Online Handwritten Script Recognition

Anoop M. Namboodiri, *Student Member*, *IEEE*, and
Anil K. Jain, *Fellow*, *IEEE*

Abstract—Automatic identification of handwritten script facilitates many important applications such as automatic transcription of multilingual documents and search for documents on the Web containing a particular script. The increase in usage of handheld devices which accept handwritten input has created a growing demand for algorithms that can efficiently analyze and retrieve handwritten data. This paper proposes a method to classify words and lines in an online handwritten document into one of the six major scripts: *Arabic*, *Cyrillic*, *Devnagari*, *Han*, *Hebrew*, *or Roman*. The classification is based on 11 different spatial and temporal features extracted from the strokes of the words. The proposed system attains an overall classification accuracy of 87.1 percent at the word level with 5-fold cross validation on a data set containing 13,379 words. The classification accuracy improves to 95 percent as the number of words in the test sample is increased to five, and to 95.5 percent for complete text lines consisting of an average of seven words.

Index Terms—Document understanding, handwritten script identification, online document, evidence accumulation, feature design.

1 Introduction

With the increase in popularity of portable computing devices such as PDAs and handheld computers [1], [2], nonkeyboard-based methods for data entry are receiving more attention in the research communities and commercial sector. The most promising options are pen-based and voice-based inputs. Digitizing devices like SmartBoards [3] and computing platforms such as the IBM Thinkpad TransNote [4] and Tablet PCs [5], have a pen-based user interface. Such devices, which generate handwritten documents with online or dynamic (temporal) information, require efficient algorithms for processing and retrieving handwritten data. Online documents may be written in different languages and scripts.[1] A single document page in itself may contain text written in multiple scripts. For example, a document in English may have some annotations or edits in another language. Most of the text recognition algorithms are designed to work with a particular script and treat any input text as being written only in the script under consideration. Therefore, an online document analyzer must first identify the script before employing a particular algorithm for text recognition. Fig. 1 shows an example of a document page containing six different scripts. Note that a typical multilingual, online document contains two or three scripts [6]. The document page shown in Fig. 1 was created by us for illustrating the results. Note that the writing style in English is mixed, with both cursive and handprint words. For lexicon-based recognizers, it may be helpful to identify the specific language of the text if the same script is used by multiple languages.

A script is defined as a graphic form of a writing system [7]. Different scripts may follow the same writing system. For example, the *alphabetic* system is adopted by scripts like Roman and Greek, and the *phonetic-alphabetic* system is adopted by most Indian scripts, including Devnagari. A specific script like Roman may be used by multiple languages such as English, German, and French. Detailed studies of the history and evolution of scripts can be found in Jensen [8] and Lo [9]. The six scripts considered in this work, named Arabic, Cyrillic, Devnagari, Han, Hebrew, and Roman, cover the languages used by a majority of the world population [10] (see Fig. 1). The general class of Han-based scripts include Chinese, Japanese, and Korean (we do not consider Kana or Han-Gul). Devnagari script is used by many Indian languages, including Hindi, Sanskrit, Marathi, and Rajasthani. Arabic script is used by Arabic, Farsi, Urdu, etc. Roman script is used by many European languages like English, German, French, and Italian. We attempt to solve the problem of script recognition, where either an entire document or a part of a document (up to word level) is classified into one of the six scripts mentioned above, with the aim of facilitating text recognition or retrieval. The problem of identifying the actual language often involves recognizing the text and identifying specific words or sequence of characters, which is beyond the scope of this paper.

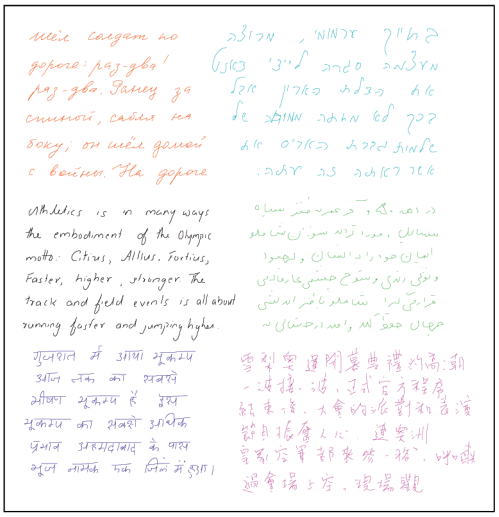


Fig. 1. A multiscript online document containing Cyrillic, Hebrew, Roman, Arabic, Devnagari, and Han scripts.

Most of the published work on automatic script recognition deals with offline documents, i.e., documents which are either handwritten or printed on a paper and then scanned to obtain a two-dimensional digital representation. For printed documents, Hochberg et al. [11] used cluster-based templates for discriminating 13 different scripts. Spitz [12] proposed a language identification scheme where the words of 26 different languages are first classified into Han-based and Latin-based scripts. The actual languages are identified using projection profiles of words and character shapes. Jain and Zhong [13] used Gabor filter-based texture features to segment a page into regions containing Han and Roman scripts. Pal and Chaudhuri [14] have developed a system for identifying Indian scripts using horizontal projection profiles and looking for the presence or absence of specific shapes in different scripts. Other approaches to script and language identification in printed documents are reported in [15], [16], [17]. There have been very few attempts on handwritten script identification in offline documents. Hochberg et al. [18] used features of connected components to classify six different scripts (Arabic, Chinese, Cyrillic, Devnagari, Japanese, and Roman) and reported a classification accuracy of 88 percent on document pages.

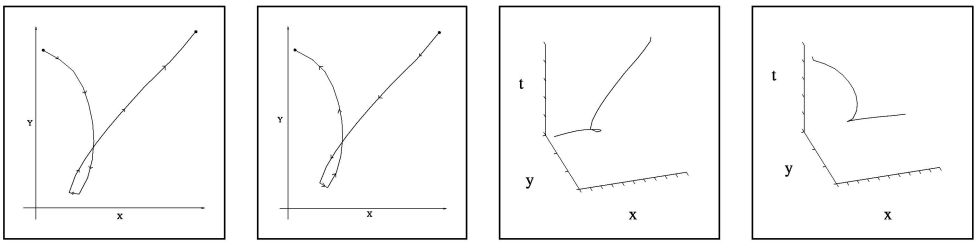
There are a few important aspects of online documents that enable us to process them in a fundamentally different way than offline documents. The most important characteristic of online documents is that they capture the temporal sequence of strokes[2] while writing the document (see Fig. 2). This allows us to analyze

1. Multiple languages may use the same script. See explanation in the next paragraph.

2. A stroke is defined as the locus of the tip of the pen from pen-down to the next pen-up position.

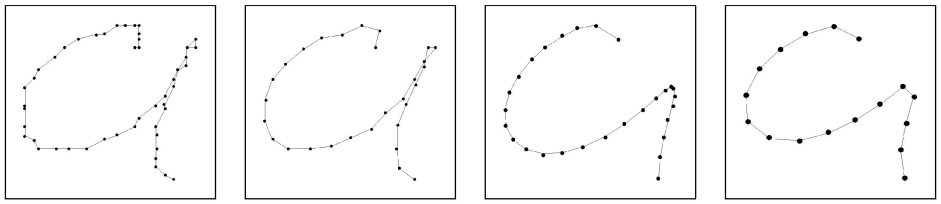
• *The authors are with the Department of Computer Science and Engineering, Michigan State University, East Lansing, MI 48824. E-mail: {anoop, jain}@cse.msu.edu.*

Manuscript received 7 Feb. 2002; revised 31 Aug. 2002; accepted 1 Nov. 2002. Recommended for acceptance by L. Vincent.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 115864.



(a) (b) (c) (d)

Fig. 2. Online script variability. The character "r" written in two different styles. The offline representations ((a) and (b)) look similar, but the temporal variations make them look very different ((c) and (d)). The writing direction is indicated using arrows in (a) and (b). The vertical axes in (c) and (d) represent time.



(a) (b) (c) (d)

Fig. 3. Preprocessing. (a) An input stroke. The dots represent the sampling points. (b) The stroke after equidistant resampling. (c) Result of lowpass filtering. (d) The final result after second resampling.

the individual strokes and use the additional temporal information for both script identification as well as text recognition.

In the case of online documents, segmentation of foreground from the background is a relatively simple task as the captured data, i.e., the (x, y) coordinates of the locus of the stylus, defines the characters and any other point on the page belongs to the background. We use stroke properties as well as the spatial and temporal information of a collection of strokes to identify the script used in the document. Unfortunately, the temporal information also introduces additional variability to the handwritten characters, which creates large intraclass variations of strokes in each of the script classes. Fig. 2 shows two samples of the character "r," with and without the temporal information. Even though the spatial representations in Figs. 2a and 2b look similar, the temporal differences introduce large intraclass variability in the online script, as shown in Figs. 2c and 2d.

Currently, there are a few algorithms available for online text recognition for individual scripts, but there have been no attempts to automatically recognize the script in online documents. The only work in processing multilingual online documents that we are aware of is by Lee et al. [6], which attempts to do recognition of multiple languages simultaneously using a hierarchical Hidden Markov Model. A script identification system can improve the utility and performance of online data capturing devices, and also aid in the search and retrieval of handwritten documents on the Internet containing a specific script.

2 Data Collection and Preprocessing

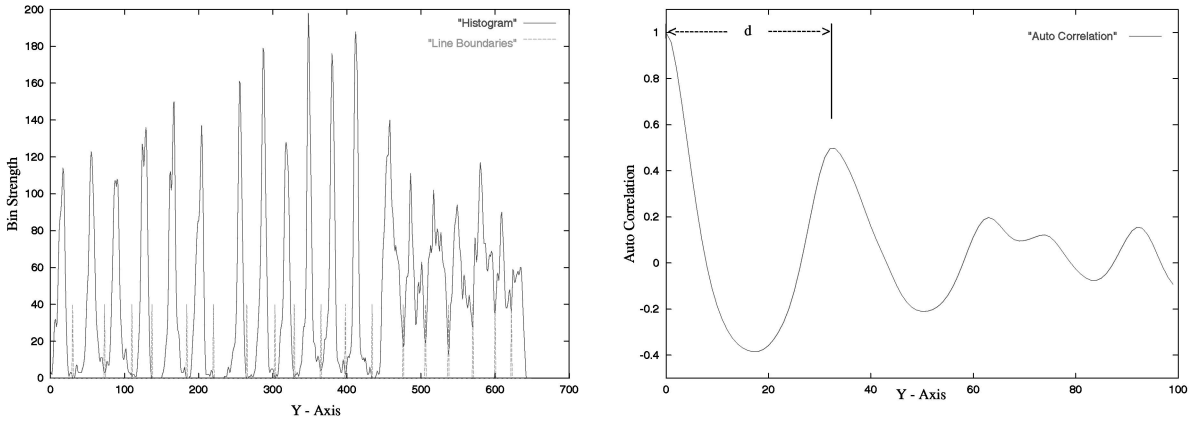
The data used in this paper was collected using the *CrossPad*®.[3] The *CrossPad* has a pen and paper interface along with the ability to digitally capture the (x, y) position of the pen tip using an *RF* transmitter embedded in the pen. The pen position is sampled at a constant rate of 132 samples per second and the device has a resolution of $0.1\ mm$ along the x and y axes. The data was collected on ruled paper with an interline distance of $8.75\ mm$, although some writers wrote on alternate lines. We must point out that the actual device for data collection is not important as long as it can generate a temporal sequence of x and y positions of the pen tip. However, the writing styles of people may vary considerably on different writing surfaces and the script classifier may require training on different surfaces.

During preprocessing, the individual strokes are resampled to make the sampled points equidistant. This helps to reduce the variations in scripts due to different writing speeds and to avoid anomalous cases such as having a large number of samples at the same position when the user holds the pen down at a point. The strokes are then smoothed using a Gaussian (lowpass) filter. The x and y coordinates of the sequence of points are independently filtered by convolution with one-dimensional Gaussian kernels. This reduces noise due to pen vibrations and errors in the sensing mechanism. The individual strokes are again resampled to make the points equidistant. The resampling distances in both cases were set to 10 pixels and the standard deviation of the Gaussian was set to 6, both determined experimentally for the data. During the lowpass filtering and resampling operations, the *critical points* in a stroke are retained. A critical point is defined as a point in the stroke where the x or y direction of the stroke reverses (changes sign), in addition to the extreme points (pen-up or pen-down) of the stroke. Fig. 3 shows an example of preprocessing the online character *a* (consisting of a single stroke).

TABLE 1
Number of Pages, Lines, and Words of Data Collected for Each of the Six Scripts in the Database

Script	Number of Writers	Number of Pages	Number of Lines	Number of Words
Arabic	15	15	209	1423
Cyrillic	10	10	276	1002
Devnagari	12	18	382	3173
Han	12	15	282	1981
Hebrew	10	10	284	2261
Roman	45	45	722	3539

3. Crosspad was manufactured by IBM and A.T. Cross [19].



(a) (b)

Fig. 4. Interline distance estimation. (a) The y-axis projection of the document in Fig. 1, and the text-line boundaries. (b) Autocorrelation of the projection used to estimate the value of d.

Each user was asked to write one page of text in a particular script, with each page containing approximately 20 lines of text. No restriction was imposed on the content or style of writing (cursive or handprint). The writers consisted of college graduate students, professors, and employees in private companies and businessmen. The details of the database used for this work are given in Table 1. Multiple pages from some of the writers were collected at different times.

2.1 Line and Word Detection

The data available to a script recognizer is usually a complete handwritten page or a subset of it. To recognize the script of individual lines or words in the page, we first need to segment the page into lines and words. The problem of text line identification in online documents has been attempted before [20]. To identify the individual lines, first the interline distance is estimated. The interline distance, d, is defined as the distance between successive peaks in the autocorrelation of the y-axis projection of the text. Fig. 4a shows the y-axis projection of the document in Fig. 1, and Fig. 4b shows the autocorrelation of the projection. The interline distance estimate is indicated in Fig. 4b as d.

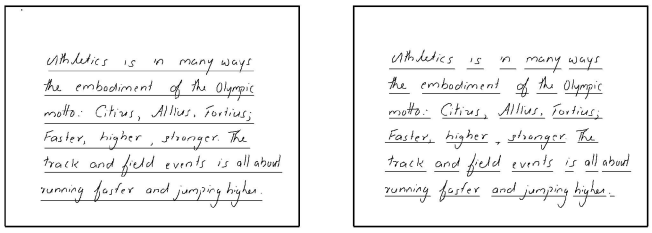
Lines are identified by finding valleys in the projection, keeping the interline distance as a guiding factor. To avoid local minima, we choose only those points, which have the smallest magnitude within a window of width d, as valleys. The text-line boundaries identified for the document in Fig. 1 are shown in Fig. 4a, along with the y-axis projection. Once the line boundaries are obtained, the text is divided into lines by collecting all the strokes which fall in between two successive line boundaries. The temporal information from stroke order is used to disambiguate strokes which fall across line boundaries and to correctly group small strokes, which may fall into an adjacent line. Temporal information is also used to split lines in pages with multicolumn text (e.g., the document in Fig. 1). Fig. 5a shows the output of our line detection algorithm for part of the multicolumn document in Fig. 1.

A word is defined as a set of strokes that overlap horizontally. The segmentation of a line into words is done using an x-axis projection of the text in the line. The valleys in the projection are noted as word boundaries and the strokes which fall between two boundaries are collected and labeled as a word. The minimum width of a valley in the projection for word segmentation was experimentally determined as 30 pixels for the resolution of the digitizing device used ($0.1\ mm$). Fig. 5b shows the output of our word detection algorithm for the document in Fig. 1.

3 Feature Extraction

Each sample or pattern that we attempt to classify is either a word or a set of contiguous words in a line. It is helpful to study the general properties of each of the six scripts for feature extraction.

1. *Arabic*: Arabic is written from right to left within a line and the lines are written from top to bottom. A typical Arabic character contains a relatively long main stroke which is drawn from right to left, along with one to three dots. The character set contains three long vowels. Short markings (diacritics) may be added to the main character to indicate short vowels [21]. Due to these diacritical marks and the dots in the script, the length of the strokes vary considerably.
2. *Cyrillic*: Cyrillic script looks very similar to the cursive Roman script. The most distinctive features of Cyrillic script, compared to Roman script are: 1) individual characters, connected together in a word, form one long stroke, and 2) the absence of delayed strokes (see Fig. 7). Delayed strokes cause movement of the pen in the direction opposite to the regular writing direction.
3. *Devnagari*: The most important characteristic of Devnagari script is the horizontal line present at the top of each word, called "*Shirorekha*" (see Fig. 6). These lines are usually drawn after the word is written and hence are similar to delayed strokes in Roman script. The words are written from left to right in a line.
4. *Han*: Characters of Han script are composed of multiple short strokes. The strokes are usually drawn from top to bottom and left to right within a character. The direction of writing of words in a line is either left to right or top to bottom. The database used in this study contains Han script text of the former type (horizontal text lines).



(a) (b)

Fig. 5. Identifying *text lines* and *words* in the document in Fig. 1. Individual (a) *text lines* and (b) *words* detected by our algorithm are underlined.

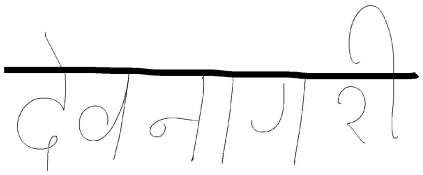


Fig. 6. The word "*devnagari*" written in Devnagari script. The *Shirorekha* is shown in bold.

Fig. 7. The word "*trait*" contains three *delayed strokes*, shown as bold dotted lines here.

5. *Hebrew*: Words in a line of Hebrew script are written from right to left and, hence, the script is temporally similar to Arabic. The most distinguishing factor of Hebrew from Arabic is that the strokes are more uniform in length in the former.
6. *Roman*: Roman script has the same writing direction as Cyrillic, Devnagari, and Han scripts. We have already noted the distinguishing features of these scripts compared to the Roman script. In addition, the length of the strokes tends to fall between that of Devnagari and Cyrillic scripts.

The features are extracted either from the individual strokes or from a collection of strokes. Here, we describe the features and their method of computation. The features are presented in the order of their saliency in the final classifier (as determined by the feature selection algorithm described in Section 5).

1. *Horizontal Interstroke Direction (HID)*: This is the sum of the horizontal directions between the starting points of consecutive strokes in the pattern. The feature essentially captures the writing direction within a line.

$$HID = \sum_{i=1}^{n-r} dir(i, i+r),$$

where

$$dir(i,j) = \begin{cases} +1 & X_{start}(stroke_i) < X_{start}(stroke_j) \\ -1 & \text{otherwise,} \end{cases}$$

where $X_{start}(.)$ denotes the x coordinate of the pen-down position of the stroke, n is the number of strokes in the pattern, and r is set to 3 to reduce errors due to abrupt changes in direction between successive strokes. The value of HID falls in the range $[r-n, n-r]$.

2. *Average Stroke Length (ASL)*: As described in Section 2, each stroke is resampled during preprocessing so that the sample points are equidistant. Hence, the number of sample points in a stroke is used as a measure of its length. The Average Stroke Length is defined as the average length of the individual strokes in the pattern.

$$ASL = \frac{1}{n}\sum_{i=1}^{n} length(stroke_i),$$

where n is the number of strokes in the pattern. The value of ASL is a real number which falls in the range $[1.0, R_0]$, where the value of R_0 depends on the resampling distance used during preprocessing ($R_0 = 600$ in our experiments).

3. *Shirorekha Strength*: This feature measures the strength of the horizontal line component in the pattern using the Hough transform. The value of this feature is computed as:

$$ShirorekhaStrength = \frac{\sum\limits_{\forall\, r, -10<\theta<10} H(r,\theta)}{\sum\limits_{\forall\, r,\theta} H(r,\theta)},$$

where $H(r,\theta)$ denotes the number of votes in the (r,θ)th bin in the two-dimensional Hough transform space. The Hough transform can be computed efficiently for dynamic data by considering only the sample points. The numerator is the sum of the bins corresponding to line orientations between -10° and 10° and the denominator is the sum of all the bins in the Hough transform space. Note that it is difficult to constraint the values of r in the transform space due to variations introduced by sampling and the handwriting itself. The value of *Shirorekha Strength* is a real number which falls in the range $[0.0, 1.0]$.

4. *Shirorekha Confidence*: We compute a confidence measure for a stroke being a Shirorekha (see Fig. 6). Each stroke in the pattern is inspected for three different properties of a Shirorekha; Shirorekhas span the width of a word, always occur at the top of the word, and are horizontal. Hence, the confidence (C) of a stroke (s) is computed as:

$$C(s) = \frac{width(s)}{width(pattern)} * \frac{\bar{Y}(s)}{height(pattern)} * \left(1 - \frac{height(s)}{width(s)}\right),$$

where $width(s)$ refers to the length along the x-axis (horizontal), $height(s)$ is the length of a stroke along the y-axis (vertical), and $\bar{Y}(s)$ is the average of the y-coordinates of the stroke points. Note that $0 \le C(s) \le 1$ for strokes with $height < width$. For vertical strokes, the value of $C(s)$ will be negative. To avoid abnormal scaling of the values of this feature, $C(s)$ was set to zero, if its computed value is negative. For an n-stroke pattern, the *Shirorekha Confidence* is computed as the maximum value for C among all its component strokes.

5. *Stroke Density*: This is the number of strokes per unit length (x-axis) of the pattern. Note that the Han script is written using short strokes, while Roman and Cyrillic are written using longer strokes.

$$\text{Stroke Density} = \frac{n}{width(pattern)},$$

where n is the number of strokes in the pattern. The value of *Stroke Density* is a real number and can vary within the range $(0.0, R_1)$, where R_1 is a positive real number. In

(a)

1, 51.75, 0.15, 0.036, 0.023, 2.27, 0, 0.5, −1.0, 3.0, 1891.69

(b)

Fig. 8. (a) The strokes of the word "page." Each stroke is shown in a different color. (b) The feature vector extracted from it is shown.

TABLE 2
Error Rate of the *k*-Nearest Neighbor Classifier with Different Values of *k*

k	1	3	5	7	9	11
Error	17.45%	15.93%	15.36%	15.28%	15.18%	15.20%

practice, the value of R_1 was observed to be 0.5, with a mean of 0.031 and a standard deviation of 0.029.

6. *Aspect Ratio*: This is the ratio of the width to the height of a pattern.

$$AspectRatio(pattern) = \frac{width(pattern)}{height(pattern)}.$$

The value of *Aspect Ratio* is a real number and can vary within the range $(0.0, R_2)$, where R_2 is a positive real number. In practice, the value of R_2 was observed to be 5.0. Note that this feature is more meaningful for word-level classification than for the classification of a complete line.

7. *Reverse Distance*: This is the distance by which the pen moves in the direction opposite to the normal writing direction. The normal writing direction is different for different scripts as we noted at the beginning of this section. The value of *Reverse Distance* is a nonnegative integer and its observed values were in the range $[0, 1200]$.

8. *Average Horizontal Stroke Direction*: Horizontal Stroke Direction (HD) of a stroke, s, can be understood as the horizontal direction from the start of the stroke to its end. Formally, we define $HD(s)$ as:

$$HD(s) = \begin{cases} +1 & X_{pen-down}(s) < X_{pen-up}(s) \\ -1 & \text{otherwise,} \end{cases}$$

where $X_{pen-down}(.)$ and $X_{pen-up}(.)$ are the x-coordinates of the pen-down and pen-up positions, respectively. For an n-stroke pattern, the Average Horizontal Stroke Direction is computed as the average of the HD values of its component strokes. The value of *Average Horizontal Stroke Direction* falls in the range $[-1.0, 1.0]$.

9. *Average Vertical Stroke Direction*: It is defined similar to the Average Horizontal Stroke Direction. The Vertical Direction (VD) of a single stroke s is defined as:

$$VD(s) = \begin{cases} +1 & Y_{pen-down}(s) < Y_{pen-up}(s) \\ -1 & \text{otherwise,} \end{cases}$$

where $Y_{pen-down}(.)$ and $Y_{pen-up}(.)$ are the y-coordinates of the pen-down and pen-up positions, respectively. For an n-stroke pattern, the Average Vertical Stroke Direction is computed as the average of the VD values of its component strokes. The value of *Average Vertical Stroke Direction* falls in the range $[-1.0, 1.0]$.

10. *Vertical Interstroke Direction (VID)*: The Vertical Interstroke Direction is defined as:

$$VID = \sum_{i=1}^{n-1} dir(i, i+1),$$

where

$$dir(i, j) = \begin{cases} +1 & \bar{Y}(stroke_i) < \bar{Y}(stroke_j) \\ -1 & \text{otherwise.} \end{cases}$$

$\bar{Y}(s)$ is the average of the y-coordinates of the stroke points and n is the number of strokes in the pattern. The value of *VID* is an integer and falls in the range $(1-n, n-1)$.

11. *Variance of Stroke Length*: This is the variance in sample lengths of individual strokes within a pattern. The value is of *Variance of Stroke Length* is a nonnegative integer and its observed value was between 0 and $250,000$ in our experiments.

TABLE 3
The Error Rates for Different Classifiers with 5-Fold Cross-Validation Using 11 Features

Classifier	Remarks	Error Rate
Nearest Neighbor	No Normalization	35.8 %
Nearest Neighbor	Normalized Features	17.6 %
5-Nearest Neighbor	Normalized Features	15.4 %
Bayes Quadratic	Gaussian with Full Covariance	22.9 %
Mixture of Gaussian Distr.	Diagonal Covariance	25.5 %
Decision Tree	C5.0	16.1 %
Neural Network	One hidden layer with 25 Nodes	14.0 %
SVM	RBF Kernel	13.5 %

Fig. 8 shows the word "page" and the feature vector extracted from it.

4 Classifier Design

Experiments were conducted with different classifiers to determine the one which performs the best for the problem in hand. Here, we describe the details of each of the classifiers employed in our experiments.

1. *k-Nearest Neighbor (KNN) classifier*: The distance between two feature vectors was computed using the Euclidean distance metric. Since the features computed differ drastically in their range of values (see Section 3), a linear

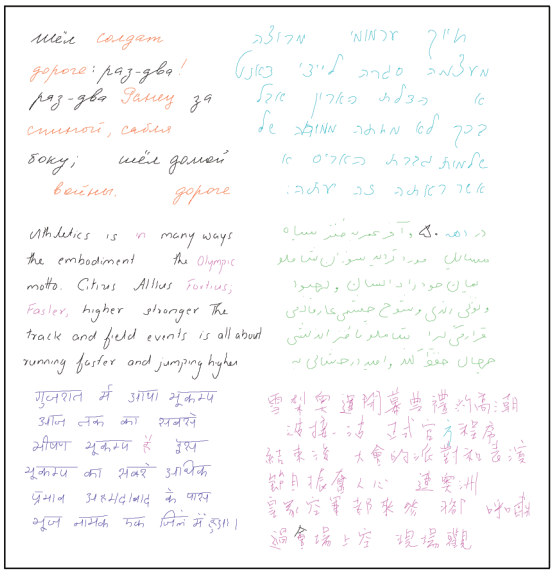


Fig. 9. The figures shows the classification results of the document page in Fig. 1. The color scheme to denote each script is the same as that used in Fig. 1.

transformation was applied to every feature to make their mean zero, and variance unity, in the training set. This transformation is referred to as *normalization* in the remainder of this paper. Experiments were conducted with different values of k, ranging from 1 to 11. Table 2 summarizes the results of the experiment.

2. *Bayes Quadratic classifier*: The distribution of the feature vectors for each of the classes was assumed to be Gaussian, and the mean and the covariance matrix were estimated from the training data. A Bayesian classifier, with equal priors was used to classify the test patterns into one of the six script classes. The features were normalized as described before.
3. *Bayesian classifier with Mixture of Gaussian Densities*: The densities of each of the six classes were assumed to be a mixture of Gaussians. The number of Gaussians within a mixture and their parameters were estimated using the EM algorithm [22]. (four to six Gaussians per script). The classifier itself is similar to the *Bayes Quadratic* classifier. The features were normalized.
4. *Decision Tree-based classifier*: The decision tree-based classifier partitions the feature space into a number of subregions by splitting the feature space, using one feature at a time (axis-parallel splits). The regions are split until each subregion contains patterns of only one class with a small number of possible outliers. The decision tree classifiers had around 800 nodes in the tree. We used the *C5.0* package to train and test the decision tree classifier.
5. *Neural Network-based classifier*: We used a three-layer neural network (one hidden layer) for the script classifier. The input layer contains 11 nodes, corresponding to each of the features in the feature vector. The output layer contains six nodes, corresponding to the six script classes. The number of nodes in the hidden layer was experimentally determined as 25. Increasing the number of hidden nodes above 25 gave little improvement in classification accuracy. During the training phase, the desired output for the node corresponding to the label of the pattern is set to 1, and all other node outputs are set to 0. When a test pattern is fed to a trained neural net, the class corresponding to output node with highest output value is determined to be the result of the classification.
6. *Support Vector Machine (SVM)*: Support vector machines map an n-dimensional feature vector to an m-dimensional feature space ($m > n$), with the assumption that the patterns belonging to different classes are linearly separable in the m-dimensional feature space. The mapping is implicitly defined by a *kernel function*. We used the SVM-Torch package to conduct the experiments and used the linear, polynomial and radial basis function (RBF) kernels. The classifier using RBF kernel performed best among the three.

Table 3 compares the performance of different classifiers with all the 11 features described in Section 3. The data set containing $13,379$ words was randomly divided into five (approximately equal) groups and a 5-fold cross validation was done for each of the classifiers. The error rates reported are the averages of these five trials. We notice that the *KNN* ($k = 5$), neural net, and *SVM*-based classifiers give similar performances.

It is difficult to compare these results with any of the reported results in the existing literature, as the script identification problem for online documents has not been attempted before. In the case of offline documents, Hochberg et al. [18] reported an overall accuracy of 88 percent for script identification at the page level, using six different scripts. Our algorithm classifies all of the 108 test pages correctly. However, one should note that the offline and online script identification problems involve very different challenges. The above comparison is made only for the purpose of understanding the relative complexities of the two similar looking problems. Fig. 9 shows the output of script recognition of the document in Fig. 1.

4.1 Combining Multiple Classifiers

The results of the different classifiers may be combined to obtain better classification accuracy. The results can be combined at different stages in the classification process. We have used a confidence level fusion technique where each classifier generates a confidence score for each of the six scripts. The confidence score is a number in the range $[0, 1]$, where 0 indicates that the test pattern is least likely to be of the script associated with the score, while a confidence score of 1 indicates that the test pattern is most likely to be the corresponding script. The confidence scores generated by the individual classifiers are summed and normalized to the range $[0, 1]$ to generate the final confidence score. The script which has the highest score is selected as the true class. In our experiments, we combined the results obtained from *SVM*, *KNN* (k = 5), and neural network classifiers. For SVM classifier, the confidence was generated from the output of the individual (two class) classifiers. The confidence score for the KNN classifier was computed as the proportion of neighbors which belong to the decided class. The output value of the node corresponding to the decided class gives the confidence value for the neural net-based classifier. The combined classifier could attain an accuracy of 87.1 percent on 5-fold cross-validation. The standard deviation of error over the cross-validation runs was 0.3 percent. Table 4 gives the confusion matrix for the combined script classifier which discriminates individual text lines.

5 Feature Selection

An interesting question to ask in any classification system is, how good are the available features, for the purpose of classification. A very effective, although suboptimal, way to determine the best subset of features for classification, given a particular classifier, is the sequential floating search method (SFSM) [23], [24]. The features described in Section 3 are ordered according to their contribution to classification accuracy. The plot in Fig. 10 shows the increase in performance as each feature is added by the SFSM. None of the features were eliminated during the selection process even though removing the features 10 and 11 does not considerably degrade the performance of the classifier.

In addition to classification using the features described in Section 3, experiments with dimensionality reduction using PCA and LDA techniques were conducted. The application of PCA reduced the classification accuracy to 78 percent. This is probably due to the difference in the scales of various features. Applying LDA resulted in approximately the same classification accuracy. However, LDA was able to attain this using only eight extracted features instead of the 11 input features.

6 Classification of Contiguous Words and Text Lines

In many practical applications, contiguous words belonging to the same script are available for classification. We expect that the script recognition accuracy will improve as the number of consecutive words of text in a test sample increases. In the case of online script recognition, this boils down to the number of words of text that is required to make an accurate prediction. The plot in Fig. 11 shows the increase in accuracy of the combined classifier as a function of the number of words in a test sample. A set of words was considered as a single pattern for classification in this case. We notice that with five words, we can make a highly accurate (95 percent) classification of the script of the text. The script classification accuracy improves to 95.5 percent when we use an

TABLE 4
Confusion Matrix of the Combined Script Classifier for Text Lines

	Classified					
True	Arabic	Devnagari	Han	Roman	Cyrillic	Hebrew
Arabic	204			1		4
Devnagari		373	1	7		1
Han			276	5	1	
Roman		3	1	696	23	
Cyrillic	1	3	1	39	230	1
Hebrew	5					279

Overall accuracy was 95.5 percent on 2,155 individual text lines.

entire text line, consisting of an average of seven words. The error in accuracy estimate is about 1 percent as indicated by a standard deviation of 0.5 percent. The accuracy of prediction of the script of a single word also depends on the length of the word. A measure of word length, which can be employed in the case of online data, is the number of strokes in the word. The classification improves considerably as the number of strokes in the word increases (up to 89 percent for 5-stroke words). These results give us an indication of the improvement in performance as the amount of data increases (evidence accumulation).

7 Conclusions and Future Work

We have presented a script identification algorithm to recognize six major scripts in an online document. The aim is to facilitate text recognition and to allow script-based retrieval of online handwritten documents. The classification is done at the word level, which allows us to detect individual words of a particular script present within the text of another script. The classification accuracies reported here are much higher than those reported in the case of script identification of offline handwritten documents, although the reader should bear in mind that the complexities of the two problems are different.

One of the main areas of improvement in the above algorithm is to develop a method for accurately identifying text lines and words in a document. We are currently working on developing statistical methods for robust segmentation of online documents. The script classification algorithm can also be extended to do page segmentation, when different regions of the handwritten text are in different scripts.

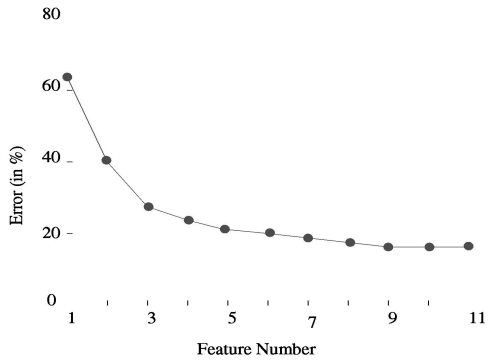


Fig. 10. Classification performance as individual features are added by the sequential floating search method.

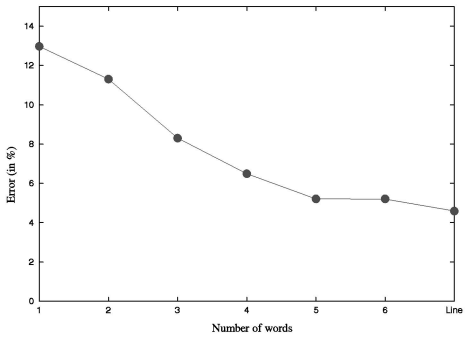


Fig. 11. Classification performance with increasing number of words in a test sample.

Acknowledgments

This research was supported by the IBM University Partnership Program.

References

[1] A History of PDAs, http://www.pdawear.com/news/article_pda_beginning.htm, 2003.
[2] *Pen Computing Magazine: PenWindows,* http://www.pencomputing.com/PenWindows/index.html, 2003.
[3] Smart Technologies Inc. Homepage, http://www.smarttech.com/, 2003.
[4] IBM ThinkPad TransNote, http://www-132.ibm.com/content/search/transnote.html, 2003.
[5] Windows XP Tablet PC Edition Homepage, http://www.microsoft.com/windowsxp/tabletpc/default.asp, 2003.
[6] J.J. Lee and J.H. Kim, "A Unified Network-Based Approach for Online Recognition of Multi-Lingual Cursive Handwritings," *Proc. Fifth Int'l Workshop Frontiers in Handwriting Recognition,* pp. 393-397, Sept. 1996.
[7] F. Coulmas, *The Blackwell Encyclopedia of Writing Systems.* Malden, Mass.: Blackwell Publishers, 1999.
[8] H. Jensen, *Sign, Symbol, and Script: An Account of Man's Effort to Write.* third ed. London: George Allen and Unwin, 1970.
[9] L.K. Lo AncientScripts.com, http://www.ancientscripts.com/, 2003.
[10] A. Nakanishi, *Writing Systems of the World.* Tokyo: Charles E. Tuttle Company, 1999.
[11] J. Hochberg, P. Kelly, T. Thomas, and L. Kerns, "Automatic Script Identification from Document Images Using Cluster-Based Templates," *Proc. Third Int'l Conf. Document Analysis and Recognition,* pp. 378-381, Aug. 1995.
[12] A.L. Spitz, "Determination of the Script and Language Content of Document Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 3, pp. 235-245, Mar. 1997.
[13] A.K. Jain and Y. Zhong, "Page Segmentation Using Texture Analysis," *Pattern Recognition,* vol. 29, pp. 743-770, May 1996.
[14] U. Pal and B.B. Chaudhuri, "Script Line Separation from Indian Multi-Script Documents," *Proc. Fifth Int'l Conf. Document Analysis and Recognition,* Sept. 1999.
[15] C.Y. Suen, S. Bergler, N. Nobile, B. Waked, C.P. Nadal, and A. Bloch, "Categorizing Document Images Into Script and Language Classes," *Proc. Int'l Conf. Advances in Pattern Recognition,* pp. 297-306, Nov. 1998.
[16] C.L. Tan, P.Y. Leong, and S. He, "Language Identification in Multilingual Documents," *Proc. Int'l Symp. Intelligent Multimedia and Distance Education,* Aug. 1999.
[17] G.S. Peake and T.N. Tan, "Script and Language Identification from Document Images," *Proc. Third Asian Conf. Computer Vision,* pp. 96-104, Jan. 1998.
[18] J. Hochberg, K. Bowers, M. Cannon, and P. Kelly, "Script and Language Identification for Handwritten Document Images," *Int'l J. Document Analysis and Recognition,* vol. 2, pp. 45-52, Feb. 1999.
[19] IBM Pen Technologies, http://www.research.ibm.com/handwriting/, 2003.
[20] E.H. Ratzlaff, "Inter-Line Distance Estimation and Text Line Extraction for Unconstrained Online Handwriting," *Proc. Seventh Int'l Workshop Frontiers in Handwriting Recognition,* Sept. 2000.
[21] The Art of Arabic Calligraphy, http://www.sakkal.com/ArtArabicCalligraphy.html, 2003.
[22] M.T. Figueiredo and A.K. Jain, "Unsupervised Selection and Estimation of Finite Mixture Models," *Proc. 15th Int'l Conf. Pattern Recognition,* pp. 87-90, Sept. 2000.
[23] A.K. Jain and D. Zongker, "Feature-Selection: Evaluation, Application, and Small Sample Performance," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 19, no. 2, pp. 153-158, Feb. 1997.
[24] R. Duda, P. Hart, and D. Stork, *Pattern Classifcation and Scene Analysis.* second ed. New York: John Wiley and Sons, 2001.