Writing Speed Normalization for On-Line Handwritten Text Recognition [*]

Moisés Pastor, Alejandro Toselli and Enrique Vidal
Institut Tecnològic d'Informàtica
Universitat Politècnica de València
Camí de Vera s/n 46071 València. Spain.
{moises,evidal,ahector}@iti.upv.es

Abstract

Pen-based interfaces aim at improving the man-machine interaction of many portable systems. While statistical models can be used to learn pen position sequences, they suffer from the huge variability exhibited by the speed of writing. To improve performance, invariance to the writing speed is needed. Trace segmentation is a technique that can be used to normalize the writing speed. This method is controlled by a parameter called resampling distance. A study of the resampling distance is presented here, along with an other approximation to the writing speed normalization called "derivatives normalization". The improvement using trace segmentation was 19.3% relative to the baseline, whilst the improvement using derivatives normalization was 47.3% relative.

1 Introduction

A critical feature of current mobile computing devices (e.g., personal assistants and communicators, laptops, etc.) is the user interface. Keyboards can not be reduced in size significantly before they become awkward to use; however, if they are not shrunk in size, they lose their portability. In this context, pen-based interfaces have become essential because of their simple size scalability.

Many continuous writing recognizers follow a statistical approach. In this approach, every stationary point sequence in the text is approximated by a statistical law, typically a mixture of Gaussian densities. Successive stationary regions on the other hand, are modeled by states of a suitable Hidden Markov Model (HMM). The parameters of these models are estimated from sample data using well-known techniques, such as Expectation-Maximization. Usually, not enough data is generally available to cope with the large variability exhibited by real data. Current works put special effort in reducing this variability through elaborate preprocessing methods to take the maximum advantage of the data available.

In on-line handwriting systems, the sample data is obtained from a temporal sampling of the pen position. For two writings of the same word, the point coordinates can change greatly depending on the writing speed. This speed has huge differences between writers, also for the very same writer and even within the same word. An approach to solve this problem, called *trace segmentation* consists in a resampling operation that redistribute data points (originally sampled in equal time intervals) to enforce even spacing between them. This way the word will have the same points at the same places independently of the speed the word was written.

Trace segmentation has been employed as a preprocessing phase for on-line handwriting recognition by several authors [8, 9, 6, 13, 12, 4, 7]. In [14], trace segmentation is used not only for speed invariance, but also to reduce the size of the samples and speed up the recognition time.

The main drawback of the *trace segmentation* is that introduces a new parameter, the *resampling distance* (distance between two consecutive points) which needs to be tuned. It is greatly desirable to have a criterion to choose the value of the resampling distance instead of setting it up empirically. Some works [3, 1, 2] calculate this resampling distance as a function of the core high of the word to recognize. However the module of derivatives, which explain the writing speed (see section 2) do not remains constant between different writings as the resampling distance depends on the size of the characters.

As an alternative to trace segmentation, we propose to normalize the derivatives by the their module. Derivatives explain both the direction and the speed of the trace. If the module of transformed derivatives becomes

[*]Work partially funded by *TIRIG* project. TIC 2003-08496-C

constant (equal to 1), the representation will be invariant to the writing speed, while keeping the direction information.

There are also on-line handwriting applications where it is not a good idea to perform the writing speed normalization as, for example, signature verification [11]. In this kind of application writing speed plays an important role in writing identification methods.

The paper is organized as follows. In section 2, two methods for writing speed normalization are explained: *trace segmentation* and *derivative normalization*. Section 3 describes the on-line handwritten word recognition system. Section 4 describes the on-line corpora used in the experimental phase. Experimental results are presented in section 5 and conclusions are drawn in the final section.

2 Writing Speed Normalization

2.1 Trace segmentation

Let $S = \{(x_0, y_0), (x_1, y_1)...(x_{n-1}, y_{n-1})\}$ be a sequence of data points (a stroke) sampled with a fixed time interval τ. *Trace segmentation* consists in resampling S using a fixed distance interval, α (rather than a fixed time interval τ). The coordinates of the resampled points are obtained by (linear) interpolation/decimation. A more detailed explanation of this procedure is given by the following algorithm:

```
1:  **In:** $S = \{(x_0, y_0), \cdots, (x_{n-1}, y_{n-1})\}, \alpha$
2:  **Out:** $\hat{S} = \{(\hat{x}_0, \hat{y}_0), \cdots, (\hat{x}_{n-1}, \hat{y}_{n-1})\}$
3:  $L_0 = 0$ // accumulated stroke length
4:  **for** $i = 1..(n-1)$ **do**
5:      $L_i = L_{i-1} + \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$
6:  $m = \lfloor L_{n-1}/\alpha \rfloor$ //number of out points
7:  $(\hat{x}_0, \hat{y}_0) = (x_0, y_0)$ //the first point
8:  j=1
9:  **for** $p = 1..(m-2)$ **do**
10:     **while** $L_j < p \cdot \alpha$ **do**
11:         j++
12:     $C = (p \cdot \alpha - L_{n-1})/(L_n - L_{n-1})$
13:     $\hat{x}_p = x_{n-1} + (x_n - x_{n-1}) \cdot C$
14:     $\hat{y}_p = y_{n-1} + (y_n - x_{n-1}) \cdot C$
15: $(\hat{x}_{m-1}, \hat{y}_{m-1}) = (x_{n-1}, y_{n-1})$ //the last point
16: return $\hat{S}$
```

For each point in the original writing sequence, the length of the path from the origin up to this point is calculated. The number of output points is the total stroke length divided by the *resampling distance* α. To exactly preserve the original length of a stroke, the first and last point are maintained in place. Linear interpolation is used to find the position of each new output point. Some examples of a tracesegmented word, are shown in figure 1.

An effect of *trace segmentation* is the normalization of derivatives (see Section 3.2). As distance between points remains equal for all the points, the module of derivatives (writing speed), M is constant too. This way, the range of the derivatives is normalized between 0 and M.

$$M = \sqrt{x'^2 + y'^2} \quad (1)$$
$$x' = \sqrt{M^2 - y'^2} \rightarrow x' \in [0, M] \quad (2)$$
$$y' = \sqrt{M^2 - x'^2} \rightarrow y' \in [0, M] \quad (3)$$

2.2 Derivatives Normalization

The main disadvantage of the *trace segmentation* method is that it introduces a new parameter which needs to be tuned. If the resampling distance, α, becomes large, the amount of data points decreases and some information might be lost. If α is too small (upsampling the data points), the amount of data points increases but the amount of relevant information remains constant.

Having this in mind, as an approximation to avoid tuning, the writing speed can be normalized, rather than using *trace segmentation*, by normalizing the derivatives by the derivative module in each point, as it is shown in equations 4. The module of normalized derivatives will be constant (equal to 1) and the system becomes invariant to the writing speed.

$$\hat{x}' = \frac{x'}{\sqrt{x'^2 + y'^2}} \qquad \hat{y}' = \frac{y'}{\sqrt{x'^2 + y'^2}} \quad (4)$$

where $\hat{x}', \hat{y}' \in [0..1]$

3 The on-line handwritten word recognition system

The on-line handwritten recognition system used here accepts as input a time ordered sequence of data points and outputs a sequence of words hypothesis. Three different modules take part in this system: preprocessing, feature extraction and recognizer module

3.1 Preprocessing

The on-line input data stream consists of a sequence of strokes. A stroke consist on a sequence of coordinates ordered in time (x_t, y_t). There are two kinds of strokes: pen-down strokes (also referred to as visible strokes) acquired with the digital pen touching the pad surface, and pen-up strokes acquired without touching it. In this work only pen-down strokes are considered.

The preprocessing of each sample involves four processes: repeated points elimination, noise reduction, size normalization and writing speed normalization (*trace segmentation*, if it is the case). Noise in handwritten strokes is due to erratic hand motions and inaccuracy of the digitalization process. In order to reduce noise, we employ a smoothing technique consisting in replacing every point (x_t, y_t) in the trajectory by the mean value of its neighbors [3].

It is important to remark that the temporal order of the data points is preserved throughout all preprocessing steps.

3.2 Feature extraction

Once the original coordinate sequences have been preprocessed, they are transformed into new temporal sequences of 6-dimensional real-valued feature vectors. The six features computed for each sample point are:

- **Normalized Vertical Position** y_{N_t}: each sample sequence, (x, y), are scaled and translated to obtain new y_{N_t} values in the range $[0, 100]$, preserving the original aspect-ratio of the sample.

- **First Derivatives** : x' and y' calculated using the method given in [15]:

$$x_t{}' = \frac{\sum_{i=1}^{2} i \cdot (x_{t+i} - x_{t-i})}{2 \cdot \sum_{i=1}^{2} i^2} \quad (5)$$

$$y_t{}' = \frac{\sum_{i=1}^{2} i \cdot (y_{t+i} - y_{t-i})}{2 \cdot \sum_{i=1}^{2} i^2} \quad (6)$$

  where the range of i defines the window size which determines the number of neighbor points involved in the computation.

- **Second derivatives**: x'' and y'' computed in the same way as the first derivatives but using x' and $y\prime$ instead of x and y ($\hat{x}'$ and $\hat{y}\prime$ in the case of *normalized derivatives*).

- **Curvature**: k_t, is the inverse of the radius of the curve in each point. The curvature is calculated as follows:

$$k_t = \frac{x' \cdot y'' - x'' \cdot y'}{\left(x'^2 + y'^2\right)^{3/2}} \quad (7)$$

The coordinate x is not used as a feature because of x range for different instances of the same character, can vary greatly depending on the position of the character into a word.

3.3 The recognition module

Hidden Markov Models (HMMs, see [10]) are used to estimate the probability for a sequence of feature vectors which characterize the stroke time evolution of a given handwritten text.

Each word to be recognized is represented as a sequence of characters where each character itself is modeled by a continuous left-to-right HMM. Basically, each character HMM is a stochastic finite-state device that models the succession, along time, of feature vectors computed for instances of this character.

Each HMM state generates feature vectors following an adequate parametric probabilistic law; typically, a *mixture of Gaussian densities*. The adequate number of densities in the mixture per state, as well as the number of HMM states, need to be tuned empirically and it may be conditioned by the available amount of training data.

Once an HMM *topology* (number of states, the transitions between them), and number of densities per state, has been adopted, the model parameters can be easily trained from samples of handwritten words, along with the transcription of these words, into the corresponding sequence of characters. This training process is carried out using a well known instance of the EM algorithm called *forward-backward* or *Baum-Welch reestimation* [5].

The recognition of an unknown sequence of feature vectors $\mathbf{f}_0^{m-1} = \{f_0 \dots f_{m-1}\}$ is formulated as the problem of finding a word $\hat{w}$, represented by a sequence of character HMMs, that maximizes:

$$\hat{w} = \arg\max_{w \in W} p(\mathbf{f}_0^{m-1} | w, \theta) P(w) \quad (8)$$

Where W is the set of words (dictionary) and θ is the trained set of HMM parameters. $P(w)$ is the a-priori probability of the word w. In this work all words have the same a-priory probability. The optimization problem (formula 8) can be solved using the well known *Viterbi* algorithm [5].

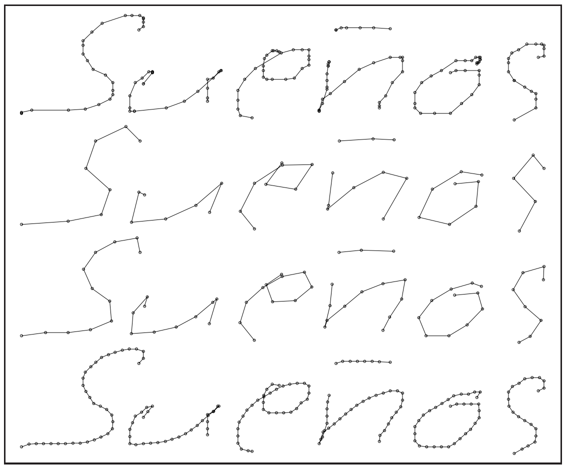

Figure 1. From top to bottom: original image, trace segmentation with resampling distance $\alpha = 60$, with $\alpha = 40$ and with $\alpha = 13$ (the best performer). Note: in the case of *Derivative Normalization* **approximation, points remain at their position, the image produced is similar to the original one.**

	State Number					
	8	10	12	14	16	18
Base	16.3	**15.0**	15.6	15.1	16.7	21.3
8	23.9	18.9	17.4	14.5	**14.4**	14.4
10	19.7	17.1	15.3	14.2	**13.8**	14.2
12	20.5	17.6	15.2	**14.5**	14.7	15.6
13	17.7	15.2	**12.1**	12.9	14.2	15.5
14	15.8	14.4	**12.4**	13.0	13.1	15.6
16	15.5	14.3	**14.0**	16.6	16.6	20.7
18	18.2	15.8	**14.5**	15.4	19.9	28.7
20	17.5	16.8	**14.7**	18.2	26.8	40.4
DN	9.3	**7.9**	8.9	11.8	16.3	19.9

Table 1. Classification error percentage for the baseline (Base), trace segmentation with several resampling distances, and for the derivatives normalization method(DN). Each column present the results for a given number of HMM states (SN). Best results for each row are typeset in boldface.

4 On-line handwritten corpora

An experimental on-line handwriting text corpora was acquired at the *Universitat Politècnica de València*, using a digital tablet which recorded pen locations at 400 dots per inch. The tablet had also a mechanism to detect whether the pen is touching the tablet surface or not, in order to detect pen-down and pen-up states respectively. This corpora comprises a set of isolated handwritten characters as well as a set of handwritten words, The later was used in the experiments described in the next section. The corpus is composed of 2964 on-line Spanish words handwritten by 19 different writers. 2028 randomly selected samples from 13 writers were used for training and 936 from the rest of the writers were left for testing.

5 Experiments and results

The performance of the system is shown in table 1. The baseline (*Base* labeled row) shows the results obtained without any kind of writing speed normalization. Rows labeled from 8 to 20 are the results for the corresponding resampling distances. In the row labeled *DN* (Normalized Derivatives), the performance for the alternative writing speed normalization method is presented.

The best result without writing speed normalization was 15% classification error. The improvement provided by *trace segmentation* is 19.3% relative. The best result obtained for *derivatives normalization* is 7.9% which represents an 47.3% of improvement respect to the baseline.

The *trace segmentation* method moves the points from their original locations. This would introduce some deformation in the characters. *Derivatives normalization* leave the points at their original locations while making the system invariant to the writing speed. According to the results it seems, that de original distribution of the points is important. Discriminant parts of characters are written in a lower speed, more accurately than others. That would mean that the density of points within a word is proportional to the importance of each part.

Trace segmentation exhibits stability with respect to the number of HMM states. The number of states depends on the number of stationary regions in the points sequence. If there are enough number of points (at least one per state), changes on the resampling distance do not affect to the HMM topology significantly. For the baseline, the best result was obtained for 10 states. There is no significant variation in error for 12 or 14 states. The best result for *derivatives normalization* was achieved for 10 states. This would mean that the variance intra-class has lessened, resulting in a smaller number of parameters to train.

6 Remarks and conclusions

A study of the influence of the resampling distance in *trace segmentation* is presented. The method improves the system performance by 19.3% relative. *Trace segmentation* showed to be robust with respect to the HMM topology. The drawback of this method is the introduction of a new parameter which needs to be tuned.

An other approximation to the writing speed normalization is tested, normalization of derivatives by the derivatives module. The improvement using this approximation was 47.3% relative. This method demonstrated to be better and do no introduces any new parameter to be tuned. The *derivatives normalization* seems to be a good normalization method.

References

[1] R. Gross. Run-On Recognition in an On-line Handwriting Recognition System. Technical report, University of Karlsruhe, Germany, June 1997.

[2] J. Hu, S. G. Lim, and M. K. Brown. HMM Based Writer Independent On-Line Handwritten Character and Word Recognition. In *The 6th International Workshop on Frontiers in Handwriting Recognition,*, KAIST Campus, Taejon city, Korea, Aug. 1998.

[3] S. Jaeger, S. Manke, J. Reichert, and A. Waibel. On-Line Handwriting Recognition: The NPen++ Recognizer. *International Journal on Document Analysis and Recognition*, 3(3):169–181, 2001.

[4] S. Jaeger, S. Manke, and A. Waibel. Npen++: an on-line handwriting recognition system. In *Proc. of the Seventh IWFHR*, pages 249–260, Amsterdam, The Netherlands, Sept. 2000.

[5] F. Jelinek. *Statistical Methods for Speech Recognition*. MIT Press, 1998.

[6] A. Kosmala, J. Rottland, and G. Rigoll. Improved On-Line Handwriting Recognition Using Context Dependent Hidden Markov Models. In *Proc. of the 4th International Conference Document Analysis and Recognition (ICDAR '97)*, pages 641–644, Ulm, GERMANY, Aug. 1997.

[7] C.-L. Liu, S. Jaeger, and M. Nakagawa. Online Recognition of Chinese Characters: The State-of-the-Art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(2):198–213, Feb. 2004.

[8] S. Manke, M. Finke, , and A. Waibel. NPen++: A Writer Independent, Large Vocabulary On-Line Cursive Handwriting Recognition System. In *Proc. Third International Conference on Document Analysis and Recognition*, volume 1, pages 403–408, Montréal, Canada, Aug. 1995.

[9] S. Manke, M. Finke, and A. Waibel. The use of dynamic writing information in a connectionist on-line cursive handwriting recognition system. In G. Tesauro, D. Touretzky, and T. Leen, editors, *Advances in Neural Information Processing Systems*, volume 7, pages 1093–1100. The MIT Press, 1995.

[10] L. Rabiner. A Tutorial of Hidden Markov Models and Selected Application in Speech Recognition. *Proc. IEEE*, 77:257–286, 1989.

[11] G. Rigoll and A. Kosmala. A Systematic Comparison Between On-Line and Off-Line Methods for Signature Verification with Hidden Markov Models. In *Int. Conference on Pattern Recognition (ICPR)*, pages 1755–1757, Brisbane, 1998.

[12] G. Seni, R. K. Srihari, and N. M. Nasrabadi. Large vocabulary recognition of on-line handwritten cursive words. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(7):757–762, 1996.

[13] H. Shu. A On-line Handwriting Recognition Using Hidden Markov Models. Master's thesis, Massachusetts Institute of Technology, 1996.

[14] V. Vuori, J. Laaksonen, E. Oja, and J. Kangas. Speeding Up On-line Recognition of Handwritten Characters by Pruning the Prototype Set. In *Proc. of the Sixth International Conference on Document Analysis and Recognition (ICDAR '01)*, pages 0501–0507, Seattle, Washington, Sept. 2001.

[15] S. Young, J. Odell, D. Ollason, V. Valtchev, and P. W. odland. *The HTK Book: Hidden Markov Models Toolkit V2.1*. Cambridge Research Laboratory Ltd, Mar. 1997.