

Combining On-Line and Off-Line Systems for Handwriting Recognition

Marcus Liwicki, 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 a new multiple classifier system (MCS) for recognizing notes written on a whiteboard. This MCS combines one off-line and two on-line handwriting recognition systems derived from previous work. The recognizers are all based on Hidden Markov Models but vary in the way of preprocessing and normalization. To combine the output sequences of the recognizers, we incrementally align the word sequences using a standard string matching algorithm. For deriving the final decision a voting strategy is applied. With the combination we could increase the system performance over the best individual recognizer by about 2%.

1. Introduction

The domain of handwriting recognition has traditionally been divided into on-line and off-line recognition. In off-line recognition the text to be recognized is captured by a scanner and stored as an image, while in the on-line mode the handwriting is produced by means of an electronic pen or a mouse and acquired as a time-dependent signal. Good progress has been achieved in both off-line and on-line recognition [2, 14, 16]. In this paper we consider a novel task, which is the recognition of text written on a whiteboard. A similar task has been considered in refs. [4, 13]. However, while in refs. [4, 13] a video camera was employed to capture the handwriting, we use the eBeam¹ interface which is based on infrared sensing. This system is easier to use than a video camera and it is less vulnerable to artifacts arising from poor lighting conditions, self-occlusion and low image resolution.

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. Recently, we developed an off-line [11] and an on-line [10] recognition systems for solving

this task. In the present paper we describe the combination of three different systems derived from the recognizers introduced in refs. [11, 10]. To combine the output of the recognizers, we incrementally align the word sequences using a standard string matching algorithm described in Ref. [17]. Then the word that most often occurs at a certain position is used as the final result.

The IAM-OnDB [9]² has been used for our experiments. We could significantly increase the word recognition accuracy by almost 2%, compared to the best performing stand-alone system. This results in an overall performance of about 66.8%.

The rest of the paper is organized as follows. Section 2 gives an overview of the proposed system and Section 3 introduces the main steps for preprocessing the data and extracting the features. The recognition systems are described in Section 4 and their combination is presented in Section 5. Section 6 reports on experimental results of the combination, and finally Section 7 draws some conclusions and gives an outlook to future work.

2 System Overview

The eBeam interface is used for recording the handwriting. It allows one 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. 1.

The recorded on-line data usually contain noisy points and gaps within strokes. In Fig. 2 examples of both types of distortion are shown. In the word *await* a spurious point occurs that leads to the introduction of a large artifact, i.e. two long additional strokes. Furthermore, we observe in the

¹eBeam System by Luidia, Inc. - www.e-Beam.com

²<http://www.iam.unibe.ch/fki/iamondb/>

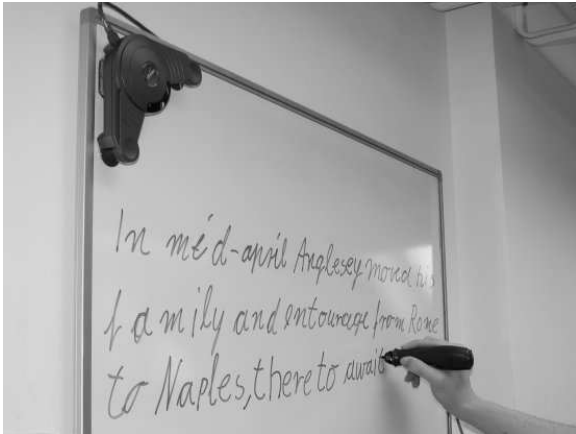


Figure 1. Illustration of the recording

first line that there are many gaps within the text, which are caused by loss of data. Thus we first apply some on-line preprocessing operations to recover from these artifacts. These operations are described in Ref. [11]. The cleaned text data is then automatically divided into lines using some simple heuristics.

After further preprocessing, normalization and feature extraction, a classifier based on Hidden Markov Models (HMMs) is applied. The next section describes the different ways of preprocessing the data in the three systems. The recognizer, which is the same for all three systems, will be explained in more detail in Section 4.

3 The Three Systems

While the three recognition systems are using the same HMM-based classifier, they differ in the way of preprocessing and feature extraction. This section gives an overview of all processing steps, starting with the segmented text lines and ending with the features which serve as input to the HMMs.

3.1 Off-Line System

Since the segmented text lines are still in the on-line format, they have to be transformed into off-line images, so that they can be used as input for the off-line recognizer. First all consecutive points within the same stroke are connected. This results in one line segment per stroke. Then the lines are dilated to a width of eight pixels. The center of each line is colored black and the pixels are getting lighter towards the periphery of each line segment. Fig. 4 shows an example of an off-line image generated from the on-line input. Compared to Figs. 2 and 3 the handwriting looks more similar to normal handwriting (see Fig. 5). In general,

In mid-april Anglesey moved his family and entourage from Rome to Naples, there to await the arrival of

Figure 2. Recorded text

In mid-april Anglesey moved his family and entourage from Rome to Naples, there to await the arrival of

Figure 3. Text after removing noise

the generation of realistic off-line data is a quite complex problem. Ref. [15] proposes methods to create images that look even more similar to scanned images. In the experiments reported in Ref. [15] the recognizer was trained and tested on computer generated images, and the best performance has been achieved using a constant thickness. During our experiments the recognition rate increased when we supplemented this simple approach with the generation of different gray values.

In mid-april Anglesey

Figure 4. Generated gray-scale image

In mid-April Anglesey

Figure 5. Off-Line image of the text shown in Fig. 4 (produced by a different writer and acquired with a document scanner)

The text lines resulting from the on-line to off-line conversion procedure are normalized with respect to skew, slant, writing width and baseline location. Normalization of the baseline location means that the body of the text line (the part which is located between the upper and lower baseline), the ascender part (located above the upper baseline), and the descender part (below the lower baseline) are vertically scaled to a predefined size each. Writing width normalization is performed by a horizontal scaling operation, the purpose of which is to scale the characters so that they

have a predefined average width.

To extract a sequence of feature vectors from the normalized images, a sliding window is used. The width of the window is one pixel, and nine geometrical features are computed at each window position. Thus an input text line is converted into a sequence of feature vectors in a 9-dimensional feature space. The nine features correspond to the following geometric quantities. The first features represent the average gray value of the pixels in the window and their center of gravity. Furthermore, the second order moment in vertical direction is used as a feature. In addition to these global features, the location of the uppermost and lowermost black pixel as well as their gradient, determined by using the neighboring windows, are taken as features. Feature number eight is the number of black-white transitions between the uppermost and lowermost pixel in an image column. Finally, the proportion of black pixels to the number of pixels between these two points is used. For a detailed description of the features see Ref. [12].

3.2 First On-Line System

The individual text lines are corrected with respect to their skew. For this purpose we perform a linear regression through all points. For slant normalization, we compute the histogram over all angles between the lines connecting two successive points of the trajectory and the horizontal line [8].

After normalization, delayed strokes, e.g. the crossing of a “t” or the dot of an “i” are removed, using simple heuristics. That is, strokes written in the upper region above already written parts, followed by a pen-movement to the right, are eliminated. Next, we perform an equidistant resampling of the point sequence, i.e. the original sequence of points is replaced by a sequence of points on the trajectory where all consecutive points have 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 this purpose, we compute two linear regressions through the minima and maxima of the y -coordinates of the strokes and remove the least fitting points. This correction step is done twice which then results in the estimated baseline (minima) and corpus line (maxima). The baseline is subtracted from all y -coordinates to make it equal to the x -axis. This also results in the three main writing areas. As the last preprocessing step, the width of the characters is normalized.

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 features belonging to this class are the following: the

pen-up/pen-down feature which indicates whether the pen-tip has been on the board at the actual point; the hat-feature which indicates if a delayed stroke has been removed at the same horizontal position; the speed; the normalized x -coordinate; the normalized y -coordinate; the cosine and sine of the writing direction; the cosine and sine of the curvature; the vicinity aspect, i.e. the aspect of the trajectory in a given on-line vicinity; the vicinity slope, i.e. the cosine and sine of the angle $\alpha(t)$ of the straight line from the first to the last vicinity point; the vicinity curliness, i.e. the length of the trajectory in the vicinity divided by $\max(\Delta x(t), \Delta y(t))$; and the vicinity linearity, i.e. the average square distance d^2 of each point in the vicinity to the straight line from the first to the last vicinity point.

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

3.3 Second On-Line System

The second on-line system has been originally described in [10]. It is basically the same system as the first on-line system up to two differences. Before applying the normalization operations, an additional step is introduced, which is important for the whiteboard data. The text lines on a whiteboard usually have no uniform skew along the whole line and the slant and size of the letters is not the same at the beginning and at the end of a line. This is caused by the fact that people stand, rather than sit, during writing and the arm does not rest on a table. Therefore the text line is split into subparts and the rest of the preprocessing is done for each subpart separately. Splitting is done at gaps that are larger than the mean gap size. Also the size of both subparts has to be greater than a predefined threshold. The slant correction is supplemented with the following method. We 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.

4 Recognition

The same recognition engine is used in the off-line and the on-line systems. An HMM is built for each of the 58 characters in the character set, which includes all small and all 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 [3] is applied. In the recognition phase, the Viterbi algorithm [6] 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 [7] 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.

5 Combination

Since the recognizers output word sequences for whole text lines and there may be a different number of words in each output sequence, these sequences are aligned into a Word Transition Network (WTN). The problem of finding the optimal alignment for n sequences is NP-complete [18]. Thus an approximate solution was taken. This solution aligns the multiple sequences incrementally by building WTNs. At the beginning, the first and the second word sequence are aligned in a single WTN, using a standard string matching algorithm described in Ref. [17]. The resulting WTN is aligned with the next word sequence giving a new WTN, which is then aligned with the next recognition result and so on. This method does not guarantee an optimal solution, but the suboptimal solution often provides a good alignment accuracy in practice. An example of an alignment of the output of our three recognizers (denoted by W_1 , W_2 and W_3) is shown in Fig. 6

After the alignment a voting module extracts the best scoring word sequence from the WTN. In this work, only

In mid-april Anglesey

W_1 : In mid-april Angle say

W_2 : It mid-april Anglesey

W_3 : I a mid-April Anglesey

$WTN_1 = W_1 + W_2 :$

In	mid-april	Angle	say
It	mid-april	Anglesey	ϵ

$WTN_2 = WTN_1 + W_3 :$

In	ϵ	mid-april	Angle	say
It	ϵ	mid-april	Anglesey	ϵ
I	a	mid-April	Anglesey	ϵ

Figure 6. Example of iteratively aligning multiple recognition results.

the number of occurrences of a word w is taken into account for making a decision. In the case of ties, the output of the best performing system on the validation set is taken. In the example of Fig. 6 it would be W_1 which yields the final output “In mid-april Anglesey”. Note that this is the correct transcription, which is not present in the recognition results of any single recognizer. For the alignment and construction of the WTN the Recognizer Output Voting Error Reduction (ROVER) of Ref. [5] is used.

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.

We used the sets of the benchmark task with the closed vocabulary IAM-OnDB-t1³. There the data is divided into four sets: one set for training; one set for validating the meta parameters of the training; a second validation set which can be used, for example, for optimizing a language model; and an independent test set. No writer appears in more than one set. Thus, a writer independent recognition task is considered. The size of the vocabulary is about 11 K. In our experiments, we did not include a language model. Thus the second validation set has not been used.

Table 1 shows the results of the three individual recognition systems. The word recognition rate is simply measured

³see benchmark task on <http://www.iam.unibe.ch/~fki/iamondb/>

System	Recognition Rate	Accuracy
Off-Line	67.9 %	61.4 %
1st On-Line	73.4 %	65.1 %
2nd On-Line	73.8 %	65.2 %

Table 1. Results on IAM-OnDB-t1 benchmark

by dividing the number of correct recognized words by the number of words in the transcription. The accuracy is calculated using the following formula:

$$acc = 1 - \frac{\#insertions + \#substitutions + \#deletions}{\#words_in_transcription}$$

For measuring the performance of the combined system, the accuracy is used since it also takes the insertions into account. If we would just use the recognition rate, we could easily increase the performance by never including any null-transition in the WTN, because this only affects the number of insertions.

System	Accuracy
Best Single System	65.2 %
Combination	66.8 %

Table 2. Accuracy of combination

The results of the final combination are shown in Table 2. The recognition accuracy could be increased by 1.6% to 66.8%, which is statistically significant ($\alpha = 0.05$).

7 Conclusions and Future Work

In this paper we presented a new multiple classifier system (MCS) for the recognition of handwritten notes written on a whiteboard. We combined one off-line and two on-line recognition systems. To combine the output sequences of the recognizers, we incrementally aligned the word sequences using a standard string matching algorithm. Then at each output position the word with the most occurrences has been used as the final result. With the MCS we could statistically significantly increase the accuracy by 1.6%.

In our future work we plan to apply other combination methods. In particular we will use several confidence measures and integrate them in the voting strategy [5]. As reported in the literature, there exist better ways of calculating confidence measures than just using the normalized likelihood or local n-best lists [1].

References

[1] R. Bertolami, M. Zimmermann, and H. Bunke. Rejection strategies for offline handwritten text line recognition. *Pattern Recognition Letters*, 27(16):2005–2012, 2006.

[2] 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.

[3] 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.

[4] G. A. Fink, M. Wienecke, and G. Sagerer. Video-based on-line handwriting recognition. In *Proc. 6th Int. Conf. on Document Analysis and Recognition*, pages 226–230, 2001.

[5] J. Fiscus. A post-processing system to yield reduced word error rates: Recognizer output voting error reduction. In *IEEE Workshop on Automatic Speech Recognition and Understanding*, pages 347–352, 1997.

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

[7] 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.

[8] 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.

[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] M. Liwicki and H. Bunke. HMM-based on-line recognition of handwritten whiteboard notes. In *Proc. 10th Int. Workshop on Frontiers in Handwriting Recognition*, pages 595–599, 2006.

[11] M. Liwicki and H. Bunke. Handwriting recognition of whiteboard notes - studying the influence of training set size and type. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 21(1):83–98, 2007.

[12] U.-V. Marti and H. Bunke. Using a statistical language model to improve the performance of an HMM-based cursive handwriting recognition system. *Int. Journal of Pattern Recognition and Artificial Intelligence*, 15:65–90, 2001.

[13] M. Munich and P. Perona. Visual input for pen-based computers. In *Proc. 3rd Int. Conf. on Pattern Recognition*, pages 33–37, 1996.

[14] 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.

[15] O. Velek, C.-L. Liu, S. Jaeger, and M. Nakagawa. An improved approach to generating realistic Kanji character images from on-line characters and its benefit to off-line recognition performance. In *Proc. 16th Int. Conf. on Pattern Recognition*, pages 588–591, 2002.

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

[17] A. Viterbi. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. on Information Theory*, 13:260–269, 1967.

[18] L. Wang and T. Jiang. On the complexity of multiple sequence alignment. *Journal of Computational Biology*, 1(4):337–348, 1994.