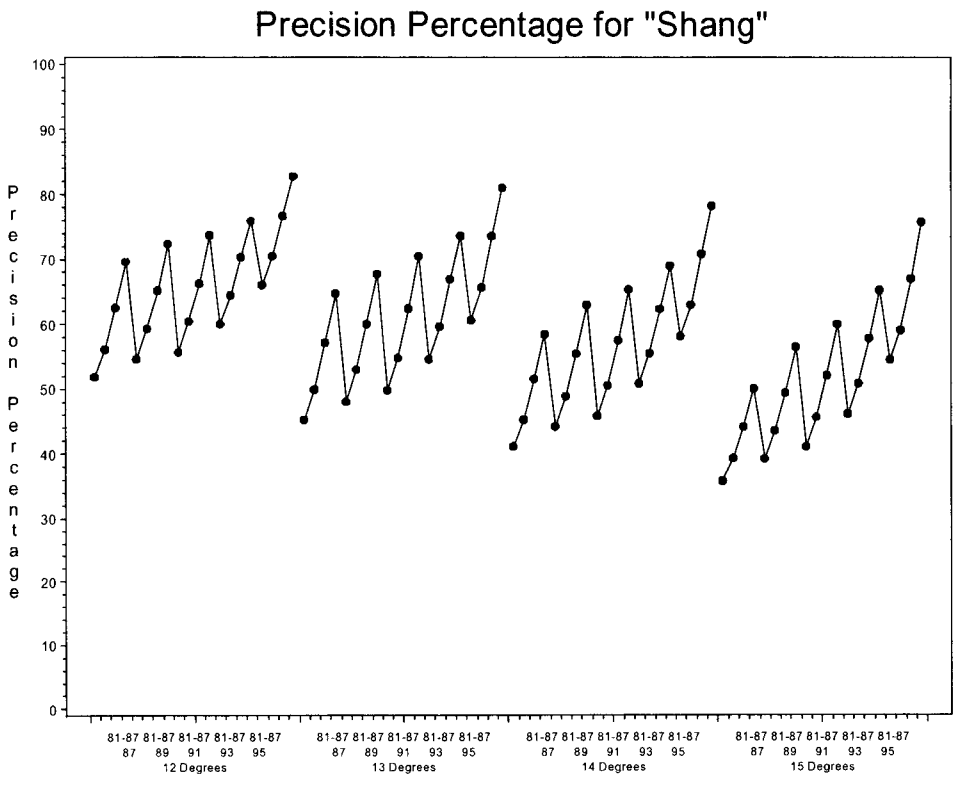


Figure 12b: Mean percentage precision for study of *shang*.

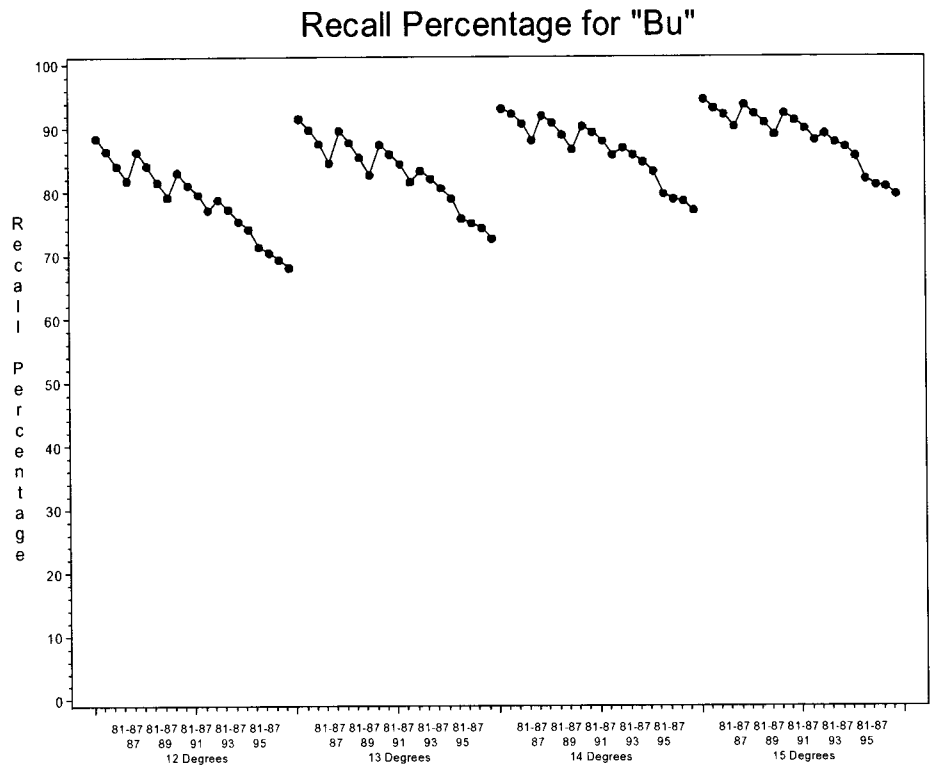


Figure 12c: Mean percentage recall for study of *bu*.

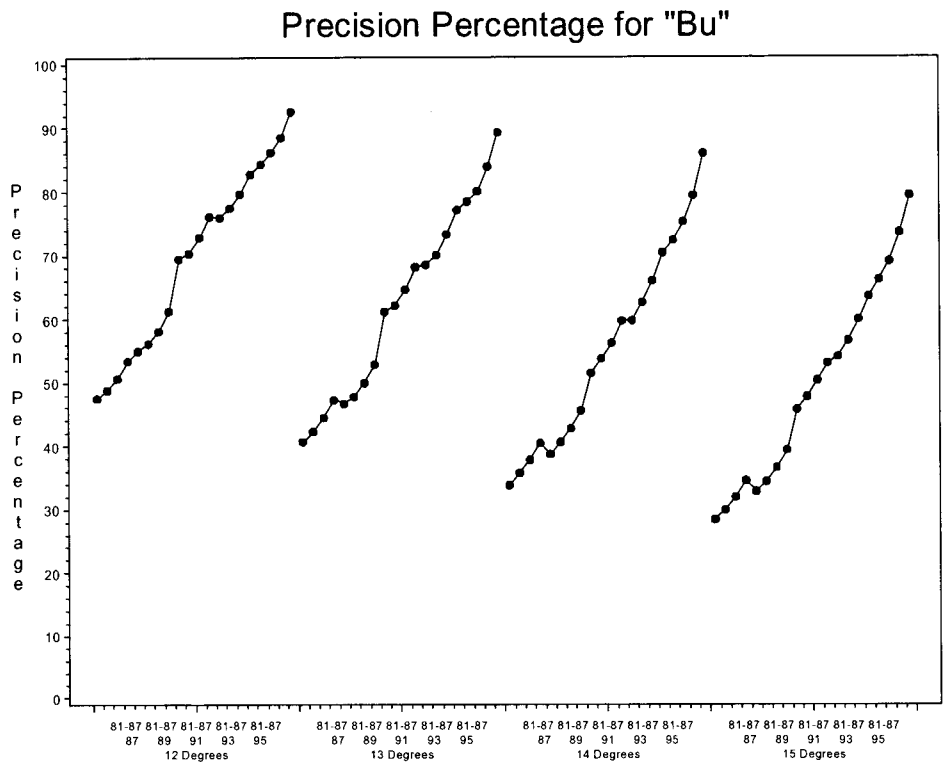


Figure 12d: Mean percentage precision for study of *bu*.

8 Conclusions

The work described in this paper represents the initial steps toward developing a viable means for Chinese word spotting (and ultimately recognition) based on Pictographic Recognition. The concept driving the Authors' approach was that it would be possible to empirically derive sets of embedded forms that characterized the Chinese word in which they were embedded. To test this idea, the Authors collected a sample of Chinese writings and focused on a subset of six simple character forms. These six forms were selected because they enabled the Authors to control for complexity during the initial evaluation. The six simple forms also presented challenges because of their similarity to each other and the high number of common embedded forms they shared.

The Authors were able to implement a system that automatically distilled embedded forms from handwritten Chinese words, compiled a database of these forms, and used these forms to recognize other versions of these same Chinese words. The factors that influenced the behavior of recognition based on embedded forms are:

1. Embedded graph topology as expressed through a code corresponding to graph isomorphism.
2. Embedded graph shape as described through Feature Angles relative to a designated node within the graph.
3. Percentage of embedding as described as the proportion of pixels from the entire Chinese word occupied by the embedded graph.

In the system implemented by the Authors, embedded graphs were shown to behave in a manner similar to the actual building blocks of written Chinese: the radicals. These pseudo-radicals empirically generated within this study confirm the concept that is possible to isolate common topological and geometric forms that transcend multiple occurrences of the same written Chinese word. And, once isolated these common embedded forms can become the foundation for word recognition.

The Authors intend to continue this work by focusing on the following topics:

1. Continuing to build a ground truth database of Chinese handwriting exemplars.
2. Applying the techniques herein described to more complex Chinese words.
3. Continuing to "tune" the recognition process based on the three parameters discussed above: Topology, Geometry and Embedding.
4. Expanding from the concept of Feature Angles to more detailed measurement data. It should be noted that Pictographic Recognition offers a wealth of feature information that was intentionally excluded from the present study that will be brought to bear on future work.
5. Applying more sophisticated scoring methods—currently used for English and Arabic recognition to Chinese. These methods permit composite scoring among numerous embeddings using a "preponderance of evidence" approach.

Techniques for Solving the Large-Scale Classification Problem in Chinese Handwriting Recognition

Fu Chang
Institute of Information Science, Academia Sinica
Taipei, Taiwan
Email: fchang@iis.sinica.edu.tw

Abstract

Due to the large number of categories, or class types, in the Chinese language, the challenge of the character recognition task is how to deal with such a large-scale problem in both the training and testing phases. This talk addresses three techniques, the combination of which has been found to be effective in solving the problem. The techniques are: 1) a prototype learning/matching method that determines the number and location of prototypes in the learning phase, and decides the candidates for each character in the testing phase; 2) support vector machines (SVM) that post-process the top-ranked candidates obtained during the prototype learning or matching process; and 3) fast feature-vector matching techniques to speed up prototype matching via decision trees and sub-vector matching. The techniques are applied to Chinese handwritten characters, expressed as feature vectors derived by extraction operations, including nonlinear normalization, directional feature extraction, and feature blurring.

1 Introduction

The support vector machine (SVM) classification method (Cortes and Vapnik [1]) represents a major development in pattern recognition research because it produces highly accurate results for optical character recognition (Vapnik [2]). In applying SVM to Chinese character recognition (CCR), however, we face a challenge due to the large number of class types (at least 3,000) in the Chinese language.

SVM is essentially a method of binary classification, in which each object is classified as one of two classes. When dealing with a multi-class classification, in which each object is classified as one of m classes, where $m > 2$, the problem must be decomposed into binary classification sub-problems and the SVM method can then be applied to the sub-problems.

There are two possible ways to decompose the problem: *one-against-others* (Bottou et al. [3]) and *one-against-one* (Knerr et al. [4] and Platt [5]). In the former approach, we train m SVM classifiers, each of which classifies a sample (character) as A or not A, where A is one of the m class types. In the latter approach, we train $m(m\text{-}1)/2$ class types, each of which classifies a sample as A or B, where A and B are any two class types. The one-against-others approach is computationally costly in terms of CCR training, since it constructs m classifiers, each derived from n training samples, where n is the number of training samples. The one-against-one approach is also costly in terms of training, since it constructs $m(m\text{-}1)/2$ classifiers; however, each classifier is derived from a smaller set of training samples.

To cope with the size of the CCR application, we propose a novel decomposition scheme. First, we use a prototype learning method (Chang et al. [6] and Chou et al. [7]) to reduce the number of training samples to a much smaller set of prototypes. We assume that Chinese characters are represented as d-dimensional feature vectors. Therefore, the resultant prototypes are also d-dimensional, and they decompose the feature-vector space into disjoint domains of attraction (DOA) [1], where the DOA of a prototype $\mathbf{p}$ is defined as the set of feature vectors that find $\mathbf{p}$ as the nearest prototype.

This decomposition scheme can be useful to SVM in the following way. In the training phase, we collect the pairs, referred to as *confusing pairs*, of class types that are the top-k_1 candidates of some training samples. We then construct SVM classifiers for these pairs. In the testing phase, when a test character $\mathbf{x}$ is given, we collect the top-k_2 candidates of $\mathbf{x}$ and apply SVM classifiers to the confusing pairs found among the candidates. We then apply a simple voting scheme to re-arrange the involved candidates of $\mathbf{x}$.

To further reduce the number of CCR candidates in the testing phase, we can use the combination of the following methods (Liu et al. [8]). The first method employs multiple decision trees to collectively determine the candidates for each test character. The second method employs sub-vector matching in a few intermediate steps using a subset of features in each step. A decision tree then decomposes the feature space into disjoint hyper-rectangles, each of which is associated

[1] They are also called Voronoi cells in the literature.

with a number of candidates. Since multiple trees are used, we rely on a voting scheme to decide the candidates for the next step, which uses sub-vectors to further reduce the computational cost of matching test characters with prototypes.

The remainder of the paper is organized as follows. Section 2 contains the learning algorithm for constructing prototypes out of training samples. In Section 3, we describe the post-processing technique that uses SVM in the training and testing phases. Section 4 discusses the methods that speed up prototype matching, and Section 5 details the experiment results. Finally, in Section 6, we present our conclusions.

2 The Prototype Learning Algorithm

The prototype learning algorithm (PLA) described in this paper is a special version of the adaptive prototype learning (APL) algorithms detailed in Chang et al. [9]. In fact, the PLA is a very simple version of APL, but unlike general APL, it does not involve any parameters, and thus avoids the rather high computational cost of searching for optimal parameter values. PLA represents a rather fast and reasonable way to decompose the feature vector space so that we do not have to spend too much time on the expensive optimization process.

We assume that there are n training samples $(\mathbf{x}_1, y_1), \ldots, (\mathbf{x}_n, y_n)$ drawn independently from the set $R^d \times \Lambda$ according to the same distribution, where $\Lambda = \{1, 2, \ldots, m\}$ is a set of labels or class types. Prototypes also lie in R^d, but they are not necessarily training samples. Moreover, each prototype is associated with a label that is also a member of Λ. Two entities (samples or prototypes) are said to be homogeneous if they have the same label, or heterogeneous if their labels are different. When a set of prototypes is given, we say that a sample $\mathbf{x}$ is absorbed, if

$$\| \mathbf{x} - \mathbf{q} \| - \| \mathbf{x} - \mathbf{p} \| > 0, \qquad (1)$$

where $\mathbf{p}$ is the nearest homogeneous prototype to $\mathbf{x}$, and $\mathbf{q}$ is the nearest heterogeneous prototype.

The steps of PLA are as follows:

P1 Initiation: For each label y, initiate a y-prototype as the average of all y-samples.

P2 Absorption Check: Check whether each sample has been absorbed. If all samples have been absorbed, terminate the process; otherwise, proceed to the next step.

P3 Prototype Augmentation: If there are still un-absorbed y-samples, select one and apply the fuzzy c-means (FCM) clustering algorithm to construct clusters, using the selected y-sample and all existing y-prototypes as seeds. Return to P2 to proceed.

We now provide the technical details.

Selection of Unabsorbed Samples in P1 and P3: In P1, a y-sample is selected as follows. We let each y-sample cast a vote to its nearest y-sample, and select the one that receives the highest number of votes. In P3, an unabsorbed y-sample is selected as follows. Let $\Psi_y = \{\mathbf{x}_i: l(\mathbf{x}_i)=y \ \& \ \mathbf{x}_i \text{ is unabsorbed}\}$. We let each member of Ψ_y cast a vote for the nearest member in this set. The selected y-sample is the member of Ψ_y that receives the highest number of votes.

Fuzzy c-means in P3: In FCM [9-10], the objective function to be minimized is

$$\sum_{i=1}^{I} \sum_{j=1}^{J} u_{ij}^{\alpha} \| \mathbf{c}_i - \mathbf{x}_j \|^2, \quad \text{for } \alpha \in (1, \infty) \qquad (2)$$

under the constraint

$$\sum_{i=1}^{I} u_{ij} = 1, \qquad \text{for } j = 1, 2, \ldots, J, \qquad (3)$$

where u_{ij} is the membership grade of sample $\mathbf{x}_j$ to prototype $\mathbf{c}_i$. Using the Lagrangian method, we can derive the following equations:

$$u_{ij} = \frac{\left(1 / \| \boldsymbol{c}_i - \boldsymbol{x}_j \|\right)^{\frac{2}{\alpha-1}}}{\sum_{k=1}^{I} \left(1 / \| \boldsymbol{c}_k - \boldsymbol{x}_j \|\right)^{\frac{2}{\alpha-1}}}, \qquad (4)$$

$$\boldsymbol{c}_i = \frac{\sum_{j=1}^{J} u_{ij}^{\alpha} \boldsymbol{x}_j}{\sum_{j=1}^{J} u_{ij}^{\alpha}}, \qquad (5)$$

for $i = 1, 2, \ldots, I$, and $j = 1, 2, \ldots, J$ respectively. FCM is a numerical method that finds a locally optimal solution for (9) and (10). Using a set of seeds as the initial solution for $\{\mathbf{c}_i\}_{i=1}^{I}$, the algorithm computes $\{u_{ij}\}_{i,j=1}^{I,J}$ and $\{\mathbf{c}_i\}_{i=1}^{I}$ iteratively. To ensure rapid convergence of FCM, we require that the process stops when the number of iterations reaches 30, or $\sum_{i=1}^{I} \| \mathbf{c}_i^{old} - \mathbf{c}_i^{new} \| = 0$.

The prototypes are thus the cluster centers computed by FCM, which are the weighted sum of all the samples. For this reason, it is possible that the iterative process in steps P1 to P3 could continue to construct new prototypes and never converge [7]. To remedy this problem, we modify P3 as follows.

Recall that in P3, we employ FCM to compute a set Λ_y of y-prototypes, using an un-absorbed y-sample $\mathbf{x}$ and existing y-prototypes as seeds. For each $\mathbf{p}$ in Λ_y, let $D_y(\mathbf{p})$ be the set of y-samples for which $\mathbf{p}$ is the nearest y-prototype. If there exists any $\mathbf{p}$ in Λ_y, for which $D_y(\mathbf{p})$ is empty, we declare $\mathbf{x}$ to be a *futile* sample. If a sample is declared *futile* in an iteration, it will not be taken as a sample in any subsequent iteration.

This modification of P3 ensures the convergence of PLA [9].

3 SVM for Post-Processing

When a set of prototypes is given, we define the top-k *candidates* of a sample $\mathbf{x}$ as the top-k class types found within the prototypes, which are sorted according to their distances from $\mathbf{x}$. We may have to look for more than k nearest prototypes to obtain the top-k candidates, since two different prototypes could bear the same class type. We then collect the pairs (C_i, C_j), where C_i and C_j are, respectively, the i^{th} and j^{th} candidates of $\mathbf{x}$ for $1 \le i, j \le k$. Note that different values of k can be chosen in the training and the testing phases. We assume that k_1 and k_2 candidates are selected in the training testing phases respectively.

For each confusing pair (C, D), the samples of C and D are labeled, respectively, as 1 and -1. The task of the SVM method is to derive from the C- and D- training samples a *decision function* $f(\mathbf{x})$ in the following form.

$$f(\mathbf{x}) = \sum_{i=1}^{I} y_i \alpha_i K(\mathbf{s}_i, \mathbf{x}), \tag{6}$$

where $\mathbf{s}_i$ are support vectors, α_i is the weight of $\mathbf{s}_i$, y_i is the label of $\mathbf{s}_i$, $i = 1, \ldots, I$, and $K(\cdot, \cdot)$ is a kernel function. The character $\mathbf{x}$ is classified as a C- or D- object, depending on whether $f(\mathbf{x})$ is positive or negative; details are given in [2]. For our application, we adopt the kernel function:

$$K(\mathbf{s}, \mathbf{x}) = (< \mathbf{s}, \mathbf{x} > + 1)^{\delta}, \tag{7}$$

where $< \mathbf{s}, \mathbf{x} >$ is the inner product of $\mathbf{s}$ and $\mathbf{x}$, and δ is the degree of the polynomial kernel function.

In the testing phase, for a character $\mathbf{x}$, we first use the prototype-matching process to find k_2 candidates for $\mathbf{x}$. We then compute the decision functions associated with all confusing pairs found within the top-k_2 candidates of $\mathbf{x}$. If a confusing pair (C, D) is found among the candidates and $\mathbf{x}$ is classified as a C-type, then C scores one unit. The candidate with the highest score is ranked first, the candidate with the second highest score is ranked second, and so on. If two candidates receive the same score, their relative positions remain the same as before. We then rearrange the involved candidates according to their assigned ranks.

Note that there are three parameters involved here: δ, k_1 and k_2. To determine their values, we require a set of samples in addition to the set used for training. This new set is called *validation data*. We use the training data to construct SVM classifiers for various values of δ and k_1, and the validation data to compute the accuracy rates of the classifiers under different values of δ and k_1, using k_2 candidates for each test sample. In so doing, we are able to find the best combination of values for δ, k_1 and k_2.

4 Acceleration of Prototype Matching

Two methods are used to speed up the matching of samples and prototypes: multiple trees and sub-vector matching. They both have advantages and disadvantages. The tree method, for example, is fast, but there is a high risk of excluding the nearest known objects from the candidate list if the list is short. In contrast, with the sub-vector matching method, there is a lower risk of minimizing the candidate list, but the computational cost is high. One way to maximize the benefits of both methods is to combine them as follows. First, we use the tree method to substantially reduce the number of candidates, and then apply the sub-vector method to the remaining candidates to find the nearest known object.

4.1 Multiple Trees

Each decision tree is a CART (Breiman et al. [10]) or a binary C4.5 tree (Quinlan [11]). On each node of the tree, the feature and the split point is chosen to maximize the reduction of impurity [10]. With such a tree structure at our disposal, we need to resolve three issues: 1) we need to grow multiple trees, instead of a single tree; 2) we need to know when to terminate the tree; and 3) we need a mechanism to retrieve candidates from the multiple trees.

With regard to the first problem, suppose that the training samples are expressed as d-dimensional feature vectors and we decide to grow t number of trees. We divide each d-dimensional vector into t times e-dimensional sub-vectors so that $e \times t \ge d$. If the equality holds, the t sub-vectors will not share any features; otherwise, they will overlap partially. The training samples are then split into t sets of sub-vectors and each set is input to a tree. When the tree-growing process has been completed, we store at each leaf the class types of the samples that have reached that leaf.

For the second problem, we do not want to grow a tree to very deep levels, since the deeper we go, the smaller the leaves and the greater the risk of losing critical candidates. Suppose that the input samples are e-dimensional vectors and the total number of training samples is n. One way to limit the growth of the tree is to simply stop splitting all nodes at level l, where $1 \le l \le e$, and count the root level as 1. However, it is unreasonable to require that all paths stop at the same level l, thereby generating leaves of various sizes. Instead, we limit the size of leaves to $u = n/2^{l-1}$ because, if a tree terminates at level l, the average leaf size of the tree will be u. Therefore, if a node contains less than u samples, we do not split it further. Since the value of u depends on that of l, we write it explicitly as $u(l)$. The optimal value of l is determined in the procedure for solving the third problem, i.e., candidate retrieval, which we now address.

To retrieve candidates from multiple trees, we first grow t trees, in which the leaf sizes are bounded from above by $u(l)$. We then assume that a training sample is input to these trees and locates on the leaf L_i of tree i, $i = 1, 2, \ldots, t$. For each class type C stored on these leaves, we define its vote count as the number of leaves

on which it is stored. We then take the candidate list as the set of C whose vote count exceeds v, i.e.,

$$Candidate\_List(l, v) = \{C \in \bigcup_{i=1}^{t} L_i : vote\_count(C) \geq v\} \quad (8)$$

The optimal values of l and v are determined by means of the validation data (cf. Section 3). We first compute the accuracy rate $R_{prototype}$, defined as the proportion of validation samples that match in class type with their nearest prototypes. This represents the accuracy rate we obtain *without* multiple trees. To obtain the accuracy rate *with* multiple trees, we first grow multiple trees that terminate at leaves no larger than $u(l)$. We then input the validation samples into the trees to obtain the accuracy rate $R_{tree}(l, v)$, defined as the proportion of validation samples whose class types fall within *Candidate_List*(l, v). We choose l and v such that

$$R_{tree}(l, v) \geq R_{prototype}. \quad (9)$$

The set Θ of (l, v) that satisfies (9) is never empty, since $(1, 1)$ is in Θ. The optimal choice of (l, v) is then the pair in Θ that maintains the minimal size of *Candidate_List*(l, v).

Displayed equations are to be centered on the column with. Standard English letters like x are to appear as x (italicized) in the text if they are used as mathematical symbols. Punctuation marks are used at the end of equations they appeared directly in the text.

4.2 Sub-Vector Matching

When we input a test character to the multiple trees, we retrieve from the trees the prototypes whose class types fall within *Candidate_List*(l, v) for the optimal value of l and v. To avoid wasting time on unlikely candidates, we take the following intermediate steps, each of which performs sub-vector matching.

The first step handles sub-vectors of length d_1, the second step handles sub-vectors of length d_2, and so on, where $d_1 < d_2 < \ldots < d$. At the end of each step, the prototypes whose distance to the unknown object falls below a certain threshold are input to the next step for further processing. In the last step, full-length vectors are matched and the nearest prototypes are output. Two elements must be determined for each step: the feature types included in the sub-vectors and the threshold.

We first sort all feature types by means of their information gain [8, 11]. Then, we employ the features of the top-d_1 ranks in the first step, and the features of the top-d_2 ranks in the second step, and so on.

In the sub-vector matching method, we must determine three elements: how many steps we need to perform, the dimension of the sub-vector to be used in each step, and the threshold associated with this dimension. Let us assume that the dimension is given and we want to determine the threshold associated with it. Again, we use validation data for this purpose. We pass the validation samples through full-vector matching as well as sub-vector matching because we want to set the threshold in such a way that the two matching approaches achieve a comparable performance.

To ensure a robust performance, we associate a threshold with each sample $\mathbf{s}$ as follows:

$$Threshold(\mathbf{s}, \lambda) = \lambda \times Avg\_Dist + (1\text{-}\lambda) \times Min\_Dist, \quad (10)$$

where λ is a value between 0 and 1, *Avg_Dist* is the average sub-vector distance between $\mathbf{x}$ and all prototypes, and *Min_Dist* is the minimum sub-vector distance. If $\mathbf{p}$ is a prototype with $\|\mathbf{p} - \mathbf{x}\|^2 < Threshold(\mathbf{s}, \lambda)$, $\mathbf{p}$ is said to be λ-acceptable. We define the *Accuracy_Rate*(λ) as the proportion of validation samples that match in class type with the nearest λ-acceptable prototypes. Let R_{full} be the proportion of validation samples that match in class type with their nearest prototype with respect to the full-vector distance. The optimal value of λ is then the smallest value for which

$$Accuracy\_Rate(\lambda) \geq R_{full}. \quad (11)$$

We now consider the number of steps to be performed, and the dimension, or the number of features, to be used in each step. We use n_e to denote the number of prototypes passed to the next step if we perform sub-vector matching with dimension e. Suppose that the dimension for step i is d_i. We describe what to do at step i+1. To perform sub-vector matching at step i+1, we need to compute $e \times n_{d_i}$ operations at step i+1 and $(d - e) \times n_e$ operations at step i+2, assuming that full-vector matching is performed in the second step; therefore, the computational complexity of these two steps is

$$C(e) = e \times n_{d_i} + (d - e) \times n_e. \quad (12)$$

Let $d_{i+1} = \arg\min_{d_i < e \leq d} C(e)$. There are two options available at step i+1. Either we perform sub-vector matching with dimension d_{i+1}, or we perform full-vector matching. The complexity of the former is $C(d_{i+1})$, while that of the latter is $(d - d_i) \times n_{d_i}$. If

$$C(d_{i+1}) < (d - d_i) \times n_{d_i}, \quad (13)$$

we adopt sub-vector matching with dimension d_{i+1}; otherwise, we adopt full-vector matching. We proceed in this fashion, until dimension d is reached at a certain step.

5. Experiment Results

We evaluated the three techniques by applying them to the ETL9B dataset, which consists of 3,036 Chinese character types. From this dataset, we took 100 samples per character type for training purposes and another 100 samples per character type as validation data. The fea-

ture extraction method we employed consisted of three basic techniques (Chang et al. [12-14]): non-linear normalization (Lee and Park [15] and Yamada et al. [16]), directional feature extraction ([12], [14]), and feature blurring (Liu et al. [17]).

If we had to train all one-against-one SVM classifiers for the 3,036 character types out of 303,600 training samples, it would have taken an estimated 32 days using a PC with a Pentium IV 2.4GHz CPU and 2GB RAM. In the testing phase, a fast method, called DAGSVM [5], would have required 31.78 seconds to recognize a character and would have had to store approximately 1.5×10^8 support vectors in the memory. If, however, we use the proposed approach, we only need to store 19,237 prototypes (6.3% of the training samples), and 11,104,041 support vectors (approximately 7% of the support vectors, if all pairs are taken as confusing pairs). It is noteworthy that we spent only 61.1 hours training both the prototypes and SVM. The three parameters δ, k_1, and k_2 (cf. Section 3) were found to be 2, 3, and 5, respectively.

By applying multiple trees and sub-vector methods to speed up the matching of test samples and prototypes, we can recognize 1418.7 characters per second, compared to 57.6 characters per second if a sample had to match all prototypes; thus, the speed-up ratio is 24.6. From the results shown in Table 1, we also observe that the acceleration methods do not cause any loss in test accuracy.

Table 1. The performance of multiple trees and sub-vector methods.

ETL9B (Number of Class Types = 3,036, Number of Prototypes = 19,237)			
	Testing Accuracy	Computing Time(s)	Computing Speed (c/s)
Un-accelerated Matching	93.66%	5,271	57.6
Multiple Trees + Sub-Vector	93.68%	214	1418.7

Using SVM for post-processing boosts the test accuracy rate from 93.68% to 96.59%, but it adds 9,537 seconds to the recognition time such that the recognition speed drops from 1418.7 to 31.1characters per second (Table 2).

Table 2: The performance of our proposed methods applied to ETL9B.

	Testing Accuracy	Computing Time (s)	Computing Speed (c/s)
Multiple Trees + Sub-Vector	93.68%	214	1418.7
Multiple Trees + Sub-Vector + SVM	96.59%	9,751	31.1

6. Conclusions

We have proposed a combination of three methods to solve the Chinese handwriting recognition problem: prototype learning/matching, SVM, and fast vector matching using multiple trees and sub-vectors. The prototypes and trees provide the means to decompose the feature vector space and thus help reduce the number of candidates for matching. Sub-vector matching is obviously useful because it avoids wasting time on less likely candidates. By using these techniques in the pre-processing stage, we are able to exploit the effectiveness of SVM to enhance the test accuracy of the recognition task. The combination of the techniques is not only useful for the current application, but also for many other types of classification problems that involve a large number of class types and training samples.

References

[1] C. Cortes and V. Vapnik, "Support-vector network," *Machine Learning*, vol. 20, pp. 273–297, 1995.

[2] V. Vapnik, *The Nature of Statistical Learning Theory*, Springer Verlag, New York, 1995.

[3] L. Bottou, C. Cortes, J. Denker, H. Drucker, I. Guyon, L. Jackel, Y. LeCun, U. Muller, E. Sackinger, P. Simard, and V. Vapnik, Comparison of classifier methods: A case study in handwriting digit recognition, *Proc. Int. Conf. Pattern Recognition.*, pp. 77–87, 1994.

[4] S. Knerr, L. Personnaz, and G. Dreyfus, Single-layer learning revisited: A stepwise procedure for building and training a neural network, *Neurocomputing: Algorithms, Architectures and Applications*, J. Fogelman, Ed. New York: Springer-Verlag, 1990.

[5] J. C. Platt, N. Cristianini, and J. Shawe-Taylor, Large margin DAG's for multiclass classification, *Advances in Neural Information Processing Systems*, Cambridge, MA: MIT Press, 2000, vol. 12, pp. 547–553.

[6] F. Chang, C.-C. Lin, and C.-J. Chen, Applying A Hybrid Method To Handwritten Character Recognition, *Intern. Conf. Pattern Recognition 2004*, vol. 2, pp. 529-532, Cambridge, 2004.

[7] C.-H. Chou, C.-C. Lin, Y.-H. Liu., and F. Chang, A Prototype Classification Method and Its Use in A Hybrid Solution for Multiclass Pattern Recognition, *Pattern Recognition*, vol. 39, no. 4 , pp. 624-634, 2006.

[8] Y.-H. Liu, C.-C. Lin, W.-H. Lin, and F. Chang, Accelerating Feature-Vector Matching Using Multiple-Tree and Sub-Vector Methods, *Pattern Recognition*, to appear.

[9] F. Chang, C.-C. Lin, and C.-J. Lu, Adaptive Prototype Learning Algorithms: Theoretical and Experimental Studies, *Journal of Machine Learning Research*, to appear.

[10] L. Breiman, J. H. Friedman, R. A. Olshen, and C. J. Stone, *Classification and Regression Trees*, New York: Chapman and Hall, 1984.

[11] J. R. Quinlan, Induction of Decision Tree, *Machine Learning*, vol. 1, no. 1. pp. 81-106, 1986.

[12] F. Chang, C.-C. Lin, and C.-J. Chen, "A hybrid method for multiclass classification and its application to handwritten character recognition," Institute of Information Science, Academia Sinica, Taipei, Taiwan, Tech. Rep. TR-IIS-04-016, 2004.

[13] F. Chang, C.-H. Chou, C.-C. Lin, and C.-J. Chen, "A prototype classification method and its application to handwritten character recognition," *IEEE SMC*, Hague, 2004.

[14] F. Chang, C.-C. Lin, and C.-J. Chen, "Applying a hybrid method to handwritten character recognition," in *Proc.17th Intern. Conf. Pattern Recognition*, pp. 529-532, 2004.

[15] S.-W. Lee and J.-S. Park, "Nonlinear shape normalization methods for the recognition of large-set handwritten characters," *Pattern Recognition*, vol. 27, no. 7, pp. 895-902, 1994.

[16] H. Yamada, K. Yamamoto, and T. Saito, "A nonlinear normalization method for handprinted Kanji character recognition – line density equalization," *Pattern Recognition*, vol. 23, no. 9, pp. 1023-1029, 1990.

[17] C.-L. Liu, I.-J. Kim, and J.H. Kim, "High accuracy handwritten Chinese character recognition by improved feature matching method," in *Proc. 4th Intern. Conf. Document Analysis and Recognition*, 1997, pp. 1033-1037.

Multi-Character Field Recognition for Arabic and Chinese Handwriting

Daniel Lopresti
Lehigh University
Bethlehem, PA 18015
lopresti@cse.lehigh.edu

George Nagy
Rensselaer Polytechnic
Institute DocLab
Troy, NY 12180
nagy@ecse.rpi.edu

Sharad Seth
University of Nebraska
Lincoln, NE 68588
seth@cse.unl.ed

Xiaoli Zhang
Rensselaer Polytechnic
Institute DocLab
Troy, NY 12180
zhangxl@rpi.edu

Abstract

Two methods, Symbolic Indirect Correlation (SIC) and Style Constrained Classification (SCC), are proposed for recognizing handwritten Arabic and Chinese words and phrases. SIC reassembles variable-length segments of an unknown query that match similar segments of labeled reference words. Recognition is based on the correspondence between the order of the feature vectors and of the lexical transcript in both the query and the references. SIC implicitly incorporates language context in the form of letter n-grams. SCC is based on the notion that the style (distortion or noise) of a character is a good predictor of the distortions arising in other characters, even of a different class, from the same source. It is adaptive in the sense that with a long-enough field, its accuracy converges to that of a style-specific classifier trained on the writer of the unknown query. Neither SIC nor SCC requires the query words to appear among the references.

1 Introduction

From the perspective of character recognition, Arabic and Chinese are at the opposite ends of the spectrum. The former has a small alphabet with word-position dependent allographs, is quasi-cursive, and has "diacritics", ascenders and descenders. The latter has an indefinitely large number of classes (of which only the first ~20,000 have been coded), essentially word-level symbols (many with a radical-based substructure), and fixed-pitch block characters. Arabic strokes can be approximated by arcs of circles, while most Chinese strokes are straight, with a ~1:7 range in width (like brush strokes), and a flourish at the end. Unlike Arabic, Chinese does not have deliberate loops.

They also exhibit some commonalities. Both have been incorporated in the scripts used by other languages: Arabic in Urdu and Persian, Han in Japanese and Hangul, among many others. Both have traditional roots and forms dating back several thousand years, preserved in a large body of classical manuscripts, and have undergone considerable and diverse modifications in each host language and region of the world. Nevertheless, both scripts have preserved sufficient uniformity to link cultures which can no longer understand each other's speech. Their classical forms are prized and cultivated in calligraphy, which combines visual and language arts. Neither script has upper and lower case.

Industrial strength Arabic and Chinese OCR products must also be able to recognize Latin characters, "Arabic" (Indian) numerals, and Western punctuation. This introduces additional complexity, more because of the need to handle diverse, intermingled reading orders and output codes than because of the increased number of classes.

Many thousands of papers (the very first of which, coincidentally, is [1]) have been written on Chinese character recognition. By the time of our first survey [2] much of the research was reported in Chinese and Japanese publications. In our second survey [3] we found little new research in the West. Recent research collaboration with Professor C-L Liu at the Pattern Recognition Laboratory of the Chinese Academy of Science (CASIA), visits with Professor X. Ding at Tsinghua University, and a tour of Hanwang High Technology in Beijing acquainted us with the largest concentrations of character research activity in the world and some of China's thriving OCR industry.

Research on Arabic character recognition (actually on Farsi) began in the late sixties. Scattered projects, mainly by speakers of Arabic in the West, increased until the turn of the millenium, when research began to grow exponentially. Nearly one thousand reports have already been published, mostly in English and French. Nevertheless, work on Arabic OCR lags far behind Chinese OCR because of the lack of monolithic government and market support, and of large, publicly available databases. For a recent survey of the state-of-the-art in offline Arabic handwriting recognition please see [4].

Our team has over 100 years of sustained experience in character recognition, with over one hundred research publications on this topic to our credit. In addition to intra-departmental access to native speakers and writers of both Arabic and Chinese with a background in pattern recognition and signal and image processing, we have forged strong professional bonds with many of the leading researchers in both areas. We are convinced that extrapolating successful methods from Western

(including Russian) OCR is insufficient for either Arabic or Chinese because, ideally and optimally, every glyph of an entire document must be considered simultaneously before a label is assigned to any one of them. In practice, this notion translates to field classification, where glyphs that are difficult to recognize in isolation (or that cannot be isolated/segmented) are recognized in conjunction with several others.

Because of the wide range of different problems exhibited by the two scripts, we believe that tackling both simultaneously is a valuable strategy for research that will bring benefits not only to character recognition on other scripts (like those derived from Sanskrit), but also to the wider field of pattern recognition. Below, we outline how we propose to apply field classifiers that have already proved successful on easier tasks to Arabic and Chinese documents.

We bring two orthogonal ideas to the table: Symbolic Indirect Correlation (SIC) and Style Constrained Classification (SCC). The former recognizes unknown sequences of features (possibly spanning several characters) by finding and reassembling its constituent subsequences in the feature sequence representation of labeled reference text. The unknown word(s) need not be represented in the reference set, only their lexical constituents (i.e., symbol polygrams). Style-based classification, on the other hand, has been applied to distorted but segmented patterns. It maximizes the posterior probability of the field's feature vector of same-source words or phrases given the transcript, under the constraint of source or style specific statistical dependence between all the features of the field. Over the last decade, we have developed (and published) the necessary mathematical apparatus for field classification based on both SIC and SCC.

As is customary in many-class problems, we will use a hierarchical approach to reduce the number of candidate classes to which we apply the full power of our advanced methods. We believe that top-50 classification with less than 1% error on a candidate list of several thousand Arabic words or Chinese characters is within the state of the art, and that field classification can differentiate similar candidates in this reduced list.

We are not aware of any adequate handwritten test data with full context in either Arabic or Chinese. Our proposals for the essential characteristics of such a database were presented at SDIUT05 [5]. Even though the appropriate test data is not yet available, it is still possible to initiate this research on the currently available isolated word and character collections

2 Arabic Character Recognition

Symbolic indirect correlation (SIC) is a general approach to recognition of text that cannot be reliably segmented into characters, as is the case with most offline and online handwriting in non-hieroglyphic scripts.

SIC recognition is based on local matches between unsegmented patterns at both the feature and lexical levels. At the feature level, the unknown pattern is compared to a known (reference) string of features and the results are captured in the form of a match graph. Another matching process is used to find polygram co-occurrences between the lexical transcripts of the reference string and every class to be recognized. In a second-level matching, the order of feature co-occurrences is compared to the order of polygram co-occurrence in the lexical transcript of each class and the unknown pattern is given the label of the best matching lexical class.

SIC offers distinct advantages over prevailing approaches. It avoids the usual integrated segmentation-by-recognition loop. Unlike other whole-word recognition methods, SIC does not need feature-level samples of the words to be recognized. Finally, unlike methods based on Hidden Markov Models, it does not require estimation of an enormous number of parameters by a fragile bootstrap process. Furthermore, SIC can compensate for noisy features or inaccurate feature matching by increasing the length of the reference set.

We introduced SIC in [6,7] with a representation based on ordered bipartite graphs and established its advantages through simulations with a significant amount of noise. Later investigations showed that in the presence of excessive noise, the sub-graph isomorphism based approach to the second-level matching requires an unreasonably large reference set [8,9]. A maximum-likelihood approach [10] avoids this computational bottleneck in the second-level matching. Since this method seems promising for Arabic recognition, we describe in some detail how we build candidate solutions to the query; interested readers will find full technical details in [10,11].

The second level matching assigns the labels of the best-fitting segments in the reference set to each matching segment in the query. The assignment is constrained by the order of the (possibly overlapping) matches. The probability of each candidate solution to the query is computed as follows.

With a large enough reference set, the feature matches between the query and the "words" in the reference string cover most, if not all, of any query word. Further,

a feature match may or may not occur between the query and any given reference word or phrase; the same is true also for a lexical match (bigrams or higher polygrams) between the pair. Thus, when a candidate solution to the query is built by assigning a polygram to each matched feature segment in the query, one of four possible conditions applies to the assignment with respect to every reference word. The assigned polygram:

(1) occurs in the reference word and there is a segment match (*valid match*),

(2) does not occur in the reference word but there is a segment match (*spurious match*),

(3) occurs but there is no segment match (*missed match*), and

(4) does not occur and there is no segment match (*correct reject*).

These conditional probabilities can be estimated by matching the reference words against each other. Then, they are used to estimate the likelihood of each candidate solution and the solution with the maximum likelihood is chosen.

While our work on SIC so far has been restricted to English handwriting, we believe that it would apply well to Arabic handwriting because of the many common characteristics shared by the two. These include linear order of writing, strong baseline, and three well-defined zones (ascender, descender, and median). Other unique features of Arabic writing also speak favorably for SIC:

- Connection of adjacent letters is prescribed by rigorous rules in Arabic. The resulting connected components at the sub-word level (PAWs) may themselves be connected by hasty writers. The segment-free recognition of SIC has been demonstrated to work on cursive English writing.
- Different shapes of letters at the beginning, middle, and end of a word require only that sufficient instances of each kind be included in the reference set.
- Occasionally, Arabic writing breaks from the usual right-to-left order by placing two successive characters one on top of the other. If this happens with some consistency in the writing, a feature-level match of the compound character in the unknown word and the reference string would be correlated in SIC with the corresponding bigram in the second-level matching. Similar considerations apply to the recognition of letters that are sometimes written out of sequence by Arabic writers.

In order to substantiate these claims, we have recently initiated work on applying SIC to recognize offline handwriting using a sample of images from the database of handwritten Arabic town names [12]. At this point, we have only completed the first-level matching at the feature and lexical levels. The features were adopted from English handwriting with minor variations; we expect substantial improvement with feature sets specifically developed for Arabic writing, such as the one reported in [13].

In our preliminary explorations, we selected a reference set of eight town names (numbered 2, 7, 8, 29, 45, 48, 51 and 52), transcribed by writer ae07. We chose four other town names (numbered 1, 3, 14, and 42) by the same writer as query words. The latter had good bigram coverage by the reference set. We used the Smith-Waterman algorithm [14] to find the local alignments (matches) for feature-level matching. The algorithm uses a flexible cost function that allows for mismatches, insertions, and deletions and finds the optimal sequence of such steps needed to match the two subsequences. For a given cost function, it finds the strongest matches starting at every position of the query string against the reference string. False matches abound at shorter segments hence match-score thresholds are set to minimize the likelihood of a false match. Empirically, this is found to filter out most of the matches corresponding to unigrams and character fragments. The same algorithm was adapted to lexical matching of the transcripts of the query and reference strings.

Two examples of the lexical and feature match graphs are shown in Figures 1 and 2 to convey a sense of how the SIC approach might apply to Arabic handwriting recognition. The same query word, ae07_014, is matched with the reference word ae07_002 in Figure 1 and with the reference word ae07_045 in Figure 2. In each figure, part (a) shows the lexical match graph and part (b) shows the feature match graph. The lexical matching is carried out for exact bigram and higher-order n-gram matches and, typically, results in one or two matches. The feature matching typically yields many more edges for the same pair of words, even for a threshold value that is high enough to minimize single-character matches. The strength of a match is shown as a positive-integer weight of the corresponding edge in the graph; it is indicative of the extent of the matching segments in the feature domain. In the examples we have chosen to show only the top-three candidate edges in each case. For visualization, the strength of matches is denoted by the thickness of lines and line colors (magenta, gray, and brown in the decreasing order).

Figure 1 (a) shows that there is only one lexical match in this example: the bigram (aaA laB) at position 5 in the query word matches with the same bigram at position 5 in the reference word, where the positions are counted from right-to-left in accordance with the Arabic

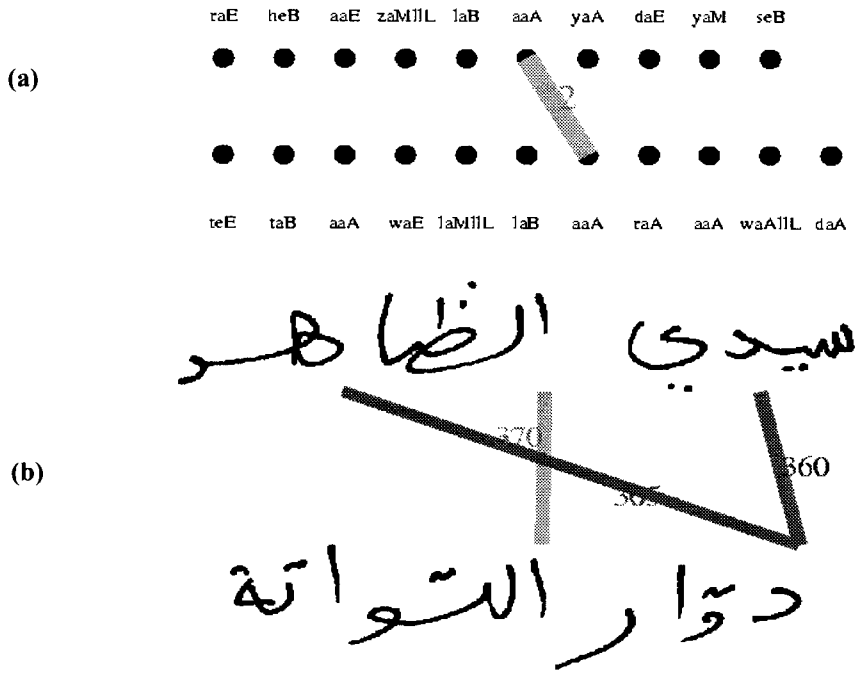


Figure 1: *Lexical and feature graphs of the query word ae07_014 and the reference word ae07_002.*

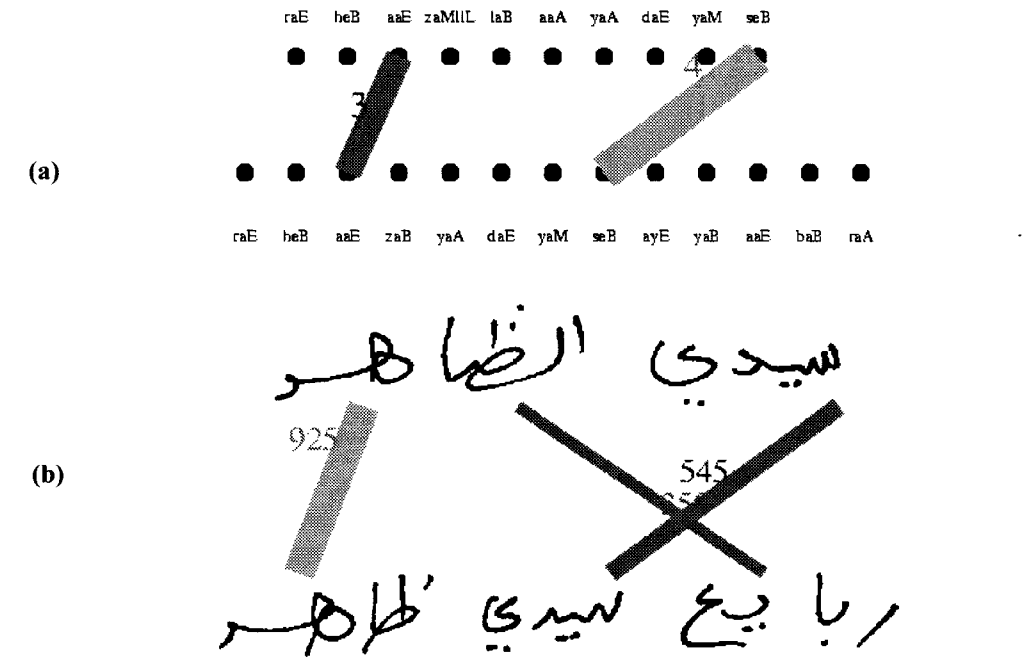


Figure 2: *Lexical and feature graphs of the query word ae07_014 and the reference word ae07_045*

writing convention. In Figure 1 (b), this lexical match is correctly identified by the strongest match, of strength 370, in the feature graph. However, the feature graph also includes two spurious edges, of strength 365 and 360 respectively, that do not have corresponding edges in the lexical graph.

In the second example, Figure 2(a) shows two lexical matches: a 4-gram at the beginning of the query word matching at position 5 in the reference word and a trigram matching at the end of both the words. Both of these are also found in the feature graph, in Figure 2(b), among the top-3 matches. However, the strongest edge, of strength 925, corresponds to the trigram and the next strongest edge, of strength 545, corresponds to the 4-gram. The third edge, of strength 250, is spurious. We note that because of both signal noise and variability in the character-widths, the strength of a correctly matched edge in the feature graph is only weakly correlated with the strength of its corresponding edge in the lexical graph.

Even though the feature set used in our examples is not particularly well adapted to Arabic, the feature matching process correctly picks the lexical matches in many cases. However, there are many spurious matches as well. Our second-level matching process, described in [10] is shown to be robust against a large number of spurious matches but at the expense of increased computation time. Therefore, we plan to explore a post-processing approach to eliminate some bad matches. The basic idea here is to use non-sequential features to screen the 2-D regions identified by every feature match. Hull [15] employs a similar idea in another context, to select candidates for whole-word recognition, even when they are printed in different fonts. Consider, for example, the feature graph in Figure 2(b) showing the top-3 matches. The image region in the query word corresponding to the weakest edge, which has the strength of 250, has a flat stroke with a diacritic mark above it while that corresponding to the reference word has diacritic marks below the stroke. It should be possible to reject this match based on 2-D features that summarize the dominant directions of each black pixel in different sub-regions of the two images.

3 Chinese Character Recognition

Almost any method can recognize neat handwritten Chinese with better than 85% accuracy, and newsprint at over 95%. Making allowance for the usual 2:1 reject/error trade-off, this implies that we must concentrate on 35% of the handwritten material and 15% of the print. These figures are based on the characters used in the People's Republic of China, which are somewhat harder to recognize than the characters used in Taiwan, Japan, and Korea because the simplifications fifty years ago removed many "redundant" strokes.

Printed Chinese characters are usually fixed-pitch, without ascenders or descenders, and all the characters fit into the same size, horizontally aligned, bounding boxes. Whether reading order is left-to-right or top-to-bottom is easily determined. Segmentation is, however, a major source of error in handwriting. Rushed writers connect and even overlap characters, do not adhere to a clear baseline, and cannot squeeze complex characters into bounding boxes that are ample for simpler characters. (However, some text produced by expert calligraphers is nearly indistinguishable from print, as exemplified in the Proceedings of the conferences on Computers and Chinese Input/Output Systems in the early 1970's.)

Both handwriting and print exhibit pairs (occasionally even triples) of characters with almost identical shapes but different meanings. (Some researchers deliberately exclude such confusion pairs from reported error statistics.) Human readers resolve such ambiguities through broad context. A far more restricted set of language constraints is also used in Chinese OCR. Dictionary (lexicon) look-up cannot be applied in the same way as in Western languages, but the extreme skew of the distribution of unigrams and of two- and three-character sequences can be readily exploited. We note, in particular, that the number of Chinese family and given names, where mistakes cannot be tolerated, is less than in most Western nations. Foreign names may be transliterated or printed in their native script.

We discussed a new approach to segmentation-free character recognition in the section on Arabic. Here we present style-constrained field classification, which is the only recourse when there is insufficient linguistic context. When we cannot read a letter, we look for easier-to-recognize instances of the same shape. Other instances of an unknown character may be easier to classify because there is less (or different) noise, or they are segmented better, or because there is more language context. Adaptive algorithms that benefit from typeface and writer intra-class consistency of this kind have been known for decades [16,17,18,19] but found their way into commercial systems only recently [20]. The scope of adaptation is typically a page: it is assumed that each page is written by a single person, or printed in a limited set of typefaces. A set of reliable prototypes is collected in a first pass, and the remaining problematic characters are recognized in one or more subsequent passes.

Some easily confused characters where adaptation can help are shown Table 1. (We use printed examples for ease of interpretation by readers who cannot read Chinese. The handwritten version of these characters are, of course, even more ambiguous.) However, adaptation works well only with long fields, where there are several samples of each class. In Chinese, much longer fields are needed than in alphabetic languages.

Table 1: *Two Han confusion pairs (ri/yue and dao/diao) in seven fonts. The left column and its transliterations are the font names. On the right are a few of the 7x6/2=21 possible different-font confusion pairs, on which conventional singlet classifier is likely to make errors.*

Chinese Font			Font Confusion
方正姚体	(fang zheng yao ti)	日 曰	
方正舒体	(fang zheng shu ti)	日 曰	日曰 / 曰曰
华文细黑	(hua wen xi hei)	日 曰	日曰 / 曰曰
华文行楷	(hua wen xing kai)	日 曰	日曰 / 曰曰
华文中宋	(hua wen zhong song)	日 曰	
新宋体	(xin song ti)	日 曰	
幼圆	(you yuan)	日 曰	
		(ri yue)	
方正姚体	(fang zheng yao ti)	刀 刁	
方正舒体	(fang zheng shu ti)	刀 刁	
华文细黑	(hua wen xi hei)	刀 刁	刀刁 / 刀刁
华文行楷	(hua wen xing kai)	刀 刁	刀刀 / 刀刁
华文中宋	(hua wen zhong song)	刀 刁	
新宋体	(xin song ti)	刀 刁	
幼圆	(you yuan)	刀 刁	
		(dao diao)	

Table 2: *Scenarios faced by a singlet classifier (above) and by a pair classifier (below), on two pairs of similar characters. When it is known that both characters are from the same style (font or writer), the confusions are more easily resolved by a style-constrained classifier that has additional information from another character in the same style.*

	Font1/ Font2	Font2/ Font1
Singlets:	日/曰	曰/日
	刁/刀	刁/刀
Pairs:	子日/曰子	子曰/日子
	刁钻/刀钻	刁钻/刀钻

We have recently demonstrated a much less intuitive aspect of local shape consistency that we call inter-class style. The underlying idea is simply that knowing how an individual writes a **g** or a **p** may help us predict how she may write a **q**. In fact, the shape of every class provides some information about every other class. In a statistical framework, we say that the features of one class are style-conditionally dependent on the features of another class. Abandoning the customary independence assumption leads to a more complex mathematical framework.

Nevertheless, the optimal maximum a posteriori (MAP) classifier can be formulated neatly [21,22,23]. In the last three years we demonstrated significant gains in accuracy through style-constrained field classification on both printed and hand-printed digits. We now propose to do the same for Chinese and Arabic. Table 2 suggests why pairs of Chinese characters are easier to classify than individual characters.

We note that printers, copiers, scanners and cameras also introduce useable style constraints. Human readers resort to field classification when necessary. Like humans, machines must also be enabled to do field classification, only when needed, because it is expensive. The number of field classes increases exponentially with field length. Whereas with numerals we used field lengths up to 5, for Chinese we propose to apply style constraints only to selected pairs and triples.

For the sake of completeness, we note that font recognition is an inefficient form of style-constrained classification. It generally requires separate features for font and class identification, is confused when fonts share some shapes, wastes statistical evidence on identifying the font when only class labels are wanted, and cannot accommodate font miscegenation.

4 Interaction

All OCR systems benefit from some human help, typically at the beginning or the end of the process. Scanning is almost always checked, because even current scanners occasionally bungle digitization. At the beginning, the operator may label some unusual characters, select a language model, or provide some general format information. He or she may also occasionally assist page segmentation. At the end, low-confidence labels are verified or corrected. When there are too many errors, the entire page may be keyed in instead of corrected.

Current OCR systems do not make the most efficient use of the operator, perhaps because such work is often outsourced and offshored. For urgent and critical applications, the operator may well be the end user. Workers with other primary missions are not likely to tolerate the repetitive routine of data entry personnel. The interaction must therefore take place wherever and whenever it is most effective and, above all, it should not be wasted. The software should attempt every task. The operator must, however, have the opportunity of correcting the result whenever necessary. Whether a particular error needs to be corrected requires the kind of judgment that at present machines lack.

We have made the case for a personal, mobile, multilingual support system at SDIUT 2004 [5], on interactive table interpretation at DAS06 [24], and on large scale document entry at DIAL06 [25]. We have argued that every operator action should result in some change in the configuration of the system that decreases the likelihood of the same situation occurring again. In other words, the system must improve with use. The defense rests.

References

[1] R. G. Casey and G. Nagy, "Recognition of Printed Chinese Characters," *IEEE Transactions on Electronic Computers,* vol. 15, pp. 91-101, February 1966.

[2] R. G. Casey and G. Nagy, "Chinese Character Recognition: A Twenty-five-year Perspective," in *Proc. International Conference on Pattern Recognition*, Rome, Italy, pp. 1023-1026, 1988.

[3] J. Kanai, Y. Liu, and G. Nagy, "An OCR-oriented Overview of Ideographic Writing Systems," in *Handbook of Character Recognition and Document Image Analysis*, H. Bunke and P. S. P. Wang, Eds.: World Scientific, 1997, pp. 285-304.

[4] L. M. Lorigo and V. Govindaraju, "Offline Arabic Handwriting Recognition: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 28, pp. 712-724, May 2006.

[5] D. Lopresti and G. Nagy, "Mobile Interactive Support System for Time-Critical Document Exploitation," in *Proc. Symposium on Document Image Understanding*, College Park, MD, pp. 111-119, 2005.

[6] G. Nagy, S. C. Seth, S. K. Mehta, and Y. Lin, "Indirect Symbolic Correlation Approach to Unsegmented Text Recognition," in *Proc. Conference on Computer Vision and Pattern Recognition Workshop on Document Image Analysis and Retrieval (DIAR'03)*, Madison, WI, p. 22, 2003.

[7] G. Nagy, D. P. Lopresti, M. Krishnamoorthy, Y. Lin, S. Seth, and S. Mehta, "A Nonparametric classifier for unsegmented text," in *Proc. SPIE*, San Jose, pp. 102-108, 2004.

[8] A. Joshi and G. Nagy, "Online Handwriting Recognition Using Time-Order of Lexical and Signal Co-Occurrences," in *Proc. 12th Biennial Conference of the International Graphonomics Society*, Salerno, Italy, pp. 201-205, 2005.

[9] D. Lopresti, A. Joshi, and G. Nagy, "Match Graph Generation for Symbolic Indirect Correlation," in *Proc. SPIE-IS&T Symposium on*

Document Recognition and Retrieval, Vol. 6067-06, San Jose, CA, 2006.

[10] A. Joshi, G. Nagy, D. Lopresti, and S. Seth, "A Maximum-Likelihood Approach to Symbolic Indirect Correlation," in *Proc. International Conference on Pattern Recognition*, Hong Kong, China, 2006.

[11] A. Joshi, "Symbolic Indirect Correlation Classifier," Rensselaer Polytechnic Institute,ECSE Department, Troy, NY, Ph. D. Thesis 2006.

[12] V. Märgner and M. Pechwitz.*IFN/ENIT-database: Database of Handwritten Arabic Words*. [Online].Available: http://www.ifnenit.com/index.htm.

[13] R. El-Hajj, L. Likforman-Sulem, and C. Mokbel, "Arabic Handwriting Recognition Using Baseline Dependant Features and Hidden Markov Modeling," in *Proc. Int. Conference on Document Analysis and Recognition, ICDAR*, pp. 893-897, 2005.

[14] T. F. Smith and M. S. Waterman, "Identification of common molecular sequences," *Journal of Molecular Biology*, vol. 147, pp. 195-197, 1981.

[15] J. J. Hull, "Incorporating Language Syntax in Visual Text Recognition with a Statistical Model," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 18, pp. 1251-1256, December 1996.

[16] G. Nagy and G. L. Shelton, "Self-Corrective Character Recognition System," *IEEE Transactions on Information Theory*, vol. 12, pp. 215-222, April 1966.

[17] H. S. Baird and G. Nagy, "A Self-correcting 100-font Classifier," in *Proc. SPIE Conference on Document Recognition and Retrieval*, San Jose, CA, pp. 106-115, 1994.

[18] Y. Xu and G. Nagy, "Prototype Extraction and Adaptive OCR," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 21, pp. 1280-1296, December 1999.

[19] T. K. Ho and G. Nagy, "OCR with no shape training," in *Proc. International Conference on Pattern Recognition*, Barcelona, Spain, pp. 27-30, 2000.

[20] I. Marosi and L. Tóth, "OCR Voting Methods for Recognizing Low Contrast Printed Documents," in *Proc. 2nd IEEE International Conference on Document Image Analysis for Libraries, DIAL 2006*, Lyon, France, pp. 108-115, 2006.

[21] S. Veeramachaneni and G. Nagy, "Adaptive classifiers for multisource OCR," *International Journal of Document Analysis and Recognition*, vol. 6, pp. 154-166, March 2004.

[22] S. Veeramachaneni and G. Nagy, "Style context with second order statistics," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 14-22, January 2005.

[23] P. Sarkar and G. Nagy, "Style consistent classification of isogenous patterns," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, pp. 88-98, January 2005.

[24] D. W. Embley, D. Lopresti, and G. Nagy, "Notes on Contemporary Table Recognition," in *Proc. Document Analysis Systems VII, 7th International Workshop, DAS 2006*, Nelson, New Zealand, pp. 164-175, 2006.

[25] G. Nagy and D. Lopresti, "Interactive Document Processing and Digital Libraries," in *Proc. 2nd IEEE International Conference on Document Image Analysis for Libraries, DIAL 2006*, Lyon, France, p. 8 pages, 2006.

Farsi Script Recognition: A Survey

Ching Y. Suen, Sara Izadi, Javad Sadri, Farshid Solimanpour
CENPARMI (Center for Pattern Recognition and Machine Intelligence), Computer Science Department, Concordia University, 1455 de Maisonneuve Blvd. West, Montreal, Quebec, Canada, H3G 1M8, Tel: (514)-848-2424-Ext:7950, Fax: (514)-848-2830
Emails:{suen, s_izadin, j_sadri, f_solima}@cs.concordia.ca

Abstract

In this paper, a brief history of the evolution of Farsi (Persian) Script and its connection to Arabic Script is presented. Similarities and dissimilarities between Farsi and Arabic scripts, from the OCR (Optical Character Recognition) point of view are discussed. Also, the state-of-the-art techniques in Farsi script recognition, as well as the challenges ahead are briefly reviewed.

1 Introduction

Farsi and Arabic are two important cursive scripts used mainly in the Middle East and some other neighboring countries. Farsi is the main language used in Iran and Afghanistan, and it is spoken by more than 110 million people, including some people in Tajikistan, and Pakistan. Arabic is spoken in all Arab countries, both in the Middle East and in Africa, and it is used by 234 million people worldwide. In Western countries, it is commonly thought that Farsi and Arabic scripts are exactly the same. However, minor yet important differences exist in their alphabets and their styles of writing. Due to these differences, a system adjusted for automatic recognition of one script might not perform well for the other one. While much research on Arabic recognition has been published and introduced internationally, most of the research in Farsi recognition has been presented only in Farsi Journals and Iranian conferences. To the best of our knowledge, research in Farsi script recognition has not yet been widely introduced to the research community.

This paper is organized as follows: Section 2 presents a brief history of the evolution of Farsi script, and its historic connection to Arabic script. In Section 4, similarities and dissimilarities among Farsi and Arabic scripts from the OCR point of view are highlighted. Section 4 presents the general structure of a Farsi Script recognition system. Sections 5-7 review some of the research efforts towards Farsi script segmentation/recognition. Section 08 introduces some of the existing databases for research on Farsi script recognition. Section 9 briefly lists some of the remaining challenges in Farsi script recognition, and draws the conclusion.

2 History of the evolution and propagation of the Farsi script

In this section, we briefly review the history of Persian Script and its connection to Arabic Script. Historically, evolution of the Persian language falls into three periods: Old, Middle, and Modern.

2.1 Old Persian Script (Old Persian cuneiform, from 550 to 330 BCE):

Old Persian script was a cuneiform type script dating from the time of the Achaemenid dynasties in Persia (6th–4th century B.C.E.) [1]. In that script, characters were made of strokes, which could be impressed upon soft materials by a stylus with an angled end. The Old Persian characters were inscribed in Syllabary. This means that each character had the value of a vowel or of a consonant plus a vowel. The old script consists of a total of 36 such characters plus 5 ideograms, one ligature of ideogram and case ending, the word-divider, and numerical symbols. Figure 1 shows the alphabet and numbers used in Old Persian script.

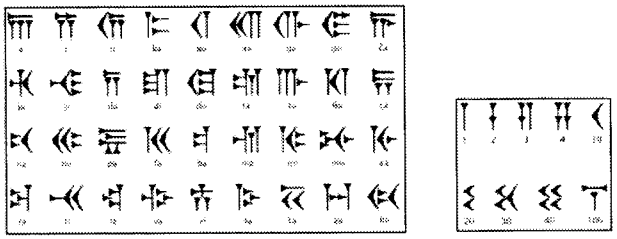


Figure 1 Old Persian script alphabet and numbers.

2.2 Middle Persian (or Pahlavi, from 300 B.C. to about 900 A.D.):

Middle Persian includes the Iranian dialects as they appeared from about 300 B.C.E. to about 900 A.D. Middle Persian is generally called Pahlavi (a derivative of the Old Persian word 'Parthian'). It was the language of quite a large body of Zoroastrian literature, the state religion of Sassanid in Iran.

Pahlavi script was originally developed from another script called Aramaic script. Book Pahlavi, the

most common form of the script, was a complicated writing system with 12 characters representing 24 sounds. The notable features of the Pahlavi script were: It was written from right to left in horizontal lines; Only some vowels were indicated; The letters used to represent those vowels had multiple pronunciations. This explains why Pahlavi script was complicated. In later forms of the Pahlavi script, attempts were made to improve the alphabet by adding diacritics and signs to the letters. An example of Pahlavi script is shown in Figure 2.

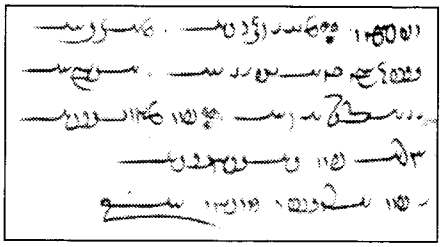


Figure 2 A sample of Pahlavi (Middle Persian) script

After the fall of the Sassanids by Arab Muslems (in the 7th century), the Pahlavi alphabet was still in use by the Zoroastrian community until the 10th century. However, because of the difficulty in reading and writing Pahlavi script, it gradually gave way to the Arabic script as the official language of politics and religion in Iran.

2.3 Modern Persian (Persian, variant of the Arabic alphabet):

After the conquest of Persia (Iran) in the 7th century (650 A.D.), the introduction of Islam brought a massive infusion of loaned words from Arabic to the Persian Language. Moreover, the Arabic alphabet was gradually adopted as the Persians' new script. In fact, at the start of the Islamic era, two types of Arabic scripts were in use. One was a rounded script called "Naskh". The other was a square and angular script called, "Kufic", considered elegant. In Iran, the skill gained in the art of calligraphy for Pahlavi script was employed in beautifying the newly adopted Arabic script. A system of handwriting script that contained several styles of writing was invented in the tenth century, which became a powerful tool in the development and standardization of cursive scripts. The new styles, although rounded, were considered of the acceptable elegancy to be used in the writing of the "Koran". Among those developed styles, "Ta'liq", a delicate script in which horizontal strokes of letters are elongated, was disseminated in the writing of Farsi. This style survived and evolved as a legible script, called "Nasta'liq", which has been commonly used in Farsi handwriting, since the 16th century. The current practice of writing Nasta'liq is heavily based on Kalhor's manner (an Iranian Calligraphist). He modified and adapted Nasta'liq to be easily used with printing machines. He also devised methods for teaching Nasta'liq, and specified clear proportional rules for it. Nasta'liq itself has a successor called "Shekasteh Nasta'liq", which is also currently in use in Iran. Examples of different styles of writing are shown in Figure 3.

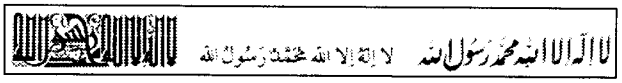


Figure 3 A sentence written in Kufic, Naskh, and Nasta'liq Styles (from left to right)

2.4 Propagation of Farsi script

Since the Islamic era, the Arabic script has been widely adopted for use in many of the world's languages, including those other than Arabic. In different regions, however, variant forms of the alphabet have been derived for the spoken languages. This adaptation includes further extensions to the alphabet and/or altered shapes of some of its letters. Propagation of the modern Persian script and Nasta'liq style to other countries dates back to 1526-1707 when the Mughal Empire used Persian as the court language during their rule over the Indian subcontinent. During this time, the Persian script and Nasta'liq style propagated in South Asia, including Pakistan, India, and Bangladesh. Their influences remain to this day. In Pakistan, Urdu is written in Nasta'liq, and this script can be seen there more than anywhere else in the world. In some cities in India, with large Urdu-speaking populations like Hyderabad, Lakhnau, and others, many street signs and such are written in Nasta'liq style. In Bangladesh, after Urdu ceased to remain an official language (1971 A.D.), only a few neighborhoods retained the influence of the Persian script and Nasta'liq style.

In Table 1, languages which are written in different forms of Arabic are shown. Among these languages, all of the Indian and Turkic languages (such as Azari, Uyghur), written in Arabic script, tended to use the adopted Persian modified alphabet or writing style.

Table 1 Languages adopted from the Arabic script

Arabic	Farsi	Malayalam	Sindhi
Baluchi	Kashmiri	Malay	Tajik
Brahui	Kirghiz	Panjabi	Uyghur
Dari	Kurdish	Pashto	Urdu
Azari			

3 Farsi and Arabic Similarities and dissimilarities

In this section, we briefly review the general characteristics and similarities/dissimilarities between Farsi and Arabic scripts that are in use today. In Iran, the evolution of the Arabic script has had some changes over time in the position of the letters and method of writing, even for similar characters. These will be further explained in the following subsections.

3.1 Common Characteristics of Farsi and Arabic scripts

The Arabic script and all of its derived forms are inherently cursive. Unlike Latin, block letters written in sequence do not stand for a word. Instead, the letters are joined along a writing line or baseline. Separating and segmentation of those letters is therefore more challenging than the separation of Latin letters. In both Persian and Arabic there are several letters that share the same basic form and differ only by a small complementary part. The complementary part could be a dot, a group of dots or a slanted bar. It can lie above, below or inside the letter.

The vowels for A, E and O have no letters and may be shown as a small diagonal under-bar stroke (for E), an over-bar stroke (for A), or a small comma (for O). Therefore, another characteristic of both Persian and Arabic scripts is the existence of some marks (diacritics) written above or below the letters. These diacritics indicate vowels, where consonant letters are connected together to make their pronunciation easier.

In both Persian and Arabic languages, the position of the character in the word and the previous character of the word (if there is any), are the factors that determine the shapes of the characters. All Persian letters (with seven exceptions)/(six in Arabic) can be connected to other letters from both the right and the left sides. The seven exceptional characters can only be connected to other letters from the right side. Therefore, if any of those seven letters appear in the middle of a word, there will be a gap in connectivity.

3.2 Farsi and Arabic Alphabets

The Farsi alphabet has four more letters than the Arabic alphabet. These letters represent phonemes that do not appear in Arabic phonology. For example, the Arabic language lacks a [p] phoneme, but Farsi has added its own letter to represent [p] in the script. 0Figure 4 shows the Farsi alphabet, which has 32 letters; however, the Arabic alphabet has only 28 letters. In the specified figure, the letters that are identified by a closed circle 0are not included in the Arabic alphabet. The letters in the Persian alphabet are derived from 18 shapes. These letters are distinguished by one dot (10 cases), two dots (3 cases) or three dots (5 cases) placed above or below the letter.

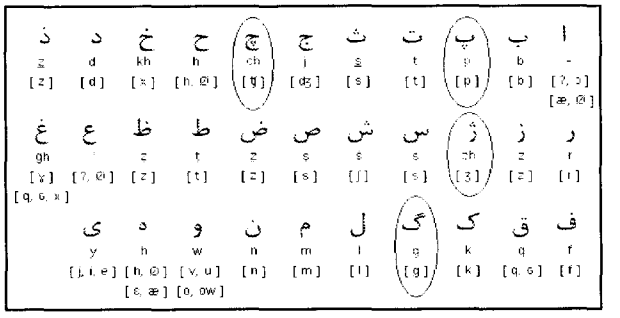


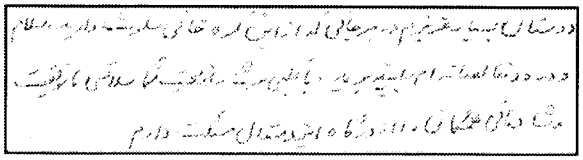
Figure 4 Farsi alphabet. Letters which have been identified by a closed circle are not included in the Arabic Alphabet.

In the Persian alphabet, similar to the Arabic alphabet, a letter can appear in up to four different forms: detached, initial, medial, and final. Table 2 shows all of these variations for all of the letters in the Persian alphabet. This characteristic of Persian and Arabic letters increases the effective size of the alphabet from 32 letters up to 114 (for Arabic, from 28 to 100).

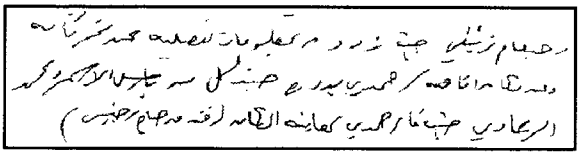
Table 2 Four different shapes of letters in the Persian alphabet.

Isolated	Initial	Medial	Final	Roman	Name	Isolated	Initial	Medial	Final	Roman	Name
ا	ا	ـا	ـا	á	alef	ص	صـ	ـصـ	ـص	ṣ	sád
ب	بـ	ـبـ	ـب	b	be	ض	ضـ	ـضـ	ـض	ḍ	zád
پ	پـ	ـپـ	ـپ	p	pe	ط	طـ	ـطـ	ـط	ṭ	tá
ت	تـ	ـتـ	ـت	t	te	ظ	ظـ	ـظـ	ـظ	ẓ	zá
ث	ثـ	ـثـ	ـث	th	se	ع	عـ	ـعـ	ـع	`	ayn
ج	جـ	ـجـ	ـج	j	jim	غ	غـ	ـغـ	ـغ	gh	ghayn
چ	چـ	ـچـ	ـچ	ch	che	ف	فـ	ـفـ	ـف	f	fe
ح	حـ	ـحـ	ـح	ḥ	ḥe	ق	قـ	ـقـ	ـق	q	qáf
خ	خـ	ـخـ	ـخ	kh	khe	ک	کـ	ـکـ	ـک	k	káf
د	د	ـد	ـد	d	dál	گ	گـ	ـگـ	ـگ	g	gáf
ذ	ذ	ـذ	ـذ	dh	zál	ل	لـ	ـلـ	ـل	l	lám
ر	ر	ـر	ـر	r	re	م	مـ	ـمـ	ـم	m	mím
ز	ز	ـز	ـز	z	ze	ن	نـ	ـنـ	ـن	n	nún
ژ	ژ	ـژ	ـژ	zh	zhe	و	و	ـو	ـو	v/ú	váv
س	سـ	ـسـ	ـس	s	sin	ه	هـ	ـهـ	ـه	h	he
ش	شـ	ـشـ	ـش	sh	shin	ی	یـ	ـیـ	ـی	y/i	ye

In order to better compare Farsi and Arabic scripts, two pieces of the texts written in these two languages are shown in Figure 5.



(a) Sample of Farsi Script



(b) Sample of Arabic Script

Figure 5 Samples of handwriting in Farsi and Arabic.

3.3 Farsi Digits

Although Persian and Arabic digits (so-called Hindi or Indian digits) look very similar, there are important differences between the Persian and Arabic handwriting of digits [2]. Persian handwritten digits normally form 13 classes of digits because of the two different ways of writing the numbers 0, 4 and 6 (Figure 6-b). At the same time, Arabic handwritten digits normally form 11 classes because of the two different ways of writing the number 3 (Figure 6-c). Also the number 5 is written differently in these two languages.

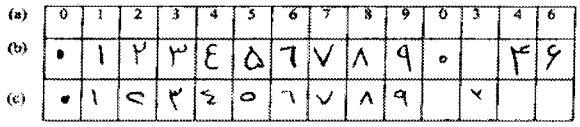


Figure 6 (a) Latin, (b) Farsi, (c) Arabic digits

4 Script recognition systems

Like other scripts, a system for the recognition of Farsi script should have an architecture as shown in Figure 7. We briefly explain each module here.

Pre-processing: This module may include some image processing operations such as binarization of an input image, noise reduction, smoothing, filling or other image transformations such as normalization.

Segmentation: This module tries to find connected or touching digits and to separate them by introducing some segmentation paths.

Recognition: In this module, there is a classifier that assigns class labels and corresponding confidence values to the segmented digits.

Verification: In some methods, there is an optional verifier that verifies the decisions made by the recognizer, and it tries to reduce the error rate of the recognizer.

In the following sections, we will review some important research activities on the development of offline/online Farsi script recognition systems.

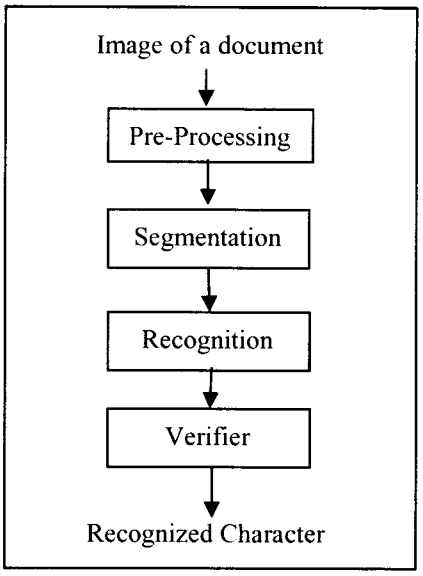


Figure 7 Block diagram of a script recognition system.

5 Pre-processing and segmentation

A typical script recognition system consists of the main blocks depicted in Section 4. Preprocessing in offline recognition usually includes noise removal, smoothing the contours, slant/skew correction and size normalization.

Here, we present a method for segmentation of handwritten numeral strings in Farsi [3] which is based on combining features from the foreground and background of the image. The novelty of the method is that extracting features in the foreground is based on a new method called skeleton tracing, and in the background it is based on combining the vertical top and bottom projection profiles and their skeletons. Based on a combination of this information, and some global information from the string image, this algorithm tries to construct the best segmentation paths for separating the touched digits.

5.1 Generating foreground features

To find feature points on the foreground (black pixels) of each connected component, a new algorithm based on skeleton tracing is introduced. First, by Zhang-Suen thinning algorithm [4], the skeleton of each connected component is extracted (see Figure 8-b). Then, on this skeleton, two points called starting and ending points (denoted by S and E, respectively) are found. In each connected component, starting, and ending points are the points with the smallest and

largest values of the x coordinates, respectively. Then, from the starting point (S), the foreground skeleton is traversed in two different directions: first clockwise, and then counter-clockwise until both traverses reach the ending point (E). In Figure 8-c, we define the traverse in the clockwise direction as top-skeleton, and the traverse in the counter-clockwise as bottom-skeleton. Actually top/bottom skeletons are subsets of pixels of the original (foreground) skeleton, which are visited during the traversals in the clockwise/counter-clockwise directions. When the algorithm traverses the top/bottom skeletons, it looks for intersection points (IPs) which are visited on the skeleton. IPs are points which have more than two connected branches. Corresponding to each visit of any intersection point in the skeletons, there is an angle where its bisector can be found. The intersections of these bisectors with the contour of the connected component are obtained, and they are denoted by □ in Figure 8-d. These points form our foreground feature points.

Figure 8 (a) Original image (4 touching 3), (b) Original skeleton, starting point (S), ending point (E) are depicted by □, (c) From starting point (S), skeleton is traversed in two different directions (clockwise: dashed arrows, and counter-clockwise: dotted arrows) to the end point (E), (d) Mapping of intersection points on the outer contour by bisectors to form foreground-features (denoted by □).

5.2 Generating background features

Considering the background (white pixels) of a connected component, its skeleton is found (see Figure 9-c). Then, on the background skeleton, all the end points are extracted (points which have one black neighbor and they are denoted by □ in Figure 9-d).

Figure 9 (a) Pre-processed and slant corrected image, (b) Background region, (c) Top/Bottom background skeletons, (d) Top/Bottom-background-skeletons after removing parts that are lower/higher than middle line.

5.3 Constructing segmentation paths

After finding all the feature points in the foreground (denoted by □ in Figure 8-d), and background features (denoted by □ in Figure 9-d), these feature points from top to bottom, or from bottom to top, are assigned together alternatively to construct all possible segmentation paths (denoted by dashed lines in Figure 10 [5].

Figure 10 Feature points on the background, and foreground (from the top and bottom) are matched, and assigned together to construct segmentation paths.

6 Offline Farsi recognition

The earliest work in this area was conducted by two Iranian researchers: Parhami and Taraghi who presented a Farsi text recognition system in 1980 ([6], [7]). Their system was demonstrated in newspaper headlines; sub-words were segmented and recognized according to features such as concavities, loops, and connectivity.

This section, briefly reviews some of the research efforts towards offline Farsi script recognition based on the types of features used for recognizing isolated Farsi digits or letters: structural features, and statistical features.

6.1 Structural features

Structural features are those such as loops, branch-points, endpoints, and dots, which are based on the instinctive aspects of writing. To compute these features the skeleton of the text image should often be extracted, as shown in Figure 11.

Figure 11 (a) Original, (b) Structural features shown on "skeleton" image extracted using the algorithm presented in [8].

One of the reasons that structural features are more common for the recognition of Farsi scripts than of Latin scripts is that the primary shape of many Farsi letters are alike and only their number of dots and the locations of their dots are different. Therefore, to differentiate such letters, structural features are used for capturing dot information explicitly. Samples of papers that have used structural features for Farsi digit recognition are listed in Table 3.

Table 3 Papers that have used structural features for Farsi script recognition.

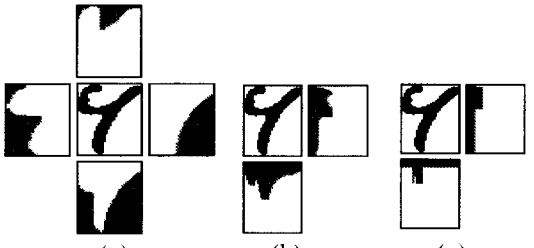
Research	PP	Classifier	Database	RR
[9]	Thinning	MLP-NN	Private Farsi Digits 4800 Samples 10 Classes	94.44%
[10]	Thinning	SVM + RBF Kernel	Private Farsi Chars 3200 Samples 20 Classes	96.92%

6.2 Statistical features

Statistical features are numerical measures, computed over the images or regions of the images. Some examples of statistical features include: pixel densities, histograms of chain code directions, moments, and Fourier descriptors. In Figure 12, samples of outer profiles, projection histograms and crossing counts are shown, which were used in [9], and [12]. Some papers that have used statistical features are listed in Table 4.

Table 4 Papers that have used statistical features for Farsi script recognition.

Research	Pre-Processing	Feature(s)	Classifier	Database	Recognition Rate
[9]	None	Projection Histogram, Outer Profiles, Crossing Counts, Size	SVM + RBF & Polynomial Kernels	Private Farsi Digits 9000 Samples	Poly.: 99.44% RBF: 99.57%
[12]	None	Projection Histogram, Outer Profiles, Crossing Counts	SVM + RBF Kernel	CENPARMI Farsi Digits 8000 Samples	97.32%
[13]	Size Normalization and Thinning	Haar wavelet, 3 resolutions	MLP-NN	Private Farsi digits except "2" & "0". 3840 Samples	Digits: 92.33% Chars: 91.81%
[14]	Size /Rotation Normalization	Fractal coding: domain coordinates, brightness offset, an affine transformation.		Private Farsi digits except "2" & "0". 3840 Samples	Digits: 91.37% Chars: 87.26%
[15]	Thinning	12-Segment template centered by the control point	MLP-NN	Private Farsi Digits 730 Samples	97.6%
[16]	Size Normalization and Thinning	Haar wavelet, 3 resolutions.	SVM + RBF Kernel	Private Farsi Digits except "2" & "0": 3840 Samples Farsi Chars : 6080 Samples	MLP/Char: 92.33% MLP/Digits: 91.81% SVM/Char: 93.75% SVM/Digits: 92.44%
[17]	Thinning	Coordinate Features/PCA	MLP-NN	Private Farsi Digits 2430 Samples	87%, 85%, 83%, and 91% when combined



(a) (b) (c)

Figure 12 (a) Outer profiles from four directions, (b) projection histograms from two directions, (c) Crossing counts from two directions.

7 Farsi online recognition

In online handwritten recognition, the process of automatic recognition takes place while a person writes. Online handwriting recognition systems deal with handwriting captured by some sort of digitizer. The electronic tablet is a typical capturing device, and contains a pointing device (usually a pen) and a sensing device. Online handwritten recognition can be used with tablet PCs, hand-held computers, personal digital assistants (PDA's), smart phones, etc. There is a comprehensive review of digitizing technology in [18]. The movement of the pen is sensed at equal time or distance intervals. The spatial position $(x(t), y(t))$ and the indications of pen-up/pen-down switching are usually recorded as online data by a software application. Dynamic or temporal information is the main difference between the online and offline data, and it addresses the number of strokes and the order of the strokes. A typical online recognition system consists of the main blocks depicted in Section 4. Here, the research in online Persian recognition is explained under this general pattern recognition system.

Preprocessing in online handwriting recognition usually includes smoothing, data reduction, normalization and de-hooking. So far, there have been few efforts in segmentation for online Persian script. By thresholding on a change of the angle along the trajectory isolated characters were segmented in [19].

A loop detection method was added in [20] to enhance the result of segmentation.

For the recognition of online Persian script, rule-based classifiers and neural networks (NN) are the mainly used classifiers. A review of these systems is presented in Sections 7.1 and 7.2.

7.1 Rule- based recognition methods

Rule-based recognition has been used in many script recognition applications. A recognition rate of 99.6% was reported by T. Al-Sheikh et al. [21] for their proposed hierarchical rule-based method for online isolated Arabic character recognition. The first level of the hierarchy was based on the number of strokes of the characters, and there was further division by using rule-based classification. Relying on an error-free segmentation, this method is unable to handle noisy data or writing variations. A hybrid system using geometrical and structural features and fuzzy techniques was presented by F.Bouslama et al. [22]. Alimi et al. [23] minimized error in an on-line recognition system for isolated Arabic characters in their proposed system. A bank of prototypes was developed for the coded characters based on directional codes and positional codes. For the recognition of a new pattern, the distance between the pattern and prototype was minimized using dynamic programming. A recognition rate of 96% was reported. Fuzzy set theory was used for both segmentation and classification in [19]. After segmentation, each segment was characterized by four features which were described by sets of fuzzy membership functions. A recognition rate of 95% was reported for the isolated characters and some character combinations. In [20], a recognition rate of 75% for the database in [24] was achieved for isolated characters by using fuzzy logic recognition.

Although rule-based classification is very successful for small numbers of classes, choosing the values of the model parameters for a large number of classes is a challenging task. As the size of symbols grows, correspondingly more rules will be needed, and hence the inference is slower and not suitable for real time applications.

7.2 Neural Network-based recognition methods

Neural networks (NN) are able to learn complex nonlinear input-output relationships, use sequential training procedures, and adapt themselves to the data. The multilayer perceptron (MLP) neural network and Self-Organizing Map (SOM/Kohonen-Network) are the two most widely used NNs in pattern recognition and classification. N. Mezghani et al. used a Kohonen neural network to recognize the basic shape (17 classes were considered) of online isolated Arabic characters [25]. The database they used consisted of 432 samples per class. By combining two separately trained Kohonen maps with a tangent vector and Fourier descriptor features, local and global character descriptions were taken into account in the classification, and a recognition rate of 88.38% was achieved. Pruning and filtering the maps, after being trained, improved the recognition accuracy to 93.54%. The authors in [26] investigated the usefulness of the Kullback–Leibler divergence and the Hellinger distance (as a replacement for the traditionally-used Euclidean distance) in similarity measurements of the feature vectors for training Kohonen maps. The best achieved recognition rate showed an improvement of 0.5% over the Euclidean distance by using the Hellinger distance. The Persian isolated characters were also recognized in [27] by using MLP NN. Isolated Persian letters were divided into 12 classes based on the number of dots, shapes and locations of their diacritics. After recognition of the complementary parts and their locations by two MLP NN's, the character's main shape would be analyzed by another NN if further recognition was needed. The reported recognition rate was 93.9% (for 4,144 letters) using the database in [24]. This method assumes that the character's main body is written in the first stroke. This assumption, although valid in most of the cases, is not always guaranteed.

8 Databases

There are many examples of widely used databases in the field of handwriting recognition such as: NIST (English isolated digits) [28], CEDAR (city names, state names, zip codes, and alphanumeric characters) [29], CENPARMI (isolated English digits) [30], UNIPEN (isolated English digits) [31], ETL9 (Japanese characters) [32], and PE92 (Korean characters) [33].

Recently, some databases were released for Arabic handwriting recognition such as IFN/ENIT Arabic city names in 2002 [34] and CENPARMI Arabic Cheques in 2003 [35]. Among the latter two, CENPARMI's database consists of legal and courtesy amounts on real cheques, and isolated digits. As stated before, the digits in Arabic and Farsi are alike; hence, CENPARMI's Indian digits and courtesy amounts database can also be used for Farsi handwritten numeral recognition. "Indian digits" are the numeric digits normally used in Farsi and Arabic writing, as opposed to the "Arabic numerals" used in Latin scripts.

There is no standard Farsi database available for

researchers, to date; however, our team has developed a standard set of handwritten databases, set to be released and presented at the 10th International Workshop on Frontiers in Handwriting Recognition (IWFHR), which will be held in France, on October 23-26, 2006 [12]. This set of databases consists of isolated Farsi digits (18000 samples), isolated Farsi letters (11900 samples), Farsi numerals (7350 samples), Farsi legal amounts (8575 samples), Farsi dates (175 samples) and a small set of English digits (3500 samples) written by 175 different writers. The English digits are included in order to capture the style of writing by those Iranians who are not native English speakers.

The only online Persian database was developed in 2004 by contribution of 124 writers [27]. One thousand of the most frequent sub-words were extracted from the online archives of two newspapers, containing more than 300,000 words as candidates for data collection. About 124 samples of one hundred of the most frequent sub-words along with isolated digits, and isolated letters were collected.

9 Conclusion and challenges ahead

This paper reviews the evolution of Farsi (Persian) script and its relation to Arabic script from the OCR (Optical Character Recognition) point of view. A brief review of the state-of-the-art techniques in offline and online Farsi script recognition as well as segmentation and databases is also presented. However, there are still many unsolved problems in Farsi and Arabic recognition systems (handwritten and typewritten).

Unfortunately, there has been no adequate standard database for offline and offline Farsi recognition available to date; therefore, there has not been any benchmark available for researchers to compare their methodologies. Methods have been mainly tested on the small and private databases collected by authors and have not always been always available to others.

Cursive handwriting makes segmentation of words and numerical strings a difficult challenge in Farsi. There has been little research in the segmentation of handwritten words and numerals in Farsi. A large part of the work in both offline and online recognition consists of isolated character recognition, which is hard to generalize for the cursive word recognition and related real world applications.

Finally, the lack of comprehensive surveys on Farsi recognition (recognition methodologies, segmentation, databases, etc.) sometimes yields to repetition of research by different groups. Hence, we hope that this work will be a good motivation for researchers in order to unify and improve future research in Farsi script recognition.

References

[1] "Wikipedia, The Free Encyclopedia," Wikimedia Foundation Inc., Aug 2006 http://en.wikipedia.org/wiki/Main_Page.

[2] J. Sadri, C. Y. Suen, T. D. Bui , " Application of Support Vector Machines for recognition of handwritten Arabic/Persian digits," *Proceeding of the Second Conference on Machine Vision and Image Processing & Applications (MVIP)*, Vol. 1, Feb. 2003, Iran, pp. 300-307.

[3] J. Sadri, C. Y. Suen, and T. D. Bui, "Segmentation of handwritten numeral strings in Farsi and English languages", *Proceedings of Third Iranian Conference on Machine Vision and Image Processing & Applications (MVIP) 2005*, Vol. 1, Feb. 2005, Tehran, Iran, pp.305-311.

[4] T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Communication ACM*, Vol. 27, No. 3, 1984, pp. 236-239.

[5] J. Sadri, C. Y. Suen, and T. D. Bui, "A new approach for segmentation and recognition of handwritten numeral strings", *Proceeding of IS&T/SPIE International Symposium on Electronic Imaging (EI)*, Vol. 5657, USA, Jan. 2005, pp. 92-100.

[6] B. Parhami and M. Taraghi, "Automatic recognition of printed Farsi texts," *Proceedings of the Conference on Pattern Recognition*, Oxford, England, 1980.

[7] B. Parhami and M. Taraghi, "Automatic recognition of printed Farsi texts," *Pattern Recognition*, Vol. 14, No. 1-6, 1981, pp. 395-403.

[8] Y. S. Chen and W. H. Hsu, "A new parallel thinning algorithm for binary image," *Proceedings of National Computer Symposium*, 1985, pp. 295-299.

[9] M. Zeyaratban, K. Faez, S. Mozzafari, M. Azvaji, "Presenting a new structural method based on partitioning thinned image for recognition of Handwritten Farsi/Arabic numerals," *Proceedings of the Third Conference on Machine Vision, Image Processing and Applications*, In Farsi, Vol. 1, Iran, Feb 2005, pp.76-82.

[10] M. Dehghan, K. Faez, "Farsi Handwritten Character Recognition with Moment Invariants," *Proceedings of 13th International Conference on Digital Signal Processing*, Vol. 2, Greece, 1997,

pp. 507-510.

[11] H. Soltanzadeh and M. Rahmati, "Recognition of Persian handwritten digits using image profiles of multiple orientations," *Pattern Recognition Letters*, 2004, Vol. 25, pp. 1569–1576.

[12] F. Solimanpour, J. Sadri, and C. Y. Suen, "Standard databases for recognition of handwritten digits, numerical strings, legal amounts, letters and dates in Farsi language", *To be appeard in the Proceedings of the 10th Int'l Workshop on Frontiers in Handwriting Recognition*, Oct 2006.

[13] A. Mowlaei, K. Faez and A.T. Haghighat, "Feature extraction with wavelet transform for recognition of isolated handwritten Farsi/Arabic characters and numerals," *Proceedings of 14th International Conference on Digital Signal Processing*, Vol. 2, July 2002, pp. 923-926.

[14] S. Mozaffari, K. Faez, and H. Rashidy-Kanan, "Recognition of isolated handwritten Farsi/Arabic alphanumeric using fractal codes," *IEEE Proceedings of Southwest Symposium on Image Analysis and Interpretation*, 2004, pp. 104-108.

[15] A. Harifi and A. Aghagolzadeh, "A new pattern for handwritten Persian/Arabic digit recognition," *International Journal of Information Technology*, Vol. 1, No. 4, 2004, pp. 293-296.

[16] A. Mowlaei and K. Faez, "Recognition of isolated handwritten Persian/Arabic characters and numerals using support vector machines," *IEEE Workshop on Neural Networks for Signal Processing*, 2003, pp. 547-554.

[17] S. H. Nabavi Karizi, R. Ebrahimpour, E. Kabir, "the application of combining classifiers in recognition of Farsi handwritten digits," *Proceedings of the Third Conference on Machine Vision, Image Processing and Applications (MVIP 2005)*, In Farsi, Vol. 1, 2005, pp. 115-119.

[18] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, In Farsi, Vol. 12, Issue 8, Aug. 1990, pp. 787 - 808

[19] R. Halavati, S. B. Souraki, and M. Soleymani, "Persian Online Handwriting Recognition *Using Fuzzy Modeling," International Fuzzy Systems Association World Congress (IFSA) 2005*, Beijing, China, July 2005.

[20] M.S. Baghshah, "A novel fuzzy approach to recognition of online Persian handwriting," *Proceedings of 5th International Conference on Intelligent Systems Design and Applications* (ISDA) 2005, Sep. 2005, pp. 268-273.

[21] T. Al-Sheikh and S. El-Taweel, "Real-time Arabic Handwritten Character recognition," *Pattern recognition*, Vol. 23, Issue. 12, 1990, pp. 1323–1332.

[22] F. Bouslama and A. Amin, "Pen-based recognition system of Arabic character utilizing structural and fuzzy techniques", *Second International Conference on Knowledge-Based Intelligent Electronic Systems*, 1998, pp 76-85.

[23] A. M. Alimi and O. A. Ghorbel, "The analysis of error in an on-line recognition system of Arabic handwritten character," *Proceedings of International Conference on Document Analysis and Recognition (ICDAR) 1995*, Vol. 2, Aug 1995, pp. 890–893.

[24] S. M. Razavi and E. Kabir, "A Data base for Online Persian Handwritten recognition," *6th Conference on Intelligent Systems*, In Farsi, Kerman, Iran, 2004.

[25] N. Mezghani, M. Cheriet, and A. Mitiche, "Combination of pruned Kohonen maps for on-line Arabic character recognition," *International Conference on Document Analysis and Retrieval (ICDAR) 2003*, Edinburgh, pp. 900-905.

[26] N. Mezghani, A. Mitiche, and M. Cheriet, "A new representation of shape and its use for superior performance in on-line Arabic character recognition by an associative memory," *International Journal on Document Analysis and Recognition (IJDAR) 2005*, Vol. 7, No. 4, 2005, pp. 201-210.

[27] S. M. Razavi and E. Kabir, "Online Persian Isolated character recognition," *The Third Conference on Machine Vision, Image Processing & Applications (MVIP) 2005*, In Farsi, Vol. 1, Tehran, Iran, Feb, 2005, pp.83-89.

[28] R. Wilkinson, J. Geist, S. Janet, P. Grother, C. Burges, R. Creecy, B. Hammond, J. Hull, N. Larsen, T. Vogl, and C. Wilson, "The first census optical character recognition systems conference #NISTIR 4912," *The U.S. Bureau of Census and the National Institute of Standards and Technology*, Gaithersburg, MD, 1992.

[29] J. Hull, "A database for handwritten text recognition research," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, May 1994, Vol. 16, Issue 5, pp. 550–554.

[30] C. Y. Suen, C. Nadal, R. Legault, T. Mai, and L.

Lam, "Computer recognition of unconstrained handwritten numerals," *Proceedings of the IEEE*, Vol. 7, Issue 80, 1992, Pages 1162–1180.

[31] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet, "Unipen project of on-line data exchange and benchmarks," *Proceedings of the 12th IAPR Int. Conf on Pattern Recognition*, Oct. 1994, pp. 29–33.

[32] F. Jelinek, "Self-organized language modeling for speech recognition," In A. Waibel and K.-F. Lee, editors, *Readings in Speech Recognition*, Morgan Kaufmann Publishers Inc., 1990, pp. 450–506.

[33] D. Kim, Y. Hwang, S. Park, E. Kim, S. Paek, and S. Bang, "Handwritten Korean character image database PE92," *Proceedings of the Second Int. Conference on Document Analysis and Recognition*, 1993, pp. 470–473.

[34] M. Pechwitz, S. Snoussi Maddouri, V. Märgner, N. Ellouze , and H. Amiri, "IFN/ENIT-database of handwritten Arabic words", *Proceedings of Colloque International Francophone sur l'Écrit et le Document (CIFE)*, 2002, pp. 129-136.

[35] Y. Al-Ohali, M. Cheriet, and C. Suen, "Databases for recognition of handwritten Arabic cheques," *Pattern Recognition*, Vol. 36, 2003, pp. 111-121.

Arabic Handwriting Recognition and Application to Ancient Documents

Liana M. Lorigo
Center for Unified Biometrics and Sensors
Department of Computer Science and Engineering
University at Buffalo
Amherst, NY 14228

Abstract

The automatic recognition of text on scanned images has enabled many applications such as searching for words in large volumes of documents, automatic sorting of postal mail, and convenient editing of previously printed documents. The domain of handwriting in the Arabic script presents unique technical challenges and has been addressed more recently than other domains. First, this document describes the state of the art in Arabic handwriting recognition (AHR). The description includes background, technical methods, and discussion of trends in the field. Second, this document describes current related projects at the Center for Unified Biometrics and Sensors (CUBS) at the University at Buffalo. They focus on recognition and analysis for ancient handwritten Arabic documents. Challenges are illustrated.

1 Introduction

Many different methods for Arabic handwriting recognition have been proposed and applied to various types of images. The first portion of this paper provides a review of representative methods and background on aspects of Arabic handwriting recognition. It largely follows from [1]. Background includes the tasks of skeletonization, contour extraction, and baseline detection. Structural and statistical features will be explained, likewise holistic and segmentation-based paradigms. Statistical recognition engines will be described briefly. Specific AHR algorithms are described and compared.

A recognition system can be either "on-line" or "off-line". It is "on-line" if the sequence of points traced by the pen is available, as with electronic personal data assistants that require the user to write on a screen. It is "off-line" if it is applied to images of previously written text. This discussion is restricted to off-line systems.

The second portion of this paper describes current projects at the Center for Unified Biometrics and Sensors (CUBS) at the University at Buffalo that focus on ancient handwritten Arabic documents.

2 Background

2.1 Algorithm Strategies

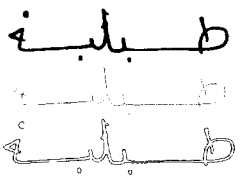


Figure 1: An image, skeleton (using [2]), contour.

The image is often converted to a more concise representation prior to recognition (Figure 1). A *skeleton* is a one-pixel thick representation showing the centerlines of the text. Skeletonization, or "thinning", facilitates shape classification and feature detection. An alternative is the Freeman chain code of the border ("contour") of the text [2, 3]. Chain code stores the absolute position of the first pixel and the relative positions of successive pixels along the contour. Difficulties with thinning include possible mislocalization of features and ambiguities particular to each algorithm. The contour approach avoids these difficulties since no shape information is lost.

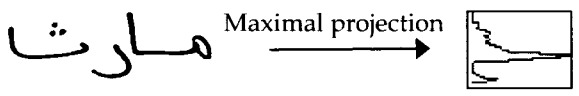


Figure 2: Projection for baseline detection.

A common step is the detection of the baseline. The standard approach is the projection of the binary image of a word or line of text onto a vertical line. The baseline can be detected as the maximal peak (Figure 2). Other approaches were presented by Pechwitz and Märgner [4] and Farooq *et al.* [5].

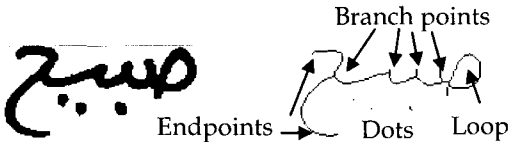


Figure 3: Image, structural features on skeleton [2].

The distinction between structural features and statistical features is helpful in developing and evaluating methods. *Structural* features are intuitive aspects of writing, such as loops, branch-points, endpoints, and dots. They are often, but not necessarily, computed from a skeleton of the text image as shown in Figure 3. Many Arabic letters share common primary shapes, differing only in the number of dots and whether the dots are above or below the primary shape. Structural features are a natural method for capturing dot information explicitly, which is required to differentiate such letters. This perspective may be a reason that structural features remain more common for the recognition of Arabic script than for that of Latin script. *Statistical* features are numerical measures computed over images or regions of images. They include, but are not limited to, pixel densities, histograms of chain code directions, moments, and Fourier descriptors.

Both image-level [6, 7] and structural [8-10] features have been applied to AHR (respective examples: [11], [12-14]). The former places the burden of processing on the recognizer, while the latter involves more processing at the feature detection stage. The pixel-based approach is more common for machine-print than for handwritten Arabic, which often uses structural or hybrid approaches. This situation may be due to the greater variability in handwriting. More training data would be needed for image-level features to model handwritten character shapes than printed character shapes. Conclusions about the comparative efficacy of the approaches for handwriting are not yet possible because large testing databases are not yet used throughout the field.

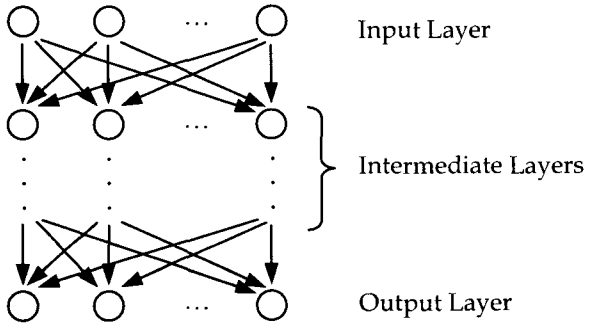


Figure 4: ANNs may have any number of elements (circles) per layer and any number of layers.

Many methodologies from the fields of machine learning and pattern recognition are used in AHR. Artificial neural networks (ANNs) consist of simple processing elements and a high degree of interconnection [15]. The weights within the elements are learned from training data. The elements are organized into an initial input layer, intermediate "hidden" layers, and a final output layer (Figure 4). The final layer gives a character or word choice in this task.

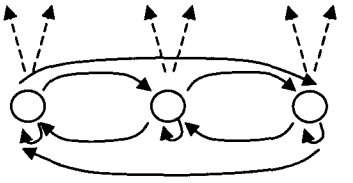


Figure 5: HMM with three states, two possible symbols at each.

Hidden Markov models (HMMs) are also used [16]. Conventional HMMs model one-dimensional sequences of data and contain states and probabilities for transitioning between them according to an observed sequence of data or "observations". Assume that, at each time step, the system was in one of n possible states and produced one of m possible observation symbols, the choice depending on probabilities associated with that state (Figure 5). The goal is to reconstruct the state sequence ("path") from the observations to learn the meaning of the data. For OCR, the observations could be sets of pixel values and states could represent parts of letters. An alternative to finding a path in a single model is accepting the most probable of several models. See [17] for a 2000 survey of the use of HMMs in Arabic character recognition.

Approaches can be either "holistic" or segmentation-based. "Holistic" means that words are processed as a whole without segmentation into characters or strokes [18]. In segmentation-based approaches, whole or partial characters are recognized individually after they have been extracted from the text.

2.2 Frames

A popular general approach for machine-print recognition and handwriting recognition is the frame-based approach. "Frames" are vertical strips of the image of a single line of text (Figure 6). Frames are considered sequentially along the line of text in the direction of left to right (e.g., English) or right to left (e.g. Arabic) depending on the script. Treating each frame as a single observation enables the two-dimensional image to be considered as a one-dimensional signal. This perspective enables the direct application of techniques used previously for speech recognition, especially HMMs. In a

straightforward example, the intensities of the pixels in the frames can comprise the observation feature vector.

Figure 6: Illustration of frame-based approach.

BBN explored this approach for script-independent character recognition of machine-printed text. Pixel intensities are used directly, without knowledge of letter or glyph shapes is needed. The same recognizer, which followed from their speech recognition work, was trained and tested for many scripts including Latin, Arabic, Chinese, and Japanese [6, 19, 20]. Only the lexicon, language model, and training data depended on the language. In 1999 they presented a system for English and Arabic in which the lexicon was unlimited [7]. The DARPA Arabic OCR Corpus was used for testing.

3 Recognition Methods

The recognition process can be partitioned into the sequence of pre-processing, conversion to new representation, segmentation, feature extraction, and recognition (Table 1, Figure 7). "Features" are not necessarily structural or pre-computed, but are any values passed to the recognizer. This framework is a guide for discussion only, and some systems do not follow it precisely. Feature extraction and recognition steps are described below. Approaches are organized according to the features and recognizers used.

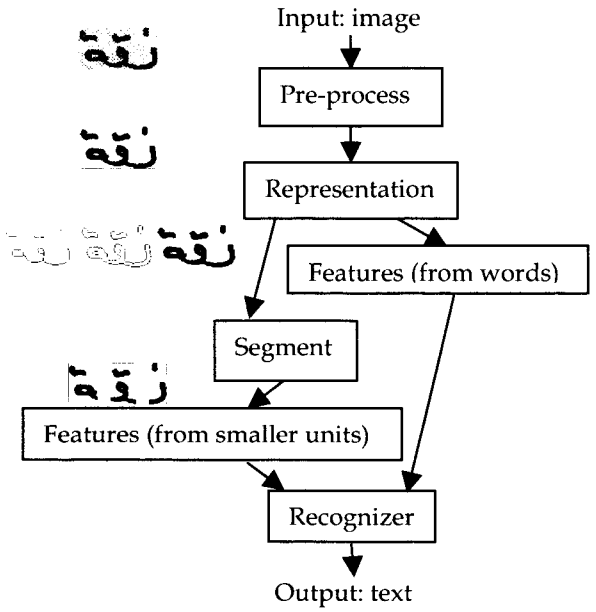


Figure 7: Generalized Framework for AHR.

Component	Description
Pre-processing	Noise removal, text detection, similar
Representation	Skeletons, contours, pixels, or other
Segmentation	Partitioning words or sub-words into characters, strokes, or other units
Features	Shape attributes, pixels, or other information passed to the recognizer
Recognizer	Algorithm that identifies text

Table 1:Components of Recognition Framework

3.1 Features

Table 2 lists the features used in the various algorithms, and system descriptions follow.

Publication	Features
El-Hajj 2005 [21]	Pixel densities, transitions, and concavities in frames and with respect to baselines; frame derivatives; centers of gravity with respect to baselines
Mozaffari 2005 [22]	Average and variance of X and Y changes in portions of the skeleton
Safabakhsh 2005 [23]	Fourier descriptors, numbers of loops, height/width ratios, pixel densities, positions of right and left connections
Alma'adeed 2004 [24]	Ascenders, descenders, structural features, frame-based features
Haraty 2002-2004 [25-28]	Loops, end/turning/junction points, widths, heights, row/column transitions and densities, contour heights
Souici-Meslati, Farah 2004 [13, 29, 30]	Loops, dots, connected components, ascenders, descenders
Amin 2003 [12]	Loops, dots, hamza, lines, open curves, and their relationships
Clocksin 2003 [31]	Moment functions applied to image and polar transform image
Khorsheed 2003 [14]	Dots, loops, endpoints, branch/cross/turning points, lines
Pechwitz 2003 [11]	Pixel intensites from pre-processed image
Snoussi Maddouri 2002 [32]	Ascenders, descenders, loops, dots, sub-word positions, normalized Fourier descriptors
Dehghan 2001 [33]	Histograms of slopes along contour
Dehghani 2001 [34]	Slope, curvature, number of active pixels using a two-dimensional pattern
Fahmy 2001 [35]	End/turning/junction points, complementary characters, loops
Miled 2001 [36]	Concavities and inclinations for

	ascenders and descenders, densities and shapes for diacritics
Abuhaiba 1994, 1998 [37, 38]	Dots, loops, strokes, links, directions, intersections
Amin 1996 [39]	Loop positions and types, line positions and directions
Olivier 1996 [40]	Loops and relative locations, heights, and sizes of parts of characters
Al-Yousefi 1992 [41]	Statistics from moments of horizontal and vertical projections
Goraine 1992 [42]	Stroke directions, sub-word positions, loops, secondary strokes, dots
Almuallim 1987 [43]	Strokes' endpoints, lengths, frames, connection points, others

Table 2: Features

In 1987, Almuallim and Yamaguchi proposed one of the first methods for Arabic handwriting recognition [43]. It used the skeleton representation and structural features for word recognition. Words were segmented into "strokes" which were classified and combined into characters according to the features. The recognizer was the set of classification rules. The method achieved a recognition rate of 91% on 400 words by two writers. To our knowledge, the method was the first to focus on text that was not pre-segmented.

In 1992, Al-Yousefi and Udpa introduced a statistical approach for the recognition of isolated Arabic characters [41]. It included the segmentation of each character into primary and secondary parts (such as dots and small markings) and normalization by moments of vertical and horizontal projections. The features were nine measurements of kurtosis, skew, and relationships of moments, and the recognizer was a quadratic Bayesian classifier. Test data included machine-printed and handwritten characters, but only 10 samples were used on the handwritten side.

Goraine *et al.* presented a structural approach in 1992 [42]. It operated on whole words and was applied to typewritten and handwritten words. After segmentation points were estimated from skeletons, structural features and a rule-based recognizer identified each letter. A dictionary was used to confirm or correct the results. In the handwriting recognition test, the system obtained a 90% recognition rate on 180 words comprised of about 600 characters. The three writers were asked to write neatly in a pre-specified font.

Related work by Clocksin and Fernando in 2003 addressed the domain of Syriac manuscripts [31]. Also a West Semitic language, Syriac is less grammatically complex than Arabic and was a primary language for theology, science, and literature from the 3rd century AD to the 7th century AD. The system used full image representation of individual characters and sets of features based on moments. A segmentation method based on vertical and horizontal projections and run-lengths was described, but recognition rates were given for pre-segmented characters. The recognizer was a support vector machine with tenfold cross-validation. The highest rate attained was 91% using features from the character image and its polar transform image.

In 2005, El-Hajj *et al.* demonstrated the benefit of features based on upper and lower baselines, within the context of frame-based features with an HMM recognizer [21]. This context was used in the BBN system for machine-print discussed above [6]. El-Hajj *et al.*, however, included features measuring densities, transitions, and concavities in zones defined by the detected baselines. The system was tested on the IFN/ENIT database minus those names that have fewer than eight images, leaving 21,500 images for testing. For each of four experiments, the system was trained on three of the four image sets and tested on the remaining set. Recognition rates ranged from 85.45% to 87.20%. In their experiments, the addition of the baseline-dependent features to similar measurements that do not use those zones significantly improved recognition.

Also in 2005, Mozaffari *et al.* proposed a method for the recognition of Arabic numeric characters which is structural and also uses statistical features [22]. Endpoints and intersection points were detected on a skeleton then used to partition it into primitives. Eight statistical features were computed on each primitive, the features for all primitives were concatenated, and the result was normalized for length. Nearest-neighbor was used for classification. Eight digits were tested, and 280 image of each were used for training and 200 for testing. The digits were written by over 200 writers collectively. The recognition rate was 94.44%.

3.2 Recognition Engine

Table 3 lists some recognizers used in AHR systems.

Publication	**Recognition Engine**
El-Hajj 2005 [21]	Character HMMs of four states, jumps of at most one state, and a mixture of three Gaussians
Mozaffari 2005 [22]	Nearest-neighbor
Safabakhsh 2005 [23]	Continuous-density variable-duration HMM
Alma'adeed 2004 [24]	Rules with structural features and HMM with frame-based features
Farah 2004 [29]	Parallel combination of ANN, K-nearest-neighbor, fuzzy K-NN
Haraty 2002-2004 [25-28]	Multiple neural networks
Souici-Meslati 2004 [13, 30]	ANNs constructed from rules
Al-Shaher 2003 [44]	Expectation-maximization
Amin 2003 [12]	Rules via logic programming
Clocksin 2003 [31]	Support vector machine with

	tenfold cross-validation
Khorsheed 2003 [14]	One HMM from 32 character HMMs with unlimited jumps
Pechwitz 2003 [11]	Word models made from 160 semi-continuous character HMMs
Snoussi Maddouri 2002 [32]	Four-layer transparent neural network
Dehghan 2001 [33]	One discrete HMM for each class
Dehghani 2001 [34]	Two HMMs per character, for horizontal/vertical projections
Fahmy 2001 [35]	Neural network
Miled 2001 [36]	Planar HMMs
Abuhaiba 1998 [37]	Fuzzy sequential machine
Amin 1996 [39]	Five-layer neural network
Abuhaiba 1994 [38]	Rules to fit trees to graph models
Al-Yousefi 1992 [41]	Quadratic Bayesian classifier
Goraine 1992 [42]	Rules to classify strokes, characters
Almuallim 1987 [43]	Rules to join strokes into characters

Table 3: Recognizers

3.2.1 Rules

Several previously described systems used rules for recognition [24, 38, 42, 43]. To automate this paradigm, Amin presented an automatic technique to learn rules for isolated characters (2003) [12]. Structural features including open curves in several directions were detected from the Freeman code representation of the skeleton of each character, and the relationships were determined with Inductive Logic Programming (ILP). The reader is referred to [45] for further information on ILP. Test data consisted of 40 samples of 120 different characters by different writers, with 30 character samples used for training and 10 for testing for most experiments. A character recognition rate of 86.65% was obtained.

3.2.2 Artificial Neural Networks

Many variations of ANNs have been used in this field. In 2001 Fahmy and Al Ali proposed a system based on ANNs with structural features [35]. Pre-processing steps included slope correction and slant correction. Features were detected from skeletons and fed into a neural network. A 69.7% word recognition rate was obtained on 600 words written by one writer.

Snoussi Maddouri *et al.* used a "transparent" four-layer neural network on images of words from bank checks (2002) [32]. Here, "transparency" means that the layers had intuitive meanings: primitives, letters, sub-words, and words. Pre-processing included slant correction before baseline detection. Global features including ascenders, descenders, loops, dots, and position were used for "contextual" segmentation, which refers to segmentation into zones based on features. Global features and local Fourier descriptors were fed to the network. A 97% word-level recognition rate was achieved on 2,070 images with a lexicon of 70 words. The combination of global and local features was like the mechanism by which people read. We recognize many words without examining individual letters, and if that fails we examine letters.

Haraty and Ghaddar proposed the use of two neural networks to classify previously segmented characters (2003) [26, 28]. Their method used a skeleton representation and structural and quantitative features such as the number and density of black pixels and the numbers of endpoints, loops, corner points, and branch points. On 2,132 characters, the recognition rate was over 73%. A prior segmentation system used the same representation and similar features to over-segment words and a neural network to confirm or reject proposed breakpoints [25]. Also addressed was the task of horizontal segmentation, which is needed when one letter is above another (here, "ligatures") [27]. Two networks were used, and success rates were 79% for ligature identification and 91% for validation of potential horizontal segmentation points.

3.2.3 Hidden Markov Models

HMMs have also been applied in diverse ways. Miled and Ben Amara combined the algorithm of [40] with a planar hidden Markov model (PHMM) to recognize machine-printed and handwritten words in 2001 [36]. They chose the planar model to handle the planar nature of writing and the situation in which one letter is directly above another.

In 2001, Dehghan *et al.* presented an HMM-based system whose features were histograms of Freeman chain code directions in regions of vertical frames [33]. No segmentation was used. There was one discrete HMM for each city class. The system achieved a 65% word-level recognition rate without the use of contextual information on a database of more than 17,000 images of the 198 names of cities in Iran.

A 2003 approach by Pechwitz and Märgner used 160 semi-continuous HMMs representing the characters or shapes [11]. It thinned each word and used columns of pixels in the blurred thinned image as features. The models were combined into a word model for each of 946 valid city names. The system obtained an 89% word-level recognition rate using the IFN/ENIT database (26,459 images of Tunisian city-names).

Khorsheed applied an HMM recognizer with image skeletonization to the recognition of text in an ancient manuscript (2003) [14]. No segmentation was done. One HMM was constructed from 32 individual character HMMs, each with unrestricted jump margin. Structural features were used, and recognition rate was 87% (72%) with (without) spell-check. The rate for the correct result being in the top five choices was 97% (81%). The test set was 405 character samples of a

single font, extracted from a single manuscript.

Safabakhsh and Adibi applied a continuous-density variable-duration hidden Markov model [46] to the recognition of handwritten Persian words in the Nastaaligh style (2005) [23]. This style contains many vertically overlapping letters and sloped letter sequences, which present problems for the ordering of characters and for baseline detection. Their system removed ascenders and descenders before the primary recognition stage to avoid incorrect orderings and was baseline-independent. Words were over-segmented into pseudo-characters using local minima of their upper contour, similar to [40]. Eight features were computed for each pseudo-character (Table 6). The HMM was path-discriminant and included 25 character states each of which was divided into up to four sub-states to indicate position-dependent shapes. The lexicon consisted of 50 words chosen to include all characters and compound forms, and the training set contained two 50-word scripts from each of seven writers. On a test set of two 50-word scripts from two different writers and omitting words that showed error in an earlier stage of the method, the system achieved a 69% recognition rate with 5 iterations of the recognition step and a 91% rate with 20 iterations. The rates were 52.38% and 90.48% on 21 words not in the lexicon.

3.2.4 Hybrids

In 2004 Alma'adeed *et al.* combined a rule-based recognizer with a set of HMMs to recognize words in a bank-check lexicon of 47 words [24]. Pre-processing normalized the text with respect to slant, slope, and letter height. A skeleton representation normalized for stroke width, and no segmentation was done. The rule-based engine used ascenders, descenders, and other structural features to separate the data into groups of words (reduce the lexicon), and an HMM classifier for each group used frame-based features to determine the word. To train the HMMs, words were separated into letters or sub-letters that were transformed into feature vectors and partitioned by a clustering algorithm. In testing, the feature vectors were obtained by vector-quantizing observation vectors obtained from frames of the image. The HMMs had 55 possible states, corresponding to letters or sub-letters in the dataset, and codebook sizes between 80 and 120. The system was tested on about 4,700 words collectively written by 100 writers, excluding about 10% of the words due to errors in baseline detection and pre-processing. A near 60% recognition rate was achieved. An earlier version obtained a 45% recognition rate (2002) [47].

Souici-Meslati and Sellami presented a hybrid approach to the recognition of literal amounts on bank checks in 2004 [13]. The recognizer was a neural network whose structure was defined by a rule-based method. Pre-processing included binarization, smoothing, normalization for word size, and baseline detection. The representation was Freeman chain code of the text's contour, and the features were loops, dots, connected components, ascenders, and descenders. Segmentation was not performed. Data for training and testing consisted of 480 and 1200 words respectively. The system obtained a 93% recognition rate, outperforming separate neural network and rule-based systems which each obtained a rate of approximately 85%. Also in 2004, this group proposed another method for this task [29]. The features were still structural and the representation chain code, but the recognizer differed. Three classifiers ran in parallel: neural network, k-nearest-neighbor, and fuzzy k-nearest-neighbor. The outputs were combined by word-level score summation and syntactic post-processing to obtain a valid phrase. 1200 words by 100 writers were used for training and 3,600 words for testing. The recognition rate was 96%, about 4% above the average of the individual classifiers. For both methods, the lexicon contained 48 words. Thirdly, this group presented a system to recognize city names in Algerian postal addresses (Souici *et al.*, 2004 [30]). The recognizer was a knowledge-based neural network like in [13]. The recognition rate was 92% with a 55-word lexicon. Separate training and testing datasets each contained 550 words (each of 55 words written by 10 writers).

In 2005, Farah *et al.* extended the work of [29] to study the effects of multi-classifier systems on recognition rates [48]. Separate ANNs processed structural and statistical features, and eight classifier combination methods were tested. The highest rate of 95.2% was observed with an ANN for combination. This rate surpassed the 89.3% observed with one ANN processing the two feature types collectively. Also, higher rates were achieved when both initial ANNs were trained on the same 2400-word dataset than when each was trained on half of that set. Test data was a separate set of 2400 words, and all words were literal amounts.

4 Discussion

The combination of statistical and structural information in AHR systems emerges as a theme. Knowledge-based neural network systems, frame-based HMMs whose features are guided by baselines, and statistical features detected after segmentation according to structural features are recent examples. So too is the use of purely statistical recognizers after structural features have been used to partition the words into groups for lexicon reduction.

Most systems address recognition of restrictive vocabularies such as that of words on bank checks or the names of cities in a region. Research areas include the extension to free handwriting. This will include larger scale work in lexicon reduction. Linguistic information has not been applied in this task and will be needed for the recognition of free handwriting. Morphology is the area of linguistics that investigates word formation,

including affixes, roots, and patterns. Arabic is a Semitic language and, as such, exhibits a systematic yet complex structure. This structure can assist AHR. Finally, there is large demand for recognition systems for historical documents. These documents are often handwritten and pose additional challenges. Little research has been done in this domain.

5 Ancient Documents Research

CUBS is currently exploring the recognition of ancient handwriting documents in the Arabic script. This work builds on and complements our tradition of research in handwriting recognition and recent work in recognition for the Indic scripts of Devanagari and Tamil. The challenges posed by ancient Arabic and Persian documents are illustrated on a dataset prepared by New York University's Afghanistan Digital Library. This dataset is also used to illustrate challenges of a stylized font.

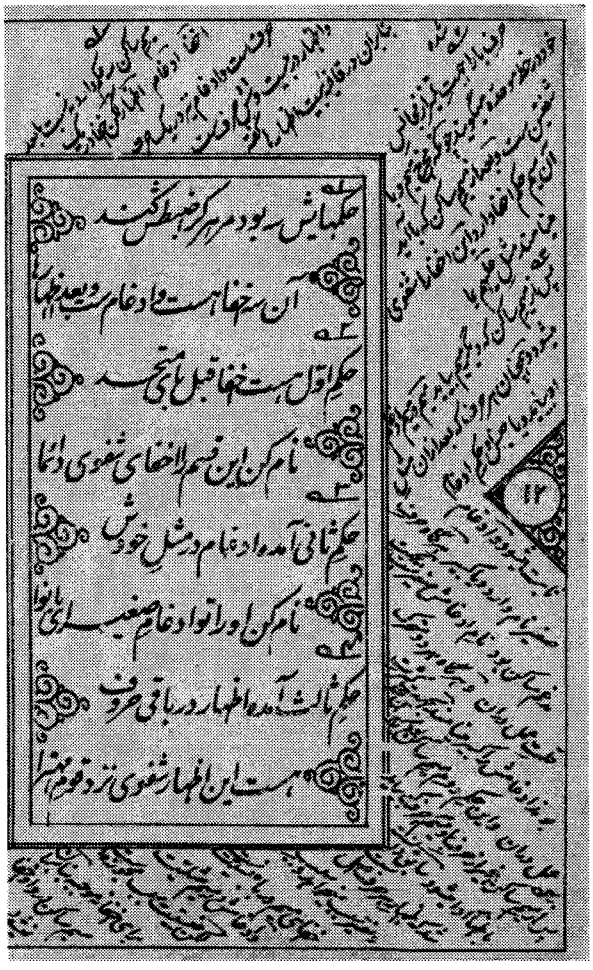


Figure 8: Sample lithograph image from Afghanistan Digital Library.

The text in the images with which we are working is generally neat and of good contrast, but other challenges exist. Figure 8 shows an image from the Afghanistan Digital Library. The page pictured is from a book. Most of the pages in the book show the basic layout of text in an ornately marked rectangle with additional text written at fixed orientations outside the rectangle. In such cases, the text areas and angles could be learned or input by a user prior to the recognition stage.

Figure 9: Script variations. (a) Diacritical marks on Arabic script. (b) Font with more slope and overlap, and non-linear writing with text above primary line.

The library also contains images of text in two scripts interspersed. In one example, the narrative is written in the Persian script, while portions are in Arabic script and fully marked with diacritics. These scripts are easily distinguishable to a person. For example, Figure 9a shows Arabic and Figure 9b Persian. They must be distinguished since letter and ligature appearances vary systematically. Note the marks in the Arabic example and the slope and overlap in the Persian example.

Work we have begun in this domain includes binarization, slant detection, segmentation, feature detection, and transcript mapping. The transcript mapping task for contemporary Arabic handwriting was a focus of ours during the past year. We have recently begun to develop it for the domain of ancient documents. Transcript mapping is the task of aligning a typed text file (e.g., ASCII file) with an image of text. This is useful for generating databases for research and for enabling an interface to highlight words of interest in a computerized search of ancient documents. For the former, we can have a text file of a pre-defined paragraph or set of paragraphs, and many people can write this text by hand. A transcript mapping program could automatically crop out the images of each word to yield a fully annotated set of images of single written words, which is needed to train a recognizer. This process alleviates the tedious effort of manually dragging a rectangle around each word in each page image and typing in the correct annotation.

For example, each connected sequence of writing can be detected and indicated with a bounding box. The mapping from bounding boxes to subwords from the text file would be trivial if each box uniquely matched a subword in the transcript. However, places where there is a break in the writing or where adjacent subwords touch cause the correct mapping to be nontrivial. Our transcript mapping algorithm is currently in submission to another conference so cannot be described here. New experiments on ancient documents are ongoing.

Acknowledgments

The work described herein is joint work with Dr. Venu Govindaraju. The images of handwritten Arabic script in Figure 1, Figure 2, Figure 3, and Figure 7 were obtained from the IFN/ENIT database.

The Afghanistan Digital Library is a joint project of New York University Libraries, NYU's Department of Middle Eastern and Islamic Studies, and the university's Hagop Kevorkian Center. It was undertaken with the support of the National Endowment for the Humanities.

References

[1]L. Lorigo and V. Govindaraju, "Off-line Arabic Handwriting Recognition: A Survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, pp. 712-724, 2006.

[2]H. Freeman, "On the Encoding of Arbitrary Geometric Configurations," *IRE Trans. Electron. Comput.*, vol. EC-10, pp. 260-268, 1961.

[3]S. Madhvanath, G. Kim, and V. Govindaraju, "Chain code processing for handwritten word recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, pp. 928-932, 1999.

[4]M. Pechwitz and V. Märgner, "Baseline estimation for Arabic handwritten words," in *Proc. 8th International Workshop on Frontiers in Handwriting Recognition*, 2002, pp. 479-484.

[5]F. Farooq, V. Govindaraju, and M. Perrone, "Pre-processing Methods for Handwritten Arabic Documents," in *Proc. International Conference on Document Analysis and Recognition*. Seoul, Korea, 2005, pp. 267-271.

[6]J. Makhoul, R. Schwartz, C. Lapre, and I. Bazzi, "A Script-Independent Methodology for Optical Character Recognition," *Pattern Recognition*, vol. 31, pp. 1285-1294, 1998.

[7]I. Bazzi, R. Schwartz, and J. Makhoul, "An Omnifont Open-Vocabulary OCR System for English and Arabic," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 21, pp. 495-504, 1999.

[8]S. A. Al-Qahtani and M. S. Khorsheed, "A HTK-Based System to Recognise Arabic Script," in *Proc. 4th IASTED International Conference on Visualization, Imaging, and Image Processing*. Marbella, Spain: ACTA Press, 2004.

[9]S. A. Al-Qahtani and M. S. Khorsheed, "An Omni-font HTK-Based Arabic Recognition System," in *Proc. 8th IASTED International Conference on Artificial Intelligence and Soft Computing*. Marbella Spain: ACTA Press, 2004.

[10]M. S. Khorsheed and W. F. Clocksin, "Structural Features of Cursive Arabic Script," in *Proc. British Machine Vision Conference*, 1999, pp. 422-431.

[11]M. Pechwitz and V. Märgner, "HMM based approach for handwritten Arabic word recognition using the IFN/ENIT - database," in *Proc. International Conference on Document Analysis and Recognition*, 2003, pp. 890-894.

[12]A. Amin, "Recognition of hand-printed characters based on structural description and inductive logic programming," *Pattern Recognition Letters*, vol. 24, pp. 3187-3196, 2003.

[13]L. Souici-Meslati and M. Sellami, "A Hybrid Approach for Arabic Literal Amounts Recognition," *The Arabian Journal for Science and Engineering*, vol. 29, pp. 177-194, 2004.

[14]M. S. Khorsheed, "Recognising handwritten Arabic manuscripts using a single hidden Markov model," *Pattern Recognition Letters*, vol. 24, pp. 2235-2242, 2003.

[15]R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, Second ed: John Wiley & Sons, Inc., 2001.

[16]L. R. Rabiner and B. H. Juang, "An Introduction to Hidden Markov Models," in *IEEE ASSP Magazine*, vol. 3, 1986, pp. 4-16.

[17]N. Ben Amara, A. Belaïd, and N. Ellouze, "Utilisation des modèles markoviens en reconnaissance de l'écriture arabe: Etat de l'art," in *Proc. Colloque International Francophone sur l'Ecrit et le Document*. Lyon, France, 2000.

[18]S. Madhvanath and V. Govindaraju, "The role of holistic paradigms in handwritten word recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 23, pp. 149-164, 2001.

[19]P. Natarajan, M. Decerbo, T. Keller, R. Schwartz, and J. Makhoul, "Porting the BBN BYBLOS OCR System to New Languages," in *Proc. Symposium on Document Image Understanding Technology*. Greenbelt, Maryland, USA, 2003, pp. 47-52.

[20]M. Decerbo, P. Natarajan, R. Prasad, E. MacRostie, and A. Ravindran, "Performance

Improvements to the BBN Byblos OCR System," in *Proc. International Conference on Document Analysis and Recognition*. Seoul, Korea, 2005, pp. 411-415.

[21]R. El-Hajj, L. Likforman-Sulem, and C. Mokbel, "Arabic Handwriting Recognition Using Baseline Dependant Features and Hidden Markov Modeling," in *Proc. International Conference on Document Analysis and Recognition*. Seoul, Korea, 2005, pp. 893-897.

[22]S. Mozaffari, K. Faez, and M. Ziaratban, "Structural Decomposition and Statistical Description of Farsi/Arabic Handwritten Numeric Characters," in *Proc. International Conference on Document Analysis and Recognition*. Seoul, Korea, 2005, pp. 237-241.

[23]R. Safabakhsh and P. Adibi, "Nastaaligh Handwritten Word Recognition Using a Continuous-Density Variable-Duration HMM," *The Arabian Journal for Science and Engineering*, vol. 30, pp. 95-118, 2005.

[24]S. Alma'adeed, C. Higgens, and D. Elliman, "Off-line recognition of handwritten Arabic words using multiple hidden Markov models," *Knowledge-Based Systems*, vol. 17, pp. 75-79, 2004.

[25]R. Haraty and A. Hamid, "Segmenting Handwritten Arabic Text," in *Proc. International Conference on Computer Science, Software Engineering, Information Technology, e-Business, and Applications*. Foz de Iguazu, Brazil, 2002.

[26]R. Haraty and C. Ghaddar, "Arabic Text Recognition," *International Arab Journal of Information Technology*, vol. 1, pp. 156-163, 2004.

[27]R. Haraty and H. El-Zabadani, "Abjad: An Off-line Arabic Handwritten Recognition System," in *Proc. International Arab Conference on Information Technology*. Doha, Qatar, 2002.

[28]R. Haraty and C. Ghaddar, "Neuro-Classification for Handwritten Arabic Text," in *Proc. ACS/IEEE International Conference on Computer Systems and Applications*. Tunis, Tunisia, 2003.

[29]N. Farah, L. Souici, L. Farah, and M. Sellami, "Arabic Words Recognition with Classifiers Combination: An Application to Literal Amounts," in *Proc. Artificial Intelligence: Methodology, Systems, and Applications*. Varna, Bulgaria, 2004, pp. 420-429.

[30]L. Souici, N. Farah, T. Sari, and M. Sellami, "Rule Based Neural Networks Construction for Handwritten Arabic City-Names Recognition," in *Proc. Artificial Intelligence: Methodology, Systems, and Applications*. Varna, Bulgaria, 2004, pp. 331-340.

[31]W. F. Clocksin and P. P. J. Fernando, "Towards automatic transcription of Syriac handwriting," in *Proc. International Conference on Image Analysis and Processing*. Mantova, Italy, 2003, pp. 664-669.

[32]S. Snoussi Maddouri, H. Amiri, A. Belaid, and C. Choisy, "Combination of Local and Global Vision Modeling for Arabic Handwritten Words Recognition," in *Proc. International Conference on Frontiers in Handwriting Recognition*, 2002, pp. 128-135.

[33]M. Dehghan, K. Faez, M. Ahmadi, and M. Shridhar, "Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM," *Pattern Recognition*, vol. 34, pp. 1057-1065, 2001.

[34]A. Dehghani, F. Shabani, and P. Nava, "Off-line Recognition of Isolated Persian Handwritten Characters Using Multiple Hidden Markov Models," in *Proc. International Conference on Information Technology: Coding and Computing*, 2001, pp. 506-510.

[35]M. M. M. Fahmy and S. Al Ali, "Automatic Recognition of Handwritten Arabic Characters Using Their Geometrical Features," *Studies in Informatics and Control Journal*, vol. 10, 2001.

[36]H. Miled and N. E. B. Amara, "Planar Markov modeling for Arabic writing recognition: advancement state," in *Proc. International Conference on Document Analysis and Recognition*, 2001, pp. 69-73.

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

[38]I. S. I. Abuhaiba, S. A. Mahmoud, and R. J. Green, "Recognition of handwritten cursive Arabic characters," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 16, pp. 664-672, 1994.

[39]A. Amin, H. Al-Sadoun, and S. Fischer, "Hand-printed Arabic character recognition system using an artificial network," *Pattern Recognition*, vol. 29, pp. 663-675, 1996.

[40]G. Olivier, H. Miled, K. Romeo, and Y.

Lecourtier, "Segmentation and coding of Arabic handwritten words," in *Proc. 13th International Conference on Pattern Recognition*, vol. 3, 1996, pp. 264-268.

[41]H. Al-Yousefi and S. S. Udpa, "Recognition of Arabic characters," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 14, pp. 853-857, 1992.

[42]H. Goraine, M. Usher, and S. Al-Emami, "Off-line Arabic character recognition," *Computer*, vol. 25, pp. 71-74, 1992.

[43]H. Almuallim and S. Yamaguchi, " A method of recognition of Arabic cursive handwriting," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 9, pp. 715-722, 1987.

[44]A. A. Al-Shaher and E. R. Hancock, "Learning mixtures of point distribution models with the EM algorithm," *Pattern Recognition*, vol. 36, pp. 2805-2818, 2003.

[45]J. R. Quinlan, "Learning Logical Definitions from Relations," *Machine Learning*, vol. 5, pp. 239-266, 1990.

[46]M.-Y. Chen, A. Kundu, and S. N. Srihari, "Variable Duration Hidden Markov Model and Morphological Segmentation for Handwritten Word Recognition," *Image Processing, IEEE Transations on*, vol. 4, pp. 1675-1688, 1995.

[47]S. Alma'adeed, C. Higgens, and D. Elliman, "Recognition of off-line handwritten Arabic words using hidden Markov model approach," in *Proc. 16th International Conference on Pattern Recognition*, vol. 3, 2002, pp. 481-484.

[48]N. Farah, A. Ennaji, T. Khadir, and M. Sellami, "Benefits of multiclassifier systems for Arabic handwritten words recognition," in *Proc. International Conference on Document Analysis and Recognition*. Seoul, Korea, 2005, pp. 222-226.

A Two Tier Arabic Offline Handwriting Recognition based on conditional joining rules

Ahmad AbdulKader
Microsoft Research
ahmadab@microsoft.com

Abstract

In this paper we present a novel approach for the recognition of offline Arabic handwritten text that is motivated by the Arabic letters' conditional joining rules. A lexicon of Arabic words can be expressed in terms of a new alphabet of PAWs (Part of Arabic Word). PAWs can be expressed in terms of letters. The recognition problem is decomposed into two problems that are solved simultaneously. To find the best matching word for an input image, a Two-Tier Beam search is performed. In Tier one the search is constrained by a letter to PAW lexicon. In Tier two, the search is constrained by a PAW to word lexicon. The searches are driven by a PAW recognizer.

Experiments conducted on the standard IFN/ENIT database [7] of handwritten Tunisian town names show word error rates of about 11%. This result is comparable to the results of the commonly used HMM based approaches.

Keywords: Offline Arabic handwriting recognition, Neural Networks, IFN/ENIT, Beam Search.

1. Introduction

The recognition of handwritten text in images, commonly known as offline handwriting recognition, is still a challenging task. Significant work still remains to be done before large scale commercially viable systems can be built. This is more so for Arabic (and other non-Latin scripts in general) than Latin scripts where less research effort has been put into solving the problem.

Most research in Arabic offline recognition has been directed to numeral and single character recognition [2]. Few examples exist where the offline recognition of Arabic words problem is addressed [6]. The availability of standard publicly available databases of handwritten Arabic text images like IFN/INIT database has encouraged more research in this area [6] [10].

For Latin scripts, HMM (Hidden Markov Model) based approaches have dominated the space of offline cursive word recognition [11] [1]. In a typical setup, a lexicon is provided to constrain the output of the recognizer. An HMM is then built for every word in the lexicon and the corresponding likelihood (probability of data being generated by the model) is computed. The most likely interpretation is then postulated to be the correct one.

In the few reported approaches to Arabic recognition, approaches very similar to the ones used in Latin were used [6]. Some attempts were made to modify the preprocessing and feature extraction phases to accommodate the different nature of the Arabic writing script. However, the author is not aware of any attempts to this date to exploit the unique properties of Arabic script for recognition purposes or to build systems that are inspired by the distinct nature of the Arabic handwriting script.

In this work, we will present an approach that exploits a key (yet often ignored) property of the Arabic writing script in building a recognition system. This property is basically the set of conditional joining rules that govern how Arabic letters are connected in cursive writing. In Section 2, we show how this property leads to the emergence of PAWs and how our approach exploits these to build a two-tier recognition system. In Section 3, we describe our recognition system in details. Section 4 reports the experimental results conducted on the publicly available IFN/ENIT database of handwritten Tunisian town names and how these compare to the results reported using alternative approaches.

A system built based on the approach described in this paper was submitted as an entry to the ICDAR05 Arabic word recognition competition [8]. The system was evaluated as the second best system on a blind test set and the best system on the non-blind test set. Further developments to the system were also done after the end of the competition. The author's remarks on the competition and the effect of the inconsistency between the training and test set distribution are provided at the end of the paper.

2. Exploiting the Arabic Writing System

Arabic (*arabī*) is the 4th or 5th most widely spoken language in the world. It is spoken by close to 300 million speakers mostly living in North Africa and South West Asia. It is the largest member of the Semitic branch of the Afro-Asiatic language family[12].

Arabic script has a distinct writing system that is significantly different from the commonly known Latin or Han based writing systems. Below is a brief history

and description of the writing system and how one of its unique properties has been exploited to build an offline word recognition system.

2.1. The Arabic writing system

The Arabic script evolved from the Nabataean Aramaic script. It has been used since the 4th century AD, but the earliest document, an inscription in Arabic, Syriac and Greek, dates from 512 AD. The Aramaic language has fewer consonants than Arabic, so during the 7th century new Arabic letters were created by adding dots to existing letters in order to avoid ambiguities. Further diacritics indicating short vowels were introduced, but are only generally used to ensure the Qur'an was read aloud without mistakes[13].

The Arabic alphabet is written from right to left and is composed of 28 basic letters. Adaptations of the script for other languages such as Persian and Urdu have additional letters. There is no difference between written and printed letters; the writing is UNICASE (i.e. the concept of upper and lower case letters does not exist).

The Arabic script is cursive, and all primary letters have conditional forms for their glyphs, depending on whether they are at the beginning, middle or end of a word. Up to four distinct forms (initial, medial, final or isolated) of a letter might be exhibited [5].

However, only six letters (و ز ر ذ د) have either an isolated or a final form and do not have initial or medial forms. If followed by another letter, these six letters do not join with it, and so the next letter can only have its initial or isolated form despite not being the initial letter of a word. This rule applies to numerals and non-Arabic letters as well. This property is often referred to as ***conditional joining***. Figure 1 shows an illustration of this property.

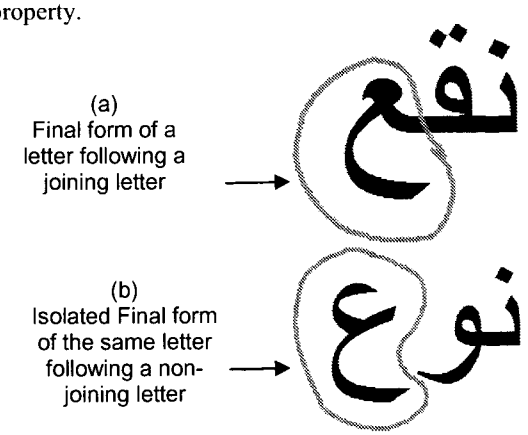


Figure 1. An illustration of the conditional joining property in Arabic script.

The conditional joining property leads to the emergence of PAWs (Part of Arabic Word). A PAW is a sequence of Arabic letters that are joined together with no exceptions. Given an Arabic word, it can be deterministically segmented into one or more PAWs.

It is worth noting that an Arabic writer must strictly abide by the conditional joining rule. Otherwise, the handwriting might be deemed unreadable. However, due to sloppiness in writing or image acquisition conditions, PAWs might end up being physically connected in an image. We empirically estimate that this happens in less than 5% of the overall PAW population. In Section 3.4, we will explain our approach for handling these cases.

2.2. A two-tier approach

Given the conditional joining property of the Arabic writing script, words can be looked at as being composed of a sequence of PAWs. In other words PAWs can be considered an alternative alphabet. The unique number of PAWs constituting a word lexicon grows sub-linearly with the number of words in the lexicon. Figure 2 shows how the number of unique PAWs grows with the size of an Arabic lexicon.

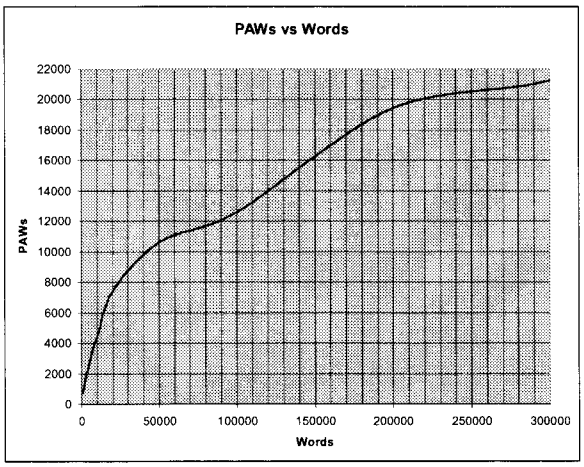


Figure 2. The number of unique PAWs in a lexicon grows sub-linearly with the number of words.

A lexicon of Arabic words can then be decomposed into two lexica. The first is a PAW to letter lexicon which lists all the unique PAWs and their spelling in terms of the letter alphabet. The second is a word to PAW lexicon that lists all the unique words and their spelling in terms of the PAW alphabet.

Consequently, the problem of finding the best matching lexicon entry to an image can be decomposed into two intertwined problems that are solved simultaneously. The first problem is finding the best possible mapping from characters to PAWs constrained by the first lexicon. The second problem is finding the best possible mapping from PAWs to words constrained by the second lexicon.

This two-tier approach has a number of useful properties. One property is that since lexicons constrain the outputs of the recognition process, a number of character recognition errors can be fixed in the PAW recognition phase. Figure 3 shows an example of this type of potential recognition errors. It is unlikely in this example that the second letter "ص" would have been proposed by a character recognizer given how poorly it is written.

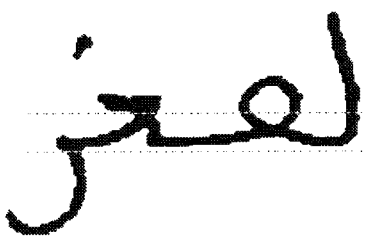


Figure 3. An example image of the لصغر PAW that is confusable with لعخر which is a valid lexicon PAW.

Another property is that PAWs end up having their own prior probabilities that can be utilized by the PAW recognizer to favor more frequently occurring PAWs. These prior probabilities can be looked at as a linguistic n-gram character model that drives the recognition process.

3. The recognition system

A block diagram of the two-tier recognition system is shown in Figure 4. In the following sections we will describe the preprocessing, normalization, segmentation, recognition and search steps in detail.

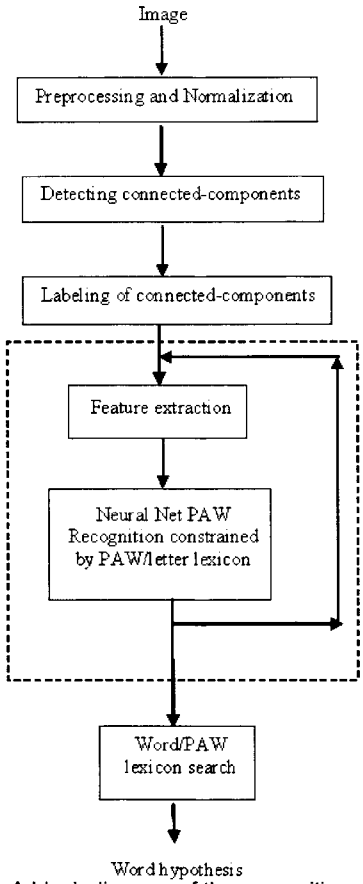


Figure 4. A block diagram of the recognition system

3.1. Preprocessing, normalization and segmentation

The images in the IFN/ENIT database had already passed through the basic processing of image binarization, cropping, word segmentation and noise reduction; we have skipped these phases in our experiments. The very first step of processing is the detection of connected-components. Connected-components whose width and height are below a certain threshold (the choice of which is not critical) are obtained. This threshold has been determined empirically throughout the experiments. This step acts as an additional noise reduction step. Connected-components are then sorted from right to left based on their rightmost point. This allows the search algorithm to sequence through the connected-components in an order that is close to the writing order.

Connected-components are then labeled as 'primary' and 'secondary'. This labeling is performed by detecting relative horizontal overlaps between connected-components and applying some safe thresholds on connected-component sizes. Each secondary connected-component has to be associated to a primary one. No secondary component can exist alone. Figure 5 shows the grouped connected-components in an image of a word. Each group of connected components contains one primary and one or more secondary connected components.

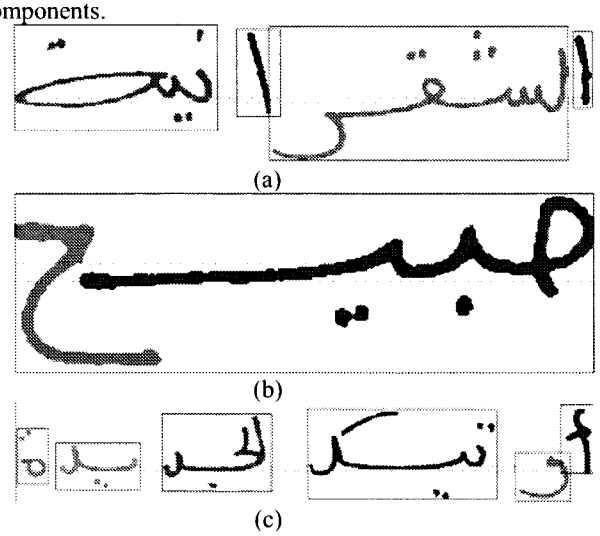


Figure 5. Three examples of grouped connected-components. (a) A case where each connected-component group is an actual PAW. (b) A case where a PAW was split into two connected-component groups. (c) A case where two PAWs were joined in one connected-component group (purple color).

In 5(a) each connected-component group corresponds to exactly one PAW. We have empirically determined that this case represents around 65% of the

overall population of words. Figure 5(b) shows a case where the two connected component groups correspond to one PAW (i.e the over-segmentation case). Over-segmentation represents around 30% or the word population. Figure 5(c) shows the case where the purple connected component-group is actually two touching PAWs. This case is not inherently handled by the proposed approach. It constitutes around 5% of the cases. We will explain in section 3.4 how it was handled. As such, a fundamental assumption of the following steps of the system is that PAWs can only occur on connected-component group boundaries.

3.2. The PAW recognizer

The IFN/ENIT database has a lexicon of 946 Tunisian town names. The number of unique PAWs in this word lexicon is 762. Although the training database might not necessarily have at least one sample of each valid word, it turns out that there is at least one sample present of every valid PAW.

Because the number of unique PAWs is relatively low for this experiment, it was decided to use a Neural Network based classifier to recognize PAWs. As the size of the word lexicon gets bigger and the number of valid PAWs grows, it might not be practical to directly use a Neural Network classifier for recognizing PAWs.

In our experiments we build two Neural Net PAW classifiers. The first classifier is a convolutional Neural Network. Convolution Neural Networks [9] has been reported to have the best accuracy on offline handwritten digits. In this type of networks, the input image is scaled to fit a fixed size grid while maintaining its aspect ratio. Since the number of letters in a PAW can vary from 1 to 8, the grid aspect ratio has to be wide enough to accommodate the widest possible PAW while still maintaining its distinctness. The second classifier is based on features extracted from the directional codes of the connected-components constituting the PAW. Each of these two classifiers has 762 outputs and was trained with training sets that reflect the prior distributions of PAWs in the word lexicon.

PAWs can exist at the start, middle and final position in a word. Additionally a PAW can constitute the whole word. We'll refer to this as the Isolated position. Figure 6 shows example PAWs at each of the four possible positions.

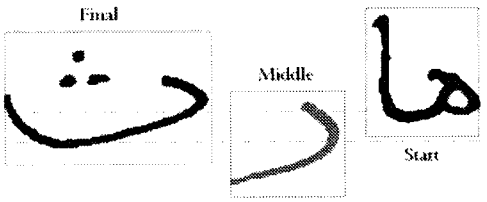


Figure 8. PAWs at Start, Middle and Final positions in the word

Since each unique PAW does not necessarily exist at all possible positions within a word: Start, Middle, Final, Isolated, it was determined that building a separate classifier for each possible position improves PAW recognition accuracy significantly.

3.3. Beam search

As mentioned above, the word lexicon can be decomposed into two lexica: A letter to PAW lexicon and a PAW to word lexicon. The letter to PAW lexicon is used to constrain the output of the PAW recognizer as mentioned above. The PAW to word recognizer is used to constrain the search for the best matching word.

Beam search is an algorithm that is an extension to the well known best-first search algorithm. Like best-first search, it uses a heuristic function to evaluate the promise of each node it examines. Beam search, however, only unfolds the first m most promising nodes at each depth, where m is a fixed number, the "beam width". It commonly used in speech recognition [3].

The Beam search is used to find the best matching word to an image using the output of PAW recognizer as a search heuristic. The search algorithm sequences through the connected-components groups and considers either starting a new PAW or adding the group to the existing PAW. The list of possible PAWs together with their corresponding posterior probabilities produced by the PAW recognizer is retained. Different connected-component group to PAW mappings are kept in a lattice of possible segmentations. After sequencing through all the groups, the best possible segmentation is evaluated and chosen to be the winning hypothesis.

For practical reasons and to make sure that the segmentation possibilities in the lattice do not explode, two heuristics are used. First, the maximum number of connected-component groups per PAW is capped at 4. This number has been determined empirically based on the training data. Second, at every step in the lattice, segmentation possibilities that have a probability that is lower than the most probable segmentation by a certain threshold are pruned. This means that theoretically, the Beam search might not produce the most probable segmentation. However, this rarely happens in practice.

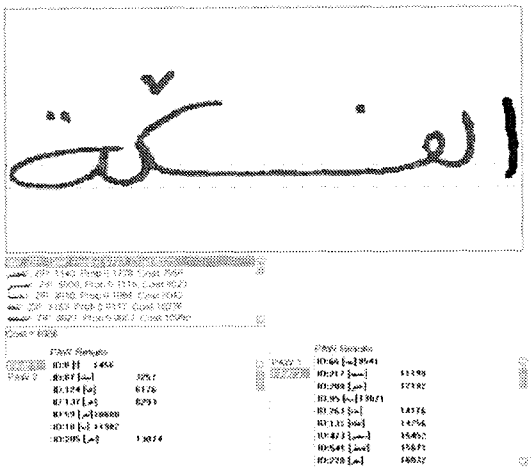


Figure 7. An recognition example showing the word recognition results in the top list and the PAW recognition results in the lower list boxes.

Figure 7 shows an example image, the final recognition results and the PAW recognition results of the two connected-component groups. Note that although the second PAW was misrecognized, the overall word was correctly recognized.

3.4. Handling exceptions

As pointed out earlier the under-segmentation case was empirically determined to constitute around 5% of the words. To handle the under-segmentation case, where more then one PAW end up being segmented as one connected-component group a final step in the process was added. The final step is triggered if the probability of the winning segmentation path in the lattice is lower than a certain threshold. This was found to be strong evidence that under-segmentation occurred. When triggered, a Viterbi search is performed on the individual PAW recognition results of the connected-component groups. In this search the edit distance between the each of the PAW to Word lexicon and the recognition results are computed. Both PAW insertions and deletions are allowed with a penalty associated with each.

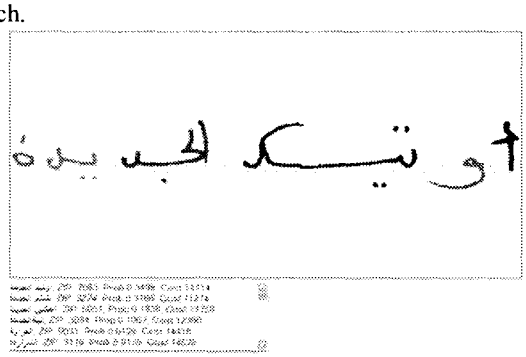


Figure 8. A recognition example of an under-segmented image. The Viterbi search that is triggered when the best Beam result is lower than a certain threshold produced the correct answer.

4. Experiments

4.1. The data set

Experiments were conducted on the publicly available IFN/ENIT database [7]. The database is split into four sets A, B, C & D. The 4 sets contain 26,459 images of segmented Tunisian town names handwritten by 411 unique writers. The total number of PAWs in the set is 115,585. For each image the ground truth information is available. The number of unique word labels is 946, and the number of unique PAW labels is 762.

Sets A, B & C were used for training and validation. Set D was used for evaluation. Set D has 6735 words handwritten by 104 unique writers none of which contributed any the words in sets A, B or C.

A widely agreed upon rule of thumb in building recognition systems is to ensure that recognizers are evaluated on a distribution similar to that of the training set. Since the 4 sets roughly have the same writer demographics, word distribution and consequently PAW distribution, this rule was upheld in our experiments.

4.2. The training process

One problem that was encountered during implementing the recognition system was getting data to train the PAW recognizer. As such, the database has word level ground truth information and does not have PAW level ground truth information. To solve this problem, we followed a bootstrapping technique similar to the bootstrapping from incomplete data in the well known Expectation-Maximization *EM* setting [4].

As mentioned in Section 3.1, our connected-component segmentation and grouping algorithm results in three different types of segmentation. For the first type, which we call *exact segmentation,* each of the resulting connected-component groups corresponds to one and exactly one PAW. Empirically, it was determined that *exact segmentation* cases constitute 65% of the total word population.

For each training sample, the number and the identity of the PAWs that constitute the sample's word label can be computed. To bootstrap the training process, a conjecture is made that for every sample in the training set where the number of connected-component groups is equal to the number of label PAWs, the identity of a specific PAW corresponds to the ground truth label of the connected-component group at the same position. This conjecture holds almost all the time. There are rare cases where PAW over-segmentation and under-segmentation occur an equal number of times in a word which results in breaking the *exact segmentation conjecture*.

As a first step, the PAW recognizers are trained on all the training samples that satisfy the *exact segmentation conjecture*, which is 65% of the training data. In subsequent steps, by using the ground truth word label and its corresponding PAWs, the PAW recognizer that was trained in the previous step is used

to segment connected-component groups into PAWs. This is done by running the same exact algorithm described in Section 3 with only one entry in the word lexicon: the ground truth. This could only work for exactly segmented and over-segmented words. And so, the under-segmented words, which constitute 5% of the training set, are excluded from the training process. The training step is analogous to maximization step in EM, while the PAW re-segmentation phase is analogous to the expectation step. This sequence (training, re-segmentation) was repeated 3 times until no significant change in the accuracy of the PAW recognizer was observed.

4.3. PAW recognition results

The results of the two individual PAW recognizers and their combined results are shown in Table 1.

Table 1. The error rates of the individual PAW recognizer and the combined PAW recognizer on set D of the IFN/ENIT database.

Recognizer	*Top 1 Errors*	*Top 10 Errors*
Convolutional Net	40.13%	12.31%
Directional Codes	32.4%	11.77%
Combined Classifier	19.98%	8.9%

4.4. Word recognition results

Table 2 shows the error rates for the overall word recognizer as measured on set D of the IFN/INIT database. It also shows the results broken down by the type of segmentation encountered in the image.

Table 2. The error rates of overall word recognizer.

Data subset	*Top 1 Errors*	*Top 10 Errors*
All data	11.06%	4.99%
Exact Segmentation	7.11%	1.67%
Over-Segmented	13.33%	4.39%
Under-Segmented	36.03%	36.03%

5. Conclusion

In this paper we have presented a novel approach to the recognition of lexicon constrained Arabic handwritten words. The approach exploits the conditional joining of letters property in Arabic writing script to decompose the problem into two problems that are solved simultaneously. Using a Neural Network based PAW recognizer a two-tier Beam search is performed to find the best matching word to the input image. Word error rates of around 11% were achieved on the publicly available IFN/ENIT database. These results are comparable to the results reported on the same set using an alternative HMM based approach [6].

5.1. The ICDAR05 competition

The same results were also reported as part of the ICDAR05 Arabic handwritten word recognition competition report. A system that implements the presented approach was ranked as the second best entry on the blind-test (whose results are not reported here since the author has no access to it) and the best entry on the non-blind test set (set D).

It is worth noting that the blind set had a different distribution of words than all published sets A, B, C & D of the database. This in turn resulted in an unexpected PAW prior distribution. This might explain that the error rate reported on the blind set is significantly higher than the non-blind set. The author is of the opinion that the competing recognizers should have been evaluated on a distribution similar to that of the training set.

6. Acknowledgment

The author would like to thank the developers of the IFN/INIT database for making it possible to evaluate different Arabic handwritten word recognition systems in an objective manner and increasing interest in Arabic handwriting recognition.

7. References

[1] A. Vinciarelli, J. Luettin. "*Off-Line Cursive Script Recognition Based on Continuous Density HMM*" International Workshop on Frontiers in Handwriting Recognition, IWFHR 2000.

[2] B. Al-Badr and S. A. Mohmond. "*Survey and bibliography of Arabic optical text recognition*". Signal Processing, 41:49–77, 1995.

[3] H. Ney, D. Mergel, A. Noll, and A. Paesler. "*Data driven search organization for continuous speech recognition. IEEE Transactions on Signal Processing*", 40(2):272--281, February 1992.

[4] J.A. Bilmes, "*A gentle tutorial of the EM algorithm and its applications to parameter estimation for Gaussian mixture and hidden Markov models*", Technical Report TR-97-021, International Computer Science Institute, Berkeley, California, 1998.

[5] K. Versteegh, "*The Arabic Language*", Edinburgh University Press, 1997.

[6] M. Pechwitz & V. Maergner, "*HMM based approach for hand- written Arabic word recognition using the IFN/ENIT database*", Proc. 7th Int. Conf. on Document Analysis and Recognition, Edinburgh, Scotland, 2003.

[7] M. Pechwitz, S. S. Maddouri, V. Maergner, N. Ellouze, and H. Amiri. "*IFN/ENIT - database of handwritten Arabic words*". In *Proc. of CIFED 2002*, pages 129–136.

[8] V. Margner, M. Pechwitz, H. E. Abed. "*ICDAR 2005 Arabic handwriting recognition competition*". Eighth International Conference on Document Analysis and Recognition, 2005. Proceedings. page(s): 70- 74 Vol. 1

[9] P. Simard, D. Steinkraus, J. C. Platt. "*Best Practices for Convolutional Neural Networks Applied to Visual Document Analysis*". ICDAR 2003: 958-962.

[10] R.A. Haraty and H.M. El-Zabadani, "*Hawwaz: An Offline Arabic Handwriting Recognition System*", International Journal of Computers and Applications - 2005.

[11] T. Steinherz, E. Rivlin, N. Intrator, "*Off-Line Cursive Script Word Recognition A Survey*". International Journal on Document Analysis and Recognition -1999, 2:90-110.
[12] Wikipedia, *"Arabic Language"* http://en.wikipedia.org/wiki/Arabic
[13] Omniglot, *"Writing System and Languages of the World"*, http://www.omniglot.com/writing/arabic.htm

Visual Processing of Arabic Handwriting. Challenges and New Directions

Mohamed Cheriet and Mokhtar Beldjehem
Laboratory for Imagery, Vision and Artificial Intelligence
École de Technologie Supérieure. University of Quebec
1100 Notre-Dame West, Montreal, Quebec, Canada, H3C 1K3
mohamed.cheriet@etsmtl.ca, mbeldjehem@livia.etsmtl.ca

Abstract

The automatic recognition of Arabic handwritten text is a problem worth solving because a solution would enable us to design, develop and use systems conveniently with a more ergonomic and flexible human-machine interaction in various ICTs fields. Thus an Arabic Handwritten OCR system should have great commercial value. In this paper we address exhaustively the most encountered problems when dealing with Arabic handwriting recognition and we present briefly lesson learnt from several serious attempts that have been undertaken in this regard. We show why, where and how morphological analysis of Arabic could be used to improve the accuracy of the Arabic handwriting recognition. In general Arabic natural language processing (NLP) could provide some error handling techniques that could be used effectively to improve the overall accuracy during the important stage that is post-processing. We give a summary of techniques that could be used to generate an Arabic lexicon and we conclude by some perspective and future trends concerning Arabic handwriting recognition research.

1 Introduction to Arabic Scripts

Arabic, one of the six official languages of the United Nations, is the mother tongue of more than 300 million people [1]. Unlike Latin-derived writing, the orientation of Arabic writing is from right-to-left.

Origin: A plausible hypothesis states that the Arabic scripts evolved from the Nabataean Aramaic scripts. It has been used since the 4th century AD, but the earliest document, an inscription in Arabic, Syriac and Greek, dates from 512 AD. The Aramaic language has fewer consonants than Arabic, so during the 7th century new Arabic letters were created by adding dots to existing letters in order to avoid ambiguities. Further diacritics indicating short vowels were introduced, but are only generally used to ensure the Qur'an was read aloud without mistakes.

There are two main types of written Arabic:

1. **Classical Arabic** - the language of the holy Qur'an (Koran) and classical literature. It is pure Arabic and differs from Modern Standard Arabic mainly in style and vocabulary, some of which remains undefined, unknown and implicit.
2. **Modern Standard Arabic (MSA)** - the universal language of the Arabic-speaking world which is understood by all Arabic speakers. In addition to pure Arabic it includes new foreign arabized words, scientific and technological terms and so on. It is the Academic Language and is the language of the vast majority of written material and of formal TV shows, lectures, etc. MSA has clear well-determined vocabulary as well as explicit well-established morphological rules and grammar rules.

Historically the holy Qur'an (Koran) was intensively used as a source (or reference) by linguists and grammarians to elicit vocabulary, morphological rules and grammar rules. It is worth mentioning that it is likely that the holy Qur'an (Koran) contains most if not all pure Arabic vocabulary as well as all Arabic morphological rules and grammar rules. However each Arabic speaking country or region also has its own variety of colloquial spoken Arabic. These colloquial varieties of Arabic appear in written form in some poetry, cartoons and comics, plays and personal letters.

Notable Features:

- As illustrated in Table 1, the Arabic alphabet contains 28 letters. Some additional letters are used in Arabic when writing place names or foreign words containing sounds which do not occur in Standard Arabic, such as /p/ or /g/.
- Words are written in horizontal lines from right to left, numerals are written from left to right.

- Most letters change form depending on whether they appear at the beginning, middle or end of a word, or on their own as illustrated in Table 1.
- Letters that can be joined are always joined in both hand-written and printed Arabic. The only exceptions to this rule are crossword puzzles and signs in which the scripts is written vertically.
- The long vowels /a:/, /i:/ and /u:/ are represented by the letters *'alif*, *yā'* and *wāw* respectively.
- As shown in Figure 1, vowel diacritics, which are used to mark short vowels and other special symbols, appear only in the holy Qur'an (Koran). They are also used, though with less consistency, in other religious texts, in classical poetry, in children textbooks and foreign learners, and occasionally in complex texts to avoid ambiguity. Sometimes the diacritics are used for decorative purposes in book titles, letterheads, name places, etc.

Table 1: Arabic Alphabet in all shapes
a) EF: End of Form b) MF: Middle of Form
c) BF: Beginning of Form d) IF: Isolated Form

Name	EF	MF	BF	IF	Name	EF	MF	BF	IF
DĀD	ـض	ـضـ	ضـ	ض	ALIF	ـا			أ
TĀ	ـط	ـطـ	طـ	ط	BĀ	ـب	ـبـ	بـ	ب
ZĀ	ـظ	ـظـ	ظـ	ظ	TĀ	ـت	ـتـ	تـ	ت
AYN	ـع	ـعـ	عـ	ع	THĀ	ـث	ـثـ	ثـ	ث
GHAYN	ـغ	ـغـ	غـ	غ	JĪM	ـج	ـجـ	جـ	ج
FĀ	ـف	ـفـ	فـ	ف	ḤĀ	ـح	ـحـ	حـ	ح
QĀF	ـق	ـقـ	قـ	ق	KHĀ	ـخ	ـخـ	خـ	خ
KĀF	ـك	ـكـ	كـ	ك	DĀL	ـد			د
LĀM	ـل	ـلـ	لـ	ل	DHĀL	ـذ			ذ
MĪM	ـم	ـمـ	مـ	م	RĀ	ـر			ر
NŪN	ـن	ـنـ	نـ	ن	ZĀY	ـز			ز
HĀ	ـه	ـهـ	هـ	ه	SĪN	ـس	ـسـ	سـ	س
WĀW	ـو			و	SHĪN	ـش	ـشـ	شـ	ش
YĀ	ـي	ـيـ	يـ	ي	SĀD	ـص	ـصـ	صـ	ص

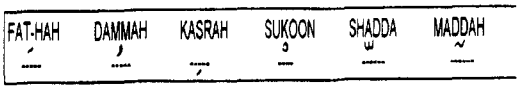


Figure 1: Diacritics Arabic Text used for "Tashkeel"

The Arabic language: Arabic is a Semitic language with about 300 million speakers in Afghanistan, Algeria, Bahrain, Chad, Cyprus, Djibouti, Egypt, Eritrea, Iran, Iraq, Jordan, Kenya, Kuwait, Lebanon, Libya, Mali, Mauritania, Morocco, Niger, Occidental Sahara, Oman, Palestine, Qatar, Saudi Arabia, Somalia, Sudan, Syria, Tajikistan, Tanzania, Tunisia, Turkey, UAE, Uzbekistan, Yemen and elsewhere in the world. There are over 30 different varieties of colloquial Arabic which include according to the degrees of their closeness:

- **Algerian**- spoken by about 75 million people in Algeria, Tunisia, Libya and Occidental Sahara.
- **Egyptian** - spoken by about 50 million people in Egypt.
- **Moroccan/Maghrebi** - spoken in Morocco by about 25 million people.
- **Sudanese** - spoken in Sudan by about 25 million people.
- **Saidi** - spoken by about 20 million people in Egypt.
- **North Levantine** - spoken in Palestine, Lebanon and Syria by about 30 million people.
- **Mesopotamian** - spoken by about 55 million people in Iraq, Iran and Syria.
- **Najdi** - spoken in Saudi Arabia, Iraq, Jordan and Syria by about 35 million people.

As illustrated in Figure 1, this system of diacritical marks is known as "Tashkeel" (vocalization). In addition as illustrated in Table 1, for some letters, dots are also present and are placed either above or beneath the letter, either single or in groups of two or three.

For their utility, effectiveness and convenience printed and/or handwritten OCR systems are widely used in many government agencies, commercial departments, research laboratories and libraries. After thirty years of intensive research, both printed and handwritten OCR products for most scripts (Latin, Hindi, Japanese, Korean and Chinese, etc.) are well developed and are available in the market. Research in Arabic handwritten recognition can be traced early to works by Amin [2-3]. Significant work has been done in the Arabic handwritten recognition area and several serious attempts have been undertaken to tackle the Arabic handwriting segmentation and recognition problem [4-13]. However, despite the availability of several Arabic printed OCR products in the market, to the best of our knowledge still except research prototypes developed for the proof of concept in Academia only, there is no operational accurate Arabic handwritten OCR commercial product available in the market. Thus an Arabic Handwritten OCR system

should have great commercial value. Following the creation of Latin handwriting Databases as in [16], similar Arabic-related efforts have been devoted to the creation of Arabic handwriting Databases [14-15] that are especially useful for training and benchmarking purposes. It seems that the degree of success achieved in Arabic handwriting recognition till now is far from the expected goal.

The structure of the rest of the paper is as follows: In section 2 we address exhaustively the most encountered problems when dealing with Arabic handwriting recognition and we present briefly lessons learnt from several attempts that have been undertaken in connection with Arabic handwritten. In section 3 in connection with the important post-processing stage we show why, where and how morphological analysis of Arabic could be used to improve the accuracy of the Arabic handwritten recognition; we deal also with Arabic natural language processing (NLP) and give briefly some techniques that could contribute to improve the accuracy of Arabic handwriting recognition. Section 4 gives briefly a summary of ideas and techniques that could be used to generate an Arabic lexicon. In section 5 we conclude and give some perspective and future trends concerning computational linguistic, natural language processing and soft computing that constitute promising Arabic handwriting recognition research.

2 Segmentation and Recognition Problems of Arabic Handwriting

It is widely accepted that machine segmentation and recognition of handwritten Arabic scripts is a difficult problem. It is not always successfully done. Beyond the idealizing assumptions, we distinguish between three categories of difficulties. Those that are due inherently to the nature and characteristics of the Arabic scripts itself; those that are the responsibility of the scripter himself and depend to the Arabic writing styles and even various available Arabic calligraphic styles that are more than dozen, some of them are shown in Figure 2; and finally those induced by the quality of the scanned document in particular for highly degraded historical documents such as in Figure 3 used for some applications of real world problems.

The first category includes the following:

- Context-dependency of the Arabic character shape, because each character changes its shape by the effect of the characters before and behind. In other terms, even the same character is scripted differently when it appears in different words, in order to be connected smoothly with the characters in front and in the rear (see Table 1 for more details).
- Cursiveness of the Arabic either printed or handwritten.
- Presence of ligatures (see Figure 4 for more details).
- It is widely accepted that segmentation of Arabic handwriting to character is not always ensured.

The second category includes the following:

- Every scripter has his own individual writing style.
- The condition and state of the scripter during writing influence significantly the writing process.

The third category includes the following:

- Ascender and descender of consecutive lines are frequently connected.
- Text not uniformly spaced.
- Lines are not straight.
- Interfering handwriting.
- Touching characters and broken (sub) words.
- Seeping of ink or faded ink.
- Noise that resulted from the scanning process (scanner background and page border marks).

Due to the complexity, anomalies and inherent specificity of the Arabic handwriting, it is widely accepted that approaches and techniques used in other languages contexts cannot be applied directly to the context of Arabic. These variability, anomalies and complexity make Arabic word decomposition (segmentation) in letters very delicate and not always ensured. In other terms, this means that first segmentation of words into letters followed by recognition of the resulting characters approach unfortunately does not work for Arabic handwritten text. Consequently, many issues in Arabic handwriting still constitute important open questions concerning:

- **Segmentation**: is it mandatory? Is it more fruitful to put research effort in trying to search for novel more sophisticated effective segmentation algorithms of Arabic handwriting or radically to bypass the segmentation (or segmentation-free)?

- **Recognition paradigms**: Which is the appropriate approach to use for Arabic handwriting? Holistic word-based approach? Local letter-based approach (or analytical)? Or hybrid approach?

- **Recognition Techniques**: Algorithm-based? Neural network-based? SVM-based? HMM-based? Symbolic-based? Expert System-based? Fuzzy-based? Possibility-based? Evidence-based? Rough set-based? Syntactic-based? Structural-based? Statistical-based? Hybrid-based? Soft computing-based and so on. In particular good techniques are those that have learning capabilities and can incorporate and take into account linguistic issues and especially have the possibility to exploit the valuable information carried out by "Tashkeel" and dots.

- **Post-processing**: Is it more interesting to keep it as an independent stage, or to integrate it totally or partly to the recognition chain? What kind of knowledge is important and how and where it is more appropriate to incorporate?

- **Architecture** of the handwritten OCR system: do we need to devise novel architectures or to reuse and adapt or extend traditional ones developed for other scripts or even for printed Arabic scripts? What are the scripts and languages that have similarities with Arabic scripts and language that have accurate handwritten OCR systems?

- **Hardware/Software Implementation**: Is it more appropriate to implement the recognition system in terms of software or in terms of a hardware chip. What is the appropriate technology? In both cases find a compromise solution that account for the environment where the OCR systems will be deployed and integrated. Is a parallel implementation possible and feasible?

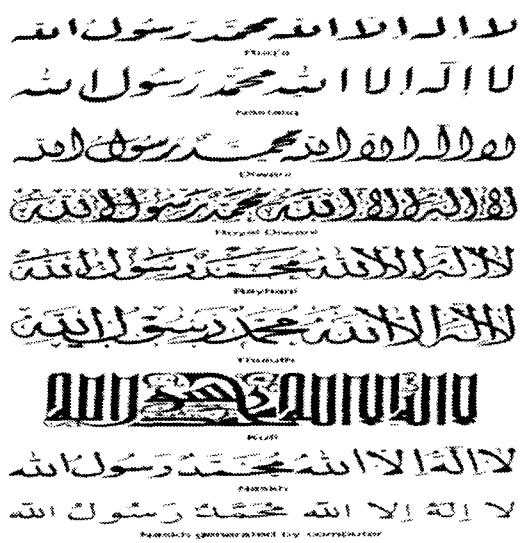


Figure 2: Some available Arabic calligraphic style

Such open questions and raised issues need an in-depth research effort that will provides answers that influence and guide the selection, conception, design and development of a cost-effective general purpose Arabic handwriting OCR system that satisfy the needs and requirements of users for a large application domain.

However, it seems that in real world problems the solution is application-dependent and that there is no unique standard solution. It is worth mentioning that all OCR research studies agree on a required mandatory pre-processing and low level segmentation of images to prepare them for the next stages of segmentation and recognition.

Figure 3: Highly degraded historical documents sample

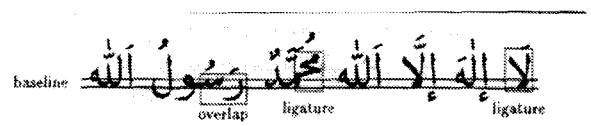


Figure 4: Some encountered ligatures in Arabic

3 Morphological Analysis, Natural Language Processing and Post-Processing

Traditionally, the bulk of post-processing stage is dedicated to error handling. In the sequel, we will investigate the contribution and the link of both *morphological analysis* and *natural language processing* (NLP) to post-processing and try to find the right place for them in a recognition system.

On one hand Arabic is highly inflected language[1]. Some kind of errors that are not caught during recognition could be fixed using morphological analysis. The Arabic word carries well around the morphology. Most Arabic words are morphologically derived from a list of roots. The root is the bare verb form; it can be trilateral, quadrilateral, or pentaliteral. Most of these roots are made up of three consonants. The Arabic language uses a root-and-pattern morphotactics; pattern can be thought of as template adhering to well-known rules. These patterns generate nouns and verbs. Roots are interdigitated with the patterns to form Arabic surface forms. Significant work has been done in the Arabic morphological analysis area. However an open question still remains on the appropriate place where to use morphology. Morphological analysis could be integrated within the recognition chain and used effectively to fix lexical errors at the word level and is well-suited for word–based recognition approaches. It can be put also in post-processing as in traditional approaches. Taking into account the size of the vocabulary on hands (small, medium, versus large), there will be no standard solution for Arabic Language for this purpose. Thus in connection with Arabic handwriting recognition there are many open questions: do we need a *morphological analyser? A stemmer*? or both?

Which approach to use to implement morphological analyser? Symbolic-based (rule-based)? Statistical-based? Or hybrid one combining both (rules in conjunction with statistics)?

Which approach is the most adequate in connection with Arabic OCR, root-based? Stem-based? Which technique is the most adequate for Arabic OCR, Automata theory? Exact or approximate matching algorithms?
And if we opt for a stemmer which stemming algorithm is the most appropriate to suit the needs Arabic OCR? n-grams based? light stemming?

On other hand natural language processing (NLP) could effectively contribute at the Arabic phrase level. Syntax analysis, semantic analysis and even pragmatic analysis could be used to implement a high-level error handling to fix errors that are not caught during recognition. It seems that the right choice is to put this in the post-processing stage once the file is edited as ASCII file within an editor. Integrating it within the recognition chain will burden the speed of recognition and hence is a bad choice. A good solution should avoid this. There are too open questions, which formalism is the most adequate to capture, represent and handle Arabic language constructs and structures, *Conceptual graphs*? *Semantic networks*? Or *definite clauses grammars (DCG)?* Or another formalism?

Contextual information brings valuable information as it is application-dependent and could be implemented within the *pragmatic analyser* that can handle both the word level and the phrase level issues. This is due to the nature of the Arabic language. Yet another open question do we need it? And if yes then where to put it?

To summarize any solution (morphological analyser, Stemmer, NLP, contextual information) should take in to account both the specificity of the Arabic language as well as understanding of the needs and requirements of Arabic handwriting OCR systems. Of course their design requires some linguistic expertise in the Arabic language.

4 Lexicon Generation

An Arabic lexicon constitutes the heart of any Arabic language processing system. Accurate words with grammatical and semantic attributes are essential and highly desirable for OCR systems as well as for machine translation, text understanding, text summarization, text generation, information retrieval, information extraction, tagging and text mining and so on. The question is how to build a high quality lexicon tailored to satisfy the requirement of Arabic OCR? A lexicon may be constructed either manually or automatically. How ever manual construction is labor-intensive, time-consuming and costly. The lexicon size might influence the choice of the recognition paradigm.

We propose to generate the dictionary automatically from an input document-making use of either some Arabic equivalent of the *WordNet* if it exists or a computer readable dictionary that is a lexical database. Bearing in mind the richness of the holy Koran, it could constitute an ideal corpus from where we may generate the Arabic lexicon.
However, a semi-automatic approach might constitute a promising and feasible even optimal solution for Arabic language. And yet another open question: do we really need a lexicon for recognition purposes? If yes, what are the pertinent attributes to include in connection with Arabic handwriting recognition?

[1] The class of languages that append inflectional morphemes to words are called inflectional languages

5 Concluding Remarks

We have addressed major problems and raised issues that constitute challenges for the effective conception, design and development of Arabic handwritten OCR systems. This study has allowed us to determine, capture and define "good" functional and non-functional requirements that we are planning to engineer in order to build a coherent and versatile implementation of an operational high-adaptability and high-accuracy Arabic Handwritten OCR system. The automatic recognition of Arabic handwritten text is a problem worth solving because a solution would enable us to design and use systems conveniently with a more ergonomic and flexible human-machine interaction in various ICTs fields. Bearing in mind that such a solution is viable and generic, it will be a building block toward developing operational multilingual OCR systems. In addition to further analysis and mature understanding of the recognition of Arabic handwriting problematic, we will look for a compromise solution that constitute designs trade-offs and account not only for the specificity of Arabic scripts and language but also for the performance, accuracy, the robustness and adaptability too.

On one hand why, how and where computational linguistics could be incorporated so as to effectively benefit more the accuracy and adaptability without affecting the efficiency of recognition is an open problem. We believe that Arabic Computational linguistics will address these challenges effectively. Computational linguistics becomes crucial to the success of intelligent OCR systems.

On another hand our purpose is to design and develop an Arabic Handwriting OCR system with learning capabilities that ensures robustness, high-adaptability, high accuracy and throughputs. It is appealing to use soft computing that helps effectively to achieve our goal. According to Zadeh [17-19] "*The exploitation of tolerance for imprecision and uncertainty underlies the remarkable human ability to understand distorted speech, decipher sloppy handwriting, comprehend nuances of natural language, summarize text, recognize and classify images and more generally, make rational decision in an environment of uncertainty and imprecision.*" This ability is what *granular hybrid soft computing* systems try to capture by learning and to emulate computationally. Granular hybrid soft computing becomes crucial to the success of intelligent OCR systems.

In general we believe that software architecture of an OCR system either printed or handwritten, be it Arabic or not, is definitely of paramount importance. In particular we are investigating the applicability of some interesting soft computing techniques from our previous work [20-27] to Arabic handwritten recognition. We propose in our future work to explore the possible joint contribution of Arabic computational linguistics, Arabic NLP and granular soft computing in building novel architectures for Arabic handwritten recognition. Cursive Latin handwriting might be also handled in a similar manner. Thus, opening the opportunity for multi-lingual OCR using the same framework. We believe that much more research effort is needed on the synergistic combination or hybridization of Arabic computational linguistics, Arabic NLP and granular hybrid soft computing. Such hybridization becomes crucial to the success of OCR systems and it is definitely an interesting and highly promising research avenue. We hope that this paper will be a trigger and/or a pointer for researching novel cost-effective hybrid architectures and especially for further incorporating of Arabic computational linguistics, Arabic NLP and granular hybrid soft computing particularly in Arabic Handwritten OCR systems and in recognition technology in general.

References

[1] A. Hani et al., Deterministic and nondeterministic flow-chart interpretations, *JASIS* **50** (6) (1999) 524–529.

[2] A. Adnan et al., Handwritten Arabic Character recognition by the IRAC system, in *Proc. Int. Conf. on Pattern Recognition*, Miami, Florida, USA, 1980, 729-731.

[3] A. Adnan et al., Recognition of Handwritten Arabic Scripts and Sentences, in *Proc. IAPR* (2), Montreal, Canada, 1984, 1055-1057.

[4] B. Al-Badr and S. Mahmoud, Survey and bibliography of Arabic Optical text recognition, *Signal Processing* **41**(1995) 49-77.

[5] S. Al-Emami and M. Usher, On-line recognition of handwritten Arabic characters, IEEE *Trans. PAMI* **12** (7) (1990) 704-710.

[6] H. Almuallim and S. Yamaguchi, A method for recognition of Arabic cursive handwriting, *IEEE Trans. PAMI* **9** (5) (1987) 715-722.

[7] H. Miled, C. Olivier, M. Cheriet et Y. Lecourtier, Coupling observation/letter for a Markovian modelisation applied to the recognition of Arabic handwriting. In *Proc. ICDAR*, Ulm, Germany, 1997, 580-583.

[8] H. Miled, M. Cheriet, C. Olivier, Multi-level Arabic Handwritten Words Recognition, SSPR/SPR (1998) 944-951.

[9] L. Souici-Meslati and M. Sellami, A Hybrid Neuro-Symbolic Approach for Arabic Handwritten Word Recognition, *JACII* **10** (1) (2006) 17-25.

[10] L. Souici-Meslati, N. Farah, T. Sari and M. Sellami, Rule Based Neural Networks Construction for Handwritten Arabic City-Names Recognition, in *Proc. AIMSA,* Verna, Bulgaria, 2004, 331-340.

[10] T.S. Al-sheikh and S. G El-Taweel, Real-time Arabic handwritten character recognition, *Pattern Recognition* **23** (12) (1990)1323-1332.

[11] I. S. I. Abuhaiba and P. Ahmed, Restoration of temporal information in off-line Arabic handwriting, *Pattern Recognition 26* (7) (1993) 1009-1017.

[12] I. S. I. Abuhaiba, S. A. Mahmoud and R. J. Green, Recognition of Handwritten Cursive Arabic Characters, *IEEE Trans. PAMI* **16**(6) (1994)664-672.

[13] S. A. Mahmoud, S. I. Abuhaiba and R. J. Green, Skeletonization of Arabic characters using clustering based skeletonization algorithm (CBSA), *Pattern Recognition* **24**(5) (1991) 453-464.

[14] Y. Al-Ohali, M. Cheriet and C. Suen, Databases for recognition of handwritten Arabic cheques, *Pattern Recognition* **36** (2003)111-121.

[15] M. Pechwitz, S. Snoussi-Maddouri, V. Margner, N. Ellouse et H. Amiri, IFN/ENIT database of handwritten Arabic words, in *Proc. Colloque Francophone International sur l'Ecrit et le Document*, Hammamet, Tunisia, 2002, 129-136.

[16] J. J. Hull, A Database for handwritten text recognition research, *IEEE Trans. PAMI* **16** (1994) 550-554.

[17] L. A. Zadeh, The Roles of Fuzzy Logic and Soft Computing in the Conception, Design, and Development of Intelligent Systems, in *Software Agents and Soft Computing: Conception and Applications*, H. S. Nwana and N. Azarmi, eds. (Springer, 1997) (LNAI 1198) 183-190.

[18] L. A. Zadeh, Soft Computing, Fuzzy Logic and Recognition Technology, in *Proc. IEEE Int. Conf. Fuzzy Syst.*, Anchorage, AK, 1998, 1678-1679.

[19] L.A. Zadeh, Some Reflections on Soft Computing, Granular Computing and their Roles in the Conception, Design and Utilization of Information/Intelligent Systems, *Soft Computing* **2**(1998)23-25.

[20] M. Beldjehem, A Contribution to the Conception, Design and Development of Hybrid Min-Max Fuzzy-Neuro Systems by Approximating Fuzzy Systems of Relational Equations: the FENNEC System, Ph.D. Thesis in Computer Science (Artificial Intelligence), University of Aix-Marseille II, 1993.

[21] M. Beldjehem, Le Système FENNEC, *Electronic BUSEFAL* **55** (1993) 95-104.

[22] M. Beldjehem, FENNEC, Un Générateur de Systèmes Neuro-Flous, in *Proc. Les Actes des Applications des Ensembles Flous, Nimes, France,* 1993, 209 -218.

[23] M. Beldjehem, the FENNEC System, in *Proc. ACM Symposium on Applied Computing (SAC), Track on fuzzy logic in Applications,* Phoenix, AZ, Marsh 1994, 126-130.

[24] M. Beldjehem, Machine learning based on the Fuzzy-Neuro Possibilistic Hybrid Approach: Design and Implementation, Electronic *BUSEFAL* **87** (2002) 95-104.

[25] M. Beldjehem, Learning IF-THEN Fuzzy Weighted Rules, in *Proc. International conference of computational intelligence,* Nicosia, North Cyprus, 2004.

[26] M. Beldjehem and M. Cheriet, Validation and Verification of Hybrid Min-Max Fuzzy Systems, In *Proc. North American Fuzzy Information Processing (NAFIPS'06)* Montreal, Canada, 2006.

[27] M. Beldjehem and M. Cheriet, Granular Processing of Arabic Handwriting, to appear.

Human Reading Based Strategies for off-line Arabic Word Recognition

A. Belaïd[1] and Ch. Choisy[2]
[1]University Nancy 2-LORIA
Campus Scientifique, 615, rue du jardin botanique
54600 Villers-Lès-Nancy, France,
Email : abelaid@loria.fr

[2]ITESOFT AIMARGUES, Parc d'Andron - Immeuble Le Séquoia, 30470 Aimargues – France
Email : Christophe.Choisy@itesoft.com

Abstract

This paper summarizes some techniques proposed for off-line Arabic word recognition. The point of view developed here concerns the human reading favoring an interactive mechanism between global memorization and local checking making easier the recognition of complex scripts as Arabic. According to this consideration, some specific papers are analyzed and their strategies commented.

1 Type of survey

Concerning Arabic recognition, several surveys have been proposed in the literature considering different points of view:

- By stressing the multiplication of the source of information, from simple classifier to a combination of them, with simple or hybrid choice of the primitives, as described by Essoukri Ben Amara and Bouslama [1].
- By considering the nature of the script: printed or handwritten, its recognition engines and its applications, like in Lorigo and Govindaraju [2].
- By pointing out the nature: symbolic or numeric of the methods as made by Amin [3].

We propose another survey based on the functioning of the human perception going from coarse to detail (i.e. local, analytical or precise). It makes it possible to better justify the choice of the observations, to order them in the classifier cascades, and to propose solutions of correction in the case of conflict or problem and gives finally a smell to all the chain of recognition.

2 Human Perception of Arabic Writing

Arabic is a calligraphic language privileging the global rendering of the whole word to the detail of the letter which is often thinned, crushed, sketched provided that it contributes to the embellishment of the unit (see Figure 1).

Figure 1: Same word written with different possible elongations as described in [1]

Thus the letter can take one to four different forms according to its position in the word. The global becomes the form to be recognized and the letter passes in the second plan favoring the total appearance (see Figure 2). The consequence is a bigger alphabet containing around 100 possible forms [1].

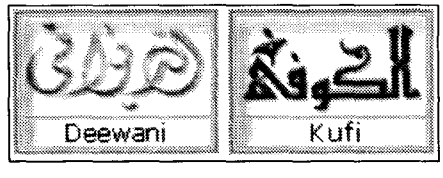


Figure 2: Examples of style fonts of Arabic as described in [4]

However, to facilitate this calligraphic reading, diacritics and accents come first to contribute to the deciphering of the letters which have very similar base shapes. Second, in order to not force too much the writer to maintain the pen lowered to calligraphy only one cursive shape, Arabic offers a decomposition in PAW (Part of Arabic Word) which introduces pauses in the writing having an influence on the recognition process. The PAWs simplify the script apprehension and make easier the linear recognition. Figure 3 gives an example of the Arabic writing complexity, with sub-words and diacritic information.

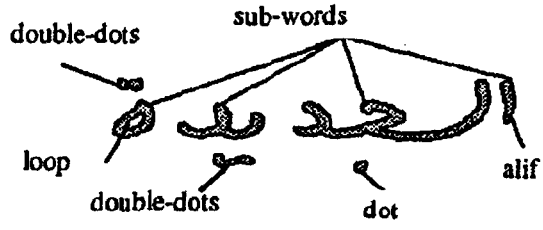


Figure 3: Arabic writing complexity: example of an handwritten word as shown in [28]

Considering the reading process itself and the perception of the writing, Arabic reading seems to be more global than syllabic. It is facilitated by chopping the word in PAWs which makes it finally semi-global.

Some psycho-cognitive experiments proved that the procedure of reading with human starts with a first global vision of the relevant characteristics. The basic experience of reading letter in and outside the word showed the "Word Superiority Effect". In order to illustrate this phenomena, Mc Clelland and Rumelhart proposed a reading model [5]. As illustrated in Figure 4, the model is based on three fundamental hypotheses: 1) the perception is operated in three different processing levels, each one of them is representing a different abstraction level, 2) the perception implies parallel processing on the visual information, 3) the related processes are interactive, i.e. bottom-up and top-down.

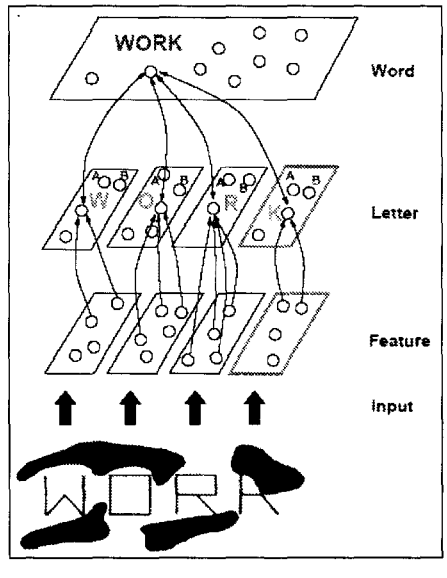


Figure 4: McClelland and Rumelhart Model

- Human builds a complete image of his environment by accumulating different sources of sensory data. In these various stages of decision-making, he proceeds by a general study of the problem. If this global vision is not sufficient, he seeks to go into details [6].
- In the case of Arabic writing, the natural "global" pattern is the PAW: words are combinations of PAWs. Furthermore, there are no clear physical limits between words: words are mainly recognized through the sense of the different PAW aggregation possibilities, where only one should give a sense for an entire sentence. PAWs could be compared to Latin syllables written separately, witch should be correctly gathered to have a meaning.

We can conclude from these standpoints that Arabic writing fits very well the reading principle of Mc Clelland and Rumelhart as it clearly privileges the "Word Superiority Effect", while adding some local perceptual information to help word understanding.

But the corresponding model has to be adapted to consider the PAW intermediate reading level and the letter distortions: PAWs introduce an intermediate global level of information, while letter shape variations make more complex their localization and their modeling.

3 Computing perception levels

All computing methodologies try to simulate a human perception level. Considering human perception of Arabic writing with the particularity of PAW, this leads to divide the classification methods in four classes:

- Global-based vision classifiers.
- Semi-global-based vision classifiers.
- Local-based vision classifiers.
- Hybrid-level classifiers.

3.1 Global-based Vision Classifiers

In this holistic approach, the word is regarded as a whole, allowing correlations on the totality of the pattern. This approach avoids a heavy task of letter localization and recognition and remains very used. However, its interest remains limited to small vocabularies or as a pre-classification step, because its complexity grows linearly with the number of word models.

This category is mainly assimilated to what is called segmentation-free approach. In fact it means that even if a segmentation is used, no local interpretation is made but information is gathered at the word level. In such an approach, one should find the best interpretation possible of a word based on an observation sequence derived from the word image without performing a meaningful segmentation first [7].

Several works on Arabic writing are almost directly derived from Latin studies. Thus the global approach leads to two questions:

- the first one is natural: "it is possible to correctly adapt classical Latin script approaches to Arabic script ?"

- the second one is: "it is possible to extract Arabic words as 'simply' as in Latin script ?"

Srihari et al. propose in [8] a handwritten Arabic word spotting system based on a feature vector similarity measure. The GSC (Gradient, Structural and Concavity) binary features previously used for Latin work in [9] give the best performances. The similarity measure is common to the two languages [9,41]. A precision of 70% is achieved at a recall of 50% when 8 writers were used for training.

The specificity of Arabic writing appears only in a particular part of the work: the word segmentation. Due to the Arabic writing nature, the authors cannot directly evaluate the gap between two consecutive PAWs to decide concerning the word limits (see Figure 5). They use then an NN on a set of 9 features. The authors indicate that one of the most relevant feature is the presence of the "Alef" letter as first letter of many Arabic words [8]. As this hypothesis is not always verified, and due to the natural homogeneous gaps betweens PAWs, the authors achieve only 60% of correct word segmentation.

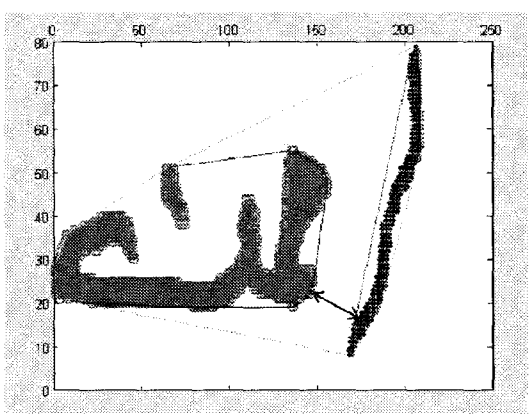


Figure 5: Gap calculation by Srihari et al. in [8]

Al-Badr et al. consider in [28] that segmenting Arabic words into letters is a too much difficult task considering the particular nature of the Arabic writing style, even if it is type printed text. It is why they propose a segmentation-free approach to recognize words. The key idea is to detect a set of shape primitives on the analyzed word, and to arrange them at best in the word space. The interpretation of each primitive depends of its context and position and the posterior probability maximization, allowing to tolerate local misrecognition. Word recognition scores varies according to if the words are clean (99.39%), degraded (95.60%) or scanned (73.13%).

This approach is not specifically dedicated to Arabic Script. Indeed the primitive shapes are very classical: lines of different lengths and orientations, corners, arcs, curves, etc. The independence with the language is so important that the authors assume that they recognize isolated word. Hence, they elude the important problem of Arabic word segmentation even though the event was underlined in the introduction of their paper.

Amin and Mansoor [29] proposed an MLP-based holistic word recognition method for handwritten Arabic words. The MLP input is a global vector composed of 6 kinds of feature vectors carefully chosen to represent globally the word, such as: number of sub-words (up to five), number of peaks of each sub-word (up to seven), number and position of complementary characters and curves within each peak with height and width of the peak. Features are dedicated to Arabic word representation, making the system very specific to the language, even though the models are classical. The recognition rate of 98% on different fonts accredits the interest of adapted language specific features.

But here also a question remains: how the words are located in the text? Even though the authors discusses the problem of PAW extraction and their interest in the word recognition, never they explain how they gather several sub-words into a whole word.

Farah et al. [10] use a battery of three NNs for word recognition, each one is fed by some specific features: statistical, structural or a mixture. Then several combination procedures are tested. The NNs used are classical MLPs with back-propagation algorithm for training. Statistical features are language-independent as they are pixel-based information: the features are the pixel density in various homogeneous zones of the image. Some of the structural features are identical to what is used in Latin systems: ascenders, descenders, loops, writing baseline. Some others are specific to Arabic writing: presence and number of diacritic dots and their position according to the baseline. Words are already isolated in the database, leading the authors to not address in this work the problem of their location. The tests on 2400 word images from 100 different writers achieve 94.93% recognition rate, for 0.97% errors and 4.10% of rejection.

Pechwitz and Märgner [19] used semi-continuous HMMs (SCHMM) representing characters or shapes, as developed by Huang [20]. For each binary image of a word, parameters are estimated after a pre-processing phase normalizing the size and the skew of the word. Then, features are collected using a sliding window approach, leading to a language-independent features (see Figure 6). As in Latin script the middle band of the writing contains the word complexity. For Arabic writing it seems better to look at 3 lines parallel to the baseline at fixed position. The Viterbi algorithm is then used for training and recognition. In training phase a

segmental k-means algorithm is performed. The recognition is done by applying a frame synchronous network Viterbi search algorithm together with a tree structured lexicon representing valid words. The models were combined into a word model for each of 946 valid city names. The system obtained 89% word-level recognition rate using the IFN/ENIT database (26,459 images of Tunisian city-names). The words are yet isolated in the database, thus this work does not have to deal with the word segmentation problem.

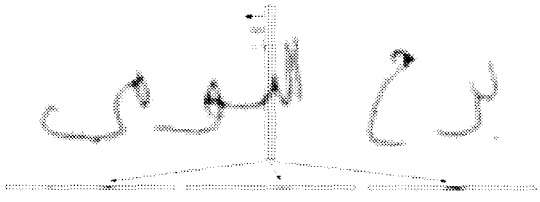


Figure 6: Feature extraction considered by Pechwitz and Märgner in [19]

Khorsheed and Clocksin propose the use of spectral features for printed Arabic word recognition [33]. As mentioned in several works the problem of word segmentation is discarded, assuming to have word images at the input. The originality of this work is in the use of a polar transformation coupled with a Fourier transform that allows to deal with rotation problems (see Figure 7). In a multi-fonts approach the system obtains 95.4% of good word classification by a simple matching with prototypes using the Euclidian distance on 1700 samples of different size, angle and translation.

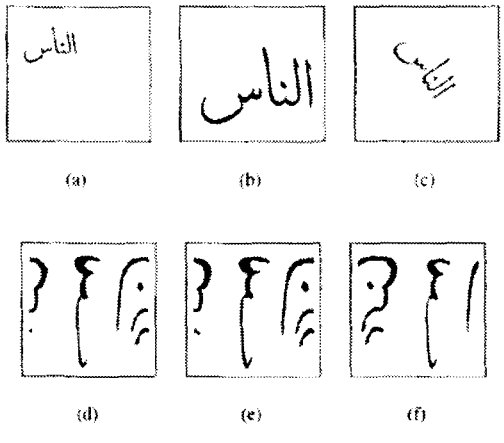


Figure 7: Polar transform of three images of the same word with different scales and rotation [33]

These works clearly accredit the word superiority principle of Mc Clelland and Rumelhart. Indeed many feature combinations and models perform very well.

But the Mc Clelland and Rumelhart model is not respected as only two levels are considered: input features level and word level. This allows to need less precision in the feature research, but the vocabulary should be limited and each word needs its own modeling.

We note that when the input uses low-level features, it is not necessary to introduce specific information, contrarily to the high-level features which need to be adapted to the writing nature. Perhaps should we extend the Mc Clelland and Rumelhart vision model with another layer, linking the pixel information with high-level features, as works the brain with the input images?

Once having a look on these different systems, we can now answer questions asked in the beginning of this section.

The first question is about the direct reuse of Latin systems on Arabic writing. This question has two answers, considering either the models, or the nature of the basic information extracted.

Considering the models, this is clear that almost Arabic systems use the same models and measures than Latin ones. We can deduce that the adaptation of the models is not necessary. Its logical because all the classical models perform information without a priori on its nature [8,10,19,28,29].

Considering the information extraction, all the approaches maintain some classical features used on Latin script. Low-level information based approaches seem able to avoid the add of specific features as they learn them directly [8,29,33]. When high level features are considered, the particularities of Arabic writing lead some authors to search for more specific features, like diacritic points, elongations and curves in the beginning or the end of words [10,29].

The second question asks about Arabic word segmentation (i.e. location in the text) possibilities. The answer is clearly no as the only work that proposes a segmentation method obtains really low segmentation results [8]: all others authors assume to deal with segmented words without approaching this problem.

Arabic word segmentation is much more difficult than for Latin writing for different reasons. The major one becomes from the PAW level which introduces a natural segmentation of the writing, with similar intra-word and inter-words gaps. This problem is underlined by Al-Badr and Haralick in [28] that indicate that the justification of Arabic text is not based on inter-word space adjustment but on elongation of some parts of words. Khedher and Abandah confirm this fact by a statistical study of 262647 Arabic words in [34], showing an average of 4.3 letters and 2.2 PAWs per word: they conclude that the PAW level is the real basic block to be processed rather than word level.

But curiously several works assume a prior word separation, without considering the difficulty of the task. Perhaps is it a reminiscence of Latin works where word segmentation is much easier? But we can think that such an hypothesis can explain why it is so few practical, industrial applications on Arabic language despite of all these good results.

3.2 Semi-global-based Vision Classifiers

The particular nature of Arabic writing allows us to describe the language in a fewer natural level: the PAW level. Indeed Arabic words are built by concatenation of several independent written parts that give another natural segmentation level. This natural segmentation allows us to refine the analysis by reducing the basic vocabulary. It is why some approaches have based their work on this level.

The principal effect of reducing the base vocabulary is the possibility to extend the dictionary. Ben Amara et al. illustrate this fact in [15] where the PAW level allows to deal with a medium vocabulary of city names, that is usually not treatable with a word global approach. The proposed system used a PHMM (Planar Hidden Markov Model) adapted to the PAW morphology. Hence, a shape is vertically decomposed into five horizontal bands, corresponding respectively from top to bottom to the ascenders, the upper diacritic dots, the writing central band, the lower diacritic dots and to the descenders. At each band is associated a super state for which corresponds an horizontal HMM modeling the concerned zone (see Figure 8).

Although the PHMM is a classical model yet used for Latin script [22], its use is clearly dedicated to Arabic writing by the integration of Arabic specificities inside the model. Arabic features are directly integrated in the PHMM by adapting the principal HMM to locate the 5 specific bands of Arabic shapes, while secondary HMMs model the stroke length variations and the diacritic information, aspects that are very specific to Arabic writing. Results are very good as the authors achieve up to 99.84% good recognition for 33168 samples from a vocabulary of 100 PAWs [31].

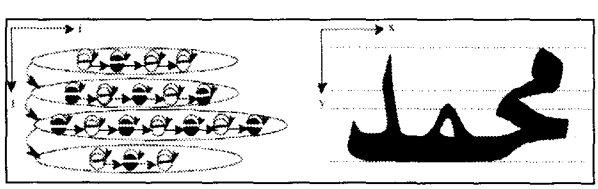


Figure 8: the PHMM architecture as defined by Ben Amara in [15,31]

Burrow confirms in [17] that Arabic word segmentation is harder than for Latin script and proposes to tackle the problem by recognising the PAWs separately. He hopped that this method will be able to cope with the large lexicon of the full database (i.e. IFN/ENIT with 946 different town names from 411 different writers). Interested only by the word shape, he considered the tracing approach as detailed in [10,30] for Latin script. He will in effect be converting an off-line representation into pseudo-on-line representation. Because of diacritics are points, their tracing does not have sense; so, they are discarded from the PAW images. Then, a PAW is transformed in an ordered series of points describing the trace.

Once K-NN classification approach is applied on each PAW, a majority vote is taken on its overall class and repeated for each PAW sample.

First results give 47% accuracy on PAWs. Refining the scoring system and adding some features including the number of dots, this allow the author to score at 74% for PAWs on correctly represented classes. Global word recognition with the add of word-global features is studied and improve greatly results, but one more time the input images are supposed to be entire word in this case: no study is made on the possibility to gather PAW information to find words in a text line.

Concerning the dependence or not of the features on the language specificity we can observe one more time that on one side low-level features are able to simply deal with feature language specificities as Amara showed (i.e. horizontal run length) by adapting the PHMM model [15,31]. On the other side, structural features proposed by Burrow need to be correctly chosen in order to reflect language specificities, and the use or not of them highly influence the results [17].

Curiously, very few work have been made on the PAW level. As assessed in the previous section, this level is however the natural global level of Arabic writing. Thus this section leads to a similar question: is it the influence of Latin works that tend to recognize whole words rather than PAWs?

This is right that for Latin human reading, the PAW level does not exist. The McClelland and Rumelhart vision model confirms this fact, as no intermediate level is given between letters and words. Two solutions exist to adapt this model: the first one is to extend it by adding a PAW layer between letters and words; the second one is to decrease the word level to PAW level, assuming that it is the real global level. The second solution has some advantages and global approaches would benefit to be applied to PAW level for several reasons:

- firstly, the PAW vocabulary is reduced according to the word vocabulary. Thus it is easier to deal with larger vocabularies.
- secondly, as PAW give a natural segmentation of word, the word representation will integrate

it in a way or another. It is thus more logical to divide the representation according to these limits.

- thirdly, it transforms the word segmentation problem into a PAW gathering problem. Now the segmentation problem has only empirical solutions, whereas the PAW gathering can use theoretical frameworks as HMM that can guarantee the optimality of the solution.

We remark that the McClelland and Rumelhart vision model could be extended to a more general approach, where the information gathering could be made recursively through as many levels as necessary. This idea is reinforced by the fact that "good readers" are able to recognize word groups rather than isolated words [44]. A level-recursive approach can then simulate this fact by gathering information through several higher-abstract structures.

3.3 Local-based Vision Classifiers

In this vision level the objective is to focus on letters or smaller entities for their interpretation. The process is thus to gather, bind, confront these entities to identify the word. But such an analysis level leads to the Sayre dilemma: to find letter limits human has to recognize them, and to recognize them human have to localize them. This problem is usually eluded by the use of implicit or explicit segmentation methods.

Fahmy and Al Ali proposed a system with structural features [11]. During a pre-processing phase, slopes and slants are corrected, then some measurements are achieved like stroke width and height of letters. Word is then normalized and encoded in a canonic form, using a skeleton coding approach used on Latin script [21] but adapted to the Arabic writing style. The word image is divided in several frames focusing on character parts, and each frame is divided into three segments. Then, classical features like turnings, junctions and loops were detected from skeletons and used as the input of an ANN. The number of inputs is 35, representing 11 features for each of the three segments of a frame, plus two inputs representing dots. One of these two inputs represents dots if they are above the baseline, while the other input represents dots if they are below the baseline. A recognition rate of 69.7% was obtained on 300 different words written by one writer, with a second writing of the 300 words for the training stage.

Let note that the system try to class the frames: there is no try to gather frames to form a complete character. Another point concerns the word segmentation: here words are separated at the writing time, eluding the difficult problem of word segmentation.

Trenkle et al. propose in [32,39] a printed text recognition system based on an over-segmentation approach (see Figure 9). A full page of text is divided in blocks and lines, and each line is segmented into atomic segments, that are part of character. During the recognition atomic segments are grouped in order to retrieve the whole character according to a Viterbi analysis. Each segment group gives 424 features obtained from horizontal and vertical projections and a edge-based chain code.

An NN is used to classify the segment groups: it has 229 outputs according to the 117 regular Arabic character forms, 80 ligature forms, 10 Arabic digits, 20 punctuation characters, and 2 rejection classes. Classification is also done by a set of decision trees. A Viterbi beam search allows to find the best decoding path on the entire line given an Arabic text model in which one are encoded the rules of Arabic typography. The model combines lexicon-free and lexicon-based approaches, with a vocabulary of 50000 common Arabic words. A dataset of 722 text images of different qualities is used for the realistic tests: the neural network achieves 89.1% of good recognition as the set of decision trees obtains 90.7% recognition rate.

This work is very complete as it addresses all of the problems of printed Arabic text recognition, from page processing and segmentation to text recognition with an ASCII output. The word segmentation problem is elegantly solved by the use of a language model to gather information. We note that the features used are low-level based, allowing to not need integration of Arabic specificities at the character level.

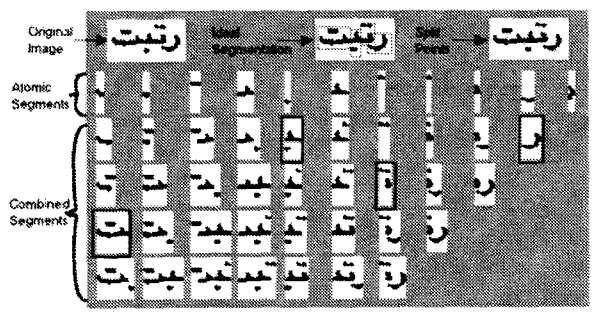


Figure 9: Over-segmentation applied by Trenkle et al. in [46,53]

In Abuhaiba et al. [14], a method for the recognition of free handwritten text is proposed. Based on the skeleton representation, the sub-words are segmented into strokes that were further segmented into "tokens". Tokens are single vertices representing dots or loops or sequences of vertices. A "fuzzy sequential machine" is employed to identify the classes. This machine is composed of sets of initial and terminal states. Stroke directions are used for entering states, and a function for transitioning between states. Tokens are either recognized directly or used to augment the recognizer.

This system achieves 55.4% of good recognition for PAWs with 17.6% of rejection, characters having 51.1% of correct answers with 29.3% of rejection. No lexicon was used, but the PAW vocabulary remains naturally limited. Although the computer used for tests is old and very slow, the approach needs very huge calculation time. As assessed by the authors commenting the relative results, the objective of this work is mainly to propose new theoretical basis and concepts.

Clocksin and Fernando propose in [12] an analytic system for Syriac manuscripts, a West Semitic language which is less grammatically complex than Arabic. The word segmentation is much simpler than for Arabic writing as the intra-word gaps seem to be clearly smaller than inter-word gaps (see Figure 10), contrarily to Arabic writing. But as the grammatical function are almost written as word prefixes or suffixes instead of as separate words, it is not possible to have a global word approach without a very huge dictionary: this language construction is close to the Arabic one. Thus the authors focus on character recognition.

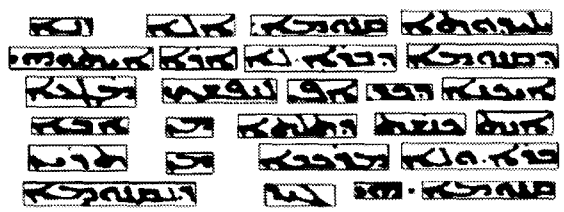


Figure 10: Syriac writing: inter-word gaps are larger than intra-word gaps [12]

A text page is then segmented into words using horizontal and vertical projections. Words are segmented into letters by over-segmenting and removing bad segmentation points, with a segmentation approach specially adapted to Syriac writing: approximately 70% of the characters are correctly segmented. Some features are extracted from character images: moments of different parts of the image and polar transformation. Classification is based on Support Vector Machine (SVM) considering a 'one against one' scheme. As in Arabic writing, a letter can have different shapes and thus can belong to different classes. The best feature combination gives 91% recognition rate for manuscript word letters, and 97% for typeset word letters.

As other previous works, this one uses low-level features, and its adaptation to the language specificities is made through the model learning. The low rate of 70% good character segmentation accredit the fact that, as for Latin scripts, a letter segmentation cannot be done correctly.

Miled et al. [18] proposed an analytical approach based on HMMs for the recognition of Tunisian state names. They integrate the notion of PAW in their system. They group letters having the same body but different diacritics in order to "solve" the problem of diacritic detection and classification. A text line is segmented into PAWs and isolated letters, then PAWs are segmented into graphemes using their upper contour and some heuristic rules, in a way that looks very similar to approaches for Latin script. Each grapheme is encoded into two vectors: the first contains topological features corresponding to human perception, like loops, openings, relative size, relative position, etc., the second one containing moment-like descriptors (here Fourier descriptors were kept). The grapheme identification is based on a K-NN classifier which obtains 84.90% of good recognition.

HMM are used to describe the word composition. Word models are built by concatenating grapheme states, considering several ways to deal with over- and sub-segmentation and a space state to deal with intra-word gaps (see Figure 11). The approach is 'flat': each word has its own HMM representation, and the input sequence is analyzed by all these models. Tests are made on isolated word images, and reach 82.52% of good recognition with a lexicon of 232 words.

This work does not focus the word separation problem, but the HMM approach can solve it at an higher representation level. An interesting aspect of this work is to consider over- and under- segmentation and to integrate the intra-word separation as a blank character: all the segmentation problems of Arabic script are then covered.

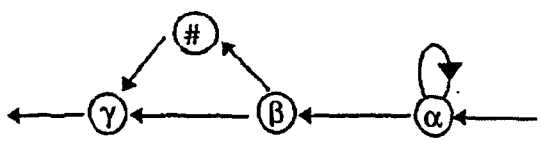


Figure 11: Word model proposed by Miled et al. [18]. α, β, and γ stand for characters, and # represent a space

Fakir et al. propose to use the Hough transform for the recognition of printed Arabic characters [38]. A full text page is digitized, noise cleaned, deskewed and segmented into lines. Then lines are segmented into words assuming that it is a bigger gap between words than between PAW. Character segmentation is based on the projection profile of the middle zone of the word with a fixed threshold that determines the breaks in the projection profile. A second segmentation is applied in order to extract diacritics.

Features are then extracted using the Hough transform, which is applied on the character skeleton to detect strokes (see Figure 12). Thus a character is

represented as a set of strokes. At recognition step the set is compared with the one of reference patterns. A second stage ends the recognition by refining the classification according to the diacritic information. 95% of correct recognition is achieved on 300 characters obtained from the segmentation process. The most common confusions are due to the thinning that brings closer some different patterns.

No information is given about the effectiveness of the segmentation itself, but 300 words were collected and only 300 letters were used for the test: this let suppose that not all the characters were correctly segmented.

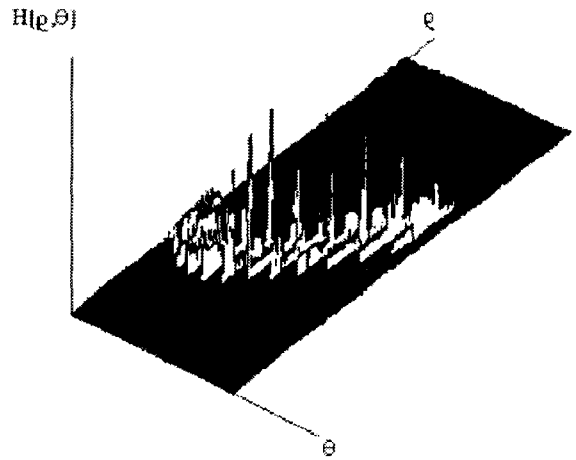


Figure 12: Array of accumulators content given by the Hough transform of a character [38]

Some researches are dedicated to character recognition. They assume that text can be segmented purely in letters. This assumption is not realistic: the segmentation problem was raised long time ago for Latin script, and many studies showed that it is globally not possible. As it seems harder to segment Arabic script, logically this segmentation will not be possible. This point is confirmed by Sari et al. [40] who propose a segmentation system dedicated to Arabic writing: they obtain only 86% of correct letter segmentation.

The interest of these works based on character recognition is reduced to the models and features used that could be integrated into others more complete works.

Alnsour and Alzoubady proposed in [35] a Neocognitron to classify handwritten characters. The input of this particular NN is composed of structural features: Freeman code chain, coordinates of starting and ending points, loops, and primitives like segments with different orientations, corners and dots. These features allow them to achieve 90% character recognition with 3.57% rejection. The system assumes a context of a handwritten Arabic document recognition software that should be able to segment words directly in letters.

Asiri and Khorsheed propose to use two different NN architectures for handwritten Arabic character recognition [36]. The first architecture has 6 output nodes, and is designed to classify the character into one of the 6 groups of similar shapes. According to this first answer, a second NN that correspond to the group will take the final decision. For all NNs the input correspond to a certain number of Haar wavelet transform coefficients. The best results are achieved for 1024 coefficients and give 88% good recognition.

For this work, character samples were collected individually: writers are asked to write isolated letters into small rectangles.

Cowell and Hussain worked on isolated Arabic printed characters [37]. A character image is normalized in 100x100 pixels, then a signature is extracted by counting the black pixels in each row and column. This signature is compared to those of a template set, and the modulus of the difference for each row and each column is summed: lower is the value, closer should be the forms.

The objective is to have a very quick matching method: here the signature matching carries 200 comparisons per template, against 10000 for a direct image matching. No clear result is given but the confusion matrix let suppose a 100% recognition rate.

As explained previously, several approaches give good results, showing that as for Latin script the analytic approach can perform very well. But such an approach presents some drawbacks. First one is the classical problem of bad segmentation that can lead to over- or under- segmentation; such errors generally lead to misclassification. More, the segmentation process has to be adapted to Arabic script, in order to take into account its specificities like vertical ligatures: thus Latin segmentation methods cannot be used efficiently without adaptation.

Some others problems lie in the approach itself, possibly accented by the language.

Thus one problem lies in the observation independence hypothesis. As letters or segments are recognized independently, any error perturbs the whole recognition process. In fact, the McClelland and Rumelhart word superiority effect is not taken into account because the word is not considered as a whole but as a sum of small parts.

Another problem is the inadequacy between segmentation and models. Indeed, the segmentation is based on structural information that is totally independent of the model nature. Thus the modeling is biased at the source by forcing the model to align on non-optimal limits. This is usually solved in two ways:

- the first way deals with the use of higher level features: as the image is interpreted, the distortions are implicitly removed. The drawback is that any bad interpretation of the image makes lose a huge quantity of information, often leading to a misclassification.
- The second way deals with the use of an implicit segmentation: thus the models are able to "choose" their best limits. Unfortunately all the models cannot be used in such an approach without exploding the calculation time.

We note that in Arabic script the notion of letter limits is very variable as horizontal stroke elongations frequently occur in letters and letter ligatures. This accredits the point of view of Choisy [42] that proposed to not search any letter limits: thus the model can focus on pertinent letter information without taking hard separation decision on the fuzzy ligature parts. This proposal fits the McClelland and Rumelhart model, where precise information position is not important but only its presence in an approximate location.

The recognition process is basically based on the use of a combination of a random field (Non Symmetric Half Plane) drawing its observation directly in the image and a HMM taking into account the column observations in the image, hence tackling the problem of length word variations. Figure 13 shows the aspect of this system applied in our Laboratory on Bangla script presenting some calligraphic similarities with Arabic.

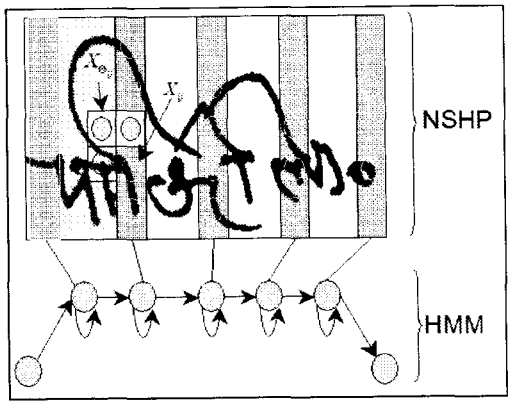


Figure 13: The NSHP-HMM system applied on Bangla script.

3.4 Hybrid-level classifiers

It is possible to combine different strategies so as to approach more the principle of human reading: the analysis must be global for a good synthesis of the information, while being based on local information suitable to make emerge this information [16,23]. Such a combination better fits the human reading, that is proved to firstly analyze global word shapes and searches for local information only to discriminate ambiguous cases.

As local-based approaches gather local information up to words, they could seem to be hybrid ones. But there is an important difference between the two approaches: hybrid approaches try to have a multi-level analysis of the writing, when local-based approaches are only based on the gathering of local information. The aim of hybrid approaches is to combine different levels of features and interpretation, leading to systems closer to the McClelland and Rumelhart proposal.

Souici et al. [24] propose a neuro-symbolic hybridization considering that people rarely, if ever, learn purely from theory or examples. A hybrid system that effectively combines symbolic knowledge with an empirical learning algorithm might be like a student who is taught using a combination of theoretical information and examples. The neural and symbolic approaches are complementary, so their integration is an interesting issue.

For that purpose, they defined a neuro-symbolic classifier for the recognition of handwritten Arabic words. First structural features are extracted from the words contained in the amounts vocabulary. Then, a symbolic knowledge base that reflects a classification of words according to their features is built. Finally, a translation algorithm (from rules to NN) is used to determine the NN architecture and to initialize its connections with specific values rather than random values, as is the case in classical NNs. This construction approach provides the network with theoretical knowledge and reduces the training stage, which remains necessary because of styles and writing conditions variability. The recognition rate varies from 83.55% (4.75% substitution) given by the rule-based approach, 85.5% (14.5% substitution) given by the NN, to 93% (7% substitution) given by the combination.

A similar approach has been applied to handwritten Arabic city-names recognition [43]. The Knowledge Based Artificial NN (KBANN) generated using translation rules is compared to a classical MLP. As the MLP obtains 80% on a 55 vocabulary words, the KBANN performs 92%. The MLP has a less complex architecture than the KBANN, but has a little more neurons.

The hybrid aspect of these work resides in the NN creation: it is based on a multi-level word description, that considers different levels of rules to classify the word according to its number of PAWs, its features and its diacritic information. Thus the network implicitly looks at different perception levels.

Maddouri et al. proposed a combination of global and local models based on a Transparent NN (TNN) [23]. This model is stemmed from the model proposed by

McClelland and Rumelhart for global reading and adapted by Côté [25] for Latin recognition. The TNN is composed of several layers where each one of them is associated to a decomposition level of the word. As Coté's TNN had three layers corresponding to features, letters and words, Maddouri extended it to take into account the Arabic PAW level. Hence the first level corresponds to features, the second to letters, the third to PAWs and the fourth to words. In each level the NN cells represents a conceptual value: primitive, letter, PAW or word (see Figure 14). Training was operated manually by fixing the weights for the cell connections. These weights are determined statistically knowing for each word the various decompositions in the three conceptual levels.

The recognition process is operated during several perceptive cycles, propagating hypotheses from the feature level to word level, looking for their association to the composition levels of the word, and retro-propagating information from the word level to refine the previous features or to extract others. More precisely, in propagation movement the global model proposes a list of structural features characterizing the presence of some letters in the word. Then it proposes a list of possible letters, PAWs and words containing these characteristics. In the back-propagation movement, the activated words and PAWs emit some hypotheses on the letters that could be present. These hypotheses conduce to research the corresponding features, or directly the letter if it has no robust feature. In this last case, a correspondence between the letter image and the corresponding printed one is performed by a local-based model using the correspondence of their Fourier descriptors, playing the role of a letter shape normalizer.

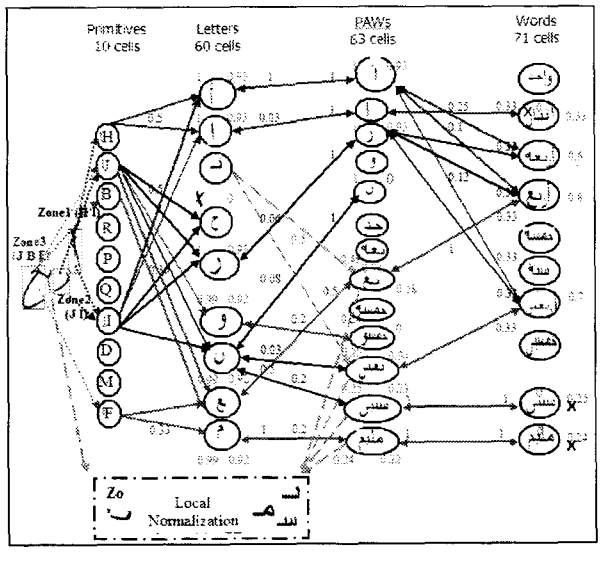


Figure 14: The TNN approach as defined by Maddouri in [23] on the word "Arbaa"

This principle was applied to PAW and word recognition, PAW recognition being made by removing the word layer. The handwritten database contains 2100 images of the 70 word vocabulary of Arabic literal amounts, containing 63 different PAWs. Using only global features, which is like to a simple propagation, PAW recognition rate is 68.42% and word recognition rate is 90%. The add of local features in next perceptive cycles permits to reach a score of 95% for PAWs and 97% for words.

The interest of this approach is to progressively refine the analysis according to the discriminative need of the word dictionary: thus for very distinct shapes a simple propagation can be sufficient, while for words having close shapes more precise information is needed to discriminate them. The "drawback" leads in the information localization, that becomes more and more difficult with the information precision, but this "problem" is inherent to the spirit of such an approach.

This work is very close to the McClelland and Rumelhart approach. The word superiority effect is raised as word recognition performs better than PAW recognition: word shape features are thus sufficient to achieve correct results. The analysis refining principle is clearly efficient, as shows the grow of the scores.

As it is very few hybrid approaches for Arabic writing, we just have a look on some other interesting works on another languages.

Pinales and Lecolinet in [26] proposed a system which is both analytical and global and emphasizes the role of high-level contextual information (see Figure 15). This model is based both on a top-down recognition scheme called backward matching and a bottom-up feature extraction process which is working in a competitive way. This approach has some similarities with the TNN proposed by Côté and Maddouri, as words "retro-propagate" their information to resolve ambiguities and complete missing letters. First results are very encouraging and show that such an approach is very pertinent.

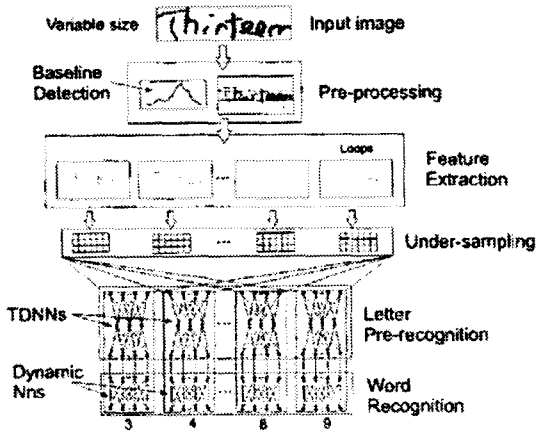


Figure 15: Pinales and Lecolinet Neural Networks combination architecture [26]

Another work has been proposed by Choisy on Latin script [16]. The approach proposes to use the elasticity property of the NSHP to normalize word images in a non-linear way. NSHP states focus pixel features according to the learned ones, giving an implicit state-based normalization. The normalized images are then analyzed by a classical Neural Network. Results show the efficiency of this approach. The drawback of this approach is the compression of the information: even though, contrary to the SDNN, there is a real adaptation of the NN input to the image, normalization is a source of information loss.

Results show that hybrid approaches are very efficient. Corroborating the McClelland and Rumelhart approach, the multi-level analysis allows to refine the analysis with more flexibility than other approaches. In particular, the principle of information focalization rather than segmenting is raised, and seems to be an important point in hybrid approaches.

We can conclude that even if it is not proved that the McClelland and Rumelhart model is the right one to simulate human reading, their approach leads to efficient reading systems. This is very interesting because it links the psychology works and the forma representations by computers. We still remark that the proposed model is clearly oriented towards Latin script reading. For Arabic script it seems necessary to extend the model with a PAW-level. Thus the McClelland and Rumelhart principle is validated but the corresponding model should be adapted to the language considered.

4 Conclusion

Several conclusions can be drawn considering all the research reported in this paper about the recognition of Arabic writing regarding the reading aspect theoretically speaking without neglecting the language characteristics.

First one is that low-level features are language independent. Once extracted (similarly for all the scripts), the training process can arrange their proximity to the language studied. At the opposite, high-level features are language-dependent, and thus need to develop specific extraction methods to retrieve all the information. Obviously, a combination of these two kinds of features should perform better, each feature level complementing the drawback of the other.

Another point is that the PAW level is very important for Arabic script modeling: contrarily to Latin script, the basic entity is not the word. Global approaches should be based on PAW. Analytical ones gain to integrate this information level. A first effect is to reduce the vocabulary complexity by gathering the information on an intermediate level.

Considering the reading approaches, hybrid ones seem to be very promising. They efficiently combine different perceptive levels, allowing to discriminate words without a complete description. In comparison with global approaches, the add of local information allows to extend the vocabulary with less confusions. Compared to local approaches, hybrid ones avoid the full-segmentation problems, and are less disturbed in case of information loss.

Another conclusion stemmed from the works themselves. In particular, two points were raised as very problematical: the segmentation in words and in letters.

The letter segmentation problem was raised long time ago for Latin scripts. Several works were made on this case, and nowadays it is commonly accepted that this problem has no solution. As Arabic writing is mainly described as more complex than Latin one, it seems obvious that a letter segmentation cannot be effective. This leads to the question: why many works are based on such an hypothesis? It seems that the experience gained on Latin languages was not totally transposed to Arabic writing.

Concerning the word segmentation (i.e. location in the text), the problem was probably hidden by the works on Latin script where word separation can be considered as a problem solved in many cases. But for Arabic writing, several works accredit the difficulty of this task. As a word analysis is interesting to show the power of the reading approaches, there is a gap between their modeling and their extraction. We think that the extraction will gain to be made by gathering PAWs through a mathematical formalism like HMM.

Globally we observe that few researches try to take into account the whole problematic of the Arabic script. Thus the word segmentation problem is mainly eluded, the PAW-level global recognition was the object of very few works, and several segmentation-based

approaches made the irrelevant hypothesis of a pure letter segmentation.

Some other problems, like elongations and vertical ligatures, are often cited in the Arabic script description, but less often taken into account in the work itself.

Thus it seems that the main experience brought from Latin works concerns the models and the features, but not the problems encountered and the processes followed to solve them. In fact, many specific problems were raised, but a lot of works consist to try another set of models, features, methods, coming from Latin works.

References

[1] N. Essoukri Ben Amara and F. Bouslama, Classification of Arabic script using multiple sources of information : State of the art and perspectives, in Int. Journal on Document Anal. And Recognition, IJDAR, 2003, 5 :195-212.

[2] L. M. Lorigo and V. Govindaraju, Offline Arabic Handwriting Recognition: A survey, in IEEE Trans. on Pat. Anal. and Mach. Int. (PAMI), vol. 28, n. 5, pp. 712-724, may 2006.

[3] A. Amin, Off-line Arabic Character Recognition: the state of the art, in Pattern Recognition, vol. 31, n. 5, pp. 517-530, 1998.

[4] http://www.islamicart.com/main/calligraphy/styles/kufi.htm

[5] J. L. McClelland and D. E. Rumelhart, An interactive activation model of context effects in letter perception, in Psychological Review, 88: pp. 375-407, 1981.

[6] J. L. McClelland and D. E. Rumelhart, Distributed memory and the representation of general and specific information, in Journal of Experimental Psychology: General, pp.159-188, 1985.

[7] T. Steinherz, E. Rivlin and N. Intrator, Off-line cursive script word recognition: a survey, IJDAR (1999) 2: 90-110.

[8] S. Srihari, H. Srinivasan, P. Babu and C. Bhole, Handwritten Arabic Word Spotting using the CEDARABIC Document Analysis System, Symposium on Document Image Understanding Technology, The Marriott Inn and Conference Center, University Maryland University College, Adelphi, Maryland November 2-4, 2005.

[9] C. I. Tomai, B. Zhang and S. N. Srihari, Discriminatory Power of Handwritten Words for Writer Recognition. Proceedings of the 17th International Conference, 23-26 Aug. 2004 Page(s):638 - 641 Vol. 2.

[10] N. Farah, M. T. Khadir and M. Sellami, Artificial neural network fusion: Application to Arabic words recognition, ESANN'2005 proceedings - European Symposium on Artificial Neural Networks, Bruges (Belgium), 27-29 April 200

[11] M.M.M. Fahmy and S. Al Ali, "Automatic Recognition of Handwritten Arabic Characters Using Their Geometrical Features," Studies in Informatics and Control J., vol. 10, 2001.

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

[13] H. Almuallim, S. Yamagochi: "A method of recognition of Arabic cursive handwriting", Pattern recognition, vol. 9, Nr 5, pp. 715–722, September 1987.

[14] I.S.I. Abuhaiba, M.J.J. Holt, and S. Datta, "Recognition of Off-Line Cursive Handwriting," Computer Vision and Image Understanding, vol. 71, pp. 19-38, 1998.

[15] N. Ben Amara, and A. Belaid, Printed PAW recognition based on planar hidden Markov models Proceedings of the 13th International Conference on Pattern Recognition, 25-29 Aug. 1996 Page(s):220 - 224 vol.2.

[16] Ch. Choisy and A. Belaïd, Coupling of a local vision by Markov field and a global vision by Neural Network for the recognition of handwritten words, in ICDAR'03, Edinburgh, 3-6 August, pp. 849-953, 2003.

[17] P. Burrow. Arabic Handwriting Recognition, Report of Master of Science School of Informatics, University of Edinburgh 2004.

[18] H. Miled, C. Olivier, M. Cheriet, Y. Lecourtier: "Coupling observation/letter for a Markovian modelisation applied to the recognition of arabic handwriting". Proc. of ICDAR'97, pp. 580–583, Ulm, Germany, 1997.

[19] M. Pechwitz and V. Maergner, HMM Based Approach for Handwritten Arabic Word Recognition Using the IFN/ENIT- Database, 7th ICDAR, Vol. 2 p. 890-894, 2003.

[20] X. D. Huang, K.F. Lee and H. Hon, "On Semi-continuous Hidden Markov Modeling, " in Proc. of the IEEE Int. Conf. Acoust., Speech, Signal Processing, pp. 689--692, April 1990.

[21] A. W. Senior et A. J. Robinson. An Off-Line Cursive Handwriting Recognition System. PAMI, 20(3) :308–321, March 1998.

[22] R. Bippus. 1-Dimensional and Pseudo 2-Dimensional HMMs for the Recognition of German Literal Amounts. ICDAR'97, vol 2, pages 487–490, Ulm, Germany, Aug. 1997.

[23] S. Snoussi Maddouri, H. Amiri, A. Belaïd and Ch. Choisy, Combination of Local and Global Vision Modeling for Arabic Handwritten Words Recognition, in Eighth IWHFR, pp. 128-132. 2002.

[24] L. Souici, and M. Sellami, "A hybrid approach for Arabic literal amounts recognition," AJSE, the Arabian Journal for Science and Engineering, Volume 29, Number 2B, pp: 177-194, October 2004

[25] M. Côté, E. Lecolinet, M. Cheriet and C. Y. Suen, Building a perception based model for reading cursive script, in ICDAR, vol. II, pp. 898-901, 1995.

[26] J. Ruiz-Pinales and E. Lecolinet, A new perceptive system for the recognition of cursive handwriting, ICPR, vol. III, pp. 53-56, 2002.

[27] B. Zhang and S. N. Srihari, "Binary vector dissimilarity measures for handwriting identification," Proceedings of the SPIE, Document Recognition and Retrieval , pp. 155–166, 2003.

[28] B. Al-Badr and R. M. Haralick, A segmentation-free approach to text recognition with application to Arabic text, in Int. Journal on Document Anal. And Recognition, IJDAR, 1998, 1:147-166.

[29] A. Amin and W. Mansoor, Recognition of Printed Arabic Text using Neural Neworks, in ICDAR'97, volume II, pp 612-615, 1997

[30] H. Nishida. An approach to integration of off-line and on-line recognition of handwriting. Pattern Recognition Letters, 16(11):1213–1219, 1995.

[31] H. Miled and N. E. Ben Amara, Planar Markov Modeling for arabic Writing Recognition : Advancement State. ICDAR 2001, volume I, pp 69-73, Seattle, 2001.

[32] J. Trenkle and A. Gillies and E. Erlandson and S. Schlosser and S. Cavin, "Advances in Arabic Text Recognition", Proceedings of the Symposium on Document Image Understanding Technology, Columbia, Maryland, 2001.

[33] M. S. Khorsheed and W. F. Clocksin, Spectral features for Arabic word recognition, ICASSP'00, page(s): 3574-3577, vol.6, 2000.

[34] M. Z. Khedher and G. Abandah, Arabic Character Recognition using Approximate Stroke Sequence, Arabic Language Resources and Evaluation Status and Prospects, LREC2002, Las Palmas de Gran Canaria, 1st June 2002

[35] A. J. Alnsour and L. M. Alzoubady, Arabic Handwritten Character Recognized by Neocognitron Artificial Neural Network, University of Sharjah Journal of Pure and Applied Sciences, vol. 3, n°2, june 2006.

[36] A. Asiri and M. S. Khorsheed, Automatic Processing of Handwritten Arabic Forms Using Neural Networks, Transactions on Engineering, Computing and Technology, vol. 7, august 2005.

[37] J. Cowell and F. Hussain, A fast recognition system for isolated Arabic character recognition IEEE Information Visualization IV2002 conference July 2002, London

[38] M. Fakir, M. M. Hassani and C. Sodeyama, On the recognition of arabic characters using Hough transform technique, Malaysian Journal of Computer Science, vol. 13, n°2, 2000

[39] A. Gillies and E. Erlandson and J. Trenkle and S. Schlosser, "Arabic Text Recognition System", Proceedings of the Symposium on Document Image Understanding Technology, Annapolis, Maryland, 1999.

[40] T. Sari, L. Souici, and M. Sellami, Off-line Handwritten Arabic Character Segmentation Algorithm: ACSA, Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR'02), p 452, Niagara-on-the-Lake, Canada, 2002.

[41] M. S. Jelodar, M. J. Fadaeieslam, N. Mozayani and M. Fazeli, A Persian OCR System using Morphological Operators, The Second Worl Enformatika Conference, WEC'05, pp 137-140, February 25-27, Istanbul, Turkey, 2005.

[42] Ch. Choisy and A. Belaid, Cross-learning in analytic word recognition without segmentation, in Int. Journal on Document Anal. And Recognition, IJDAR, 4(4): 281-289, 2002.

[43] L. Souici, N. Farah, T. Sari, and M. Sellami, "Rule Based Neural Networks Construction for Handwritten Arabic City-Names Recognition", Proceedings of the 11th International conference Artificial Intelligence: AIMSA 2004, Varna, Bulgaria, September 2004, , LNAI 3192, pp: 331-340, Springer, ISSN: 0302-9743.

[44] http://home.ican.net/~galandor/littera/syn_cor1.htm

Versatile Search of Scanned Arabic Handwriting

Sargur N. Srihari, Gregory R. Ball and Harish Srinivasan
Center of Excellence for Document Analysis and Recognition (CEDAR)
University at Buffalo, State University of New York
Amherst, New York 14228
{srihari, grball, hs32}@cedar.buffalo.edu

Abstract

Searching scanned handwritten documents is a relatively unexplored frontier for documents in any language. In the general search literature retrieval methods are described as being either image-based or text-based with the corresponding algorithms being quite different. Versatile search is defined as a framework where the query can be either a textual string or an image snippet in any language and the retrieval method is a fusion of text- and image-retrieval methods. An end-to-end versatile system known as CEDARABIC is described for searching a repository of scanned handwritten Arabic documents; in addition to being a search engine it includes several tools for image processing such as line removal, line segmentation, creating ground-truth, etc. In the search process of CEDARABIC the query can be either in English or Arabic. A UNICODE and an image query are maintained throughout the search, with the results being combined by an artificial neural network. The combination results are better than each approach alone. The results can be further improved by refining the component pieces of the framework (text transcription and image search).

1 Introduction

While searching electronic text is now a ubiquitous operation, the searching of scanned printed documents such as books is just beginning to emerge. The searching of scanned handwritten and mixed documents is a virtually unexplored area.

Processing handwritten Arabic language documents is of much current interest. One unsolved problem is a reliable method, given some query, to search for a subset among the many such documents, similarly to searching printed documents. The problem is challenging because of the unique structural features of Arabic script and the relative infancy of the field of handwriting processing.

Content-based information retrieval (CBIR) is a broad topic in information retrieval and data mining [1]. CBIR algorithms are quite different for the tasks of text retrieval and image retrieval. Correspondingly there are two approaches to searching scanned documents, stemming from the two different schools of thought. One approach is to use direct content based image retrieval (word spotting). Another is to convert the document to an electronic textual representation (ASCII for English and UNICODE for Arabic) and search it with text information retrieval methods used routinely with electronic documents. Both of these approaches can be successful under ideal circumstances, but such a situation is difficult to achieve with current technology of handwriting recognition. Image based searches do not always return correct results. Arabic handwriting recognition technology does not come close to allowing full transcriptions of unconstrained documents. However, by combining these two methods together, we achieve better performance than either on its own.

The paper describes a framework for versatile search of Arabic handwritten documents. By versatile search, we mean both versatility in the query and versatility in the search strategy–combining content based image retrieval and text-based information retrieval. Versatality in the query refers to to the query being either in textual form or electronic form. Another characteristic of versatile search is that the query can be in multiple languages such as English and Arabic. In the versatile search process both the original scanned image and the (partial) transcription are maintained at all stages. Searches proceed in parallel on both document representations. Any query is also split into both an image and a UNICODE representation which act on the corresponding instance of the document. The results from both parallel searches are combined into a single ranking of candidate documents.

The rest of the paper is organized as follows. Section 2 describes previous work in scanned document retrieval. Section 3 describes the nature of queries for versatile search. Section 4 describes the overall

system architecture as well as the process of segmentation necessary for image indexing and retrieval. Section 5 describes the image-based word shape matching approach. Section 6 describes the text-based approach that is based on character recognition. Results in the form of precision-recall with a corpus of handwritten Arabic are described in Section 7. Conclusions and future directons are given in Section 8.

2 Related Work

Searching scanned printed documents in English has had significant success. Taghva et al showed [13] that information retrieval performance continues to be high even when OCR performance is not perfect. Russell et al [6] note this can, at least in part, be attributed to redundancy and the fact that while OCR performance may commit some errors, it performs very well for English. They go on to discuss the use of handwritten and typed queries. Arabic recognition technology, however, generally performs significantly poorer than its English counterpart.

A system for directly searching scanned handwritten English handwriting was discussed in [7]. This sytem, known as CEDAR-FOX, was developed for forensic document analysis applications [12],[8]. Searching scanned Arabic text within a system known as CEDARABIC was first reported in [9]. The CEDARABIC system was based on an end-to-end software system that was previously developed for English handwriting known as CEDAR-FOX. Both systems are designed to be interactive for use by a human document examiner and have many pre-processing operations such as line and word segmentation, rule-line removal, image enhancement, etc.

3 Queries and Searches

The query can take several forms: (i) a UNICODE string of Arabic text (for example, entered on an Arabic keyboard), specifying a word or words the user wants to appear in the handwritten document, (ii) an English word or words corresponding to an idea that should appear in the Arabic document, (iii) an image of an Arabic word or words; documents should be returned that also have a representation of this Arabic word.

Word spotting algorithms start with an image query; either a full word or component prototype characters. The document is searched directly, with only minor preprocessing steps such as noise removal, etc. We take two approaches for word spotting: word shape and character shape based. In the word shape based method, features are extracted from prototype word images. This prototype can either be provided with the query, or can be looked up from a library of images based on a keyword. We then compare the features of a candidate word to the prototype words, choosing the best match. The character shape based method splits a candidate word into sequences of candidate component characters. Each sequence is matched to prototypes of the characters in the query word, and the sequence of candidate characters containing the overall maximum similarity to the prototype characters receives the highest score for that word, with the score acting as a confidence measure. The word shape based method performs well when many prototype images are present, but cannot be used if there are none available; at that point the character based method is the only available approach. In situations where both methods are applicable, their rankings are combined.

To partially transcribe documents we use several approaches of word recognition. A baseline, simple method is to perform a variation of character recognition and try to directly deduce a word. Arabic has an advantage over languages such as English because of the presence of subwords which are predictably distinct. In a second method, we compare the candidate characters against those suggested by a lexicon of words, choosing the candidate representation with the best score. Since larger lexicons generally result in poorer performance we limit the lexicon size when using such a method.

4 Framework

Figure 1 gives an overview of the versatile search framework. The key point is that both image and text queries are maintained against image and text versions of the document throughout the searching process, with their results in the end being combined with a neural network.

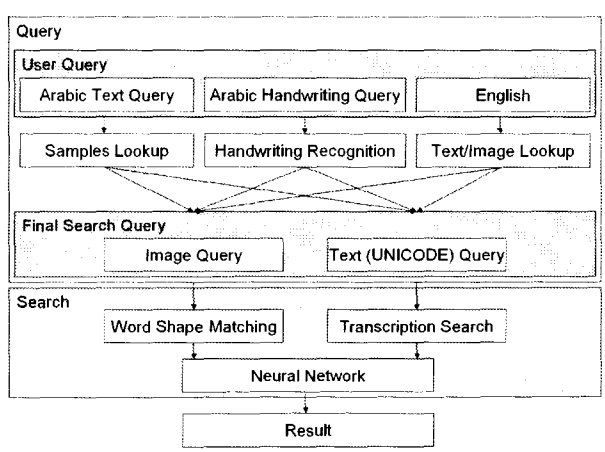


Figure 1: Versatile Search Framework

A critical common preprocessing step necessary

for both methods is segmenting a page into lines, and sometimes a line into words. A CEDARABIC representation of a segmented document is shown in Figure 2.

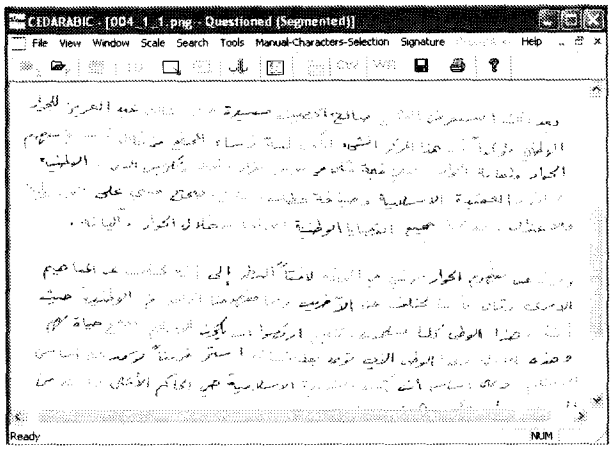


Figure 2: Segmented Document

4.1 Segmentation Algorithms

Automatic word segmentation, as presented in [10], is based on taking several features on either side of a potential segmentation point and using a neural network for deciding whether or not the segmentation is between two distinct words. Some of the differences between the tasks of segmenting Arabic script and segmenting Latin script are the presence of multiple dots above and below the main body in Arabic and the absence of upper case letters at the beginning of sentences in Arabic. The method presented was found to have an overall correctness of about 60%.

The segmentation free method attempts to perform spotting and segmentation concurrently. Rather than a candidate word image, an entire line image acts as input. The line is split into segments based on an algorithm similar to the ligature-based segmentation algorithm used in [4]. All realistic combinations of adjacent connected components are considered as potential areas where the desired word may appear. This approach more exhaustively searches a line, looking for a given word image, while at the same time keeping the number of evaluations manageable by considering only a small subset of potential regions in the image.

4.2 Automatic Word Segmentation

The process of word segmentation begins with obtaining the set of connected components for each line in the document image. Figure 3 and Figure 4 show the connected components (exterior and interior contours) of a small region of a document image. The interior contours or loops in a component

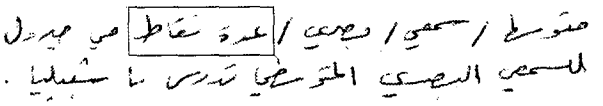


Figure 3: Region of a document image

Figure 4: Connected component exterior and interior contours

are ignored for the purpose of word segmentation as they provide no information for this purpose. The connected components are grouped into clusters, by merging minor components such as dots above and below a major component. Also particular to Arabic, many words start with the Arabic character "Alef". The presence of an "Alef" is a strong indicator that there may be a word gap between the pair of clusters. The height and width of the component are two parameters used to check if the component is the character "Alef". Figure 7 shows samples of the Arabic character "Alef". Every pair of adjacent clusters are candidates for word gaps. Nine features are extracted for these pairs of clusters and a neural network is used to determine if the gap between the pair is a word gap. The nine features are: width of the first cluster, width of second cluster, difference between the bounding box of the two clusters, flag set to 1 or 0 depending on the presence or absence of the Arabic character "Alef" in the first cluster, the same flag for the second cluster, number of components in the first cluster, number of components in the second cluster, minimum distance between the convex hulls enclosing the two clusters and the ratio between, the sum of the areas enclosed by the convex hulls of the individual clusters, to the total area inside the convex hull enclosing the clusters together. The minimum distance between convex hulls is calculated by sampling points on the convex hull for each connected component and calculating the minimum distance of all pairs of such points. Figure 5 and Figure 6 show two pairs of adjacent clusters tested for word gaps. Dotted lines indicate convex hulls around the individual clusters. Solid lines indicate convex hull around the two clusters taken together.

A neural network was trained using these nine fea-

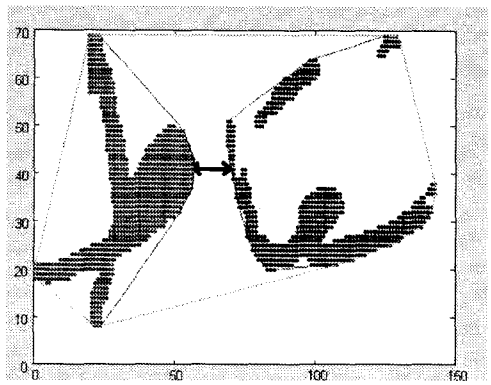


Figure 5: Gap between the convex hulls is not a word gap

Figure 6: The gap between the convex hulls is a word gap

tures with the feature vector labeled as to whether it is a word gap or not. This is similar to the neural network approach used for English postal addresses [5], but with different features.

4.2.1 Word Segmentation Performance

When applied to the document set described earlier with correctly segmented lines, the overall performance is about 60% using a set of seven segmentation features. In [10], the authors noted that a more complex set of features is expected to yield a higher level of performance.

4.3 Segmentation-free Line Processing

The segmentation free algorithm processes the words on a per line basis rather than relying on presegmented words. The algorithm can be viewed as a sequence of steps. First, the image is processed into

Figure 7: Samples of Arabic character "Alef". The height and width are two parameters that are used to detect the presence of "Alef" in the clusters.

component lines. Candidate segmentation points are generated for a given line. The line is scanned with a sliding window, generating candidate words and scoring them, as well as filtering out nearly equivalent candidates.

4.3.1 Segmentation Algorithm

The segmentation algorithm used on the line is essentially the same as the one used to generate candidate character segmentation points in candidate words in the actual spotting step. It is performed via a combination of ligatures and concavity features on an encoded contour of the components of the image. Average stroke width is estimated and used to determine the features.

Ligatures, as noted in [3] are strong candidates for segmentation points in cursive scripts. Ligatures are extracted in a similar way as in [3]–if the distance between y-coordinates of the upper half and lower half of the outer contour for a x-coordinate is less than or equal to the average stroke width, then the x-coordinate is marked as an element of a ligature. Concavity features in upper contour and convexities in the lower contour are also used to generate candidate segmentation points, which are especially useful for distinct characters which are touching, as opposed to being connected. A ligature will cause any overlapped concavity features to be ignored. For a given x-coordinate, if a concavity and convexity overlap, a segmentation point is added for that x-coordinate.

While the character based method described in [3] uses this segmentation method to split a word into candidate characters, the segmentation free line processing method uses it to split the line, the motivation being to generate candidate word regions on the line. Arabic has predicable breaks in a word based on non-connective characters. Therefore, the number of connected components in a word is predictable as well.

4.3.2 Line Scanning

The method utilizes a sliding window, starting from the left of the line and proceeding to the right (in the opposite direction to the way Arabic is written), although the direction of the scan is unimportant because all realistic combinations of connected components will be considered.

Each character class c in Arabic is associated with a minimum and a maximum durational length ($minlen(c)$ and $maxlen(c)$ respectively). These lengths are generated by segmenting a representative dataset of characters with the same segmentation algorithm, and taking the min and max for each character. Due to the nature of the Arabic character set, the upper bound for all characters is 5, not 4 as in [4].

The scanning algorithm will scan for candidate words consisting of a range of segments. For a given search word W of length n, for each character $c_i \in W$, the minimum length $minlen(W)$ considered is $\sum_{i=0}^{n-1} minlen(c_i)$ and the maximum considered length $maxlen(W)$ is $\sum_{i=0}^{n-1} maxlen(c_i)$.

The scanning algorithm starts at each segmentation point p on a line. For a given point p_i, if $i = 0$ or if $p_i.left > p_{i-1}.right$ (i.e., there is horizontal space to the left of the segmentation point) it is considered a valid start point. Similarly, for a given point p_i, if $i = max(p)$ or if $p_i.right < p_{i+1}.left$ (i.e., there is horizontal space to the right of the segmentation point) it is considered a valid endpoint. The algorithm considers candidate words to be ranges of segments between two segmentation points p_i and p_j where p_i is a valid start point, $minlen(W) \leq j - i + 1 \leq maxlen(W)$, and p_j is a valid endpoint.

While this generally results in more candidate words than the other segmentation method, since each Arabic word is only broken into a few pieces separated by whitespace it does not result in a dramatic decrease in performance.

4.3.3 Filtering

Often, a candidate word influences neighboring candidate words' scores. Neighboring candidate words are those words with overlapping segments. Often, a high scoring word will result in high scores for neighbors. The largest issue rises when the high scoring word is, in fact, an incorrect match. In this case, the incorrect choice and several of its neighboring words receive similarly good scores, pushing the rank of the actual word lower in the list. Another issue is if the word being searched for appears multiple times in a document. The best matching words' neighboring candidates depresses the second occurrence's rank. Various ways of dealing with the overlap meet with different degrees of success.

The approach taken in the current incarnation of the algorithm is to keep the candidate word that has the highest score out of the overlapping words. Unfortunately, this occasionally removes the correct word completely from the list. Alternate methods of filtering are being explored.

5 Word Shape Matching

The word segmentation is an indexing step before using the word shape matching for word retrieval. A two-step approach is employed in performing the search: (1) prototype selection: the query (English text) is used to obtain a set of handwritten samples of that word from a known set of writers (these are the prototypes), and (2) word matching: the prototypes are used to spot each occurrence of those words in the indexed document database. A ranking is performed on the entire set of test word images-where the ranking criterion is the mean similarity score between prototype words and the candidate word based on global word shape features.

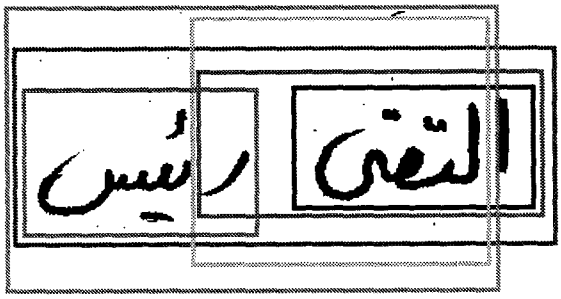


Figure 8: Candidate word regions

5.1 Prototype Selection

Prototypes which are handwritten samples of a word are obtained from an indexed (segmented) set of documents. These indexed documents contain the truth (English equivalent) for every word image. Such an indexing can be done using a transcript mapping approach such as described in [2]. Synonymous words if present in the truth are also used to obtain the prototypes. Hence queries such as "country" will result in selecting prototypes that have been truthed as "country" or "nation" etc... A dynamic programming Edit Distance algorithm is used to match the query text with the indexed word image's truth. Those with distance as zero are automatically selected as prototypes. Others can be selected manually.

5.2 Word Matching

The word matching algorithm uses a set of 1024 binary features for the word images. These binary features are compared using the correlation similarity measure 5.2.1 to obtain a similarity value between 0 and 1. This similarity score represents the extent of match between two word images. The smaller the score, the better is the match. For word spotting, every word image in the test set of documents are compared with every selected prototype and a distribution of similarity values is obtained. The distribution of similarity values is replaced by its arithmetic mean. Now every word is sorted in rank in accordance with this final mean score. Figure 9 shows an example query word image compared with the a set of 4 selected prototypes.

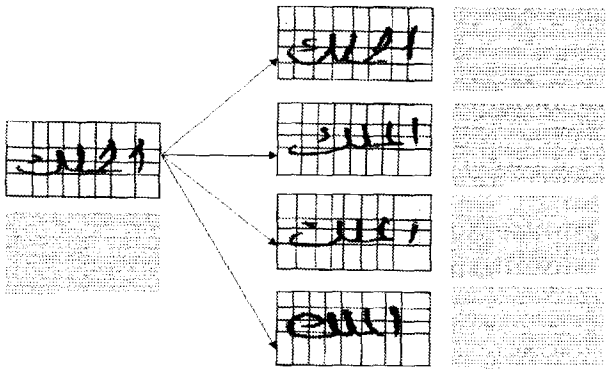


Figure 9: One query word image (left) matched against four selected prototype images. The 1024 binary GSC features are shown next to the images.

5.2.1 Similarity measure

The method of measuring the similarity or distance between two binary vectors is essential. The correlation distance performed best for GSC binary features [14] which is defined for two binary vectors X and Y , as in equation 1

$$d(X,Y) = \frac{1}{2}\left(1 - \frac{s_{11}s_{00} - s_{10}s_{01}}{[(s_{10}+s_{11})(s_{01}+s_{00})(s_{11}+s_{01})(s_{00}+s_{10})]^{\frac{1}{2}}}\right) \quad (1)$$

where s_{ij} represent the number of corresponding bits of X and Y that have values i and j.

5.3 Word-Matching Performance

The performance of the word spotter was evaluated using manually segmented Arabic documents. All experiments and results were averaged over 150 queries. For each query, a certain set of documents are used as training to provide for the template word images and rest of the documents were used for testing. Figure 14 shows the percentage of times when correct word images were retrieved within the top ranks scaled on the x-axis. More than one correct word image does exist in the test set. For example, if the word "king" occurs in document 1 twice, and we use writers 1 through 6 for training, then we have $6 * 2 = 12$ template word images to perform word spotting. The remaining test set contains $(10-6)*2 = 8$ correct matches in the test set. The top curve of the figure shows that at least one correct word is retrieved 90% of the time within the top 12 ranks. The other curves show the same when at least 2 and 3 correct words are retreived. Similar plots can be obtained when different numbers of writers are used for training.

Performance of word spotting can be specified in terms of precision and recall. If n_1 is the number of relevant words in the database, n_2 is the number of words retrieved, and n_3 is number of relevant words among those that are retrieved then $Precision = n_3/n_2$ and $Recall = n_3/n_1$.

6 Text String Matching

The approach is lexicon based; that is, it makes use of the Arabic sequences of characters for words in the lexicon in order to select sets of prototype images representing the characters forming them. A preprocessing step is a line and word segmentation process, such as the one described in [10]. The candidate word image is first split into a sequence of segments (as in Figure 10), with the ideal result being individual characters in the candidate word being separated. The segmentation algorithm used "oversegments" words in the hopes of avoiding incorrectly putting more than a single character into a segment. Segments are then rejoined and features extracted, which are in turn compared to features of prototype images of the characters. Further issues of undersegmenting unique to Arabic are dealt with using compound character classes. A score that represents the match between the lexicon and the candidate word image is then computed. The score relates to the individual character recognition scores for each of the combined segments of the word image.

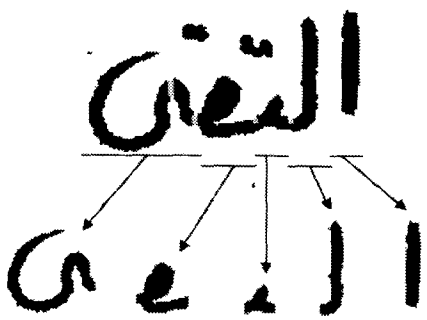


Figure 10: Arabic word "king", with segmentation points and corresponding best segments shown below

6.1 Character Dataset

Unlike the method presented in [10], this method depends on character images rather than word images to form a basis for comparison. In this method, the image is essentially cut into component characters and each character is matched for similarity, in contrast to the latter's method of matching entire word shape. Since the number of characters in Arabic is a rather small compared with the number of potential words, a library of component characters can be incorporated directly into the system. This eliminates the indexing phase of [10].

As such a character database was not readily available, a new character image dataset was derived from the existing Arabic document dataset produced from the CEDARABIC [9] project. The original dataset consisted of a collection of handwritten doc-

uments produced from a variety of authors and is described in Section 7.1. The scanned words were individualized and raw ASCII descriptions of the Arabic characters were given corresponding to the content of the word. The derived dataset consists of images of single Arabic characters and character combinations. Approximately 2,000 images of characters and character combinations in other configurations were created by allowing the ligature based segmentation algorithm to create candidate supersegments of the truthed words, and manually matching the best candidate supersegments to the corresponding character or character combination when the segmentation was successful. Both left to right and right to left versions of the writings were tested, the original right to left images producing better results (left to right versions occasionally seemed to be more prone to undersegmenting the words). The 2,000 images represent but a small fraction of potential images from this dataset. Work on extending this dataset is ongoing.

6.2 Features

WMR features for each of the character images were extracted and incorporated into the recognition engine of CEDARABIC. As described in [11], the WMR feature set consist of 74 features. Two are global features–aspect and stroke ratio of the entire character. The remaining 72 are local features. Each character image is divided into 9 subimages. The distribution of the 8 directional slopes for each subimage form this set (8 directional slopes $\times$ 9 subimages = 72 features). $F_{l_{i,j}} = s_{i,j}/N_i S_j$, $i = 1, 2, ..., 9$, $j = 0, 1, ..., 7$, where $s_{i,j}$ = number of components with slope j from subimage i, where N_i = number of components from subimage i, and $S_j = max(s_{i,j}/N_i)$. These features are the basis of comparison for the character images derived from the segmentation of words to be recognized. To date, this appears to be the first application of WMR features to Arabic recognition.

To obtain preliminary results, the base shape of a letter was mapped to all derivations of that letter. For example, the base shape of the character *beh* was mapped to *beh*, *teh*, and *theh*. The initial and medial forms of *beh* were also mapped to the initial and medial forms of *noon* and *yeh*. If separately truthed versions were available specifying explicit membership to, for example, *teh*, such characters were only included in *teh*'s set of features.

6.3 Image Processing And Segmentation

Image processing is accomplished via a method similar in part to that described in [3]. First, a chain code representation of the binary image's contours is generated. Noise removal, slant correction, and smoothing is performed. Segmentation is performed via a combination of ligatures and concavity features on an encoded contour of the components of the image. Average stroke width is estimated and used to determine the features. The number of segmentation points is kept to a minimum, but unlike in [3], the maximum number of segmentation points per character is five. WMR features are extracted from segments.

One goal of this method is to oversegment words with the hopes of eliminating undersegmentation altogether. Undersegmentation in ligature based segmentation of Arabic text, however, continues to be problematic due to the presence of character combinations and vertically separated characters. For example, some writing styles do not mark certain letters with much clarity–especially initial characters, for example initial *yeh*'s. Since the ligature based segmentation proceeds horizontally, seeking breaking points at various positions along the x-axis, the vertical "stacking" of characters cannot be dealt with simply by increasing the sensitivity of the segmentation (see Figure 11). To deal with these issues, character classes were defined corresponding to the common character and vertically occurring combinations.

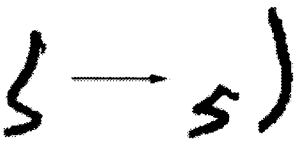


Figure 11: Left: (horizontally) unsegmentable character combination, Right: individual images

6.3.1 Preprocessing Lexicon

An Arabic word is specified as a sequence of the approximately 28 base letters. To aid recognition, a simple algorithm maps the given text to the correct variation of each character. For example, "Alef|Lam|Teh|Qaf|Alef maksura|" is mapped to "Alef$_i$|Lam$_i$|Teh$_m$|Qaf$_m$|Alef maksura$_f$|" where "i" means the letter is in the initial position, "m" means the letter is in the medial position, etc. Additional post-processing steps to the Arabic lexicon combine adjacent individual characters in appropriate positions into character combination classes. For example, "Lam$_i$|Meem$_m$|" is mapped to "Lammeem$_i$|."

The new mapping system for the 150 new character classes were incorporated into the character recognition model of CEDARABIC, replacing the support for English letters carried over from CEDAR-FOX.

6.3.2 Word Recognition

The objective is to find the best match between the lexicon and the image. In contrast to [3], up to five adjacent segments are compared to the character classes dictated as possibilities by a given lexicon entry. In the first phase of the match, the minimum Euclidean distance between the WMR features of candidate supersegments and the prototype character images is computed. In the second phase, a global optimum path is obtained using dynamic programming based on the saved minimum distances obtained in the first matching phase. The lexicon is ranked, the entries with the lowest total scores being the closest matches. Figure 10 shows the the Arabic word "king" split into its best possible segments.

Testing proceeded on the same 10 authors' documents as in [10]. Recognition was attempted on approximately 180 words written by each of the 10 authors (for a total of approximately 1,800 words). Recognition was attempted in two runs, one with a lexicon size of 20 words and one with a size of 100. The lexicon was generated from other words among the 180 being recognized. The words and the lexicons in the tests were the same for all authors.

6.4 Word Spotting

Word spotting proceeds in a very similar fashion to word recognition. In the case of word spotting, the lexicon consists only of the word being spotted. A score against this lexicon entry is generated for each candidate word in the document. The candidate words are ranked according to score, the words with the best scores are most likely to be the word being spotted.

From the documents written by the authors, 32 words were chosen at random and "spotted," in a similar fashion to the experiments performed in [10]. Note that the recall for word spotting when utilizing the expanded Arabic character classes is nearly 80% for a precision of 50, which is a significant improvement over the other methods (using simply the Arabic letters individually and the image based method described in [10]).

7 Results

7.1 Document Image Database

For evaluating the results of our methods, we used a document collection prepared from 10 different writers, each contributing 10 different full page documents in handwritten Arabic. Each document is comprised of approximately $150 - 200$ words each, with a total of $20,000$ word images in the entire database. The documents were scanned at a resolution of 300 dots per inch. This is the resolution to be used for optimal performance of the system. Figure 12 shows a sample scanned handwritten Arabic document written by writer 1.

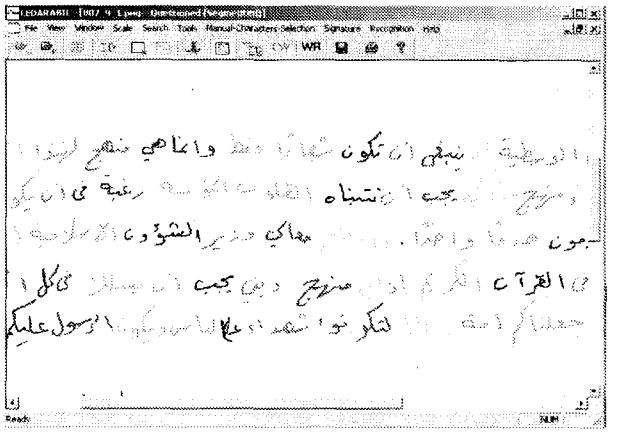


Figure 12: Sample ".arb" file opened with CEDARABIC. Each segmented word is colored differently from adjacent words.

For each of the 10 documents that were handwritten, a complete set of truth values, comprised of the alphabet sequence, meaning, and the pronunciation in that document was also given. The scanned handwritten documents' word images were mapped with the corresponding truth information (see Figure 13).

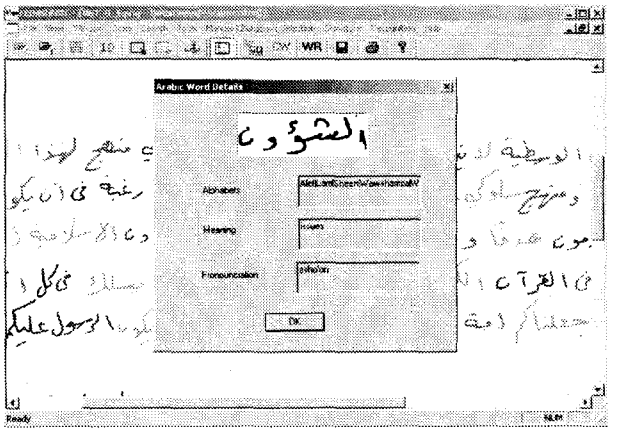


Figure 13: Word image with corresponding truth

7.2 Experiments

To test the combined method, 32 queries were issued on 13,631 records. The neural network was trained on 300 such queries, 150 positive and 150 negative query matches. All remaining records were used for testing. The combined score is a score between -1 and 1. A negative score would indicate a mismatch and a positive score a match. A 91% raw classification accuracy was observed.

Using five writers for providing prototypes and the other five for testing, using manually segmented documents, 55% precision is obtained at 50% recall

for the word shape method alone. The character based method achieves 75% precision at the same recall rate. The combined method is consistently better, resulting in about 80% precision. A comparison graph, with the word shape method using five writers, is shown in Figure 14. One search result from CEDARABIC is shown in Figure 15.

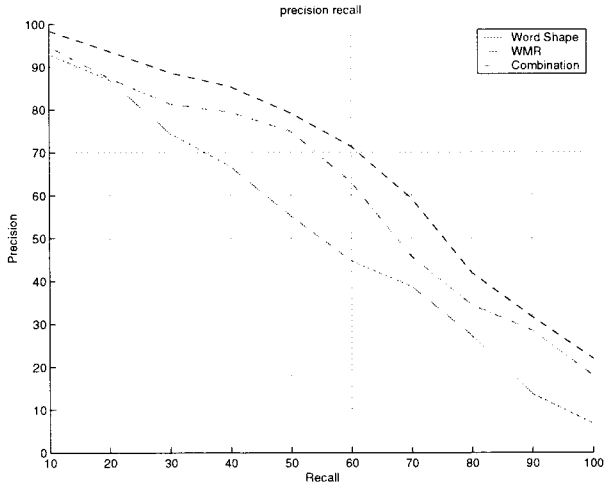


Figure 14: Precision-recall comparison of word-shape, character-shape and combination approaches.

8 Conclusion and Future Directions

Processing image and text based queries in parallel can result in higher performance than either alone. The versatile search framework presented can be applied to many document search problems. The example presented illustrates a word spotting application, but other document search strategies may experience similar performance increases. For example, a partially transcribed document could be represented as a "bag of words," with a term/document matrix. From there, latent semantic analysis TF-IDF (term frequency-inverse document frequency) weights can be used to perform traditional searches, with image search augmenting less than perfect transcription techniques. Furthermore, "plugging in" improved image or text-based search algorithms can push overall performance higher. For our experiments, we used the neural network to simply weight the two incoming scores. However, features extracted from the images may improve neural network performance.

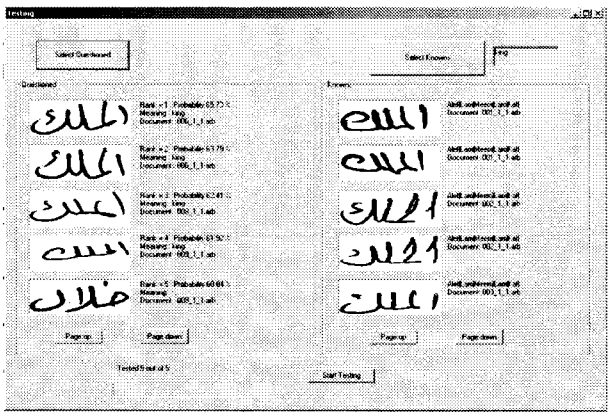


Figure 15: Word Spotting Testing Results

References

[1] D. Hand, H. Mannila, and P. Smyth. *Principles of Data Mining.* MIT Press, Cambridge, MA, 2001.

[2] C. Huang and S. N. Srihari. Mapping transcripts to handwritten text. In *Proc. Tenth International Workshop on Frontiers in Handwriting Recognition (IWFHR))*, La Boule, France, 2006. IEEE Computer Society.

[3] G. Kim. *Recognition of offline handwritten words and extension to phrase recognition.* Doctoral Dissertation, State University of New York at Buffalo, 1997.

[4] G. Kim and V. Govindaraju. A lexicon driven approach to handwritten word recognition for real-time applications. In *IEEE Transactions on Pattern Analysis and Machine Intelligence 19(4)*, pages 366–379, 1997.

[5] G. Kim, V. Govindaraju, and S. N. Srihari. A segmentation and recognition strategy for handwritten phrases. In *International Conference on Pattern Recognition, ICPR-13*, pages 510–514, 1996.

[6] G. Russell, M.P. Perrone, Yi min Chee, and Airnan Ziq. Handwritten Document Retrieval. In *Proc. Eighth International Workshop on Frontiers in Handwriting Recognition*, pages 233–238, Niagara-on-the-lake, Ontario, 2002.

[7] S. N. Srihari, C. Huang, and H. Srinivasan. A search engine for handwritten documents. In *Document Recognition and Retrieval XII: Proceedings SPIE*, pages 66–75, San Jose, CA, 2005.

[8] S. N. Srihari and Z. Shi. Forensic handwritten document retrieval system. In *Proc. Document Image Analysis for Libraries (DIAL)*, pages 188–194, Palo Alto, CA, 2004. IEEE Computer Society.

[9] S. N. Srihari, H. Srinivasan, P. Babu, and C. Bhole. Handwritten Arabic word spotting

using the CEDARABIC document analysis system. In *Proc. Symposium on Document Image Understanding Technology (SDIUT-05)*, pages 123–132, College Park, MD, Nov. 2005.

[10] S. N. Srihari, H. Srinivasan, P. Babu, and C. Bhole. Spotting words in handwritten Arabic documents. In *Document Recognition and Retrieval XIII: Proceedings SPIE*, pages 606702–1 to 606702–12, San Jose, CA, 2006.

[11] S. N. Srihari, C. I. Tomai, B. Zhang, and S. Lee. Individuality of numerals. In *Proc. Seventh International Conference on Document Analysis and Recognition(ICDAR)*, page 1096, Edinburgh, UK, 2003. IEEE Computer Society.

[12] S. N. Srihari, B. Zhang, C. Tomai, S. Lee, Z. Shi, and Y-C. Shin. A search engine for handwritten documents. In *Proc. Symposium on Document Image Understanding Technology (SDIUT-05)*, pages 67–75, Greenbelt, MD, 2003.

[13] K. Taghva, J. Borsack, and A. Condit. Results of applying probabilistic IR to OCR text. In *Research and Development in Information Retrieval*, pages 202–211, 1994.

[14] B. Zhang and S. N. Srihari. Binary vector dissimularity measures for handwriting identification. *Proceedings of the SPIE, Document Recognition and Retrieval*, pages 155–166, 2003.

Databases and Competitions
Strategies to Improve Arabic Recognition Systems

Volker Märgner **Haikal El Abed**
Technical University Braunschweig
Institute for Communications Technology
Schleinitzstrasse 22, 38106 Braunschweig,
Germany
Email: {v.maergner, elabed}@tu-bs.de

Abstract

The great success and the high recognition rates of both OCR systems and recognition systems for handwritten words are unthinkable without the availability of huge datasets of real world data. This paper gives a short survey of datasets used for recognition with special focus on their application. The main part of this paper deals with Arabic handwriting, datasets for recognition systems, and their availability. A description of different datasets and their usability is given and the results of a competition are presented. Finally, a strategy for the development of Arabic handwriting recognition systems based on datasets and competitions is presented.

1 Introduction

Machines are still far from being able to read the way humans do. Nevertheless, automatic reading of printed text has reached a very high level in many languages and applications, e.g. address reading and check reading. Powerful computers allow the execution of very efficient recognition algorithms without any special hardware. There is still a gap to fill, however for languages which are not using Latin characters. One of these languages is Arabic, which is spoken by more than 230 million people as official language in 25 countries worldwide.

We will not go into details of Arabic writing style but refer the reader who is not familiar with Arabic to [7] and [14]. The cursive style even of printed Arabic makes the segmentation into characters difficult and the extensive use of diacritics demands special methods of feature extraction and normalisation. Therefore, systems developed for Latin character based OCR can not be easily adapted to Arabic.

So far most of the successful methods for OCR and cursive script recognition are statistical methods, e.g. approaches based on Neural Nets (NN) or Hidden Markov Models (HMM). Like all methods that are based on statistical approaches they require a huge amount of data to adapt their parameters to the intended application.

Another very important aspect for recognition system development is the discussion and competition of different approaches. Only the testing of different methods on identical datasets allows for their informative comparison. Furthermore, objective quality measuring methods are necessary for ranking the systems. This also constitutes a high motivation for developing advanced methods.

This paper gives a survey of datasets, competitions, and evaluation tools necessary to improve recognition systems in general and Arabic handwritten text recognition in particular. The paper is organized as follows: In section 2 a short survey of existing datasets, competitions and evaluation tools for non Arabic text is given. This section focuses on those parts of non Arabic recognition systems that can be used to learn for Arabic text recognition. Even though many elements of recognition systems are unique and independent of the language, there are some language dependent special properties, which are discussed in this section. Section 3 describes the state of the art of existing datasets of Arabic handwritten words in some detail, followed by section 4, which presents methods and results of the first competition of Arabic handwriting recognition. Section 5 discusses several aspects important for the further development of Arabic handwriting recognition, which are summarized in section 6.

2 Datasets for Text Recognition

The first approaches for automatic character recognition were developed in the beginning of the 1930's using simple pattern matching methods on selective characters. Special, easy-to-read fonts were designed for the automatic reading on bank forms. In the 1960's mail sorting machines were able to read printed text on envelopes. A veritable leap forward was

made in the 1990's, when huge datasets were made available to the research community [1], allowing the development of statistical methods for OCR systems. Another reason for the progress in the 1990's were annual competitions of OCR accuracy, such as the workshops at the University of Nevada, e.g. [2]. Along with these developments came as a third important aspect the development of methods for measuring and comparing the quality of recognition systems (NIST, ISRI).

This work was very successful. Powerful OCR systems were realized and useful methods developed. On the other hand it became clear that the collecting and labelling of data is not only an important task but also very expensive. Since commercial companies tend to exploit their own advantage rather than accelerate development by encouraging competition, independent funding seems to be crucial.

Luckily, just as OCR development had attracted independent support, public funding was made available for the development of handwritten character and word recognition systems as well. Driven by the need of check reading and postal sorting machines real world data were made available e.g. by CEDAR [3]. NIST made available not only data for system development, but also recognition software as a benchmark for future methods. Thus it was made possible to develop and test parts of recognition systems separately, such as a character recognizer or a word-to-character segmentation method. Not having to develop entirely new systems every time, researchers were now able to focus on a certain aspect, which facilitated development significantly.

2.1 Synthetic and Artificial Data

The high costs of collecting and preparing data for building and optimizing OCR systems resulted in various cost-cutting efforts. One idea was to avoid scanning and labeling of printed text by generating synthetic data [4]. While this approach provides cheap datasets for testing and developing statistical recognition methods, it is clear that a system developed with synthetic data has to be retrained before it can be used in a real world environment.

Concerning handwriting, the development of specially designed forms for collecting artificial data made the preparation of the data for the training and testing quicker and cheaper. However, many research groups used this method to build small datasets of their own, with the outcome that the results of recognition tests performed on these diverse datasets are not comparable with the results of other research groups who used different datasets.

2.2 Competitions

Testing recognition systems with large identical datasets is crucial for the evaluation of their performance. Another challenge is their complexity, since they consist of many specialized parts solving very different tasks. While the recognition rate is a convenient measure for comparing different systems, it is a global parameter hardly significant for system component development. To improve the overall system quality it is essential to know the effectiveness of its modules.

The development of meaningful aspects of system evaluation methods was an important part of the aforementioned annual OCR tests at ISRI. The goal of these tests was not only to publicize the state-of-the-art of page reading systems, but also to provide information for improvement through competition and objective assessment. While much has been achieved concerning the evaluation problem (e.g. [15]), the availability of tools and data remains an issue for research, as pointed out in the paper [16], published 2005. For example, it is not enough to measure the quality based on the symbol output of the recognizer only by considering the word accuracy. The quality of zoning and the segmentation into words or characters is a very important feature of a recognition system, too, and should be evaluated [5]. A more general concept for evaluating modules of a system separately is presented in [18].

2.3 Requirements on Arabic Datasets

To a large degree text recognition for Arabic printed or handwritten words faces the same challenges as text recognition for Latin character based languages. Due to the fundamental differences of the characters and the writing style, however, there are some additional features to consider. In addition to the difference in writing direction, line by line from right to left, the connection of the printed characters is a great difference between Arabic and e.g. English. It is interesting to know that the Arabic set of 28 characters results in 100 different character shapes as most of the characters can appear in 4 different shapes depending on the position of the character in a word (isolated, beginning, middle, and end), but capital letters are not known. This is approximately twice the amount of the English character set, if we consider the 52 different character shapes of English, including capital letters.

The connectedness of the Arabic printed characters necessitates a more complex segmentation of a word into characters. Moreover, points and other diacritic marks are parts of the characters, which are positioned above or below the main character shape.

These characteristics of the Arabic language require a special structure of the dataset designed for training and testing Arabic recognition systems. In addition to these characteristics of printed Arabic text, even more differences can be observed in the case of Arabic handwriting, which makes the construction of a dataset and the recognition process more difficult. Figure 1

gives an example of a handwritten address written with Latin and with Arabic letters. A quite different appearance can be seen with less regularity in the text written with Arabic in comparison to the text written with Latin characters. Table 1 shows some examples of handwritten Arabic words with highlighted characteristics.

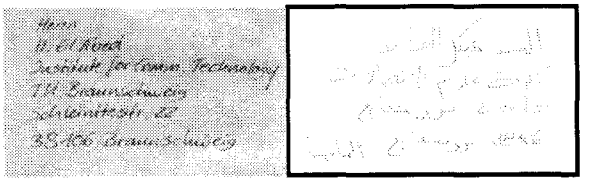
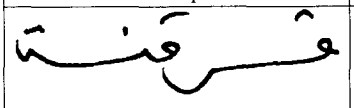
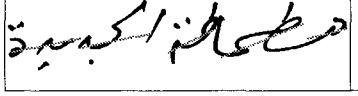


Figure 1: Handwritten address Latin and Arabic

Table 1: Examples of some typical features of Arabic handwritten words

Example	Description
	Consecutive letters within a word are typically joined together by a baseline stroke
	Characters are often stretched through lengthen the connecting line
	Ligatures replace particular character pairs or even triples
	Diacritical mark "Chadda" is used to indicate short vowels or as special form of a character
	The baseline is needed for the recognition, but sometimes difficult to find

These examples show that fundamental differences in the writing of Arabic words exist compared to English. But of course in the same way as the writing style of English differs around the world, the writing style of Arabic differs from country to country, too. A critical point is that especially the way of using ligatures differs very much.

Datasets with Arabic text or words have to consider these characteristic features. As a result, special labels have to be assigned, and the amount of differently shaped characters has to be considered in the incorporation of different words by different writers into the datasets.

3 Datasets of Arabic Text

As pointed out in section 2, datasets of typical printed or handwritten words are the most important part in the design process of a recognition system. For many years papers on Arabic printed or handwritten word recognition have been published. The first paper on Arabic OCR dates back to 1975, the first Arabic OCR system was made available in the 1990's. However, only three papers comparing OCR systems have been published so far and the newest one is already seven years old [6]. To overcome the problem of the lack of large datasets for developing Arabic OCR systems, and to motivate the research on statistical methods, a system for synthetic generation of Arabic datasets was developed according to the approach for English OCR [17]. This allows fast and simple generation of large datasets.

For Arabic handwritten word recognition the situation is not better. For many years the work published about Arabic handwriting recognition has been using small private datasets, which makes a comparison of methods more or less impossible. Another disadvantage is that very often these private datasets are too small for statistical methods to be reasonably used. Only a few datasets have been published. Recently an overview about the state of the art of Arabic offline handwriting recognition was given [7].

In the following datasets of Arabic words or text are discussed. It has to be mentioned that often databases or datasets presented in papers and used for experiments are not publicly available. The current availability of the following datasets is mentioned in each section.

3.1 ERIM Arabic Document Database

The Environmental Research Institute of Michigan (ERIM) has created a database of machine-printed Arabic documents. These images have been extracted from typewritten and typeset Arabic books and magazines and contain a wide variety of fonts and ligatures, their quality ranging from poor to good. All images were collected at 300 dpi. This database is meant to provide a training and a testing set for Arabic text recognition research. The following details are given on the ERIM website [8].

Database specifications:

- over 750 pages of Arabic text
- all data digitized on 300 dpi flatbed scanner
- grayscale page images available for some pages
- all character information stored as Unicode
- approximately 1,000,000 characters
- truth for characters as well as ligatures
- over 200 distinct Arabic ligatures
- a wide variety of fonts and data quality
- divided into Training, Statistic, and Test sets
- image format documentation included

Together with the image data, truth files are provided. All this sounds very interesting, but it was impossible for us to acquire the database. It seems that the data is no longer available.

3.2 Al-Isra Database

In a paper at the IEEE Canadian Conference on Electrical and Computer Engineering [9] a database of handwritten Arabic words, numbers, and signatures is described. The data were collected at the Al-Isra University Amman, Jordan. 500 randomly selected students contributed handwritten data to the database. The database consists of:

- 37,000 Arabic words,
- 10,000 digits (Arabian and Indian),
- 2,500 signatures, and
- 500 free-form Arabic sentences,

all saved in grayscale and black and white BMP file formats. The database was announced in 1999, but untill now no data has been published on the internet.

3.3 CENPARMI Database

At the IWFHR workshop 2000 in Amsterdam, the Centre for Pattern Recognition and Machine Intelligence (CENPARMI) in Montreal presented a database of images from 3,000 checks provided by a banking cooperation [10]. These are images of real, used checks, and the words of the legal amount have been segmented and labeled with an ASCII code sequence for each subword (PAW = part of Arabic word). Also, the courtesy amount in Indian digits has been segmented and labeled. The data are divided into training and testing sets. The database contains 29,498 subwords, 15,175 digits, and 2,499 legal and courtesy amounts.

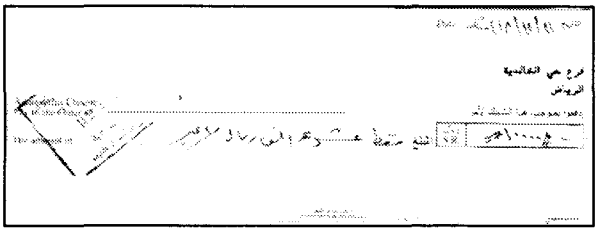


Figure 2: Example of a check from CENPARMI database.

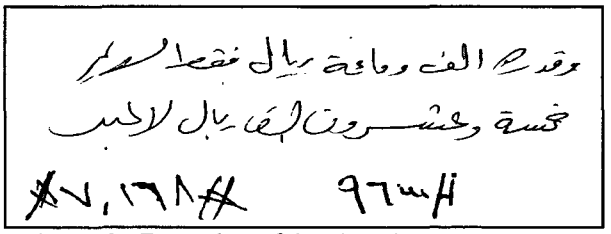


Figure 3: Examples of legal and courtesy amounts from CENPARMI database.

Figures 2 and 3 show some examples from the CENPARMI database. The advantage of this database is its availability and the fact that real world data were collected. The disadvantage is that some subwords are underrepresented so that a training of a statistical recognizer is severely restricted.

3.4 IFN/ENIT-Database

At the CIFED conference in 2002 the Institute for Communications Technology (IFN) at Technical University Braunschweig in Germany and the Ecole Nationale d'Ingénieurs de Tunis (ENIT) in Tunisia presented a database with handwritten Tunisian town names [11]. This dataset was collected on specially designed forms to make the labeling procedure as simple as possible.

The database (www.ifnenit.com) in version v1.0p2 consists of 26,459 handwritten Arabic names by 411 different writers. Written are 937 Tunisian town/village names. Each writer filled some forms with pre-selected town/village names (referred to as "names" in the following) and the corresponding post code. Ground truth was added to the image data automatically and verified manually. Figure 4 shows a dataset entry of the IFN/ENIT-database.

Image	الحمّامات
Ground truth:	
Postcode	8050
Global Word	الحمّامات
Character shape sequence	LA|LB|M|ـحـ|ـلـمـ|LE|ـمـ|LE|ـتـ
Baseline y1,y2	52,47
Baseline quality	B1 (B1=OK; B2=bad)
Quantity of words	1
Quantity of PAW's	4
Quantity of characters	8
Writing quality	W1 (W1=OK; W2=bad)

Figure 4: Example of a dataset entry

An interesting feature of this database is the very detailed labeling with the postcode as a label on word level. The character shape sequences, where each character shape depends on its position in the word, got different labels for each position. Additionally, the position of the baseline is given too as straight line. All these information, together with some additional quality measures, have been verified manually several times. The label information can be used for the training and testing of recognition systems either on word or on character level. The given baseline position can be used for the testing of baseline estimation algorithms and the dependency on the baseline accuracy.

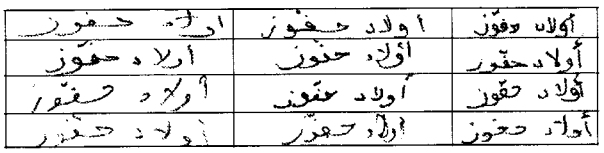


Figure 5: Examples from IFN/ENIT-database. A city name is written by 12 different writers.

Figure 5 shows some image examples from the IFN/ENIT-database. Tables 2-4 show important statistics of the IFN/ENIT-database. Table 2 shows the quantities of names, PAWs, and characters subject to the number of words in a name. It can be seen that in most cases the name consists of one word, but there are also names with four words. The overall number of words in the database is 42,510.

Table 2: Quantity of words, name images, PAWs, and characters in a name

words in a name	names	PAWs	characters
1	12,992	40,555	76,827
2	10,826	54,722	98,828
3	2,599	20,120	36,004
4	42	188	552
Total	26,459	115,585	212,211

Table 3 shows statistics of the number of PAWs in the names of the database.

Table 3: Frequency of PAWs in a name

Number of PAWs	frequency in %	Number of PAWs	frequency in %
1	2.99	6	8.24
2	15.35	7	7.32
3	17.60	8	6.04
4	24.84	>8	2.95
5	14.67		

Table 4 shows statistics of the age and the profession of the writers who contributed to the database.

Table 4: Age and profession of the writers

age	student	teacher	technician	other	Σ
≤ 20	29%	0%	0%	0%	29%
21 - 30	35.6%	4.2%	3.9%	3.9%	47.6%
31 - 40	0.2%	3.4%	4.9%	2.0%	10.5%
> 40	0%	4.1%	5.4%	3.4%	12.9%
Σ	64.8%	11.7%	14.2%	9.3%	100%

3.5 ARABASE Database

In 2005, a relational database for Arabic OCR systems was presented by N. Ben Amara et al. [12]. The authors claim to have a concept of a database to support research for On-line and Off-line Arabic handwritten and printed text recognition. All types of images of text phrases, signatures, words, characters, and digits are supposedly included in the database. Also, tools to use the database were announced. Until now only the concept has been presented, however.

4 Competitions on Arabic Handwriting

Table 5 gives an overview of the databases of Arabic printed or handwritten words we presented in this section. To our knowledge only the two bold printed databases are available for research on Arabic handwriting recognition. Even though there are some more datasets existent that are not available for public research, it is clear that there is a considerable lack of databases for developing Arabic handwriting recognition systems.

Table 5: Databases for Arabic text recognition

Name	Year	Contents	Type	Ref.
ERIM	1995	750 pages	printed	[8]
Al-Isra	1999	37,000 words 10,000 digits	handwritten	[9]
CENPARMI	**2000**	**2,499 amounts** **2,499 numbers**	**handwritten**	**[10]**
IFN/ENIT	**2002**	**26,459 names**	**handwritten**	**[11]**
ARABASE	2005	?	printed/handw.	[12]

The insights derived from the work done on OCR and handwriting recognition methods and systems for Latin character based languages in the past showed that datasets and competition are the most important prerequisites for developing recognition systems.

The IFN/ENIT-database, published 2002, is used by more than 30 research teams worldwide. This fact encouraged us to organize a competition on Arabic handwriting recognition during the ICDAR 2005 conference. Especially the groups working with the IFN/ENIT-database already were asked to submit a recognition system.

4.1 ICDAR 2005 Competition

The ICDAR 2005 competition on Arabic handwriting recognition was made on the basis of the IFN/ENIT-database. The participants build a recognizer trained with the data of the IFN/ENIT-database. The competition was then performed on an unknown, new dataset at IFN. Five groups submitted systems. The recognition results of the systems, achieved on the unknown dataset of 6,033 images, differed significantly, ranging from 15.36% up to 75.93% correctly recognized names (cf. [13]). The competition showed that existing systems achieve good results, but

the significant different performance of the participating systems also showed that still a lot of work remains to be done.

4.2 ICDAR 2007 Competition

It is planned to perform a second competition on the IFN/ENIT-dataset at ICDAR 2007. The dataset will be upgraded with the test dataset from the competition ICDAR 2005, to improve training on word level, as the words will be more equally distributed.

5 Steps to Develop Arabic Offline Recognition Systems

The state of the art of Arabic offline handwritten word recognition as presented in the sections before shows us that work remains to be done until Arabic offline handwritten word recognition systems are working as effectively as e.g. systems for the recognition of English words.

One important next step for the development of Arabic handwritten text recognition systems is the development of modules equivalent to those needed for Latin-character-based text recognition systems. The results of the first competition of Arabic handwritten word recognition systems showed us that HMM based recognizers, which are known to be very effective for cursive handwriting recognition, are also very good for Arabic handwritten word recognition. This seems to be clear as Latin character based cursive handwriting has similar features – connected characters, different shape depending on the position in a word – as Arabic handwriting. Yet, word recognizer is only one part of an offline Arabic handwriting recognition system. A complete system also needs document analysis components, as for instance image preprocessing, document segmentation, text block detection, line segmentation, word segmentation, and baseline detection. All these processing steps have to be done on real world documents, e.g. letters, bank transfer forms, insurance forms. The competition done at ICDAR 2005 was only a first step in system development, as it concerned word recognition performance only. A complete system has to take as input the paper document and perform all tasks from preprocessing to recognition automatically. Finally, an optimized Arabic text recognition system has to consider the characteristics of the Arabic language, e.g. its syntax and semantics.

5.1 Characteristics of Databases

The usefulness of databases depends on two main features: the data itself and the structure of the ground truth. The collection of data from a selected application is the first step to be done. Before the expensive and time consuming labeling process starts, a data structure for the ground truth and label information has to be defined. It should be flexible and easy to use. A hierarchical token based structure, as used in the IFN/ENIT-database, seems to be a good choice, as it is extendable and easy to use [19].

```
DEMO_001.TIF
DEMO_001.X1 POSTAL_AREAS
//MAIN POSTAL AREAS
    DEMO_001.X21 REG_NUMBER
    //POSTAL REGISTRATION NUMBERS
    DEMO_001.X22 ADDRESSES
    //ADDRESS AREAS
        DEMO_001.X31 TEXTLINES_HW
        //HANDWRITTEN TEXTLINES
            DEMO_001.X41 WORDS_HW
            // HANDWRITTEN WORDS
                DEMO_001.X51 CHARACTER_HW
                //HANDWRITTEN CHARACTERS
        DEMO_001.X32 TEXTLINES_MP
        // MACHINE PRINTED TEXTLINES
    DEMO_001.X23 BARCODE
    // BARCODES
```

Figure 6: Example of a document hierarchy file.

Figure 6 shows an example of the hierarchical structure of a document where the names of the files are given which describe the data of the associated image regions. In the IFN/ENIT-database the hierarchy consists of one level only.

The format of the region files is line-oriented with a token of three capital letters at the beginning of each line defining its contents. This format is a byte-stream, easy to read, and OS-independent. Due to the format of the region files, any additional information can be inserted by just adding an appropriate token to the list of possible tokens, without altering the format, as unknown tokens are simply ignored by the corresponding reader. Figure 7 shows an example of such a file.

```
BHE:                    /* Begin of Header */
FTY: REC                /* file type */
PIC: DEMO_001.TIF       /* name ofimage file */
NRU: 3                  /* number of regions */
DES: MAIN POSTAL AREAS/* region type */
PFN: DEMO_001.X1        /* Name of Parentfile */
EHE:                    /* End of Header */
BDA:                    /* Begin of Data */
.
BDR:                    /* Beg. of Data-Record */
OID: 3                  /* unique ID of region */
POI: 0                  /* ID of parent region */
OBB: REC 0 0 162 168    /* geom. zone-descript.*/
OBE:                    /* End of description */
LBL: RECIPIENT_ADDRESS/* Label of region */
.                       /* additional inform. */
EDR:                    /* End of Data-Record */
.
EDA:                    /* End of Data */
EOF:                    /* End of File */
```

Figure 7: Example of a region description file.

Figure 8 and 9 show examples with typical addresses on letters. The words in these examples are more differently written than in the form, designed for easy labeling, as it was the case for the collection of artificial data for the IFN/ENIT-database. Especially the vertical dimensions of the words vary significantly and the lines are partially overlapping. These features make the segmentation into words and the estimation of the baseline a difficult task.

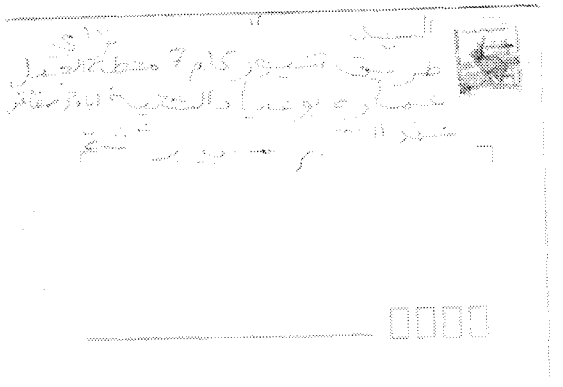


Figure 8: Example of a gray scale image of a letter.

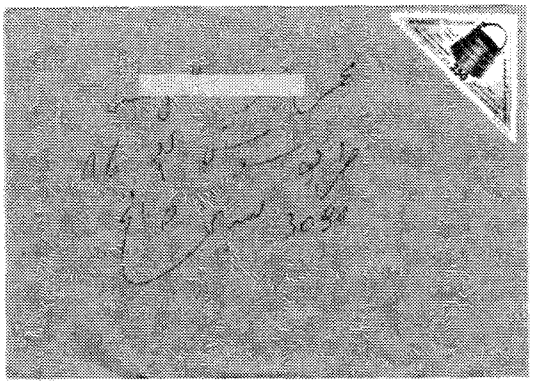


Figure 9: Another example of a gray scale image of a letter.

Therefore a huge amount of data is needed to develop appropriate methods for preprocessing and segmentation. The hierarchical database description allows also the usage of parts of the data to test modules of the whole system.

For the correct interpretation of the address additional knowledge about the different ways of writing addresses in Arabic on letters is needed, and Arabic writing style, as well as special features of the language, have to be considered, too.

Figure 10 shows another example. A form is printed with Latin characters. It is filled with words written with Latin as well as Arabic characters. Here knowledge about the structure of the form has to be available.

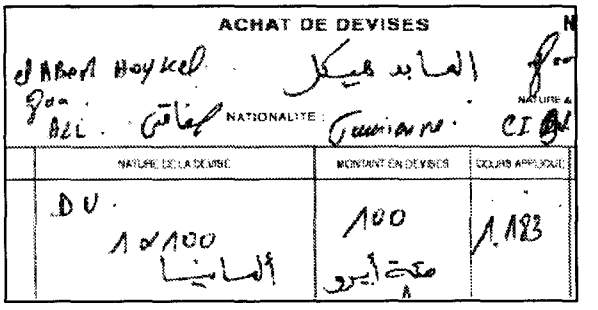


Figure 10: Example with a section of a filled form with a mixture of handwritten Latin and Arabic characters.

These examples show that using different application aspects have to be kept in mind when data for a database are collected.

5.2 Arabic Language Modules

Not only the features of Arabic handwriting have to be considered in an Arabic handwriting recognition system but also syntax and semantic of the Arabic language are needed in the recognition process. This information can be used implicitly or explicitly during the recognition process to reduce, for example the lexicon size, but also to verify the recognition result on word or sentence level. Besides using dictionary lookup, letter n-grams or even word morphology rules can be used to improve recognition results as well. The usage of language modules in recognition systems requires databases that allow the training and testing of such methods.

5.3 Competitions

Competitions organized on the basis of images of real world data are recommended. With the aforementioned hierarchical region description and flexible token based labeling of the data, a comparison of modules and systems on different levels of complexity is made possible. Address reading systems, for example, use the image of a letter as input and deliver name and address of the recipient as output. The test of a single module – e.g. text block location, word segmentation, baseline estimation, word recognition – is possible if the associated ground truth information is available in the region description.

A very important aspect of competitions is the detailed presentation and discussion of different methods, with the aim of understanding which method works better and why. The organization of a workshop to present the results and discuss the different approaches is recommended.

6 Future Work

Considering all the aspects we discussed in the previous sections the next steps which should be made to provide better Arabic handwriting recognition systems are obvious. In the following, necessary steps are listed:

- Selection of interesting fields of application for Arabic handwriting recognition.
- Collection of real world data, perform scanning and labeling of the data to construct a database.
- If real world data are not available: development of a concept to generate an artificial database by selecting people to fill forms. Scanning and labeling may be easier as the form can be specially designed.
- Organizing a workshop to discuss newest research results and present a performance evaluation of different systems or modules on the basis of a common dataset, performed by an independent group.
- Immediately making available new datasets to research teams. The format of the labels should be the same for different datasets.
- Development of performance measurements of processing and recognition modules should be considered in the workshops.
- Workshops should be repeated annually and interdisciplinary contributions encouraged.
- Exchange of methods between handwriting recognition of different languages is useful.

From all the experience on other languages these steps should help to reach the goal to develop better Arabic handwriting reading machines. In addition to this short-term objective we hope to approach as a long-term objective a better understanding of the nature of the reading process of humans in general, as this could help to reduce the amount of work to adapt a system to a new application.

7 Conclusion

In this paper we have discussed different databases and methods for evaluating recognition modules or systems. Much of the work done on text recognition systems for Latin character based text can be adapted to Arabic text recognition. For the time being, the methods used for comparing different systems can be identical to the ones developed for Latin character based OCR systems. Later, special aspects of Arabic writing should be considered. The organization of competitions with the presentation of recognition results and discussions of different approaches are the basis of a successful work on offline Arabic handwriting recognition systems.

References

[1] NIST, NIST Special Databases and Software from the Image Group, www.itl.nist.gov/iaui/vip/databases/defs/

[2] S.V. Rice, F.R. Jenkins, and T.A. Nartker, The Fifth Annual Test of OCR Accuracy, Technical Report University of Nevada, Las Vegas, 1996.

[3] www.cedar.buffalo.edu/Databases/CDROM1/

[4] T. Kanungo and R.M. Haralick, An automatic closed-loop methodology for generating character groundtruth for scanned documents, in IEEE Trans. on PAMI, 21 (2) pp 179-183, 1999.

[5] M. Thulke, A. Dengel, and V. Märgner, A General Approach to Quality Evaluation of Document Segmentation Results, in LNCS 1655, Eds: S.-W. Lee, 1999, pp. 43-57.

[6] T. Kanungo, G.A. Marton and O. Bulbul, Performance Evaluation of Two Arabic OCR Products, in *Proc. 27th AIPR Workshop, SPIE vol.3584*, 1999, pp.76–83.

[7] L.M. Lorigo and V. Govindaraju, Offline Arabic Handwriting Recognition: A Survey, IEEE Trans. on PAMI, vol. 28, no. 5, 2006, pp.712-724

[8] S. G. Sclosser, ERIM Arabic Document Database, http://documents.cfar.umd.edu/resources/databas e/ERIM_Arabic_DB.html

[9] N. Kharma, M. Ahmed and R. Ward, A new comprehensive database of handwritten Arabic words, numbers, and signatures used for OCR testing, in *Proc. of IEEE Canadian Conference on Electrical and Computer Engineering*, vol. 2, 1999, pp.766-768.

[10] Y. Al-Ohali, M. Cheriet and C. Suen, Databases for Recognition of Handwritten Arabic Cheques, Proc. of 7th *Int. Workshop on Frontiers in Handwriting Recognition*, 2000, pp. 601-606.

[11] M. Pechwitz, S. Snoussi Maddouri, V. Märgner, N. Ellouze , H. Amiri, IFN/ENIT-Database of Handwritten Arabic Words, *Proc. of CIFED'02*, 2002, pp. 129-136.

[12] N. Ben Amara, O. Mazhoud, N. Bouzrara and N. Ellouze, ARABASE: A Relational Database for Arabic OCR Systems, in The Int. Arab Journal of Information Technology, vol. 2, no. 4, 2005, pp. 259-266.

[13] V. Märgner, M. Pechwitz, and H. El Abed, ICDAR 2005 Arabic Handwriting Recognition Competition, *Proc. ICDAR*, 2005, pp. 70-74.

[14] S. Ager, Arabic alphabet, pronunciation and language, http://www.omniglot.com/writing/arabic.htm

[15] S.V. Rice, Measuring the Accuracy of Page-Reading Systems, Ph.D. dissertation, UNLV, Las Vegas, 1996

[16] T.A. Nartker, S.V. Rice, and S.E. Lumos, Software Tools and Test Data for Research and Testing of Page-Reading OCR Systems, *Proc. of IS&T/SPIE Symposium on Electronis Imaging science&Technology*, 2005

[17] V. Märgner and M. Pechwitz, Synthetic Data for Arabic OCR System Development, *Proc. of ICDAR*, 2001, pp. 1159-1163.

[18] V. Märgner, P. Karcher, and A.-K. Pawlowski, On Benchmarking of Document Analysis Systems, *Proc. of ICDAR*, 1997, pp. 331-336.

[19] R.-D. Bippus and V. Märgner, Data Structures and Tools for Document Database Generation: An Experimental System, *Proc. of ICDAR*, 1995, pp. 711-714.

[20] M. Pechwitz and V. Märgner, Baseline Estimation for Arabic Handwritten Words, *Proc. of the IWFHR*, 2002, pp. 601-606.

Paradigms in Handwriting Recognition

Venu Govindaraju

Professor, Dept of Computer Science and Engineering Associate Director, CEDAR
Director, Center for Unified Biometrics and Sensors (CUBS) University at Buffalo

From the earliest days of research, two approaches of Handwritten Word Recognition (HWR) have been identified. The first approach, often called the *analytical* approach, treats a word as a collection of simpler subunits such as characters and proceeds by segmenting the word into these units, identifying the units and building a word-level interpretation using the lexicon. The other approach treats the word as a single, indivisible entity and attempts to recognize it using features of the whole word as a whole. The latter approach is referred to as *holistic* approach, and is inspired in part by psychological studies of human reading which indicate that humans use features of word shape such as length, ascenders, and descenders in reading.

Because analytical approaches decompose the HWR into the problem of identifying a sequence of sub-units, the chief problems they face are (i) segmentation ambiguity: deciding where to segment the word image, and (ii) variability of segment shape: determining the identity of each segment. Holistic approaches circumvent these problems because they make no attempt to segment the word into sub-units. However they are faced with the problem of considering every word in the lexicon as a different class. The holistic features and the matching scheme must therefore be coarse enough to be stable across exemplars of the same word across a variety of writing styles, but fine enough to be able to distinguish exemplars of different classes.

Holistic approaches have been traditionally used in scenarios where the classes are few and fixed. For example, the check amount recognition task. Moreover, when the lexicon is small and static, it becomes possible to collect a large number of training samples of each class. Training may then be performed in the traditional sense of estimating class-conditional densities of features from the training samples or storing prototypical feature vector exemplars for each class.

When the lexicon is large or dynamic as in postal address interpretation, the ability of any given set of holistic features to distinguish between word classes is diminished. In addition, it is difficult or impossible from a practical standpoint to obtain representative samples of all word classes for training a holistic classifier. For these reasons, there is consensus among HWR researchers that the utility of the holistic approaches is either in the small static lexicon scenario or for filtering large lexicons.

By circumventing segmentation issues and treating each word as a class unto itself, holistic approaches have the potential to model effects that are unique to a class. For example, they can model co-articulation effects, i.e., the changes in the appearance of a character as a function of the shapes of neighboring characters. Further, holistic features

provide information about the word that is orthogonal to the knowledge of characters in it, and it stands to reason that the introduction of this knowledge should improve recognition. For example, a holistic approach may succeed where the writing is so poor that the individual characters cannot be distinguished but the overall shape of the word is preserved.

The central idea in analytical approaches is to identify parts of the word image as one of pre-defined set of models known to the classifier. A "circular" situation arises: in order for a piece of the word image to be identified as a model, it must first be segmented; but in order for it to be correctly segmented it must be identified as a valid model first. Different model-based methods place different emphasis on recognition and image dissection to arrive at a final segmentation such that each identified segment corresponds to one of the models. At one end of the spectrum are methods which perform an image-feature based dissection and use recognition of segments to detect and correct segmentation errors. At the other end of the spectrum are methods that use a variety of holistic features, representations, and matching methodologies.

Thus, analytical approaches transform the problem of modeling the variability in the signal at the word level to modeling it at the level of sub-models. The choice of sub-models is critical. It is difficulty to capture all of the variability of handwritten words in terms of 26 sub-models, especially when the shape of a character is a function of its neighbors. Clearly if the number of classes were finite and large amounts of training data were available, building a separate model for each word directly from the features would yield the best classification, since it would not be constrained by sub-models. In practice, these conditions are met only when the lexicon is small and static.

There are two issues that must be emphasized in this context. First, classification is only part of the problem. Given the difficulty of the task, practical recognition engines must employ multiple classification algorithms and complex strategies for combining classifiers. Second, the merit of a particular paradigm is best judged by its cost/accuracy benefits, rather than accuracy alone. An algorithm that is highly accurate at classifying words is not viable in practice if the computational cost involved is unreasonable. Conversely, an algorithm, such as the holistic recognizer with relatively low accuracy may prove beneficial if used in conjunction with more accurate algorithms, and if the additional computational burden is relatively small.

We will describe in detail two analytical methods (word model based recognition and character model based recognition) and one holistic method. In the word model based recognition, all lexicon entries are treated as word models and matched against the input. The entry with the best match is the top choice. In character model based recognition, segments are matched against individual characters without using any contextual information implied by the lexicon. Word hypothesis are generated by the character recognition results. If the best hypothesis is found in the lexicon, the recognition is done; otherwise the second best hypothesis is generated and tested, and so on. Therefore, the lexicon plays an active role in the first strategy but a passive role in the second.

The different challenges of handwriting recognition in various document analysis systems such as postal, medical forms, and bank check applications will be explored and suitable paradigms discussed.
We will draw from the following key fifteen papers published in our lab over the last ten years. These papers roughly cover the entire spectrum of handwriting recognition paradigms and roughly capture the evolution of handwriting recognition technology.

.

1. G. Kim and V. Govindaraju, "*A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications*", IEEE TPAMI 19(4):366-379, 1997.

A fast method of handwritten word recognition suitable for real-time applications is described. Preprocessing, segmentation, and feature extraction are implemented using a chaincode representation of the word contour. Dynamic matching between characters in a lexicon entry and the segments of the word image is used to rank the lexicon.

2. G. Kim and V. Govindaraju, "*Bank Check Recognition Using Cross Validation between Legal and Courtesy Amounts*", IJPRAI 11(4):657-674, 1997.

A bank check reading system using cross validation of both the legal and courtesy amounts is presented. The list of possible amounts generated by the word segmentation hypothesis (of the legal amount) is used as lexicon for the courtesy amount recognition.

3. G. Kim and V. Govindaraju, "*Handwritten Phrase Recognition as Applied to Street Name Images*", PR 31(1):41-51, 1998.

This paper describes a method for recognition of street name phrases. A neural network is designed to segment words in a phrase using distances between components and style of writing. The network learns the type of spacing that one should expect between different pairs of characters.

4. G. Kim, V. Govindaraju, and S. Srihari, "*Architecture for Handwritten Text Recognition Systems*", IJDAR 31(1):41-51, 1998.

This paper describes the stages in recognizing complete pages of handwriting. Text line detection, text line extraction, noise removal, and other image analysis tasks constitute the first stage. The second stage is about segmenting the words from the text line and recognizing them. Postprocessing using linguistic constraints is the final stage.

5. S. Madhvanath, G. Kim, and V. Govindaraju, "*Chaincode Processing For Handwritten Word Recognition*", IEEE TPAMI 21(9):928-932, 1999.

Contour representations are an efficient way of coding handwritten binary word images. The paper addresses the challenges of dealing with the one-dimensional chaincode structure.

6. S. Madhvanath, E. Kleinberg, and V. Govindaraju, "*Holistic Verifcation of Handwritten Phrases*", IEEE TPAMI 21(12):1344-1356, 1999.

This paper describes a system for rapid verification of handwritten street name phrases using perceptual holistic features.

7. J. Park, Venu Govindaraju, S. N. Srihari, "*OCR in a Hierarchical Feature Space*", IEEE TPAMI. 22(4): 400-407, 2000.

This paper describes a recursive character recognition methodology that uses features at different resolutions, from coarse to fine-grained. The degree of resolution necessary to classify a character is determined adaptively.

8. S. Madhvanath and V. Govindaraju, "*The Role of Holistic Paradigms in Handwritten Word Recognition*", IEEE TPAMI 23(2):149-164, 2001.

Theories of reading provide evidence for the existence of a parallel holistic reading process in both developing and skilled readers. Approaches to recognition are characterized as forming a continuous spectrum based upon the visual complexity of the unit of recognition employed.

9. S. Madhvanath, K. Sundar and V. Govindaraju, "*Syntactic Methodology of Pruning Large Lexicons in Cursive Script Recognition*", PR 34(1):37-46, 2001.

This paper presents a holistic technique for pruning of large lexicons for recognition of offline cursive script words. The technique involves extraction of downward pen strokes and elastic matching.

10. V. Govindaraju, P. Slavik and H. Xue, "*Use of Lexicon Density in Evaluating Word Recognizers*", IEEE TPAMI 24(6):789-800, 2002.

This paper develops the notion of lexicon density as an alternative metric to measure the expected accuracy of handwritten word recognizers instead of the commonly used lexicon size. Lexicon density is dependent on the word recognizer and is useful in applications such as address interpretation.

11. J. Park and V. Govindaraju, "*Use of Adaptive Segmentation in Handwritten Phrase Recognition*", PR 35(1): 245-252, 2002.

A method of estimating the number of characters in a handwritten word without actually recognizing the characters is presented. Word segments and subsets of lexicons are generated as hypotheses and verified by a lexicon driven word recognizer.

12. S. Setlur, A. Lawson, V. Govindaraju, and S. Srihari, "*Large Scale Address Recognition Systems*", IJDAR 4(3):154-169, 2002.

This paper describes the issues involved in the design of a system for evaluating improvements in the performance of a real-time address recognition system being used by the United States Postal Service for processing mail-piece images.

13. H. Xue and V. Govindaraju, "*On the Dependence of Handwritten Word Recognizers on Lexicons*", IEEE TPAMI 24(12):1553-1564, 2002.

This paper presents a model that statistically discovers the relation between a word recognizer and the lexicon. It uses model parameters that capture a recognizer's ability of distinguishing characters and its sensitivity to lexicon size.

14. H. Xue and V. Govindaraju, "*Stochastic Models Combining Discrete Symbols and Continuous Attributes in Handwriting Recognition*", IEEE TPAMI 2005.

This paper describes the application of stochastic finite state automata to the recognition of handwritten words. A framework for modeling the discrete features and their continuous attributes is described.

15. R. Milewski and V. Govindaraju, "*Automatic Recognition of Handwritten Medical Forms for Search Engines*", IEEE TPAMI 2006 – under review.

This paper describes an integration of handwriting recognition, natural language processing, contextual knowledge representation, and information retrieval to read handwritten text in medical forms. A bootstrapping strategy is used whereby a few characters in words are recognized (without a lexicon) to categorize a form. The smaller lexicon corresponding to the category is then used to enable lexicon driven word recognition.

Multi-lingual Offline Handwriting Recognition using Hidden Markov Models

Prem Natarajan, Shirin Saleem, Rohit Prasad, Ehry MacRostie, and Krishna Subramanian

BBN Technologies
pnataraj@bbn.com

Abstract

This paper introduces a script-independent methodology for multi-lingual offline handwriting recognition (OHR) based on the use of hidden Markov models (HMM). The OHR methodology is an extension of our script-independent approach for OCR of machine-printed text images. The feature extraction, training, and recognition components of the system are all designed to be script independent. The HMM training and recognition components are based on our Byblos continuous speech recognition system. The HMM parameters are estimated automatically from the training data, without the need for laborious hand-written rules. The system does not require pre-segmentation of the data, neither at the word level nor at the character level. Thus, the system is able to handle languages with cursive handwritten scripts in a straightforward manner.

The script independence of the system is demonstrated in two languages with different types of script: English and Chinese. Results from an initial set of experiments are presented in order to demonstrate the viability of the proposed methodology.

Keywords: Offline handwriting recognition, optical character recognition, hidden Markov models, segmentation-free recognition, script independence, language independence.

1 Introduction

Most offline handwriting recognition (OHR) systems are designed for a particular script or language. In this paper, we introduce an approach to OHR that, in principle and by design, is script-independent and can be used for the vast majority of the world's languages. In particular, the core feature extraction, training, and recognition components remain the same for all languages; only the data-specific components, such as the dictionary and the language model, depend on the specific language. Except for the pre-processing and feature extraction components, which are specific to OCR and OHR, the training and recognition components are taken without significant modification from our continuous speech recognition (CSR) system, which is called the BBN Byblos CSR system. Hence, we call our OHR system, the BBN Byblos OHR system.

The basic modeling paradigm we employ is that of hidden Markov models (HMM) [1]. HMMs are capable of modeling the variability of a feature vector as a function of one independent variable. In speech [2], there is one natural independent variable: time. In OHR and OCR, there are two independent variables since text images are two-dimensional (2 D), so 1 D HMMs cannot be used directly. We structure the OHR problem as a combination of two 1-D pattern recognition tasks. The first task, also called line finding, is to locate the individual lines of text on a page, and the second task is to recognize the text content of each line.

Even the OHR problem at the level of a single line is in truth 2 D as well; however, we make it into a 1 D problem by extracting a feature vector that is a function of only one dimension (usually horizontal position). The feature vector is extracted from narrow vertical strips along each line of text [32]. The fact that the feature vector we extract does not depend on the script being recognized is one reason that our approach is script-independent. The other reason is that the HMM modeling approach itself does not change with the script being recognized. In particular, the fact that there is no separate character segmentation component, neither in training nor in recognition, allows the same system to recognize scripts where the characters are separate or connected. To demonstrate the script-independence of our approach, we present OHR results in two languages with different scripts: English and Chinese. English presents the challenge of dealing with a cursive script, and Chinese presents the challenge of dealing with a large number of characters.

There have been a number of research efforts that use HMMs in off-line printed and handwriting recognition [3-31]. In all these efforts, the recognition of only a single language or script is attempted. From a methodological perspective, our approach differs from other approaches principally in the emphasis we place

on script independence.

In Section 2 we present the theoretical framework of the HMM paradigm. Section 3 gives a description of the BBN Byblos OHR system. Sections 2 and 3 closely follow the related Sections in [32] and are summarized here for the convenience of the reader. In Section 4 we present experimental recognition results on English and Chinese, and we conclude with a summary in Section 5.

2 Theoretical Framework

2.3 Problem Formulation

We represent a line of text from a scanned image by a sequence of feature vectors X. The aim is to find the sequence of characters that maximizes P(C|X), the probability of a sequence of characters C given the feature vector sequence X. Using Bayes' rule, P(C|X) may be written as

$$P(C|X) = P(X|C)\,P(C)/P(X). \quad (1)$$

We call P(X|C) the feature model and P(C) the language model (or grammar). P(X|C) is a model of the feature vector sequence X, given a sequence of characters C, and is approximated as the product of the component probabilities, P(Xi|ci), where Xi is the sequence of feature vectors that corresponds to character ci. The feature model for each character is given by a specific HMM.

P(C), the language model, is the prior probability of a sequence of characters C; it basically provides a soft constraint on allowable character sequences. The language model used in the Byblos OCR system is an n-gram Markov model which computes P(C) by multiplying the probabilities of consecutive groups of n characters or words.

P(X) in (1) is the a priori probability of the data and does not depend on C; therefore, we can maximize P(C|X) by maximizing the product P(X|C) P(C).

2.2 Hidden Markov Models

A hidden Markov model (HMM) [1] is essentially a Markov chain with one significant difference: in a Markov chain a state is associated with a unique, deterministic output value whereas in an HMM each state is associated with a probability distribution over all possible output values. (We use the word "output" because Markov models are generally thought of as generative models that produce the observed data as output.) Figure 1 shows a simple 4-state HMM with transitions and their probabilities, and the output probability distribution associated with each of the four states. These probability distributions are defined over the feature vector x, which is a high-dimensional vector. The model shown in Figure 1 is known as a left-to-right model because there is a flow from left to right as one traverses the model in producing the output sequence.

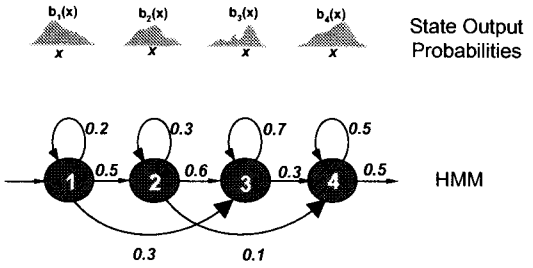


Figure 1: An example of a 4-state, left-to-right, hidden Markov model (HMM) of the type used in the Byblos OCR system. It includes one-state skips and self loops.

Given a sequence of feature vectors that was extracted from a line of text, the OHR or OCR problem is to find the sequence of states or characters that "generated" the observed sequence of feature vectors. However, because of the probabilistic nature of the output that is generated by a state, almost any sequence of states could, in principle, generate the observed output. Because it is not possible to uniquely map a sequence of feature vectors to a sequence of states/characters, the sequence of states that actually generated the vectors is hidden from the observer – hence the term hidden Markov model. Nevertheless, we can compute the probability that the observed sequence of feature vectors could have been generated by a particular sequence of states. Of particular interest is the sequence of states that has the highest probability of having generated the observed feature-vector sequence. By using the Markov property of the HMM, it is possible to find that optimal state sequence very efficiently using the Viterbi algorithm [32] or other search algorithms [33, 34]. The resulting sequence of characters is taken as the output of the recognition component.

3 Byblos OHR System

The Byblos OHR system is based upon the Byblos OCR system. Therefore, in the following, we first provide a brief review of the Byblos OCR system. This review is followed by a discussion of some features that are unique to the OHR task. For a more detailed description of the OCR system the reader is referred to [32].

3.1 Review of Byblos OCR System

The Byblos OCR system is a statistical, HMM-based recognition system that uses the Byblos HMM engine [32, 36-39] which was originally developed for speech recognition at BBN. At present, the OCR system handles single columns of text containing one or more

paragraphs. A separate page layout analysis module segments each input image into single column text zones. Figure 2 shows a block diagram of the system. As indicated in the diagram, the OCR system can be sub-divided into two basic functional components: training and recognition. Both training and recognition have the same pre-processing and feature extraction stages

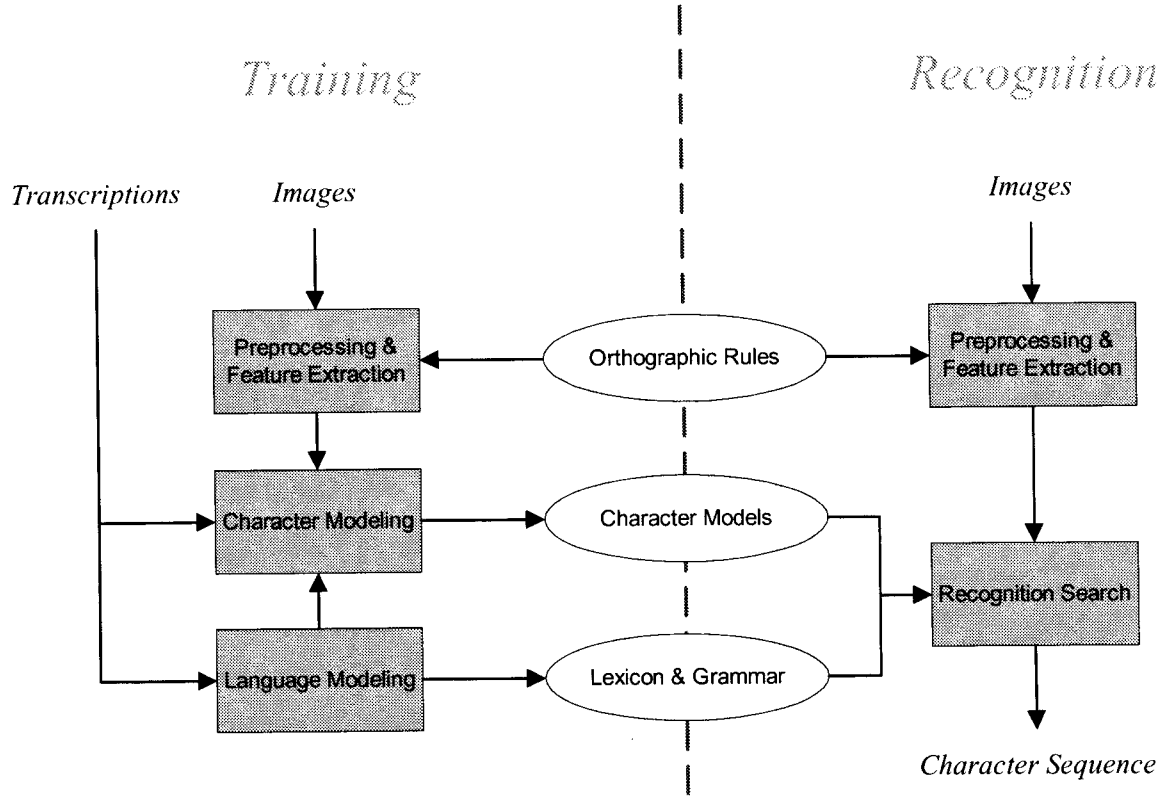


Figure 2: Block diagram of BBN OCR system.

3.1.1 Pre-processing

The pre-processing stage has two functions: skew removal and line finding. We assume that the image has been skewed owing to a rotation of the text during scanning. The details of our de-skewing algorithm are presented in [32]. After de-skewing, the image is segmented into lines of text using either an HMM-based or a connected-component based algorithm. The choice of the particular line finding algorithm is made based upon the characteristics of the data. Details of the HMM line finding procedure are presented in [38]. The end result of the line finding program is that each line of text is bounded at the top by one baseline and at the bottom by another.

3.1.2 Feature Extraction

For each line of text (as determined by the line finding process), features are computed from a sequence of overlapped windows. For each window, also called a frame, several features are computed. The feature extraction program typically computes a total of 81 features per frame. We use Linear Discriminant Analysis (LDA) [35] to reduce the number of features per frame from 81 to, typically, 15. The decision to use 15 LDA features was made empirically after running a set of experiments and choosing the number of features that resulted in the minimum character error rate. The resulting vector of 15 LDA features is a compact numerical representation of the data in the frame, and is the feature vector used in our recognition experiments. For a detailed description of the feature extraction procedure please refer to [32].

3.1.3 Training

The OCR system models each character with a multi-state, left-to-right HMM; the model for a word is the concatenation of the models for the characters in the word. Each state has an associated output probability distribution over the features, as shown in Figure 2. Each output probability distribution is modeled as a weighted sum of Gaussians, or what is called a Gaussian mixture. A Gaussian mixture is completely parameterized by the means and variances of the

component Gaussians, along with the weight of each Gaussian in the mixture. The number of states and the allowable transitions are system parameters that can be set. For our experiments we have used 14-state, left-to-right HMMs.

Training – the process of estimating the parameters (transition probabilities and feature probability distributions) of each of the character HMMs – is performed using what has been known alternately as the Baum-Welch [36], forward-backward, or expectation-maximization (EM) algorithm [37, 38], which iteratively aligns the feature vectors with the character models to obtain maximum likelihood estimates of HMM parameters. The algorithm is guaranteed to converge to a local maximum of the likelihood function. The feature probability distributions in our system are characterized by the means, variances, and weights of the Gaussian mixtures.

Depending on the amount of available training data, it may not be possible to get robust estimates of all the HMM parameters for all the characters. That is one reason why we use LDA to reduce the size of our feature vector. Another method that is used to reduce the total number of parameters in the system is to share some of the Gaussians across different character models. In particular, three such methods are available in our Byblos system: the Tied Mixture (TM) mode, the Character Tied Mixture (CTM) mode, and the State Tied Mixture Mode (STM) [44-46]. In the TM mode we train only one set of Gaussians (referred to as a codebook of Gaussians or just a codebook) that is shared between all states of all character models. The individuality of each state output probability distribution is characterized solely by the specific component mixture weights. In the CTM mode we train one codebook of Gaussians for each character model; the Gaussians in a character codebook are thus shared among the states of the model for that character but there is no sharing across characters. The CTM mode offers a greater number of free parameters and with it the possibility of better performance, subject to the availability of sufficient data for training all the parameters. For purposes of clarity, the STM mode is explained in Section 4.1.2 (English Experimental Results).

The training process is performed as follows. We assume that, for each line of text, we are given the corresponding ground truth, which is simply the sequence of characters on that line. Note that no information is given about the location of each character on the line; that is, no pre-segmentation is necessary. The training algorithm automatically and iteratively aligns the sequence of feature vectors along the line of text with the sequence of character models. This is why this method does not care whether the characters are connected or not and why it can handle languages with connected script as well as other scripts where the characters are not connected. It is also why touching characters due to fax or other degradations do not require any special handling with this method.

3.1.4 Recognition

After pre-processing a line of text and performing feature extraction, as described above, the recognition process consists in a search for the sequence of character models that has the highest probability of having generated the observed sequence of feature vectors, given the trained character models, a possible word lexicon, and a statistical language model of the possible character or word sequences. The recognition search is a two-pass [34, 35] (a forward pass and a backward pass) beam search for the most likely sequence of characters. The width of the search beam is a system parameter that can be set. Typically, lowering the beam width increases the speed but degrades the accuracy of recognition. The forward pass is an approximate but efficient procedure for generating a small list of character sequences that are possible candidates for being the most likely sequence. The backward pass is a more detailed search for the most likely character sequence within this small list. Even though we typically use the same set of character HMMs in the forward and backward passes, the search program itself does not impose any such constraint, i.e., if needed we can use two different sets of character HMMs, one in the forward and the other in the backward pass.

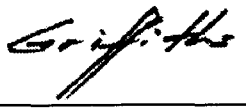
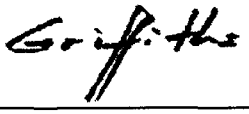
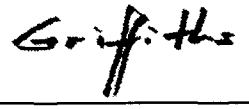
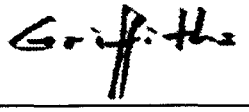
The use of a word lexicon during recognition is optional, and its use generally results in a lower error rate. The lexicon is estimated from a suitably large text corpus. The language model, which provides the probability of any character or word sequence, is also estimated from the same corpus. Note that the text corpus here does not require the presence of corresponding images; only the sequence of words in the text is needed.

3.2 Slant Correction for OHR

Handwritten text exhibits many differences from machine-printed text, all of which make the OHR task much harder than the recognition of machine-printed text. Critical differences include slanted writing, relative differences in the sizes of various characters, non-linear baseline even within a single word, and trailing strokes that hang above or below neighboring characters. These differences need to be either normalized, or modeled effectively in order to achieve reasonably high recognition accuracy.

The OHR effort reported in this paper was initiated recently and the system is currently under development. Recognizing the importance of slant normalization in handwriting recognition, our first modification of the OCR pre-processing component has been to incorporate a slant correction algorithm that works on each

Table 1: Slant correction example.

Original Image	
Iteration 1	
Iteration 2	
Iteration 3	
Iteration 4	

connected component within a page of text. Typically a connected component is a word or a fraction of a word. Other enhancements to the pre-processing (various types of normalization) and feature extraction (new features for handwriting) procedures will be designed and implemented as the work progresses. We conclude this section with a brief description of our slant correction algorithm.

The goal of the slant correction step is to eliminate any slant in each word and to make the vertical strokes perpendicular to the baseline. To normalize the slant, we estimate and then apply a non-linear, 2-D transform to each connected component (CC) within an input (black-white) text image.

The estimation of the non-linear transform is based upon the approach presented in [47]. We apply the slant correction procedure iteratively until the estimated slant is below a certain threshold. A section of the original image and the slant corrected images for each of four successive iterations of the correction procedure are shown in Table 1.

Table 2: (a) Section of original image, (b) Image after slant correction.

(a)	
(b)	

While repeated application of the transform progressively reduces the slant of text, a closer inspection of the image indicates that it the perimeter of the text progressively more jagged. A section of the original image and slant corrected image after four applications of the non-linear transform are shown in Table 2. An area of future work is to improve the slant correction procedure to reduce the manifestation of such jagged perimeters.

4 Experimental Results

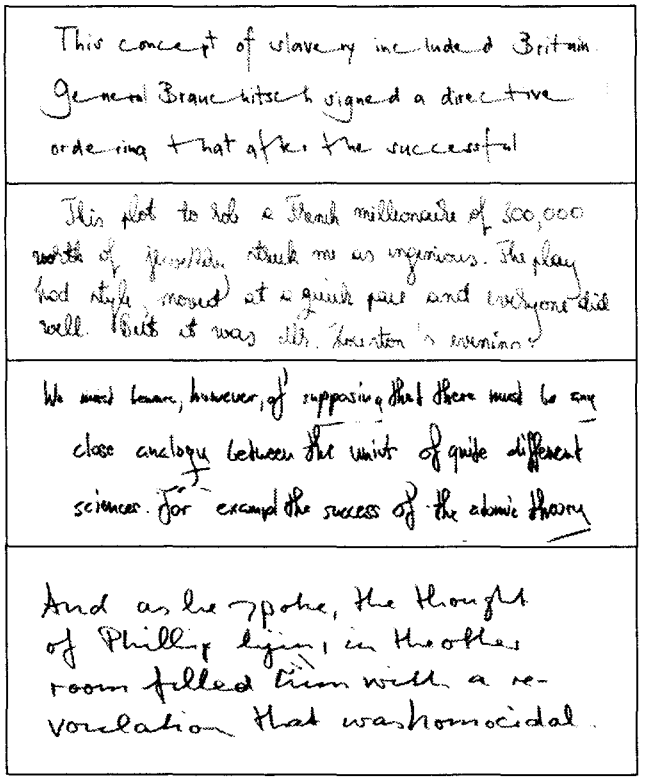
In this section we present the results of some early exploratory experiments in which we have applied our Byblos Offline Handwriting Recognition (OHR) system to handwritten text in English and in Chinese. The goal of these experiments is to establish the viability of the proposed approach for recognizing handwritten text. System performance is measured using the word error rate (WER) or the character error rate (CER), defined as:

$$\text{WER (CER)} = \frac{\text{deletions} + \text{insertions} + \text{substitutions}}{\text{total number of words (characters) in reference}}$$

where the reference is the manually generated ground truth.

In the following, we present, in separate sub-sections, the data sets, experimental procedure, and results for English and Chinese OHR.

Table 3: Sample images from the IAM database.



4.1 English OHR

4.1.1 Corpus

For our English OHR experiments, we used the IAM database [48]. The IAM English database consists of unconstrained handwritten English sentences from the LOB corpus [49]. The database was collected by distributing forms with printed text to writers, and having them write the text on the forms in their own handwriting. A total of 1539 images from 657 different writers, scanned at a resolution of 300 dpi were used in our experiments. For our experiments, we split the corpus into three sets: a training set, a development set, and a held-out test set. In dividing up the corpus into the three sets, we ensured that no writer appears both in test and in training (i.e., a writer-independent test condition). Further, in order to ensure that the language models are fair, all handwritten samples of a form are either assigned entirely to a single set (training, dev, or test). In other words, a particular passage of text is either in training or in (dev) test but never in both. Table 3 contains samples from the IAM database. The samples illustrate the tremendous diversity in writing styles contained within the IAM database.

Table 4 lists the characteristics of the training, dev, and test sets that we used in our experiments. The IAM database includes annotations of the bounding boxes for each word. For the experiments reported in this paper, we used the word bounding box information to determine the top and bottom of the lines of text within each image.

4.1.2 English Results

We ran a series of six training and recognition experiments on the IAM database in order to exercise different capabilities within our OHR system and to characterize their impact on the error rate.

The first experiment that we ran used only the machine-printed versions of the forms. We trained a set of individual English character HMMs using the machine-printed training data. Training was performed using 14-state context-independent HMMs. For any given character, the characters to its left and right define its context. For example, in the word "cat", character "a" is said to be in the context of c and t, whereas in the word "halibut", the same character "a" is in the context of an "h" and an "l". In machine-printed text the shape of a character is typically unaffected by its context, and it is, therefore, appropriate to model each character with a single, context-independent HMM. In a context-independent configuration, a single HMM is shared across all contexts of a character. We used a CTM configuration with a separate codebook of 256 Gaussians for each character HMM.

Recognition was performed using the trained character HMMs and a character tri-gram language model (LM).

Table 4: Characteristics of training, dev, and test sets for English experiments.

Characteristic	Train	Development	Test
Number of Images	1239	150	150
Number of Lines	10726	1316	1311
Total Number of Words	95512	11632	11308
Unique Number of Words	10217	3135	3093
Number of Writers	506	108	97

We did not use a word lexicon during recognition or during post-processing. We obtained a word error rate (WER) of 0.9% on the machine-printed test set. The fact that all the forms in the IAM database were rendered in a single font contributed to the low WER.

In our second experiment we trained a set of context-independent character HMMs using the IAM handwritten training data set. The HMM and language model configuration was kept the same as in our first experiment. The trained models were tested against the handwritten test data and yielded a WER of 68.50%, with an associated CER of 40.10%. We then performed a third experiment in which we replaced the character tri-gram language model with a word tri-gram language model. With a word LM, the WER dropped to 52.80% and the CER dropped to 33.80%.

Unlike machine-printed data, handwritten text exhibits a certain amount of context dependency. In other words, the shape of a particular character glyph can vary based on the character that precedes it and the character that follows it. Therefore, we ran a fourth experiment in which we trained context-dependent character HMMs. In the context-dependent configuration a separate HMM is used for each context. Using context-dependent HMMs and a word LM, we obtained a WER of 49.30% and a CER of 31.00%.

In our fifth experiment, we used a separate set of Gaussians for each state of all the context-dependent HMMs associated with a particular character. For example, we estimated a set of 128 Gaussians for the first state of all HMMs associated with the character "a", and a separate set of 128 Gaussians for the second state of all HMMs for "a", and so forth. This configuration, referred to as the state tied mixture (STM) configuration, provides a better model for the structural evolution of a character glyph in the direction of writing. With the STM model, we obtained a WER of 46.1% and a CER of 28.1%.

After exercising the various modeling capabilities of our system, we ran a final experiment in which we repeated the fifth experiment using slant corrected versions of the training and test images instead of the

raw versions from the database. Testing on the slant corrected test images using context-dependent HMMs in a 128 Gaussian STM configuration, we obtained a WER of 40.1% and a CER of 23.3%. The results of the six experiments are summarized in Table 5.

We used 10K word vocabulary derived from the IAM training transcriptions alone. The out-of-vocabulary (OOV) rate on the test set for the 10K IAM vocabulary was 10.01%. We also ran an experiment using a 30K word vocabulary that included words from our Broadcast News speech corpus. The OOV rate using the 30K vocabulary was 4.49%, but the WER and % correct did not change significantly. The only other work on this database using a HMM-based approach with statistical language models is [28]. A direct comparison of our results with that in [28] is not possible because of differences in the training and test sets. Nevertheless, using the results reported in [28] as a generic benchmark, our English OHR performance is clearly state-of-the-art.

Table 5: Summary of English recognition results.

Data Type	HMM Configuration	Language Model	CER - %	WER - %
Machine print	Context-independent, CTM	Char 3-gram	0.2	0.9
Handwriting	Context-independent, CTM	Char 3-gram	40.1	68.5
Handwriting	Context-independent, CTM	Word 3-gram	33.8	52.8
Handwriting	Context-dependent, CTM	Word 3-gram	31.0	49.3
Handwriting	Context-dependent, STM	Word 3-gram	28.1	46.1
Handwriting	Context-dependent, STM Slant corrected images	Word 3-gram	23.3	40.1

4.2 Chinese OHR

4.2.1 Chinese Handwriting Databases

For Chinese we used two different corpora for our experiments: the ETL9B corpus and the BBN Chinese Handwriting (BBN-CH) database.

The ETL9B database is published by the Japanese Technical Committee for Optical Character Recognition. The database was first released in 1993 and many researchers have reported recognition performance on that database. ETL9B contains 200 instances each of 71 Hiragana and 2,965 Kanji characters for a total of 3,036 unique characters. Following the split in [29], we used the first 20 instances of each character as the development set, the last 20 instances of each character as our test set, and the remaining 160 instances as our training set.

The second Chinese corpus that we used is the BBN Chinese Handwriting (BBN-CH) database that was collected at BBN by distributing randomly selected pages from a machine-printed corpus consisting of scanned pages from books and newspapers to 6 different authors. The authors were instructed to replicate the printed pages in their own handwriting. Each author on average contributed about 17 pages of handwritten data. The data was scanned at a resolution of 300 dpi, and manually transcribed using the source printed text as the reference.

Data from 5 authors was set aside for training. Since the amount of handwritten training data available is limited, the models were bootstrapped with data from the machine printed corpus.

The test set consists of a few pages from the 5 authors in training, as well as data from the remaining author who was not used for training. In the experimental results section we report the results for writer-dependent and writer-independent test conditions. Table 5 summarizes the characteristics of the database used for the experiments.

4.2.2 Results on ETL Database

In order to assess the viability of our OHR approach for Chinese handwriting recognition, we ran one experiment using images from the ETL9B database. The ETL9B database was collected by distributing sheets with clearly marked squares within which the writers were instructed to write a single character. Each character is scanned into a 64x63 black-white bitmap image. The images are all clean (we could not find any extraneous noise in the character images) and a visual review indicates that writers were careful in maintaining a high degree of legibility. A review of the relevant literature [29-31] indicates that a common

practice is to assume prior knowledge of the fact that each image contains only a single character. Many authors [29-31] have reported on the ELT9B database and the reported recognition rates for a single system are greater than 96%. These recognition rates, which are unusually high in the context of an unconstrained handwriting recognition task, seem to stem from the facts that the characters in the database are carefully written and clearly segmented, the images are clean, and many reported techniques leverage characteristics specific to the database.

We configured our Byblos OHR system with a 14-state HMM for each of the 3,036 characters. No slant correction was applied and our script-independent features were computed directly from the raw images in the corpus. Due to the nature of the corpus, a unigram language model with a uniform probability distribution was used. We used 160 instances of each of the characters for training and tested on 20 instances of each character. Our first experiment with no modification to the machine-printed training and recognition procedure yielded a recognition accuracy of 83%. While our recognition accuracy on this dataset is significantly lower than that reported in the literature, our initial goal was simply to establish the viability of our approach. Given the unique nature of the ETL9B database, we are unsure whether improvements obtained on this database would carry over to a more general, or generic, unconstrained handwriting situation. As a result, having established viability, we did not perform any additional experiments or optimizations on this database.

4.2.3 Results on BBN-CH Database

We ran two training experiments using the BBN-CH database. In the first experiment we trained models using the 77-page 5 writer handwritten data set, and in the second we trained models using the 5 writer handwritten data set along with the 352-page machine-printed training data set. In the second experiment, we trained separate HMMs for machine-printed and handwritten versions of each character.

Table 6: Characteristics of the training and test sets for the BBN CH Database.

Characteristic	Train-HW	Train-MP	Test-HW
Number of images	77	352	(8)+20
Number of lines	1106	11399	(101)+311
Total Number of characters	24220	348147	(2134)+6966
Number of unique characters	1641	4457	(601)+977
Number of writers	5	-	(5)+1

The test set for all experiments consists of pages written by 1 writer not included in the training set, as well as a few pages written by writers who are represented in the training set (but the test pages themselves were not included in the training set).

As can be seen from the results in Table 7, the writer-independent error rate on this set is 62.4% - a surprisingly high error rate in the context of much lower WER observed for English. We validated the writer-independent error rate by running a series of 6 jack-knifing experiments in each of which we set aside data from one writer for testing (a different writer each time) and used the data from the remaining writers for training. The jack-knifing experiment yielded a CER of 65.9% for the writer-independent condition. After an initial analysis of the results, we hypothesized that the primary cause for the high writer-independent (and, for that matter, the writer-in-training) error rate is the lack of adequate training data. The fact that adding machine-printed data, which on the average exhibits strikingly different characteristics than handwritten data helped improve performance is one good indicator that we have inadequate handwritten training data.

To further explore the correlation of CER with training data, we categorized the handwritten training characters into five categories based upon their frequency of occurrence in the training set. Table 8 shows the five categories along with the associated CER and Correct percentages. As can be seen from the data in Table 8, the correlation between the frequency of the character in training and the associated error rate is striking – as the number of training instances increases, the CER goes down. For 50 training samples or more, the reduction in CER is dramatic.

Table 7: Chinese Recognition Results.

Training Data	Test Data	CER – %	Correct – %
77-page Handwritten Set	Writers-in-training	46.7	55.6
	Fair Writer	66.7	37.7
77-Page Handwritten Set with Jack-knifing	Writers-in-training	N/A	N/A
	Fair Writer (average over 6 runs)	65.9	37.1
352-page Machine-printed plus 77-page Handwritten Set	Writers-in-training	44.1	57.5
	Fair Writer	62.4	39.8

Based on the analysis above, we intend to collect additional data in order to acquire more handwritten instances of the characters for use in training. Our estimate is that availability of more than 100 instances of each handwritten character represents the best possible condition for further research.

5 Summary and future Work

In this paper we have introduced a script-independent methodology for multi-lingual offline handwriting recognition based on the use of hidden Markov models (HMM) to model characters. This methodology is an extension of our approach to the OCR of machine-printed text. While HMMs offer an inherently language-independent framework, it is the combination of that framework with a script-independent feature extraction procedure that has enabled the development of a truly script-independent OHR/OCR system. The framework allows us to incorporate other constraints, such as a language model, into the recognition process so as to improve overall system performance.

At this early stage in the development of our OHR technology, the two salient differences between our OCR and OHR systems are in the use of slant normalization pre-processing and context-dependent HMMs. The experimental results on English demonstrate the fundamental strength of our approach for writer-independent, script-independent offline handwriting recognition. To the best of our knowledge the best previous results on the IAM database were reported in a recent June 2006 paper [28]. While the training and test sets used by the authors of [28] are no doubt different from ours, the fact that our results are better than those reported in [28] indicate that, at the very least, it is safe to assert that our OHR approach delivers state-of-the-art performance in English.

Table 8: Number of characters, % CER (on writer-independent test set), and % Correct (on writer-independent test set) in each category.

Category	Number of Training Samples	CER – %	Correct – %
1	> 100	28.5	75.48
2	50 – 99	48.4	55.18
3	10 – 49	73.9	28.83
4	5 – 9	96.9	3.92
5	1 – 4	99.6	0.45

In the case of Chinese, we have demonstrated the viability of our approach using the ETL9B data set. Furthermore, the results on the unconstrained handwritten images in the BBN-CH corpus indicate that given adequate data, reasonable recognition accuracy on clean data can be obtained.

Notwithstanding the recognition performance reported in this paper, it is clear that for the vast majority of real-world data, offline handwriting recognition continues to remain an open research problem. Nevertheless, we believe that the approach outlined in this paper offers a powerful framework for tackling that problem. Potential areas of future work include the development of more robust and discriminative features, better normalization techniques, and the development of better language modeling techniques.

Acknowledgements

The authors are grateful to the creators of the IAM and ETL databases and for making them available to the document research community.

References

[1] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," Proc. IEEE, Vol. 77, 257-286, 1989.

[2] J. Makhoul and R. Schwartz, "State of the Art in Continuous Speech Recognition," Proc. Natl. Acad. Sci. USA, Vol. 92, 9956-9963, 1995.

[3] A. Kundu and P. Bahl, "Recognition of handwritten script: a hidden Markov model based approach," Pattern Recognition, Vol. 22, 283-297, 1989.

[4] E. Levin and R. Pieraccini, "Dynamic planar warping for optical character recognition," IEEE Int. Conf. Acoustics, Speech, Signal Processing, San Francisco, CA, Vol. III, 149-152, 1992.

[5] J.A. Vlontzos and S.Y. Kung, "Hidden Markov models for character recognition," IEEE Trans. Image Processing, Vol. 1, 539-543, 1992.

[6] O.E. Agazzi and S. Kuo, "Hidden Markov model based optical character recognition in the presence of deterministic transformations," Pattern Recognition, Vol. 26, 1813-1826, 1993.

[7] M.-Y. Chen, A. Kundu, and S.N. Srihari, "Handwritten word recognition using continuous density variable duration hidden Markov model," Int. Conf. Acoustics, Speech, Signal Processing, Minneapolis, MN, Vol. 5, 105-108, 1993.

[8] C.B. Bose and S.-S. Kuo, "Connected and degraded text recognition using hidden Markov model," Pattern Recognition, Vol. 27, 1345-1363, 1994.

[9] J. Rocha and T. Pavlidis, "Character recognition without segmentation," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 17, 903-909, 1995.

[10] H. Bunke, M. Roth, and E.G. Schukat-Talamazzini, "Off-line cursive handwriting recognition using hidden Markov models," Pattern Recognition, Vol. 28, 1399-1413, 1995.

[11] C. Oh and W.S. Kim, "Off-line recognition of handwritten Korean and alphanumeric characters using hidden Markov models," Proc. Int. Conf. Document Analysis and Recognition, Montreal, Canada, Vol. 2, 815-818, 1995.

[12] J.C. Anigbogu and A. Belaid, "Hidden Markov models in text recognition," Int. J. Pattern Recognition and Artificial Intelligence, Vol. 9, 925-958, 1995.

[13] R.G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 18, 690-706, 1996.

[14] W.S. Kim and R.-H. Park, "Off-line recognition of handwritten Korean and alphanumeric characters using hidden Markov models," Pattern Recognition, Vol. 29, 845-858, 1996.

[15] H.-S. Park and S.-W. Lee, "Off-line recognition of large-set handwritten characters with multiple hidden Markov models," Pattern Recognition, Vol. 29, 231-244, 1996.

[16] M. Allam, "Segmentation versus segmentation-free for recognizing Arabic text," Proc. SPIE, Vol. 2422, 228-235, 1995.

[17] N. Ben Amara and A. Belaid, "Printed PAW recognition based on planar hidden Markov models," 13th Int. Conf. Pattern Recognition, Vienna, Austria, Vol. II, 220-224, 1996.

[18] F.T. Yarman-Vural and A. Atici, "A heuristic algorithm for optical character recognition of Arabic script," Proc. SPIE, Vol. 2727, Part 2, 725-736, 1996.

[19] A. Kaltenmeier, T. Caesar, J.M. Gloger, and E. Mandler, "Sophisticated topology of hidden Markov models for cursive script recognition," Proc. Int. Conf. Document Analysis and Recognition, Tsukuba City, Japan, 139-142, 1993.

[20] W. Cho, S.-W. Lee, and J.H. Kim, "Modeling and recognition of cursive words with hidden Markov models," Pattern Recognition, Vol. 28, 1941-1953, 1995.

[21] M. Mohamed and P. Gader, "Handwritten word recognition using segmentation-free hidden Markov modeling and segmentation-based dynamic programming techniques," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 18, 548-554, 1996.

[22] A.J. Elms and J. Illingworth, "Modelling polyfont printed characters with HMMs and a shift invariant Hamming distance," Proc. Int. Conf. Document Analysis and Recognition, Montreal, Canada, 504-507, 1995.

[23] K. Aas and L. Eikvil, "Text page recognition using grey-level features and hidden Markov models," Pattern Recognition, Vol. 29, 977-985, 1996.

[24] A. Kornai, "Experimental HMM-based postal OCR system," Proc. Int. Conf. Acoustics, Speech, Signal Processing, Munich, Germany, Vol. 4, 3177-3180, 1997.

[25] B. Al-Badr and S. Mahmoud, "Survey and bibliography of Arabic optical text recognition," Signal Processing, Vol. 41, 49-77, 1995.

[26] T. Starner, J. Makhoul, R. Schwartz, and G. Chou, "On-line Cursive Handwriting Recognition Using Speech Recognition Methods," IEEE Int. Conf. Acoustics, Speech, Signal Processing, Adelaide, Australia, Vol. V, 125-128, 1994.

[27] H. Park and S. Lee, "A Truly 2-D Hidden Markov

Model," Pattern Recognition, Vol. 31, No. 12, 1849-1864, 1998.

[28] A. Vinciarelli, S. Bengio, and H. Bunke, "Offline Recognition of Unconstrained Handwritten Texts Using HMMs and Statistical Language Models," IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 26, No. 6, June 2004, pp. 709-720.

[29] C-L. Liu and K. Marukawa, "Global Shape Normalization for Handwritten Chinese Character Recognition: A New Method," Proceedings of the 9th International Workshop on Frontiers in Handwriting Recognition (IWFHR-9), 2004.

[30] T. Wu and S. Ma, "Feature Extraction by Hierarchical Overlapped Elastic Meshing for Handwritten Chinese Character Recognition," Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR), 2003.

[31] Y. Y. Tang, L. Tu, J. Liu, S. Lee, W Lin, and I. Shyu, "Offline Recognition of Chinese Handwriting by Multifeature and Multilevel Classification," IEEE. Trans. On Pattern Analysis and Machine Intelligence, Vol 20, No. 5, May 1998.

[32] P. Natarajan, Z. Lu, I. Bazzi, R. Schwartz, and J. Makhoul, "Multilingual Machine Printed OCR," International Journal of Pattern Recognition and Artificial Intelligence, 15:1, 2001, pp. 43–63.

[33] G.D. Forney, "The Viterbi Algorithm," Proc. IEEE, Vol. 61, 268-278, 1973.

[34] S. .Austin, R. Schwartz, and P. Placeway, "The Forward-Backward Search Algorithm," IEEE Int. Conf. Acoustics, Speech, Signal Processing, Toronto, Canada, Vol. V, 697-700, 1991.

[35] R. Schwartz, L. Nguyen, and J. Makhoul, "Multiple-Pass Search Strategies," in Automatic Speech and Speaker Recognition: Advanced Topics, C-H. Lee, F.K. Soong, K.K. Paliwal, Eds., Kluwer Academic Publishers, 429-456, 1996.

[36] L. Nguyen, T. Anastasakos, F. Kubala, C. LaPre, J. Makhoul, R. Schwartz, N. Yuan, G. Zavaliagkos, and Y. Zhao, "The 1994 BBN/BYBLOS Speech Recognition System," Proc. ARPA Spoken Language Systems Technology Workshop, Austin, TX, Morgan Kaufmann Publishers, 77-81, 1995.

[37] J. Makhoul, R. Schwartz, C. LaPre, and I. Bazzi, "A Script-Independent Methodology for Optical Character Recognition," Pattern Recognition, Vol. 31, No. 9, 1285-1294, 1998.

[38] Z. Lu, R. Schwartz, and C. Raphael, "Script-Independent, HMM-based Text Line Finding for OCR," Int. Conf. Pattern Recognition, Barcelona, Spain, Vol. 4, 551-554, Sept. 2000 .

[39] J. Makhoul and R. Schwartz, "Language-Independent and Segmentation-Free Optical Character Recognition," U.S. Patent No. 5933525, August 3, 1999.

[40] K. Fukunaga, Introduction to Statistical Pattern Recognition, Academic Press, New York, Chapter 10, Second Edition, 1990.

[41] L.E.Baum, "An inequality and associated maximization technique in statistical estimation for probabilistic functions of Markov processes," Inequalities, Vol. 3, 1-8, 1972.

[42] A.P. Dempster, N.M. Laird, and D.B. Rubin, "Maximum-likelihood from incomplete data via the EM algorithm," J. Royal Statist. Soc. Ser. B (methodological), Vol. 39, 1-38, 1977.

[43] R.A. Redner and H.F. Walker, "Mixture densities, maximum likelihood and the EM algorithm," SIAM Review, Vol. 26, 195-239, 1984.

[44] J. Bellegarda and D. Nahamoo, "Tied Mixture Continuous Parameter Models for Large Vocabulary Isolated Speech Recognition," IEEE Int. Conf. Acoustics, Speech, Signal Processing, Glasgow, Scotland, Vol. 1,13-16, 1989.

[45] X.D. Huang and M.A. Jack, "Semi-continuous Hidden Markov Models for Speech Recognition," Computer Speech and Language, Vol. 3, 1989.

[46] Z. Lu, R. Schwartz, P. Natarajan, I. Bazzi, and J. Makhoul, "Advances in the BBN BYBLOS OCR System," Proc. Of Intl. Conf. Doc. Analysis and Recognition, 337-340, Bangalore, India, 1999.

[47] Yong Haur Tay, Pierre Michel Lallican, Marzuki Khalid, Christian Viard-Gaudin, Stefan Knerr, "An Offline Cursive Handwritten Word Recognition System", Proceedings of IEEE Region 10 Conference, 2001

[48] U. Marti and H. Bunke., "A full English sentence database for off-line handwriting recognition," Proc. of the 5th Int. Conf. on Document Analysis and Recognition, ICDAR'99, Bangalore, 1999, pages 705 – 708.

[49] S. Johansson, G.N. Leech, and H. Goodluck, "Manual of Information to accompany the Lancaster-Oslo/Bergen Corpus of British English, for use with digital Computers," Department of English, University of Oslo, Norway, 1978.

Demonstrations and Poster Abstracts

Introduction to Hanwang Arabic Handwriting Recognition

Hanwang Technology
Beijing, P.R. China, http://www.hw99.com/english/
Contact: Xingyu Niu, nxy@hanwang.com.cn

This software demonstrates the online Arabic handwriting recognition technology of Hanwang, running on Windows. The main features of this technology are as follows.

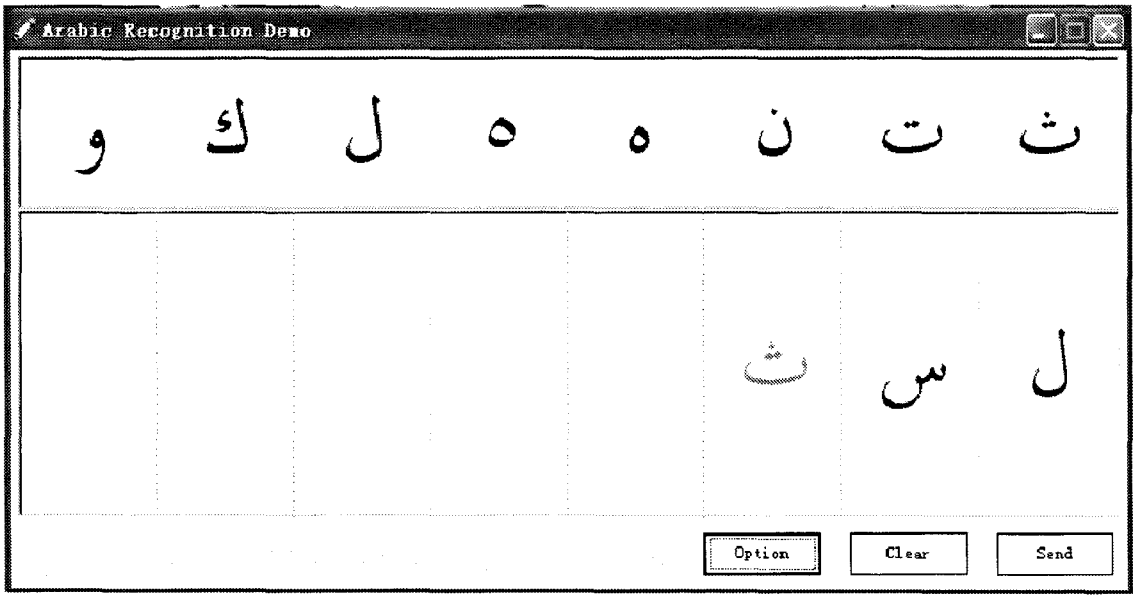
Superior Recognition Accuracy

The software supports the recognition of naturally shaped Arabic letters (28), Arabic-Indic numbers (10) and Arabic punctuations (6), as well as English letters (52), digits (10), punctuation/symbols (37), and gestures (4). It provides highly accurate handwriting recognition, and the user can input any symbol in the whole character set without need to change the writing mode and to designate the writing area. The recognition accuracy on a Hanwang internal test set is 96%.

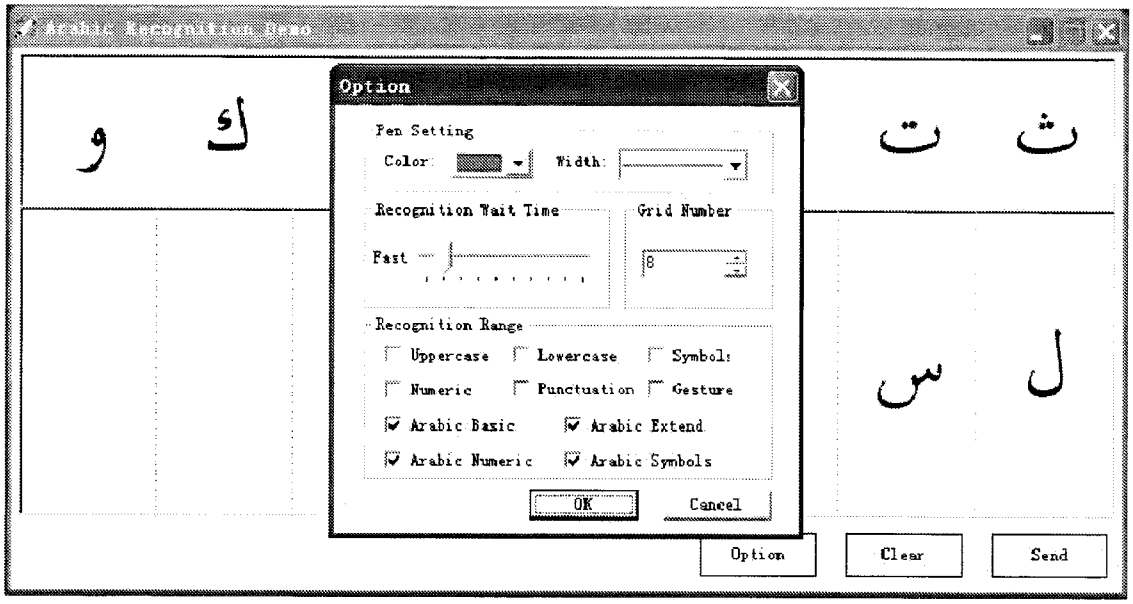
Hanwang will add to the software a function of Arabic word prediction in the fall of 2006, and realize the Arabic cursive word recognition module in spring of 2007.

Smart UI Design

The UI is designed based on Arabic writing style. It allows the user to write easily, to select candidates, and to send the recognition results to an editor. See the picture as below.



The software also provides options for the user to customize the UI style. The user can change the color of pen trajectory, the number of writing boxes, the character set, etc. See the picture as below.



Portability to Various Platforms

The software is written in ANSI C for portability and easy implementation on many different OS platforms, especially, on embedded platforms with limited computing resource. It consumes only 14K bytes RAM and 200K bytes ROM/Flash. The recognition speed is as high as 40 characters/second on platform of WinCE 4.0 + Xscale 206MHz.

About Hanwang Technology Company, Ltd.

Hanwang Technology, founded in 1993, is the largest intelligent recognition software provider in China, with focus on handwriting recognition, OCR, and machine vision applications. Currently, the handwriting recognition technology of Hanwang supports multiple languages including Chinese (simplified and traditional), Latin, Korean, Japanese, Thai and Arabic, and runs on different OS platforms such as Windows, WinCE, Linux, Symbian, etc. Hanwang's partners include Microsoft, Nokia, Sony-Ericsson, Samsung, LG, Lenovo. Currently, about 40,000,000 users are using Hanwang technologies and products worldwide.

Using Graphs to Bridge from Static Documents to Dynamic Handwriting Recognition for Handwritten Arabic Documents

Mark A. Walch*, and Rami Safadi**
*The Gannon Technologies Group, Alexandria, Virginia
**Sakhr Software, USA, 1100 Connecticut Avenue. Washington, DC 20036

The Presentation will provide a high-level overview of a method for bringing the power of Dynamic Handwriting Recognition to the analysis of handwritten Arabic documents. The presentation will focus on a particular existing Dynamic Recognition technology and an existing technology for "bridging" between this technology and Static Recognition of images.

This demonstration presents a method for extending Dynamic Handwriting Recognition technology to process handwritten Arabic language documents.

Dynamic Handwriting Recognition provides real time interpretation of handwritten strokes and is used primarily within Tablet PC or PDA environments. In the Tablet PC application, the dynamic recognizer receives real-time handwriting data provided directly by the writer using a stylus. Based on the movements of the stylus, a digital representation of handwritten strokes can be captured as words are written. These strokes we stored as data are frequently referenced as "digital ink". Digital ink consists of geometric plots of the strokes, stroke sequence, pen pressure, pen angle, and the like. Of these features, the ones most important for recognition are the geometry of the strokes and the sequence in which they were written. Because of its rich feature set, Dynamic Handwriting Recognition has achieved very significant accuracy levels.

Unlike Dynamic Recognition where user input is captured in real time, Static Handwriting Recognition involves capturing data from images of scanned documents. Static Recognition also draws upon fewer possible features than Dynamic Recognition. For instance, many Dynamic Recognition features that can be captured while the actual writing is taking place—stroke direction, stroke sequence, pen pressure and the like—are not present in scanned images and, thus, not available for Static Recognition. Thus, Dynamic Recognition technology is not directly applicable for Static Recognition tasks such as text conversion from scanned documents.

Graphs, offer the basis for a data structure that can enable Dynamic Recognition engines to perform Static Recognition on handwritten images. Graphs can be used in support of Static Recognition by creating mathematical representations from writing samples that couples the topology and geometry of writing with other information not directly available from images—such as stroke sequence. In fact, handwriting's structural form of connected pen strokes is well suited to be represented by graphs. The proposed demonstration and presentation will discuss a process for extracting graphs from static handwriting and attaching to these graphs data indicating the stroke sequence a writer would use to produce the graph. This combination of graphs and stroke sequence

provides sufficient data to generate the digital ink that permits the Dynamic Recognition process.

The demonstration also discusses a method for generating a database of graphs coupled with attendant stroke information. Included is a method for extracting graphs from a static image of a scanned document, extracting graphs from these images and matching these graphs with reference graphs stored in the database to obtain the stroke sequence information. Once this matching has been made and the static graphs have been augmented with stroke sequence information they serve as a means for bringing the power of Dynamic Recognition technology to static images.

Experiences with handwritten Arabic shows that many handwritten words can be described through very few graphs. For instance, only 20 unique graphs cover 90 percent of written Arabic characters and 35 unique graphs cover 95%. These are all very manageable numbers. A database of these graphs with attendant stroke sequence information provides the bridge for bringing Dynamic Handwriting Recognition to static handwritten documents.

Arabic Handwriting Recognition at MITRE

Amlan Kundu, Tom Hines, Jon Phillips and Ben Huyck
The MITRE Corporation, 7515 Colshire Drive, McLean, VA 22102-7508

Many of the Arabic script documents gathered in Afghanistan, Iraq, and the War on Terror are handwritten. Limited technology currently exists for recognizing offline handwritten Arabic. The number of documents to be translated far exceeds the manpower available to process them quickly. We must assemble technology solutions for digitizing, translating, and triaging the data so that we can deliver high-priority documents to analysts more rapidly.

Objectives

Our objective is to develop techniques to rapidly modify existing handwriting recognition algorithms to process Arabic and other low-density languages. We will solidify the methodology on handwritten Arabic and fine tune it on Pashto or another low-density language.

Activities

We will publicly release collected training data and corpus generation tools. Our objective is to improve performance by investigating search and matching algorithms, cross-lingual feature typologies, algorithms for segmentation and feature extraction, alternatives for diacritic handling, and a multi-pass processing scheme. We are investigating task-focused recognition and extensions to enable dynamic swapping of language models.

Impact

By taking a systematic approach to rapid development of international character recognition systems, MITRE will be prepared to help overcome the next language processing crisis. Prototypes for handwritten Arabic script recognition and corpus generation and corpora of handwritten Arabic will be made available for technology transfer. Technical reports and white papers published under this research program will advance the state of the art.

CEDARABIC - A System For Processing Handwritten Arabic Documents

Sargur N. Srihari, Gregory R. Ball, and Harish Srinivasan
Center of Excellence for Document Analysis and Recognition (CEDAR)
University at Buffalo, State University of New York
Amherst, New York 14228
srihari@cedar.buffalo.edu

CEDARABIC (سيدرابك) is an end-to-end interactive system designed for the analysis of handwritten Arabic documents. CEDARABIC's abilities are generally analogous to those found in a predecessor system known as CEDAR-FOX, which was developed for forensic examination of documents by U.S. law enforcement. Being a software branch of the CEDAR-FOX system, CEDARABIC also has capabilities for image processing operations such as enhancement, thresholding, underline removal, etc., as well as for writer verification, writer identification and signature verification. The interfaces have been modified to be appropriate for analyzing Arabic language documents by an English speaking user. The system includes a GUI for various functions including scanning, selecting a region of interest, feature computation, ground truthing (manual segmentation and entering word truth), searching of a word based on its English equivalent, etc. CEDARABIC has the ability to search and recognize handwritten Arabic documents and has tools for document transcription, useful in building corpora for further research.

At present, scanned document retrieval is the principal function of CEDARABIC. Documents can be input to CEDARABIC via scanning, or it can act on image files directly. Searching documents is accomplished by selecting a search strategy and entering a search query. The query may be an English word, such as 'king,' an ASCII representation of an Arabic word, such as 'Alef|Lam|Meem|Lam|Kaf|,' or by UNICODE input of an Arabic word, such as 'الملك.' The query can also be an image, such as a handwritten word selected from a document.

Depending on the strategy, documents may first be indexed with an automated segmentation module, or the searches may proceed directly on unindexed documents using a segmentation free approach. Different search strategies make use of what information is available. One strategy extracts images corresponding to handwritten versions of the query and recognizes based on the shape of the word. In this strategy the prototypes (template word images) corresponding to the Arabic word for the queried text, but must be documents which are set aside as training documents. The template word images serve as a knowledge base to learn the writing styles of the particular word and then use this to identify other similar words purely based on matching the image. Another strategy searches directly by using prototype character images, which has the benefit of requiring no training prototypes. A set of test documents can now be specified, and the tool will look for the occurrence of the particular word in every document specified and rank the retrieved words in the order of probability of match.

GEDI

Ground Truth Editor and Document Interface

Michael Roth Stefan Jaeger David Doermann

LAMP Lab at University of Maryland at College Park, College Park, MD 20742
{mroth, jaeger,doermann}@umiacs.umd.edu

GEDI is a ground truth editor that gives users the ability to label documents. Its basic structure involves two types of files, a .tif image file and its associated .xml text file. After a series of image files are scanned into a computer they can be opened simultaneously as a folder. Once the image files are opened, the interface creates a .xml text file for each individual image file. The purpose of the .xml file is to store specific attributes, set by the user, for the given image file.

The interface allows a user to select from the image files that are opened and then create boxes, also known as zones, around desired portions of the image file. These zones can have user defined attributes that set it apart from other types of zones. For example, on might label a signature with different attributes than a stamp. Even if they contain the same attribute names, such as "quality" for example, the values of the attributes themselves might differ. In addition to there being attributes for each zone, the folder as a whole may have attributes such as a writer ID, what hand the documents were written with and the author's age. These attributes pertain to all the image files. Therefore, in each of the individual .xml text files, the document's zones are listed, with their attributes as well as the folder's attributes.

The user defined attributes are stored in a workspace file that is altered when a user either adds or removes attributes or types of zones. The internal data structures retrieve their information from the workspace file. Each folder has its own workspace. When a folder is initially opened, the associated workspace is loaded too. If the user wishes to include a workspace from a different folder, a merge operation can be performed. This would keep the existing workspace and incorporate the new workspace attributes and zones into the current workspace file. The workspace file is stored as a .xml text file and maintains the same format as each individual .xml file.

Labeling the data on a document can be made easy by setting shortcut keys to tasks which may set an attribute to a certain value or the desired zone type. There are functionalities within GEDI that make it even easier to use. A feature called, TEXTLINEGT, permits the user to label lines of text by typing in the text included within the surrounding zone. The user then segments this TEXTLINEGT zone by line, word, or character (Figure 1).

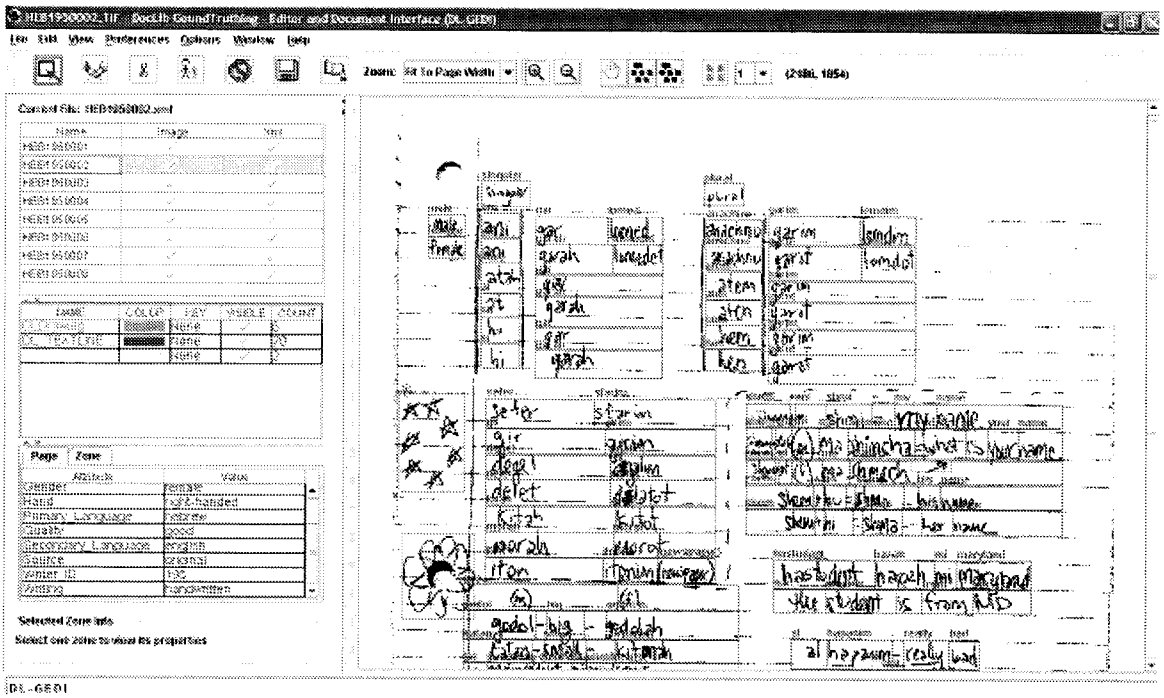


Figure 1 – This depicts the use of TEXTLINEGT as it was used to segment lines of Hebrew text into lines and words. Also, zones were created around drawings as well.

A hierarchy between zones is also evident. Visual lines can be seen by toggling the parent/child option. Should the user want to discontinue future zones from maintaining a hierarchy, they can by toggling a hierarchy option. Hierarchy between two zones can be created or destroyed at any point.

Lastly, when an attribute is created, it may be set to have a flag that makes it an "arbitrary attribute". This will allow a user to reassign the attributes value on the fly without having to add the new value to the value list. In other words, this value that was created on the fly will exist only in the image's individual .xml file and not in the main workspace file. It is also possible to rename attribute names and values and propagate these changes throughout a folder.

Word Spotting in PCR Forms

Faisal Farooq and Venu Govindaraju
Dept of Computer Science and Engineering, CEDAR,
Center for Unified Biometrics and Sensors (CUBS)
University at Buffalo

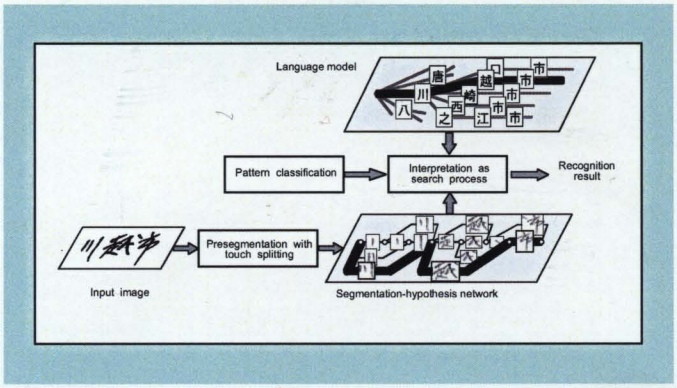
In New York State all patients who enter the Emergency Medical System (EMS) are tracked through their pre-hospital care to the emergency room using the PreHospital Care Report (PCR). The PCR is used to gather vital patient information and has manifold purpose: (i) it is used by the emergency room staff in providing appropriate care, (ii) it is to facilitate record keeping about the patient and the care provided, and (iii) to afford large regional EMS (such as WREMS, Western Regional Emergency Medical Systems.) to gain trend knowledge through macro analysis. Our goal is to automate the collection of data from the PCR and enable efficient maintenance, and dissemination of information. The task proposed is quite challenging for several reasons: (i) handwritten responses are very loosely constrained in terms of writing style, format of response, and choice of text, (ii) medical lexicons of words are very large (about 50,000 entries), iii) forms are often very noisy due to irrepressible emergency situations. This leads to the automatic transcription of the forms very difficult. We will be demonstrating a recognition free system for word spotting in the PCR forms. Thus, given query terms in plain text, we spot these keywords in the PCR form and present them to the user with a friendly color coded scheme based on relevance of the results to the query terms.

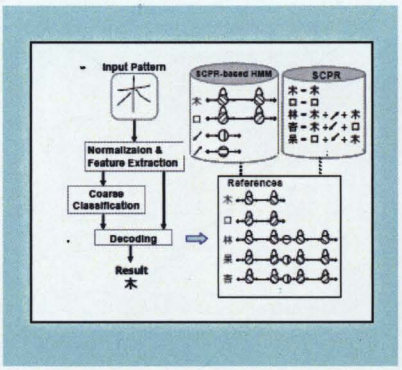
Discriminating Arabic Handwriting from Machine Print

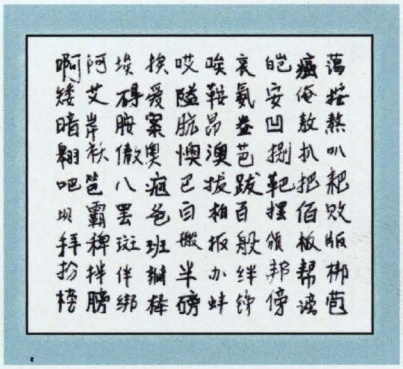
Faisal Farooq and Venu Govindaraju
Dept of Computer Science and Engineering, CEDAR,
Center for Unified Biometrics and Sensors (CUBS)
University at Buffalo

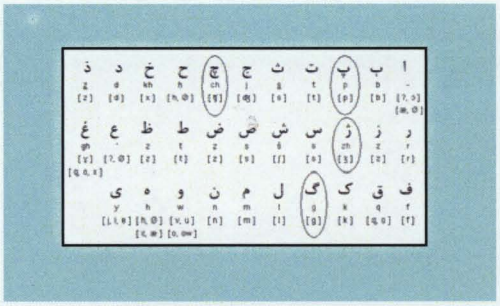
The processing of document images prior to recognition plays a significant part in the development of Handwriting Recognition systems. In a document that has both machine printed and handwritten text, it is important to distinguish between the two prior to feature extraction. It also finds application in postal automation, bank check reading and more recently in forensic sciences. This task is challenging in Arabic because the script is cursive in both machine print and handwriting and there are no special rules that can easily distinguish them. The methods for differentiating machine printed and handwritten text in the literature do not readily apply to the Arabic script because of their script dependent rules and features. We will be demonstrating our system for discriminating Arabic machine print from handwritten text. The system is based on Gabor features followed by classification using an expectation-maximization based probabilistic neural network that avoids over-fitting even with limited training data. In addition our system is very general and easily trainable given training data for multiple scripts.

Author Index


Language model
Pattern classification
Interpretation as search process
Recognition result
Presegmentation with touch splitting
Input image
Segmentation-hypothesis network


Input Pattern
SCPR-based HMM
SCPR
Normalization & Feature Extraction
Coarse Classification
Decoding
References
Result






نقع