HMM-Based On-Line Recognition of Handwritten Whiteboard Notes

Marcus Liwicki and Horst Bunke
Institute of Computer Science and Applied Mathematics
University of Bern, Neubrückstrasse 10, CH-3012 Bern, Switzerland
{liwicki, bunke}@iam.unibe.ch

Abstract

In this paper we present an on-line recognition system for handwritten texts acquired from a whiteboard. This input modality has received relatively little attention in the handwriting recognition community in the past. The system proposed in this paper uses state-of-the-art normalization and feature extraction strategies to transform a handwritten text line into a sequence of feature vectors. Additional preprocessing techniques are introduced, which significantly increase the word recognition rate. For classification, Hidden Markov Models are used together with a statistical language model. In writer independent experiments we achieved word recognition rates of 67.3% on the test set when no language model is used, and 70.8% by including a language model.

1. Introduction

Although the problem of handwriting recognition has been considered for more than 30 years [1, 11, 15], there are still many open issues, especially in the task of unconstrained handwritten sentences recognition. This domain is traditionally divided into on-line and off-line recognition. In on-line recognition a time ordered sequence of coordinates, representing the movement of the tip of pen, is captured, while in the off-line mode only the image of the text is available.

In this paper we consider an on-line recognition problem, namely the recognition of notes written on a whiteboard. This is a relatively new task. As people stand, rather than sit, during writing and the arm does not rest on a table, handwriting rendered on a whiteboard is different from handwriting produced with a pen on a writing tablet. It has been observed that the baseline usually cannot be interpolated with a simple polynomial up to degree 2. Furthermore, the size and width of the characters become smaller the more the writer moves to the right. In Fig. 1 some examples of handwritten text lines acquired from a whiteboard are shown. The problems mentioned before require special processing steps in addition to the usual preprocessing.

Despite some additional difficulty, the whiteboard modality is important in several applications, such as the documentation of lectures or meetings. In the particular application underlying this paper we aim at developing

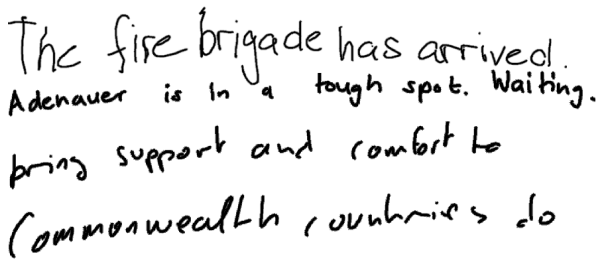

Figure 1. Examples of handwritten texts acquired from a whiteboard

a handwriting recognition system to be used in a smart meeting room scenario [18], in our case the smart meeting room developed in the IM2 project [10]. Smart meeting rooms usually have multiple acquisition devices, such as microphones, cameras, electronic tablets, and a whiteboard. In order to allow for indexing and browsing the data [19], an automatic transcription of the recorded data is needed.

Recently a first recognition system for whiteboard data has been described [8]. Since it is based on an off-line handwritten text recognizer, the recorded on-line data have to be transformed into images of the text lines. This may cause a loss of information which is available in the on-line data. The system presented in this paper works in the on-line mode. It uses state-of-the-art preprocessing and feature extraction methods with additional preprocessing steps specially developed for the characteristics of whiteboard data. For classification Hidden Markov Models (HMMs) are used. As a postprocessing step, a bigram language model is included.

In our writer independent experiments on the IAM-OnDB [9][1], we achieved a word recognition rate of up to 70.8%. The additional preprocessing steps, namely the splitting of a text line into subparts and an improved slant correction method, led to a significant improvement. We also tested the recognition system on the UNIPEN database [5], but due to missing benchmark tasks [17], we can not compare our results with previous work.

The rest of the paper is organized as follows. Section 2 gives an overview of the proposed system. In Section 3

[1] http://www.iam.unibe.ch/~fki/iamondb/

Figure 2. Illustration of the recording

the main steps for preprocessing the data are presented. The feature extraction is described in Section 4. Section 5 briefly describes the recognition system. Experiments and results are presented in Section 6, and finally Section 7 draws some conclusions and gives an outlook for future work.

2. System Overview

The eBeam interface[2] is used for recording the handwriting. It allows us to write on a whiteboard with a normal pen in a special casing, which sends infrared signals to a triangular receiver mounted in one of the corners of the whiteboard. The acquisition interface outputs a sequence of (x,y)-coordinates representing the location of the tip of the pen together with a time stamp for each location. The data is in xml-format and the frame rate of the recordings varies from 30 to 70 frames per second. An illustration is shown in Fig. 2.

The system described in this paper consists of four main modules: the on-line preprocessing, where noise in the raw data is reduced and the text line is normalized with respect to skew, slant, width and height; the feature extraction, where the sequence of points is transformed into a sequence of feature vectors; the recognition, where the HMM-based classifier generates an n-best list of word sequences; and the post-processing, where a statistical language model is applied to improve the results generated by the HMM.

3. Preprocessing

Before feature extraction can be applied, the recorded data has to be normalized. This is a very important step in handwriting recognition systems, because the styles of the writers differ with respect to skew, slant, height and width of the characters. Most of the preprocessing steps are similar to state-of-the-art on-line handwriting recognition systems [6, 11, 13]. In the literature there is no standard way of preprocessing the data, but many systems use similar techniques.

[2] eBeam System by Luidia, Inc. - www.e-Beam.com

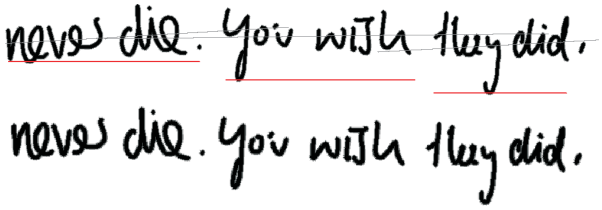

Figure 3. Breaking a text line into components and skew correction

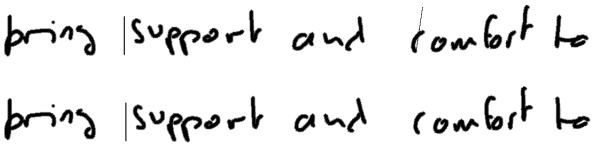

Figure 4. Slant correction with different slant for each component

The recorded on-line data usually contain noisy points and gaps within strokes, which are caused by loss of data. Thus we first apply the on-line preprocessing of Ref. [8] to recover from these artifacts. The cleaned text data is then automatically divided into lines using some simple heuristics.

Before applying the standard normalization steps, an additional step is introduced, which is important for handling the whiteboard data. As stated in Section 1, the text lines on a whiteboard usually have no common skew and slant and the size of the letters is not the same at the beginning and at the end of a line. Therefore the text line is split into subparts and the rest of the preprocessing is done for each part separately. The line is only split at gaps that are larger than the mean gap size. Also the size of both subparts has to be greater than a predefined threshold. An example splitting is indicated below the first text line in Fig. 3.

Next the subparts are corrected with respect to their skew. For that purpose we perform a linear regression through all the points. This process is illustrated in Fig. 3 with the resulting text line shown in the lower part.

For slant normalization, we compute the histogram over all angles subtended by the lines connecting two successive points of the trajectory and the horizontal line [6]. We additionally weight the histogram values with a Gaussian with its mean at the vertical angle, and the variance empirically set. This is beneficial because some words are not properly corrected if a single long straight line is drawn in horizontal direction, which results in a large histogram value. We also smooth each histogram entry with its direct neighbors using the window (0.25, 0.5, 0.25), because in some cases the correct slant is at the border of two angle intervals and a single peak at another interval may be slightly higher. This single peak will become smaller after smoothing. Figure 4 shows a text line with two subparts before and after slant correction.

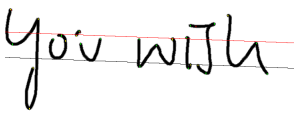

Figure 5. Baseline and corpus line of an example part of a text line

The removing of delayed strokes, e.g. the crossings of a "t" or the dots of an "i" is done using simple heuristics, i.e. strokes written in the upper region above already written parts, followed by a pen-movement to the right, are removed. The equidistant resampling of the point sequence is also done in the same way as in the literature, i.e. the sequence of points is replaced by a sequence of points on the trajectory having the same distance to each other. The optimal value for the distance has been empirically optimized. This step is needed because different writers write at a different speed.

The next important step is the computation of the baseline and the corpus line. For that purpose, the minima and maxima of the y-coordinates of the strokes are calculated. Then two linear regressions through the minima and maxima are computed with the constraint that the two resulting lines have to have the same slope. After the regression lines have been determined the least fitting points are removed and another linear regression is performed. This correction step is done twice which then results in the estimated baseline (minima) and corpus line (maxima). Figure 5 illustrates the estimated baseline and the corpus line of an example word sequence. The baseline is subtracted from all y-coordinates to make it equal to the x-axis. With the two lines the text is divided into three areas: The upper area, which mainly contains the ascenders of the letters; the median area, where the corpus of the letters is present; and the lower area with the descenders of some letters. These three areas are normalized to predefined heights.

As the last preprocessing step, the width of the characters is normalized. First, the number of characters is estimated as a fraction of the number of strokes crossing the horizontal line between the baseline and the corpus line. The text is then horizontally scaled according to this value. This preprocessing step is needed because the x-coordinate after high-pass filtering is taken as a feature (see Section 4) and the angles would differ for different widths.

4. Feature Extraction

The set of extracted features can be divided into two classes. The first class consists of features extracted for each point considering the neighbors with respect to time. The second class takes the off-line matrix representation into account.

The features of the first class are the following:

- *pen-up/pen-down:* a boolean variable indicating whether the pen-tip touches the board or not. Consecutive strokes are connected with straight lines where this feature has the value *false*.

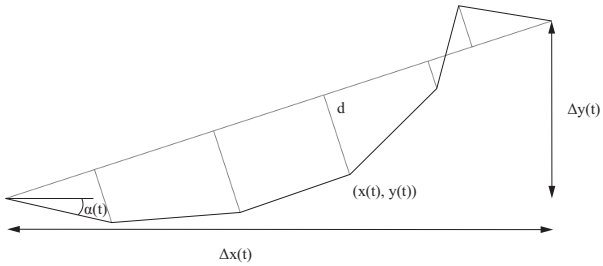

Figure 6. Features of the vicinity

- *hat-feature:* this binary feature indicates if a delayed stroke has been removed which has the same horizontal position as the considered point.
- *speed:* the velocity is computed before resampling and then interpolated.
- *x-coordinate:* the x-position is taken after high-pass filtering, i.e. after subtracting a moving average from the real horizontal position.
- *y-coordinate:* the vertical position of the point after normalization.
- *writing direction:* the cosine and sine of the angle between the line segment starting at the point and the x-axis.
- *curvature:* the cosine and sine of the angle between the lines to the previous and the next point.
- *vicinity aspect:* the aspect of the trajectory (see Fig. 6):
$$\frac{\Delta y(t) - \Delta x(t)}{\Delta y(t) + \Delta x(t)}$$
- *vicinity slope:* cosine and sine of the angle $\alpha(t)$ of the straight line from the first to the last vicinity point (see Fig. 6).
- *vicinity curliness:* the length of the trajectory in the vicinity divided by $\max(\Delta x(t), \Delta y(t))$ (see Fig. 6).
- *vicinity linearity:* the average square distance d^2 of each point in the vicinity to the straight line from the first to the last vicinity point (see Fig. 6).

The features of the second class are all computed using a two-dimensional matrix representing the off-line version of the data. The following features are used:

- *ascenders/descenders:* the number of points above/below the corpus line with a minimal distance to it, whose x-coordinates are in the vicinity of the point. The distances are set to a predefined fraction of the corpus height.
- *context map:* the two-dimensional vicinity of the point is transformed to a 3×3 map. The resulting nine values are taken as features.

5. Recognition

An HMM is built for each of the 58 characters in the character set, which includes all small and capital letters together with some other special characters, e.g. punctuation marks. In all HMMs the linear topology is used, i.e. there are only two transitions per state, one to itself and one to the next state. In the emitting states, the observation probability distributions are estimated by mixtures of Gaussian components. In other words, continuous HMMs are used. The character models are concatenated to represent words and sequences of words. For training, the Baum-Welch algorithm [2] is applied. In the recognition phase, the Viterbi algorithm [3] is used to find the most probable word sequence. As a consequence, the difficult task of explicitly segmenting a line of text into isolated words is avoided, and the segmentation is obtained as a byproduct of the Viterbi decoding applied in the recognition phase. The output of the recognizer is a sequence of words. In the experiments described in Section 6, the recognition rate will always be measured on the word level.

In [4] it has been pointed out that the number of Gaussians and training iterations have an effect on the recognition results of an HMM recognizer. Often the optimal value increases with the amount of training data because more variations are encountered. The system described in this paper has been trained with up to 36 Gaussian components and the classifier that performed best on a validation set has been taken as the final one in each of the experiments described in Section 6.

Since the system proposed in this paper is performing handwritten text recognition on text lines and not only on single words, it is reasonable to integrate a statistical language model. In this paper we consider bigram language models. Bigram language models represent a special case of the more general statistical n-gram language models [12].

N-gram language models are based on the observation that we are often able to guess the next word when we are reading a given text. In other words, the probability of a word is highly depending on the previous text. In the case of bigram language models the previous text is approximated by the last word and the dependency is modeled by the probability $p(w_i|w_{i-1})$, where w_i represents the considered word and w_{i-1} stands for the previous word. The probability $p(W)$ of a text line $W = (w_1, \ldots, w_n)$ can then be computed as follows:

$$p(W) = p(w_1) \left(\prod_{i=2}^{n} p(w_i|w_{i-1}) \right) \quad (1)$$

The most likely word sequence $\hat{W} = (w_1, \ldots, w_m)$ for a given observation sequence X is computed in the following way [20]:

$$\hat{W} = \underset{W}{\operatorname{argmax}} \ \log p(X|W) + \alpha \ \log p(W) + m \ \beta \quad (2)$$

Table 1. Performance of different recognition systems on the validation set.

system	Performance
preproc. and features as described in [13]	50.0
all features of first class (See Sect. 4)	50.5
all features	65.3
all features, new preproc.	66.7

According to Eq. 2 the optical model $p(X|W)$, which is the result of the HMM decoding, is combined with the likelihood $p(W)$ obtained from the language model. Because the HMM system and the language model produce only approximations of probabilities, two additional parameters, α and β, are used to compensate the deficiencies and to control the integration of the language model. Parameter α is called Grammar Scale Factor (GSF) and weights the influence of the language model against the optical model. The term Word Insertion Penalty (WIP) is used for parameter β which prevents the system from oversegmentation resp. undersegmentation. That is, the larger the value of β is, the higher is the system's tendency to split a text into many short words. In an attempt to optimize the system described in this paper we have included a statistical language model with these two parameters.

6. Experiments and Results

For our first experiments, we used the IAM-OnDB, a large on-line handwriting database consisting of handwriting samples acquired from a whiteboard [9]. This database contains 86,272 word instances from a 11,050 word dictionary written down in 13,040 text lines.

To make our results comparable to previous experiments on the IAM-OnDB, we used the same experimental setup as it has been done, for an off-line recognizer, in Ref. [8] and performed 5-fold cross-validation. The test set in all experiments consists of 6,204 word instances in 1,258 lines produced by 20 different writers. All tests were conducted using a dictionary consisting of the 2,337 words that appear in the test set. The language model was generated from the LOB-corpus [7].

First we trained several recognition systems on a subset of the IAM-OnDB consisting of the 20 writers mentioned above. The different recognition systems result from different preprocessing steps and feature sets. Table 1 shows the results of the systems on the validation set. It can be observed that the inclusion of the off-line information (rows 3 and 4) leads to a substantial improvement of the word recognition rate. Note that "convetional preprocessing" denotes the preprocessing steps of Sect. 3 which are known from literature, and "new preprocessing" includes the splitting into subparts and the improved slant correction.

In Table 2 the results on the test set are shown. The last row shows the results of the same recognition system as used in the second row, but trained on a larger training set of about 50k words in about 7,800 text lines written by 132 writers. The improvement resulting from the larger training set is quite substantial. Yet the improvement of

Table 2. Performance on the test set (LM = language model).

system	without LM	including a LM
conventional preproc.	62.3	64.8
new preproc.	63.6	66.4
trained on large set	67.3	70.8

the new preprocessing methods (63.6% vs. 62.3% without and 66.4%vs. 64.8% including the language model) is already statistically significant at the 95% level.

We also conducted writer independent experiments on the publicly available UNIPEN database [5]. For that purpose, we used several datasets from Benchmark#8 of train_r01_v07. The results varied from 20% to above 90% depending on the set, the dictionary size and the length of the text lines. Unfortunately, there exists no benchmark task for handwritten text line recognition, which is a known problem [17]. Thus we are not able to compare any of these results with previous work.

7. Conclusions and Future Work

In this paper we presented a new HMM-based on-line recognition system for handwritten texts acquired from a whiteboard. The recognition of notes written on a whiteboard is a relatively new task. Whiteboard data differ from usual pen-input data as people stand, rather than sit, during writing and the arm does not rest on a table. Clearly, the whiteboard modality is important for several applications, such as recording lectures or meetings. Particularly, we aim at developing a handwriting recognition system for a smart meeting room scenario.

While the proposed system uses some state-of-the-art methods for preprocessing and feature extraction, we also introduced some new preprocessing steps especially for whiteboard data. These preprocessing techniques could significantly increase the word recognition rate. The best performance in our writer independent experiments is 70.8% on the test set. As one could expect, the recognition rate is significantly higher than recognition rates of a state-of-the-art off-line recognition system tested under the same conditions [9], which only reaches 66.4%.

In future we plan to further increase the performance of the recognizer by improving the statistical language model. It is also planned to combine the presented on-line recognition system with the off-line recognizer of Ref. [8]. From such a combination, an improved recognition performance can be expected [14, 16].

We also plan to test the recognition system on other databases. Once a benchmark task for the UNIPEN database is available, we plan to use it for testing our system. Additional future work includes the recognition of graphical symbols, tables, and gestures.

Acknowledgments

This work was supported by the Swiss National Science Foundation program “Interactive Multimodal Information Management (IM)2” in the Individual Project “Visual/Video Processing”, as part of NCCR.

References

[1] H. Bunke. Recognition of cursive roman handwriting - past present and future. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, volume 1, pages 448–459, 2003.

[2] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *Journal of Royal Statistical Society B*, 39(1):1–38, 1977.

[3] G. D. Forney. The Viterbi algorithm. In *Proc. IEEE*, volume 61, pages 268–278, 1973.

[4] S. Günter and H. Bunke. HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components. *Pattern Recognition*, 37:2069–2079, 2004.

[5] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Proc. 12th Int. Conf. on Pattern Recognition*, pages 29–33, 1994.

[6] S. Jaeger, S. Manke, J. Reichert, and A. Waibel. Online handwriting recognition: the NPen++ recognizer. *Int. Journal on Document Analysis and Recognition*, 3(3):169–180, 2001.

[7] S. Johansson. *The tagged LOB Corpus: User's Manual*. Norwegian Computing Centre for the Humanities, Norway, 1986.

[8] M. Liwicki and H. Bunke. Handwriting recognition of whiteboard notes. In *Proc. 12th Conf. of the International Graphonomics Society*, pages 118–122, 2005.

[9] M. Liwicki and H. Bunke. IAM-OnDB - an on-line English sentence database acquired from handwritten text on a whiteboard. In *Proc. 8th Int. Conf. on Document Analysis and Recognition*, volume 2, pages 956–961, 2005.

[10] D. Moore. The IDIAP smart meeting room. Technical report, IDIAP-Com, 2002.

[11] R. Plamondon and S. N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Trans. Pattern Anal. Mach. Intell.*, 22(1):63–84, 2000.

[12] R. Rosenfeld. Two decades of statistical language modeling: Where do we go from here? In *Proc. IEEE 88*, volume 88, pages 1270–1278, 2000.

[13] M. Schenkel, I. Guyon, and D. Henderson. On-line cursive script recognition using time delay neural networks and hidden Markov models. *Machine Vision and Applications*, 8:215–223, 1995.

[14] O. Velek, S. Jäger, and M. Nakagawa. Accumulated-recognition-rate normalization for combining multiple on/off-line Japanese character classifiers tested on a large database. In *Proc. 4th Multiple Classifier Systems*, pages 196–205, 2003.

[15] A. Vinciarelli. A survey on off-line cursive script recognition. *Pattern Recognition*, 35(7):1433–1446, 2002.

[16] A. Vinciarelli and M. Perrone. Combining online and offline handwriting recognition. In *Proc. 7th Int. Conf. on Document Analysis and Recognition*, pages 844–848, 2003.

[17] L. Vuurpijl, R. Niels, M. van Erp, L. Schomaker, , and E. Ratzlaff. Verifying the unipen devset. In *Proc. 9th Int. Workshop on Frontiers in Handwriting Recognition*, pages 586–591, 2004.

[18] A. Waibel, T. Schultz, M. Bett, R. Malkin, I. Rogina, R. Stiefelhagen, and J. Yang. Smart: The smart meeting room task at ISL. In *Proc. IEEE ICASSP*, volume 4, pages 752–755, 2003.

[19] P. Wellner, M. Flynn, and M. Guillemot. Browsing recorded meetings with Ferret. In *Machine Learning for Multimodal Interaction*, pages 12–21, 2004.

[20] M. Zimmermann and H. Bunke. Optimizing the integration of a statistical language model in HMM-based offline handwritten text recognition. In *Proc. 17th Int. Conf. on Pattern Recognition*, pages 541–544, 2004.