IDIAP RESEARCH REPORT

IDIAP

Martigny - Valais - Suisse

Off-Line Cursive Script Recognition Based on Continuous Density HMM

Alessandro Vinciarelli [a] Juergen Luettin [a]

IDIAP–RR 99-25

December 1999

PUBLISHED IN

Proceedings of 7^th International Workshop On Frontiers in Handwriting Recognition, pp.493-498, September 2000.

Dalle Molle Institute for Perceptual Artificial Intelligence • P.O.Box 592 • Martigny • Valais • Switzerland

phone +41 − 27 − 721 77 11
fax +41 − 27 − 721 77 12
e-mail secretariat@idiap.ch
internet http://www.idiap.ch

[a] IDIAP

IDIAP Research Report 99-25

Off-Line Cursive Script Recognition Based on Continuous Density HMM

Alessandro Vinciarelli Juergen Luettin

December 1999

published in

Proceedings of 7^th International Workshop On Frontiers in Handwriting Recognition, pp.493-498, September 2000.

Abstract. A system for off-line cursive script recognition is presented. A new normalization technique (based on statistical methods) to compensate for the variability of writing style is described. The key problem of segmentation is avoided by applying a sliding window on the handwritten words. A feature vector is extracted from each frame isolated by the window. The feature vectors are used as observations in letter-oriented continuous density HMMs that perform the recognition. Feature extraction and modeling techniques are illustrated. In order to allow the comparison of the results, the system has been trained and tested using the same data and experimental conditions as in other published works. The performance of the system is evaluated in terms of character and word (with and without lexicon) recognition rate. Results comparable to those of more complex systems have been achieved.

1 Introduction

The off-line cursive script recognition (CSR) problem has been deeply studied in the last ten years. Although the range of proposed methods is very wide, these can be classified depending on two key properties: the size and nature of the lexicon involved, and wheter or not a segmentation stage is present [14].
The lexicon is related to the application that involves the recognizer. A system that reads town names in postal addresses must cope with lexica of size between 10 and 1000 words (the actual size is determined, for each word, by the ambiguity on the recognition of the last one, two or three digits of the corresponding zip code) [2][4][7][8]. A check amount recognizer needs a small dictionary of legal amounts (amounts in letters) [6][11]. In both cases above, the data are produced by many writers with different writing styles. The development of applications for transcription of personal notes, historical documents or manuscripts involves very large lexica (in principle the whole dictionary of the language of the documents) and data often produced by only a single writer [1][9][13].
The segmentation consists in finding the ligatures between the single characters in a word so that each one of them can be separately recognized. Such task is difficult and error prone, since a character cannot be recognized before having been segmented, but cannot be segmented before having been recognized. This is referred as the Sayre's paradox [14].
The use of a sliding window makes the segmentation unnecessary: by sliding the window column by column from left to right, frames of fixed width are extracted from the image. From each frame, a feature vector is extracted and the sequence of these observations so obtained is used as the representation of the word. For each word in the lexicon, a HMM is created by concatenating single letter models. The use of letter models makes the system flexible with respect to changes in dictionary, and makes the use of large lexica possible since it does'nt require training examples of each word.
The paper is organized as follows: section 2 presents the preprocessing and normalization techniques, sections 3 and 4 describe the feature extraction process and the HMM based recognition, respectively, and the final section 5 illustrates results and conlusions.

2 Preprocessing and normalization

The preprocessing works directly on the raw data and its task is the processing of the input images to a form suitable for the recognition process. In our case, the data consist of handwritten word gray level images scanned at 300 dpi (see figure 1a) and the only necessary preprocessing operation is a binarization performed with the Otsu method [5].
The normalization step aims to correct global characteristics of the word introduced by the writing style, in particular, slant and slope. We developed a new normalization technique that is completely adaptive and does not use any heuristic parameter.
A horizontal density (number of foreground pixels in a row) threshold t is calculated with the Otsu method to distinguish between rows in the core region (the character body area), that are expected to have higher density, and rows in the ascender or descender regions, that are expected to have lower density. A row belongs to the core region if its horizontal density is above t and to the other regions otherwise. The procedure is illustrated in figure 1b. However, as can be seen in the same figure, long horizontal strokes in the ascenders or descenders of a word can create areas with density above the threshold that can be confused with the core region. To find the actual core region, the number of foreground pixels in each area with density above the threshold is calculated. The area that contains most pixels is considered as core region.
Most of the methods to find the reference lines presented in the literature are based on the horizontal density histogram analysis. Maxima, minima and first derivative peaks are used to distinguish between different regions. Such features are sensitive to local strokes and can determine noise that must be eliminated by heuristic rules. The Otsu method avoids this problem because the calculation of the threshold t is performed using the density distribution, so that local strokes become statistically not

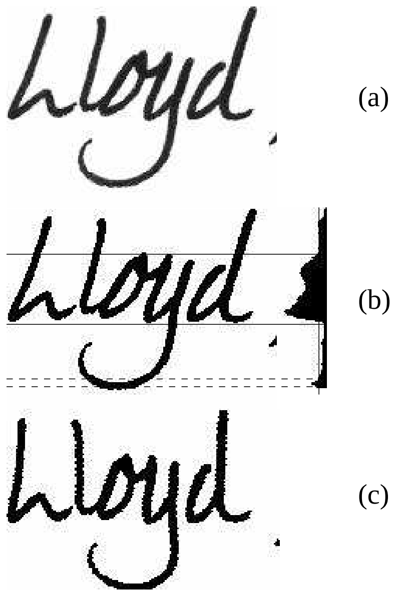

Figure 1: Preprocessing. The original image (a) is binarized. The lines in (b) enclose areas where the horizontal density values are above the threshold. The lowest area is a false candidate for core region. After having found the actual core region, the image is desloped and deslanted (c).

relevant. The only heuristic needed is the simple one above described.
The distribution of the density values is also used in [3] to deslope the image: a horizontal density histogram is calculated in many directions and for each one, the entropy $E = -\sum_{i=1}^{N} p_i \log p_i$, where p_i is the frequence of the density value i and N is the total number of density values represented, is calculated. When the slope of the direction of the histogram is close to the slope of the word, the projection of the core region is compact and not smooth, then fewer p_i will be significantly different than 0. This makes the entropy low and, when its minimum is reached, the slope is found.
In our work, to estimate the line on which the writer implicitly alignes the word (usually called *lower base-line*), the minima of the lower contour are used. Since minima belonging to descenders must not be used in such operation, the average distance between the detected minima and the core region limit is calculated. The minima that have a distance higher than the average are discarded. The estimate of the lower base-line is then obtained with the least square method. To eliminate the slope, the image is rotated until the estimated lower base-line is horizontal.
The slant correction technique is applied to the desloped image. While most other techniques for slant correction estimate the slant by averaging over the direction of near vertical strokes, our approach is based on a function S_α that gives a measure of the slant absence across the word. The calculation of such function rely on the vertical density histogram that is easier to be obtained than the direction of the strokes. For each angle α in a reasonable interval (in our case $[-15^o, 15^o]$), a shear transform t_α is applied and the following histogram is calculated:

$$H_\alpha(m) = \frac{h_\alpha(m)}{\Delta y(m)} \qquad 0 \leq m < nCol \tag{1}$$

where $h_\alpha(m)$ is the value of the vertical density histogram of the image shear transformed by the angle α, $\Delta y(m)$ is the difference between the maximum and minimum y coordinates of the foreground pixels in the m column and $nCol$ is the number of columns in the image. The number of foreground pixels of each column is divided by the distance between the highest and the lowest pixel giving $H_\alpha(m) = 1$, if the column contains a continous stroke, and $H_\alpha(m) \in [0, 1[$ otherwise.

Then, the following quantity is computed:

$$S(\alpha) = \sum_{\{i:H_\alpha(i)=1\}} h_\alpha(i)^2 \quad (2)$$

The value $\hat{\alpha}$ for wich $S(\alpha)$ is maximum, is assumed as slant estimate and the corresponding shear trasform $t_{\hat{\alpha}}$, when applied on the desloped original image, gives the deslanted image. The result of the process is displayed in figure 1c.

3 Feature extraction

When moving across the image, the sliding window isolates $nCol - width$ frames ($width$ is the width of the window).
The feature set used is based on the distribution of pixels across the frame. The window blindly isolates the frames and their content is, in many cases, meaningless. To avoid the noise due to high variability in shapes, a flou, but robust with respect to noise, representation has been adopted. The dimension of the feature vectors is low (see below) and this is an advantage when few training material is available.
Each word is processed separately and the frames have the same height as the bounding box of the word image. If in a word there are ascenders (and not descenders), the core region will be located at the bottom of the frame, viceversa if there are descenders (and not ascenders). If both ascenders and descenders are present, the core region will be located at the center of the window. This strongly affects the distribution of pixels across the frame and might create noise in the features. To avoid this problem, at each window position, the feature extraction process is applied only to the area that actually contains foreground pixels (As alternative solution, a fixed height window centered on the core region could be used, but this approach is probably more suitable when working on lines of handwritten text instead of single words).
Such area is partitioned into 16 non overlapping cells (arranged in a 4×4 grid), then the mumber n_i of foreground pixels in each cell i is calculated. The vector:

$$\mathbf{f} = (f_1, f_2, \ldots, f_{16})^T \quad (3)$$

where

$$f_i = \frac{n_i}{\sum_{j=1}^{16} n_j} \quad (4)$$

represents the feature vector. This low level feature set is very robust and its implementation needs little effort.

4 HMM based word recognition

The limited space does not allow an exhaustive presentation of Hidden Markov Models, for a good introduction, see [12].
A HMM has been trained for each letter and the model for a word is obtained by concatenation of single letter models. Since very few examples of capital letters are present in the database, only small letters have been modeled and the capital letters are labeled as small ones. In general, only the first letter of a word is capital, so the recognition can still be performed using the other characters. In some case, the capital letter has the same appearance as the small one except for the size, then, since the features are robust with respect to changes in size, there is no difference, from the recognition point of view, between the two versions of the same character.
The models are trained by maximizing the likelihood (the conditional probability of the observation

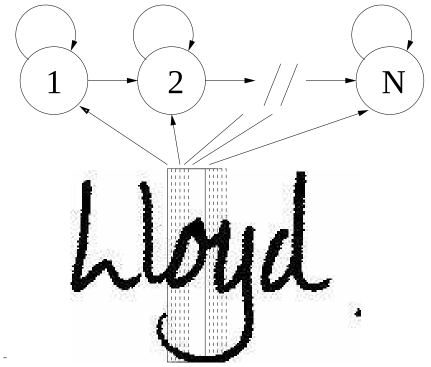

Figure 2: Markov modeling. The following frames are overlapped. The solid frame is the first position of the window on the letter, the dashed frames are the following ones. Each frame individuated by a window position belongs to a state in a letter model. In our system, $N = 9$ and the window width is 10. Only self transitions or transitions to the next state are allowed.

sequence, given the model). The technique used is known as Baum-Welch method or Expectation-Maximization (EM) [12].
The training is embedded, i.e. all the letters in a word model are trained at the same moment. The training initialization needs to know only the sequence of the letters, not their exact position in the example, and this is a great advantage because it avoids the problem of labeling each observation in the training set with the letter and position. Such task (that is needed in segmentation based approaches) would be long, time consuming and often error prone.
The number of states in the model is the same for all letters. The topology is left-right, allowing only self-transitions or transitions to the next state:

$$\begin{array}{ll} a_{ij} \geq 0 & \text{for} \quad j = i, j = i+1 \\ a_{ij} = 0 & \text{otherwise} \end{array} \tag{5}$$

In segmentation based approaches, the observations are feature vectors extracted from strokes that are supposed to be characters or parts of them. Such vectors can be discrete and belong to a finite set of observations, but even when they are continuous, they are expected to form clusters that are related to the segmented elements they are extracted from. A Vector Quantization can then reduce the observations to a sequence of discrete symbols.
When using a sliding window, the observations are extracted from parts of the word that are blindly isolated, they are so variable, that a reduction to a finite set of symbols seems difficult. In our case the features are expected to follow a continuous distribution and are modeled with mixtures of gaussians. Our motivation to this approach is as follows: the observations corresponding to the left and right part of the window are affected by noise since they are extracted from frames containing parts of the neighboring letters. This might make recognition more robust to letters written in different context. On the other hand, the observations extracted from frames in the central part of the window always represent the character that is modeled. This section will therefore contain most discriminant information in the likelihood calculation.

5 Results and conclusions

For training and testing, we used a database collected by Senior and Robinson[13] that is publicly available on the web[1]. The data consist of 4053 words extracted from a text that belongs to the LOB

[1] ftp://svr-ftp.eng.cam.ac.uk/pub/data

Corpus and has been written by a single person.
The experimental conditions described in [13] have been reproduced: the database has been divided in training, validation and test set (2360, 675 and 1016 words respectively). The lexicon size is 1334. The performance has been measured in terms of correctly classified letters and words (with and without lexicon). When using a lexicon, the classification is constrained to those letter combinations that form words present in the lexicon (model discriminant approach). To recognize words without lexicon, a model composed by all letter models is built. The initial probability distribution is uniform across the first states of each letter model. When the final state of a letter model is reached, a transition to the first state of any letter model is permitted. The most likely path (path discriminant approach) is found by the Viterbi algorithm. If the sequence of the states corresponds to the handwritten word, then the classification is correct.
Several systems have been trained and tested to find the optimal configuration. State numbers from 1 to 10 and windows 10, 12 and 16 pixel wide have been used. The observations have been modeled with mixtures of gaussians with 1, 2 and 4 components. The best results have been achieved by a system with 9-state letter models, a window 10 pixel wide and a gaussian mixture with 2 components. The performance is reported in table 1.

The observation of the words uncorrectly classified, showed that horizontal strokes, in ascenders or descenders, wider than the character they belong to (see the lower part of the y in fig. 2) can cause noise in the frames of the neighboring letters. When such strokes are present it is possible to find empty rows between the base line (upper line) and the bottom (top) of the frame. The first empty rows below the base line and above the upper line are then detected and only the area between them is used to extract the features. The best performance for this technique has been achieved by a system with 9-state letter models, a window 10 pixel wide and a gaussian mixture with 1 component.

Table 1: Results. The first column reports the character recognition rate, the second one the word recognition rate without lexicon and the third one the word recognition rate with lexicon.

character(%)	word (%)	word+lexicon(%)
67.55	10.88	82.45

The best results presented by Senior and Robinson in [13] are 93.4% with lexicon and 58.9% without lexicon, no data are available at the character level. The system described in [13] is more complex than ours: the preprocessing involves a skeletonization process, each frame is coded by a 72-dimensional vector (including low level features, loop, junctions and other element detection, snake features). A recurrent neural network is used to compute the observation probability and language modeling techniques have been applied. Parts of characters have been modeled separately and then connected to form a letter model (multistate letter models). A non optimal configuration of this system (when no language models are included and letter models are not multistate) recognizes 83.7% of the words with lexicon (no data are available without lexicon). This non-optimal system by Senior and Robinson has performance close to our system.

Table 2: Results after eliminating noise caused by horizontal strokes in ascenders and descenders. The first column reports the character recognition rate, the second one the word recognition rate without lexicon and the third one the word recognition rate with lexicon.

character(%)	word (%)	word+lexicon(%)
71.78	15.90	83.57

A system similar to ours can be found in [9]. A sliding window with fixed height span the handwritten text line by line. Each frame is used as feature vector and a Principal Component Analysis has been used to reduce the dimensionality. For each word in the lexicon, a continuous densith HMM has been

built and the recognition is performed by the Viterbi Algorithm.
Continuous density HMMs are also used in [10], where a segmentation free system is used in conjunction with a segmentation based one in order to increase the performance. In the segmentation free system features are extracted from each column and a normalization is done so that each character is represented by 24 columns. The best matching between the words in the lexicon and the observation sequence is found with the Viterbi Algorithm. Different models have been created for upper and lower case letters. For each word, two models have been built, the first is obtained by concatenating the models of the upper case characters, the second by concatenating the upper case model for the first letter and the lower case models for the other letters (since the data used for experiments are town names, only these two possibilities are expected).
The work presented in this article is still ongoing and the reported results are not final. However the performance can be compared with that reported in other works. A new normalization technique has been developed and applied that avoids heuristics and that is based on statistical methods. A feature extraction process simple and easily implementable has been used and the observations have been modeled with continuous density HMM.
The use of relatively simple techniques leaves space for further significant improvements by refining all the processing steps.

References

[1] H. Bunke, M. Roth, and E. Schukat-Talamazzini. Off-line cursive handwriting recognition using hidden markov models. *Pattern Recognition*, 28(9):1399–1413, September 1995.

[2] W. Chen, P. Gader, and H. Shi. Lexicon-driven handwritten word recognition using optimal linear combinations of order statistics. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(1):77–82, January 1999.

[3] M. Cote', E. Lecolinet, M. Cheriet, and C. Y. Suen. Automatic reading of cursive scripts using a reading model and perceptual concepts - the PERCEPTO system. *International Journal of Document Analysis and Recognition*, 1(1):3–17, january 1998.

[4] A. El-Yacoubi, M. Gilloux, R. Sabourin, and C. Suen. An HMM-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(8):752–760, August 1999.

[5] R. M. Haralick and L. G. Shapiro. *Computer and Robot Vision*. Addison Wesley, USA, 1992.

[6] S. Knerr, E. Augustin, O. Baret, and D. Price. Hidden markov model based word recognition and its application to legal amount reading on french checks. *Computer Vision and Image Understanding*, 70(3):404–419, June 1998.

[7] A. Kundu, Y. He, and M. Che. Alternatives to variable duration HMM in handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1275–1280, November 1998.

[8] S. Madhvanath, G. Kim, and V. Govindaraju. Chaincode contour processing for handwritten word recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(9):928–932, September 1999.

[9] U. Marti and H. Bunke. Towards general cursive script recognition. In *Proceedings of Sixth Int. workshop on Frontiers in Handwriting Recognition*, pages 379–388, Korea, 1998.

[10] M. Mohamed and P. Gader. Handwritten word recognition using segmentation-free hidden markov modeling and segmentation-based dynamic programming techniques. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(5):548–554, May 1996.

[11] T. Paquet and Y. Lecourtier. Recognition of handwritten sentences using a restricted lexicon. *Pattern Recognition*, 26(3):391–407, 1993.

[12] L. Rabiner and B. H. Juang. *Fundamentals of Speech Processing*. Prentice Hall, Englewodd Cliffs, NJ, 1992.

[13] A. W. Senior and A. J. Robinson. An off-line cursive handwriting recognition system. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):309–321, March 1998.

[14] T. Steinherz, E. Rivlin, and N. Intrator. Off-line cursive script word recognition - a survey. *International Journal of Document Analysis and Recognition*, 2(2):1–33, February 1999.