HMM-based Offline Recognition of Handwritten Words Crossed out with Different Kinds of Strokes

L. Likforman-Sulem

ENST- Telecom ParisTech & CNRS LTCI, 46 rue Barrault, 75013 Paris, France
laurence.likforman@enst.fr

A. Vinciarelli

IDIAP Research Institute CP592, 1920 Martigny (Switzerland)
vincia@idiap.ch

Abstract

In this work, we investigate the recognition of words that have been crossed-out by the writers and are thus degraded. The degradation consists of one or more ink strokes that span the whole word length and simulate the signs that writers use to cross out the words. The simulated strokes are superimposed to the original clean word images. We considered two types of strokes: wave-trajectory strokes created with splines curves and line-trajectory strokes generated with the delta-lognormal model of rapid line movements. The experiments have been performed using a recognition system based on hidden Markov models and the results show that the performance decrease is moderate for single writer data and light strokes, but severe for multiple writer data.

1 Introduction

There is an increasing interest for the analysis of raw handwritten material: historical documents, literary manuscripts (such as author personal drafts), mail sent to companies, letters, etc. In some cases the material is of interest for philologists, historians or other experts in humanities. In other cases, the material is of interest for forensic experts that look for legally relevant evidences. In both cases, words crossed out by the writers using different kinds of strokes cannot be neglected and carry important information. For this reason, this work investigates the effect of strokes used for crossing out the words on the recognition rate of a system based on hidden Markov models (HMM).

In order to perform extensive tests, the crossing out strokes have been simulated by superimposing lines generated with two different approaches (see Figure 1) to clean word images: the first approach aims at simulating horizontal handwritten lines and it is based on the delta-lognormal model of rapid line movements. The second approach aims at simulating wave-like handwritten strokes and it is based on spline curves. The main advantage of using simulated strokes is that it is possible to compare the performances obtained over word images that are both clean and degraded after having been crossed out.

To our knowledge, the problem of recognizing crossed-out words has not been addressed so far in the literature. However, similar problems have been investigated as a form of preprocessing for digital images of documents: in [7] guidelines on bank checks are removed before the recognition; in electronic documents, zig-zag strokes are detected in order to erase the underlying graphical object [5]; large crosses appearing in author drafts may be removed by Graphical User Interface and Kalman filtering such as in [6].

In this work, there is no attempt to detect the crossing out strokes. The words are recognized as if they were clean and the experiments show to what extent the recognition system is robust with respect to the superimposed strokes. The results show that, at least in the case of single writer data where the strokes are not too heavy, the performance degradation is low enough to be tolerated in applications like information retrieval or text categorization, where performances are good even in presence of high error rates [12].

The rest of this paper is organized as follows: Section 2 presents the models used to synthesize the crossing out simulation, Section 3 describes the recognition system used in the experiments, Section 4 shows experiments and results, and Section 5 draws some conclusions.

2 Crossing-out Simulation

The crossing out is simulated using two approaches: the first superimposes to clean word images wave-trajectory strokes created with control points and spline curves. The second superimposes to clean word images horizontal lines generated with the delta-lognormal model

L1 L2 L3 W

Figure 1. Crossing-out strokes. The picture shows the different kinds of strokes superimposed to the words: L1-3 means 1 to 3 horizontal lines and W means wave-trajectory crossing-out stroke.

of rapid line movements (see Figure 1).

2.1 The Wave Model

The wave-trajectory strokes are synthesized within the so-called *core zone*, i.e. the area containing the character bodies, because the writers tend to write the crossing-out strokes in such area. The core zone position is searched using the profile-based method described in [2]. The average stroke width w is also searched from the run-length histogram of writing pixels.

The parameters used for synthesizing wave-trajectory strokes are shown in Table 1. A number n of points are regularly placed within the core zone, alternatively in the top, middle and bottom part. The (x, y) coordinates of these points are then randomly shifted according to a Gaussian law $\mathcal{N}(0, \sigma_s^2)$. The resulting points are the control points of the B-spline curve (see Fig. 2-left).

In addition, a local width is assigned to each control point. The spline curve is drawn through the control points and the local width on the curve is interpolated linearly from the two nearest control points (see Fig. 2-right).

In our experiments, we set $\sigma_s = 1.2$ and $\sigma_w = 1$. The number n of control points ranges from $n = 5$ to $n = 13$ according to word length. The value of the parameters has been set empirically to produce realistic images.

2.2 The Line Model

The line-trajectory strokes are synthesized again within the word core zone (see above). These strokes are produced as rapid line movements according to the delta-lognormal model proposed in [9][10]. The trajectories are circular elements characterized by the parameters shown in Table 1.

The magnitude $v(t)$ of the velocity along the trajectory results from the subtraction of two lognormal laws. These laws correspond to the input commands of the agonist and antagonist neuromuscular systems respectively. These input commands are of magnitudes D_1 and D_2 respectively. They are fed at time t_0 and the responses of each system occur with logtime delay μ_1 and μ_2, and logresponse time

Table 1. Model parameters for synthesizing crossing out strokes

Wave Model	
n	number of control points
y_{min} y_{max}	y core zone positions
σ_s	shift standard deviation
w	average stroke width
σ_w	stroke width standard deviation
Line Model	
P_0	initial position
θ_0	initial angle
C_0	curvature
D_1 D_2	magnitudes of input commands
t_0	initial time
μ_1 μ_2	global time delays
σ_1 σ_2	neuromuscular response time

σ_1 and σ_2 respectively. The resulting velocity profile is bell-shaped as shown in Fig. 3.

$$v(t) = \frac{D_1}{\sigma_1\sqrt{2\pi}(t-t_0)} e^{\frac{(-ln(t-t_0)-\mu_1)^2}{2\sigma_1^2}} - \frac{D_2}{\sigma_2\sqrt{2\pi}(t-t_0)} e^{\frac{(-ln(t-t_0)-\mu_2)^2}{2\sigma_2^2}}$$

The direction $\theta(t)$ of the velocity is :

$$\theta(t) = \theta_0 + C_0 \int_{t_0}^{t} v(u)du$$

where C_0 is the curvature and θ_0 the initial angle.

P_0 is the inital position of the trajectory and $D_1 - D_2$ corresponds to the trajectory length. From velocity magnitude $v(t)$, velocity direction $\theta(t)$ and initial position $P_0(x_0, y_0)$, trajectory points are obtained by calculating the integral of velocity components in the xy-plane as shown in eq. 1. Trajectory points are darker or lighter according to velocity, producing a more natural effect.

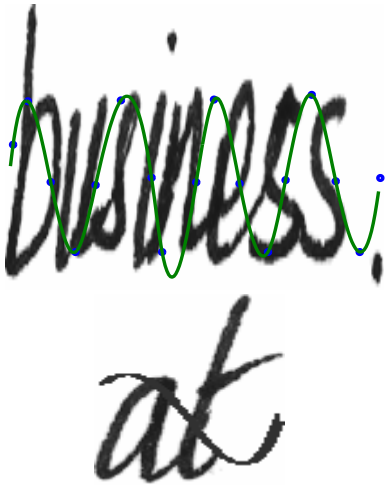

Figure 2. Wave model: control points and spline curve (top). Crossing-out stroke width interpolated between control points (bottom).

$$\begin{aligned} x(t) &= x_0 + \int_0^t v(\tau) cos\ \theta(\tau) d\tau \\ y(t) &= y_0 + \int_0^t v(\tau) cos\ \theta(\tau) d\tau \end{aligned} \quad (1)$$

We use the line model to superimpose crossing out strokes to word core zones. One to three strokes have been synthesized, increasing the degradation level. For synthesizing the strokes shown in Fig. 1, we set $D_1 = 544.6$, $D_2 = D_1 - word\ length$, $\mu_1 = -1.634$, $\sigma_1 = 0.448$, $\mu_2 = -1.333$, $\sigma_2 = 0.236$, $t_0 = 0.132$, $C_0 = 9.10^{-4}$ and $\theta_0 = \pi/20$.
These parameters provide quasi horizontal lines which span through the entire word. The timing parameters $(\mu_1, \sigma_1, \mu_2, \sigma_2)$ are close to those used in [3] and produce trajectory durations of less than 1 s.

3 The Recognition Approach

A full description of the system used in this work can be found in [13]. The system first binarizes the images of the words, then applies a normalization algorithm where slant and slope are removed (see [14] for details). The normalized word is converted into a sequence $X = (\vec{x}_1, \ldots, \vec{x}_T)$ of feature vectors using a *sliding window approach*: a window of predefined width shifts column by column from left to right and, at each position, the content of the window is converted into a vector through a feature extraction process. The feature extraction is as follows: the window content is split into 16 non-overlapping cells arranged in a 4×4 grid and the feature vector contains a feature for each cell. The i^{th} feature accounts for the percentage of foreground pixels in the window that lie in cell i.

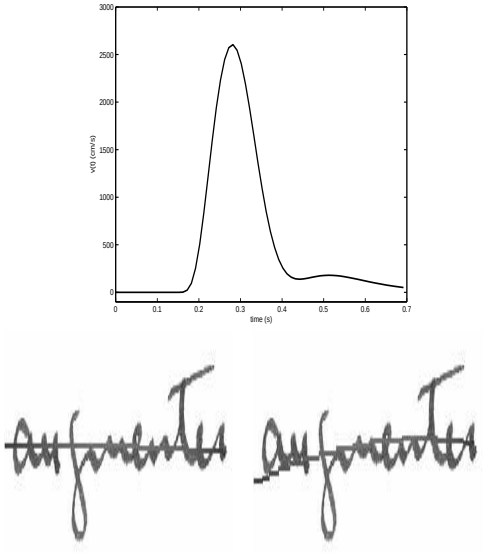

Figure 3. Line Model: Velocity profile (top). Synthesized Line Strokes (bottom).

Once the sequence X is given, the recognition process can be thought of as finding the word model $\hat{W}$ that maximizes the likelihood:

$$\hat{W} = \arg\max_{W \in L} p(X|W) \quad (2)$$

where L is the dictionary, i.e. the list of words that can be given as output of the recognition process. The likelihood $p(X|W)$ is estimated using a continuous density left-right hidden Markov model [4] where the emission probabilities are mixtures of Gaussians [1]. Each word in the dictionary corresponds to a model and each model is built by concatenating single letter models. The system requires the setting of two hyperparameters: the first is the number S of states in the letter models (the same for all models), the second is the number G of Gaussians in the mixtures (the same number for all states in all models).

4 Experiments and Results

This section presents the data used in our experiments, the details of the system training and the recognition results achieved over both clean and crossed-out data.

4.1 The data

The experiments of this work have been performed over two datasets that will be referred to as *Cambridge* and *Bern*. The Cambridge database is a collection of 4053 single writer words originally presented in [11]. The dataset has been split into training (2675 words) and test set (1378 words). The Bern database is a collection of 12198 hand-

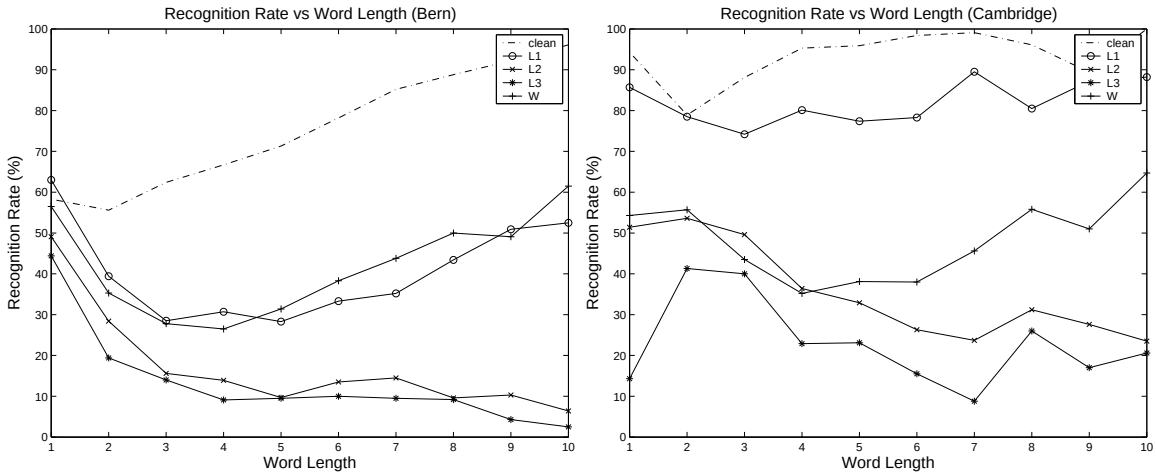

Figure 4. The plots report the recognition rate as a function of the word length for both Bern (left plot) and Cambridge (right plot) databases.

written words extracted from the IAM database [8], a collection of handwritten pages corresponding to the texts in the Brown Corpus. The data is multi-writer and the words used in this work have been segmented manually. The dataset is split into training (8013 words) and test set (4185 words).

4.2 HMM Training

The models are trained using the Baum-Welch algorithm, a technique that implements the Expectation Maximization method in the case of the HMMs. The training involves the setting of two hyperparameters (see Section 3), i.e. the number S of states in the letter models and the number G of Gaussians in the mixtures. The hyperparameters are set using the crossvalidation method: HMMs corresponding to all the pairs (S, G) with S and G belonging to a predefined range are trained over a subset of the training set and tested over the remaining part of the training set (the so-called *validation set*). The pair that leads to the best recognition results is retained as optimal. In this way the hyperparameters are set using data completely independent of the test set. In the case of the Cambridge database, $S \in [10, 14]$ and $G \in [10, 15]$, and the system selected during the crossvalidation corresponds to $S = 12$ and $G = 12$. In the case of the Bern database, $S \in [19, 23]$ and $G \in [10, 25]$, and the system selected during the crossvalidation corresponds to $S = 20$ and $G = 20$.

The models are trained only over clean material, but they are tested over both clean and noisy material, i.e. when the deletion strokes are both absent and present. The reason is that the goal of this work is to measure the performance degradation induced by the crossing out strokes and the robustness of the system.

Table 2. Recognition results. The table reports the recognition rate for the clean data and for the different kinds of deletion strokes.

database	clean	L1	L2	L3	W
Bern	70.1	34.6	16.2	12.5	35.0
Cambridge	91.9	79.9	39.0	27.2	45.7

4.3 Recognition Results

Table 2 reports the recognition results over both Bern and Cambridge databases. For the first dataset the lexicon size is 100 and for the second is 1000. The first column of the table reports the results obtained over the clean data, while the other columns report the results when different crossing out strokes are superimposed to the data. Columns L1, L2 and L3 correspond to strokes composed of one, two and three horizontal lines respectively (see Figure 1). Column W corresponds to a wave-like crossing out stroke. All the results are reported in terms of recognition rate, i.e. percentage of correctly recognized test samples. The maximum relative degradation is 70.1 percent for the Cambridge database, 82.2 percent for the Bern database and it is achieved in both cases when three horizontal deletion strokes are added. However, the single writer system is more robust when the degradation is less severe. In fact, the relative decrease when adding one horizontal line is around 50 percent for the Bern database, but it is just 13.0 percent for the Cambridge collection.

In both cases, the degradation induced by the wave-like crossing out strokes is lower than the one induced by L2 and L3. The reason is probably that the horizontal strokes produce noise in all characters, then for each letter there is a mismatch between training and test condi-

tions. On the contrary, in the case of the wave-like strokes some letters are touched only marginally and, for each word, there are some characters for which there is a better matching between training and test conditions.

Figure 4 shows the results as a function of the word length. In general, HMM-based systems tend to recognize better longer words because longer observation sequences match with less ambiguity the word models. This is what can be observed in the curves related to the recognition of clean words in Figure 4: the curve increases monotonically for the Bern database and shows a performance drop for the Cambridge database, but this is due to statistical fluctuations (words longer than six letters appear with low frequency in the Cambridge test set).

Once the crossing out strokes are superimposed, the behavior of the system is different. In fact, the length seems to help only for words with more than five letters and only for L1 and W. For the other kinds of crossing out strokes the performance decreases monotonically as the length of the words increases. The probable reason is that the likelihood of the paths in the trellis used for the Viterbi algorithm tends to become more uniform in presence of multiple deletion strokes. In fact, the mismatch between training and test conditions makes equally likely all the models and it is harder for the good model to have a likelihood higher than the others. In this way, when the number of paths increases, i.e. when the words are longer, the probability of passing through the right path decreases.

5 Conclusion

This work has presented experiments where an HMM-based offline word recognition system has been tested over images affected by deletion strokes. The reason is that in many application domains (see Section 1) the words deleted by the writers are important and it is necessary to recognize them. The results show that the system is robust with respect to the strokes only to a limited extent, in particular when the data is single-writer and when the strokes are not too heavy. In fact, in the case of single-writer data the performance of the system remains, for lighter deletion strokes, above or around 50 percent and this is enough for applications like information retrieval or text categorization [12]. However, in the case of multiple writer data the degradation is always too high and the system does not appear to be robust with respect to the noise.

Since many application domains involve only single writer data (e.g. author drafts), the approach proposed here can still be useful, especially when the recognition is just a step towards higher level applications like retrieval or categorization (see above). Future work can include the training of the models over words with deletion strokes to reduce the mismatch between training and test conditions, and a system capable of detecting the presence of deletion strokes and to adapt the models to the kind of detected deletion stroke.

References

[1] C. Bishop., *Pattern Recognition and Machine Learning*, Springer Verlag, 2006.

[2] M. Blumenstein, C. Cheng and X. Liu, "New preprocessing techniques for handwritten word recognition", *Proceedings of Vizualization, Imaging and Image Processing*, 2002, pp 480–484.

[3] M. Djioua and R. Plamondon, "The generation of velocity profiles with an artificial simulator", *International Journal of Pattern Recognition and Artificial Intelligence*, 18(7):1207–1219, 2004.

[4] F. Jelinek, *Satistical methods for speech recognition*, MIT Press, 1994.

[5] L. Julia and C. Faure, "Pattern recognition and beautification for a pen based interface", *Proceedings of the International Conference on Document Analysis and Recognition*, 1995, pp 58–61.

[6] E. Lecolinet, F. Role, L. Likforman-Sulem, J.-L. Lebrave and L. Robert, "An integrated reading and editing environment for scholarly research on literary works and their handwritten sources", *Proceedings of the ACM International Conference on Digital Libraries*, 1998, pp 144–151.

[7] K. Liu, C. Suen, M. Cheriet, J. Said, C. Nadal and Y. Tang, "Automatic extraction of baselines and data from check images", *International Journal of Pattern Recognition and Artificial Intelligence*, 11(4):675–697, 1997.

[8] U. Marti and H. Bunke, "The IAM-database: an English sentence database for offline handwriting recognition", *International Journal on Document Analysis and Recognition*, 5(1):39–46, 2002.

[9] R. Plamondon, "The generation of rapid human movements I, A delta-lognormal law", *Technical Report*, EPM/RT(93-4), 1993.

[10] R. Plamondon and W. Guerfali, "The generation of rapid human movements I, A delta-lognormal law", *Biological Cybernetics*, 78(93-4):119–132, 1998.

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

[12] A. Vinciarelli, "Noisy Text Categorization", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(12):1882–1895, 2005.

[13] 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*, 26(6):709–720, 2004.

[14] A. Vinciarelli and J. Lüettin, "A new normalization technique for cursive handwritten words", *Pattern Recognition Letters*, 22(9):1043–1050, 2001.