Line-Members – a Novel Feature in On-Line Whiteboard Note Recognition

Joachim Schenk and Johannes Lenz and Gerhard Rigoll
Institute for Human-Machine Communication
Technische Universität München
Theresienstraße 90, 80333 München
{schenk, lej, rigoll}@mmk.ei.tum.de

Abstract

Confusion-matrices show that character mix-ups between similar looking letters differing in size rather than in shape (like "s" and "S" or "e" and "l"), as well as between tall letters (such as "M" and "t") and small case letters (such as "s" and "a") and vice versa can occur in on-line whiteboard note recognition. This paper introduces a novel feature called "line-member" feature that adds discriminance to the feature vector. Thereby, for certain sample points the script line association is estimated using the Viterbi algorithm and taken as a feature.

As our experiments indicate, a relative improvement of $r = 3.3\,\%$ in character level and $r = 3.4\,\%$ in word level accuracy compared to a baseline system without the novel "line-member" feature can be achieved. In addition, the character confusion as described above can be reduced.

Keywords: On-line handwriting recognition, continuous HMM, line-member, feature extraction, preprocessing

1. Introduction

In Hidden-Markov-Model (HMM, [10]) based handwriting recognition systems, each symbol (commonly each character) is represented by a single HMM (either discrete or continuous). Words are recognized by combining character-HMMs using a dictionary. While high recognition rates are reported for *isolated word* recognition systems [4], performance considerably drops when it comes to unconstrained handwritten *sentence* recognition [8]: the lack of previous word segmentation introduces new variability and therefore requires more sophisticated character recognizers. An even more demanding task is the recognition of handwritten whiteboard notes as introduced in [8]. The conditions described in [8] may characterize the problem of on-line whiteboard note recognition as "difficult".

As mentioned above, the smallest entity usually recognized in handwriting recognition is the character. It is therefore reasonable to assume that increasing the performance of *character* recognition accuracy results in an improvement in *word* recognition accuracy. Investigations of character confusion show[1] that characters which differ in size rather than in shape (like "s" and "S" or "e"' and "l") can be confused, see 1 left.

Figure 1. Illustration of character confusion without script lines (left) and relative size description with script lines (right). Small characters are typeset *italic*.

Handwritten text can be separated into certain script lines (see e. g. [1; 5]), as shown in Fig. 2. The top line, the corpus line, the base line and the bottom line are (ideally) defined by the top of tall letters (such as "M" and "t"), the top of lower case letters (such as "s" and "a"), the base line points, and the bottom of characters such as "p", respectively [1].

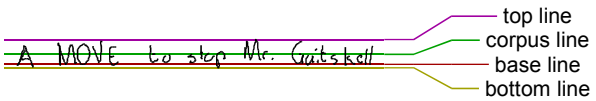

Figure 2. Script lines as e. g. defined in [1; 5]. Script sample taken from IAM-onDB [7].

[1]These investigations were performed on the baseline system presented in this paper.

From: Proc. ICFHR 2008, Montreal, Canada
ISBN 1-895193-03-6

As illustrated in Fig. 1 left, even for a human observer, without any context, it is impossible to distinguish between "s" and "S" or "e" and "l". However, if the script lines limiting the characters are given, the relative size is known and these characters are distinguishable from each other. Following this reasoning, in this paper we introduce a novel feature describing the script line association of certain sample points of a character. This aims to reduce the confusion between characters of similar shape and different size. In order to decide whether a sample point lies on a script line, and if so on which script line, the position and characteristics of each script line must be known. However, as mentioned above, the script lines are defined by the associated sample points of the character. In other words: to find the exact characteristics of the script lines it must be known which sample points belong to which line. One way to cope with this problem is to assume straight script lines or simple polynomial curves of low order [1; 8].

In this paper we contribute to the resolution of the above paradox by first finding sample points that are potential candidates for defining the script lines. Afterwards a trellis is built holding all script line association hypotheses of these points. The least costly path through that trellis is found by applying the Viterbi-algorithm [11]. Finally we iteratively refine the script line association and obtain script lines which may have any characteristic (i. e. they may have any bend). Subsequently, each sample point's script line association is used as a "line-member" feature.

The rest of this paper is structured as follows: the next section gives a brief overview of the baseline system we used as well as a set of standard features which are used for on-line handwritten whiteboard note recognition. In Sec. 3 the novel "line-member" feature is described and discussed in detail. In an experimental section (Sec. 4) the influence of the novel feature on the word level accuracy is examined. Finally, conclusions are drawn and an outlook for further work is given in Sec. 5.

2. System Overview

The handwritten whiteboard data is recorded using the eBeam-System and heuristically segmented into lines [8]. Due to the recording system's limitations the sampling is neither space nor time equidistant. In fact, the sample rate f_s differs in the range of $f_s = 30\,\mathrm{Hz}, \ldots, 70\,\mathrm{Hz}$. Resampling is therefore performed as a first preprocessing step in order to achieve a space-equidistant sampling of the handwritten data. Following this, a histogram-based skew- and slant-correction is performed as described in [6]. Finally all text lines are normalized to meet a distance of "one" between the corpus and the base line. For this initial estimate the corpus and base line are assumed to be parallel and horizontal. The position of both lines is estimated using a histogram-based projection approach similar to [2].

Following the preprocessing, features are extracted from the three-dimensional sample vector $s_t = (x(t), y(t), p(t))^{\mathrm{T}}$ in order to derive a 25-dimensional feature vector $\mathbf{f}_t = (f_1(t), \ldots, f_{25}(t))$. The 25^{th} feature (f_{25}), describing the "line membership" of certain sample points, is explained in Sec. 3. The remaining 24 state-of-the-art *on-line* and *off-line* features for handwriting recognition [5; 8] used in this paper are briefly explained below.

The extracted on-line features are: The pen's "pressure" (f_1), indicating whether or not the pen touches the whiteboard surface. A velocity equivalent (f_2) which is computed *before* resampling and is later interpolated according to the resampling factors. The x- and y-coordinate ($f_{3,4}$) after resampling, whereby the y-coordinate is highpass filtered by subtraction of the moving average. The "writing direction" ($f_{5,6}$), i. e. the angle α of spatially resampled and normalized strokes, coded as $\sin\alpha$ and $\cos\alpha$ and the "curvature" ($f_{7,8}$), i. e. the difference of consecutive angles $\Delta\alpha = \alpha_t - \alpha_{t-1}$, coded as $\sin\Delta\alpha$ and $\cos\Delta\alpha$.

On-line features describing the relation of the current sample point $\mathbf{s}_t$ to its neighbors as described in [5; 8], and altered if needed, are: a logarithmic transformation of the "vicinity aspect" v (the aspect of the trajectory between the points $\mathbf{s}_{t-\tau}$ and $\mathbf{s}_t$, whereby $\tau < t$ denotes the τ^{th} sample point before $\mathbf{s}_t$), $f_9 = \mathrm{sign}(v) \cdot \log(1 + |v|)$, the "vicinity slope" ($f_{10,11}$), i. e. the angle φ between the line $[\mathbf{s}_{t-\tau}, \mathbf{s}_t]$ and the bottom line, coded as $\sin\varphi$ and $\cos\varphi$; the "vicinity curliness" (f_{12}), the length of the trajectory normalized by $\max(|\Delta x|; |\Delta y|)$; and the average square distance to each point in the trajectory and the line $[\mathbf{s}_{t-\tau}, \mathbf{s}_t]$.

The so-called *off-line* features form the second class. We extracted: a 3×3 "context map" (f_{14-22}), i. e. a subsampled bitmap slid along the pen's trajectory to incorporate a 30×30 partition of the currently written letter's actual image and the "ascenders" and "descenders" ($f_{23,24}$), i. e. the number of pixels above respectively beneath the current sample point $\mathbf{s}(t)$.

The handwritten data is recognized by a classifier based on continuous character Hidden Markov Models (HMMs, see [10]) trained on these features.

3. Line Members

In this section the novel "line-member" feature is introduced and the choice of potential sample points affected by this feature is explained. Then a trellis representation of all line-assignment hypotheses and

the estimation of the best line-assignment via the Viterbi algorithm [11] is derived. Finally, a further enhancement of the basic line-assignment algorithm is given.

3.1. Feature Definition and Candidate Reduction

As explained in the introduction, characters which differ in size rather than in shape can be confused. To overcome this problem, a feature characterizing the line-assignment of each of the T sample points $\mathbf{s}(t)$ of a text line $\mathbf{S} = [\mathbf{s}(1), \ldots, \mathbf{s}(T)]$ is defined. This "line-member" feature (f_{25}) is given by

$$f_{25} = \begin{cases} 0 & \text{if } \mathbf{s}(t) \text{ lies on } \textit{no} \text{ line} \\ 1 & \text{if } \mathbf{s}(t) \text{ lies on top line} \\ 2 & \text{if } \mathbf{s}(t) \text{ lies on corpus line} \\ 3 & \text{if } \mathbf{s}(t) \text{ lies on base line} \\ 4 & \text{if } \mathbf{s}(t) \text{ lies on bottom line,} \end{cases} \quad (1)$$

assuming $N_\mathrm{l} = 4$ lines. The assignment of the T sample points to the N_l script lines involves something of a paradox: while the position and characteristics of each script line must be known in order to assign the sample points to the lines, the sample point assignment *defines* the characteristics of the script lines.

Each sample point $\mathbf{s}(t)$ may be assigned either to any of the N_l script lines or to no line, leading to $N_\mathrm{tot} = (N_\mathrm{l} + 1)^T$ different mappings. In case of $T \approx 100$, which is the average number of sample points per line of text in the database (see Sec. 4), this results in $N_\mathrm{tot} \approx 7.9 \cdot 10^{69}$ mappings to be investigated. To lower the number of possible mappings the number of potential line-member candidates is reduced. Meeting the script line definitions given in the introduction we use *spacial* extreme points[2] $\mathbf{s}_\mathrm{ext}(n) \in \mathcal{S}_\mathrm{ext}$. For the extreme points, local minima and local maxima are extracted from the text line $\mathbf{S}$ according to

$$\begin{aligned} \mathcal{S}_\mathrm{min} &= \{\mathbf{s}(t) \,|\, y(t) < y(t-1) \wedge y(t) < y(t+1)\} \\ \mathcal{S}_\mathrm{max} &= \{\mathbf{s}(t) \,|\, y(t) > y(t-1) \wedge y(t) > y(t+1)\}, \end{aligned} \quad (2)$$

with $2 \leq t \leq T-1$. The extreme points then are $\mathcal{S}_\mathrm{ext} = \mathcal{S}_\mathrm{min} \cup \mathcal{S}_\mathrm{max}$, with

$$\begin{aligned} \mathbf{s}_\mathrm{min}(n) &\in \mathcal{S}_\mathrm{min},\ 1 \leq n \leq N_\mathrm{min} = |\mathcal{S}_\mathrm{min}|, \\ \mathbf{s}_\mathrm{max}(n) &\in \mathcal{S}_\mathrm{max},\ 1 \leq n \leq N_\mathrm{max} = |\mathcal{S}_\mathrm{max}|, \end{aligned} \quad (3)$$

with $N_\mathrm{ext} = N_\mathrm{min} + N_\mathrm{max}$ extreme points.

[2]Some authors recommend estimation of extreme points in the velocity domain rather than in the spacial domain. However as mentioned in Sec. 2 the sample rate of the recording system differs in a wide range, inhibiting a robust estimation of extreme points in the velocity domain.

3.2. Line-Assignment

After the candidate reduction, the question arises as to how the extreme points shall be assigned to the lines. For each script line l an initial y-position $c_m(0)$, $1 \leq m \leq N_\mathrm{l}$ where each $c_m(0)$ belongs to $\mathbf{c}(0) = (c_1(0), \ldots, c_{N_\mathrm{l}}(0))$ is defined:

$$\mathbf{c}(0) = \big(\max_{1 \leq n \leq N_\mathrm{ext}} y_{\mathrm{ext},n},\ 1,\ 0,\ \min_{1 \leq n \leq N_\mathrm{ext}} y_{\mathrm{ext},n} \big)^\mathrm{T}, \quad (4)$$

as the line of text is normalized to a corpus-base line distance of "one" during preprocessing (see Sec. 2). The absolute distance between the current sample point $\mathbf{s}_\mathrm{ext}(n)$ and the script line l is

$$m(l, n) = |y(n) - c_l(0)|. \quad (5)$$

In case of horizontal script lines the assignment is a simple nearest neighbor search leading to

$$\hat{f}_{25}(n) = \operatorname*{argmin}_{1 \leq l \leq N_\mathrm{l}} m(l, n). \quad (6)$$

However, the simple assignment described in Eq. 6 is not valid for handwritten whiteboard notes: as pointed out in [8], the script lines cannot be approximated by a polynomial of degree up to two, i. e. the possibility exists that they are *not* straight lines. Each sample point assigned to a specific script line alters its characteristics. In addition, more than one script line can be a reasonable assignment for the current sample point, whereas the following sample points may decide whether or not this assignment is valid. In the next section we therefore introduce a trellis-based script line-assignment using so-called hypotheses.

3.3. Trellis representation

For each extreme point $\mathbf{s}_\mathrm{ext}(n)$, $\mathbf{s}_\mathrm{ext}(n) \in \mathcal{S}_\mathrm{ext}$, $1 \leq n \leq N_\mathrm{ext}$ an association (hypotheses) to *every* script line k ($1 \leq k \leq N_\mathrm{l}$) is assumed and represented in the nodes $\mathbf{c}(k, n) = (c_1(k, n), \ldots, c_{N_\mathrm{l}}(k, n))^\mathrm{T}$, $1 \leq k \leq N_\mathrm{l}$, $1 \leq n \leq N_\mathrm{ext}$ of a $N_\mathrm{l} \times N_\mathrm{ext}$ trellis, where $c_l(k, n)$ holds the current absolute y-position of script line l if $\mathbf{s}_\mathrm{ext}(n)$ is assigned to the script line k. The variation in the characteristics of the script line k (expressed as the absolute value of the change in its y-position) introduced by the association of the current sample point $s_\mathrm{ext}(n)$ is captured by some metric $M_n(k)$ (which is more formally defined in Eqs. 7 and 8). A high value of $M_n(k)$ indicates a high variation of the text lines whereas a small value describes horizontally running script lines.

Transitions in the trellis are made with respect to the assignment of the previous extreme point, e. g. if $\mathbf{c}(1, n-1)$ is followed by $\mathbf{c}(2, n)$, $\mathbf{s}_\mathrm{ext}(n-1)$ is assigned to the top line *and* $\mathbf{s}_\mathrm{ext}(n)$ is assigned to the corpus

line. Therefore the current line-assignment is depends directly one the preceding sample point's assignment. The transitions are made with respect to minimizing the metric $M_n(k)$ for each trellis node $\mathbf{c}(k,n)$ which describes the accumulated absolute variation of each script line. The final script line-assignment is derived from the path through the trellis yielding the smallest metric $M_{N_{\text{ext}}}(k)$. Therefore a path variable $\psi_n(k)$ is used to store the preceding trellis node. Hence, the metric $M_n(k)$, the path variable $\psi_n(k)$ and the trellis nodes $\mathbf{c}(k,n)$ are derived as follows: first the metric $M(k,1)$, the path variable $\psi(k,n)$, and the trellis nodes $\mathbf{c}(k,1)$ corresponding to the first extreme point $\mathbf{s}_{\text{ext}}(1)$ are initialized using Eq. 4:

$$\begin{aligned} M_1(k) &= |c_k(0) - y_1| \\ \psi_1(k) &= 0 \\ c_i(k,1) &= \begin{cases} y_{\text{ext},1} & \text{if } k = i \\ c_i(0) & \text{otherwise} \end{cases}, \quad \begin{matrix} 1 \le k \le N_{\text{l}} \\ 1 \le i \le N_{\text{l}}. \end{matrix} \end{aligned} \tag{7}$$

The metric $M_n(k)$, the path variable $\psi_n(k)$, and the trellis nodes $\mathbf{c}(k,n)$ of the successive extreme points $\mathbf{s}_{\text{ext}}(n)$, $1 \le n \le N_{\text{ext}}$ are recursively defined by:

$$\begin{aligned} M_n(k) &= \min_{1 \le j \le N_{\text{l}}} \left(M_{n-1}(j) + |c_k(j,n-1) - y_{\text{ext}}(n)| \right) \\ \psi_n(k) &= \operatorname*{argmin}_{1 \le j \le N_{\text{l}}} \left(M_{n-1}(j) + |c_k(j,n-1) - y_{\text{ext}}(n)| \right) \\ c_i(k,n) &= \begin{cases} y_{\text{ext}}(n) & \text{if } k = i \\ c_i(\psi_n(k), n-1) & \text{otherwise,} \end{cases} \end{aligned} \tag{8}$$

with $2 \le n \le N_{\text{ext}}$ and $1 \le i,k \le N_{\text{l}}$. The updating defined by Eq. 8 is illustrated in Fig. 3 for an exemplary transition from $\mathbf{c}(4,n-1)$ to $\mathbf{c}(2,n)$ resulting in a minimum weight. The final script line-assignment

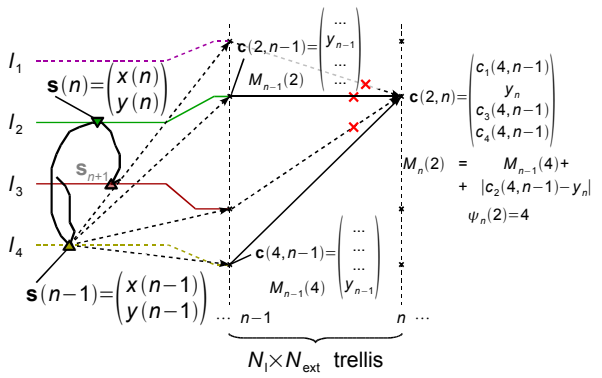

Figure 3. The current trellis node $\mathbf{c}(k,n)$ of each script line k is iteratively updated by the current extreme point $\mathbf{s}_{\text{ext}}(n)$ and the preceding trellis node $\mathbf{v}(k,n-1)$.

$f_{25}(n)$ is found via backtracking:

$$\begin{aligned} f_{25}(N_{\text{ext}}) &= \operatorname*{argmin}_{1 \le l \le N_{\text{l}}} M_{N_{\text{ext}}}(l) \\ f_{25}(n) &= \psi_{n+1}\left(f_{25}(n+1)\right), \; n = N_{\text{ext}} - 1, \ldots, 1. \end{aligned} \tag{9}$$

This implements the Viterbi algorithm [10; 11].

The script line-assignment as performed by adequately applying Eqs. 7, 8, and 9 also allows the incorporation of constraints on the characteristics of the script lines. One reasonable constraint is that script lines may not cross or lie on each other. This is expressed by the condition

$$c_1(k,n) > \ldots > c_{N_{\text{l}}}(k,n), \quad \begin{matrix} 1 \le k \le N_{\text{l}}, \\ 1 \le n \le N_{\text{ext}}. \end{matrix} \tag{10}$$

Trellis paths violating Eq. 10 are omitted.

3.4. System refinement

The basic trellis approach as introduced in the above section and explicitly expressed by Eqs. 7, 8, and 9 performs a script line-assignment for *each* extreme point $\mathbf{s}_{\text{ext}}(n)$, whether or not the current point lies on any script line. This leads to mis-characterized script lines. Furthermore, due to this wrong characterization succeeding points may be assigned to wrong lines, perpetuating the wrong characterization, and a "burst error" occurs as shown in Fig. 4.

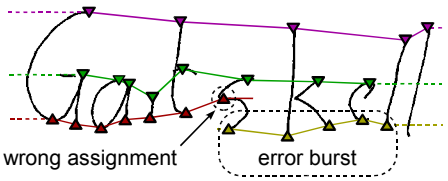

Figure 4. Burst error of successive extreme points after previous assignment of one false candidate (circle). Script sample taken from IAM-onDB [7].

To avoid the forced assignment of each extreme point, the following assumptions are made:

1. the sample points contained in $\mathcal{S}_{\min}$ belong to the bottom and base line, whereas $\mathcal{S}_{\max}$ consists of sample points on the corpus and top line and
2. for each tuple of lines (bottom and base line; corpus and top line) a main line exists which holds most of the sample points. The base line and the corpus line are assumed to be the main line for $\mathcal{S}_{\min}$ and $\mathcal{S}_{\max}$, respectively.

The first assumption is motivated by the definition of the script lines as given in the introduction and is commonly used, see [1; 5; 8]. The second assumption is based on the fact that most characters contain the

base and corpus line, whereas the bottom and top line are shared by fewer characters.

With these common-sense refinements the line-assignment as explained above is performed on the two separate sets $\mathcal{S}_{\min}$ and $\mathcal{S}_{\max}$ as defined in Eq. 3, still taking all script line hypotheses for each minimum or maximum into account. Thereby minima which belong to the top of tall letters (e. g. "J") are absorbed by either the corpus or top line and not assigned to the relevant bottom or base line.

For both sets the number of sample points assigned to the main line is counted. After this first assignment, both sets are iteratively reduced by one sample point and the assignment is repeated on the reduced sets. In cases where the number of sample points assigned to the main lines is higher than for the initial assignment, the omitted sample point leads to an improvement and the sets are further reduced. Otherwise the initial line-assignment is used. The following listing describes the extended algorithm for the script line-assignment in further detail:

Algorithm: Extended script line-assignment
Data: $\mathcal{S}_{\min}, \mathcal{S}_{\max}$
Result: script line-assignment
forall the $\mathcal{S} \in \{\mathcal{S}_{min}, \mathcal{S}_{min}\}$ **do**
  $\mathcal{S}_{\text{proc}} = \mathcal{S} = \{\mathbf{s}(1), \ldots, \mathbf{s}(|\mathcal{S}_{\text{proc}}|)\}$, imp $= 1$;
  while *imp* $= 1$ **do**
    $N_{\text{main}}(0) = \underset{\text{Eqs. 7, 8, and 9}}{\text{LineAssignment}}(\mathcal{S}_{\text{proc}})$;
    forall the $i \in \{1, \ldots, |\mathcal{S}_{proc}|\}$ **do**
      $N_{\text{main}}(i) =$ LineAssignment $(\{\mathcal{S}_{\text{proc}} \setminus \mathbf{s}(i)\})$; (Eqs. 7, 8, and 9)
    if $\hat{i} = \underset{0 \le i \le |\mathcal{S}_{proc}|}{\text{argmax}} \; N_{main}(i) == 0$ **then**
      imp $= 0$
    else
      $\mathcal{S}_{\text{proc}} = \left\{\mathcal{S}_{\text{proc}} \setminus \mathbf{s}(\hat{i})\right\}$

Figure 5 shows the exemplary trellis that results if the basic algorithm presented in Sec. 3.3 as well as the refinement from this section are applied on (a fraction of) a text line. Note the parallel paths: when two parallel paths merge for the current sample point, a final decision on the script line-assignment of all *preceding* sample points is made.

4. Experimental Results

The experiments presented in this section are conducted on a database of handwritten heuristically line-segmented whiteboard notes (IAM-OnDB, [7]). Comparability of the results is provided by using the settings of the writer-independent IAM-onDB-t1 benchmark, consisting of 56 different characters and a 11 k dictionary which also provides well defined writer-disjunct sets (one for training, two for validation, and one for test). For our experiments the same HMM topology as in [8] is used. However, in contrast to [8], just 32 Gaussians were used in both systems discussed here. The parameters of the recognition systems are optimized by evaluating the combination of both validation sets. The *character level* accuracy on the validation sets is shown in Tab. 1. The final tests are conducted on the test set of the IAM-onDB-t1 benchmark. Again, the results are shown in Tab. 1.

Table 1. Character level and word level accuracies for three systems: without/with novel line-member feature and a continuous system [9]).

	A_{b}: without novel feature	A_{lm}: with novel feature	[9]
char. ACC	61.2 %	63.3 %	—
word ACC	62.6 %	64.8 %	65.2 %

Table 2. Absolute count of character confusions of selected character pairs without/with novel feature.

System		e ↔ l	s ↔ S	a ↔ d
without	novel feature	388	392	156
with		139	193	90

The first system, our baseline system, uses the 24 features introduced in Sec. 2. On the character level we achieve $a_{\text{b}} = 61.2\,\%$ accuracy, and $A_{\text{b}} = 62.6\,\%$ accuracy on the word level, see Tab. 1. Additionally we conduct a character confusion analysis on the character level (performed on the combination of both validation sets). The absolute counts of mutual confusions of selected character pairs are shown in Tab. 2.

We next evaluate the second system using the 24 standard features and the novel line-member feature as explained in Sec. 3. Again a confusion analysis is performed on the character level (see Tab. 2). As intended, the absolute count of mutual character confusions drops. However, there are still some confusions. This is due to improper feature extraction: in some cases the sample points are not assigned to the correct script lines. Still, a peak character level accuracy of $a_{\text{lm}} = 63.3\,\%$ (a relative improvement of $r = 3.3\,\%$ compared to the baseline system) and a word level accuracy of $A_{\text{lm}} = 64.8\,\%$ (a relative improvement of $r = 3.4\,\%$) can be reported (see Tab. 1).

When comparing our results with a different, recently published system [9], using different features and more Gaussians for the continuous HMM based recognition system, both of our systems are outper-

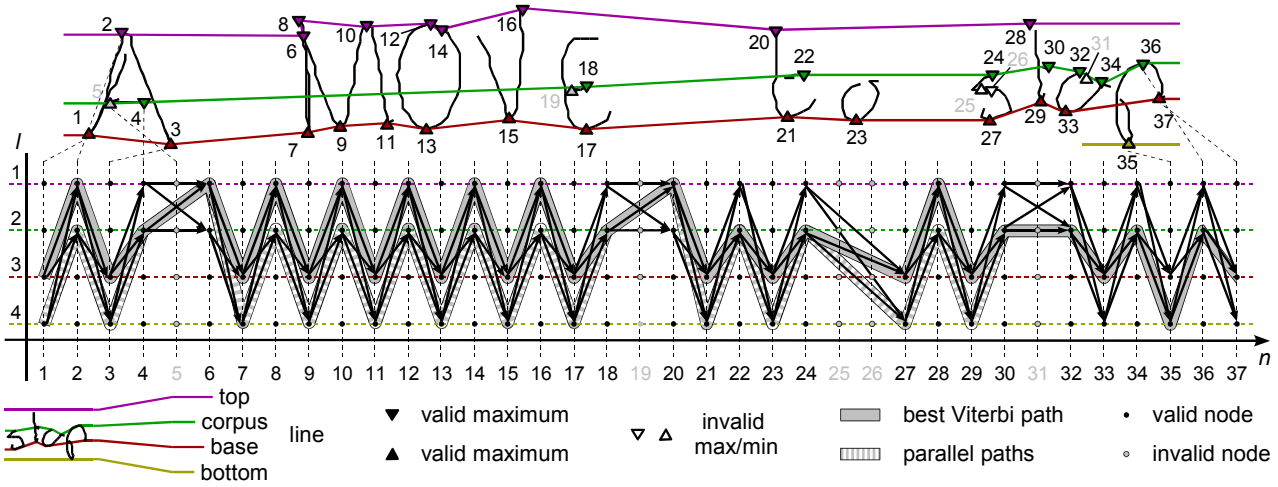

Figure 5. Line membership of a fraction of a text line ([7]) applying the algorithms presented in Secs. 3.3 and 3.4.

formed (see Tab. 1). There may be several reasons for this. First, a slightly altered feature set is used. Second, as stated in [3], the performance of continuous HMM based on-line handwriting recognition systems is influenced by the number of both Gaussian mixtures and training iterations. The exact training process of the system in [9] could not be completely duplicated and is probably mainly responsible for the difference in performance.

5. Conclusions and Outlook

In this paper we introduced a novel feature describing the line membership of certain sample points, wherein the best script line association for selected sample points is found by the Viterbi algorithm and used as the feature. Our experiments show that the absolute count of confusions between characters which differ in size rather than in shape can be reduced. A baseline system which does not use the novel feature was outperformed by $r = 3.3\,\%$ relative in character level and $r = 3.4\,\%$ relative in word level accuracy. However, while producing better results than the baseline system, the best results of a state-of-the-art system could not be reached ($r = -0.6\,\%$ relative). This is mainly due to the different number of Gaussians which have been used for the continuous HMM recognition system. Nevertheless, due to the improvement between our comparable systems, we encourage the consideration of this feature.

In future work, different metrics (such as the ascending slope rather than the absolute y-position of the script lines) will be investigated. Also we plan to give a baseline with hand annotated script line associations for certain sample points.

Acknowledgments

The authors thank M. Liwicki for providing the final benchmark's lattice and G. Weinberg for comments.

References

[1] Y. Bengio and Y. Cun, "Word Normalization for On-Line Handwritten Word Recognition", *Proc. of the IC Pattern Rec.*, pp 409–413, 1994.

[2] R. Bozinovic and S. Srihari, "Off-Line Cursive Script Word Recognition", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 11(1):68–83, 1989.

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

[4] J. Schenk and G. Rigoll, "Novel Hybrid NN/HMM Modelling Techniques for On-Line Handwriting Recognition", *Proc. of the Int. Workshop on Frontiers in Handwriting Rec.*, pp 619–623, 2006.

[5] S. Jaeger, S. Manke, J. Reichert and A. Waibel, "The NPen++ Recognizer", *Int. J. on Document Analysis and Rec.*, 3:169–180, 2001.

[6] E. Kavallieratou, N. Fakotakis and G. Kokkinakis, "New Algorithms for Skewing Correction and Slant Removal on Word-Level", *Proc. of the Int. Conf. ECS*, 2:1159–1162, 1999.

[7] M. Liwicki and H. Bunke, "IAM-OnDB - an On-Line English Sentence Database Acquired from Handwritten Text on a Whiteboard", *Proc. of the Int. Conf. on Document Analysis and Rec.*, 2:1159–1162, 2005.

[8] M. Liwicki and H. Bunke, "HMM-Based On-Line Recognition of Handwritten Whiteboard Notes", *Proc. of the Int. Workshop on Frontiers in Handwriting Rec.*, pp 595–599, 2006.

[9] M. Liwicki and H. Bunke, "Combining On-Line and Off-Line Systems for Handwriting Recognition", *Proc. of the Int. Conf. on Document Analysis and Rec.*, pp 372–376, 2007.

[10] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *Proc. of the IEEE*, 77(2):257–285, February 1989.

[11] A. Viterbi, "Error Bounds for Convolutional Codes and an Asymptotically Optimum Decoding Algorithm", *IEEE Transactions on Information Theory*, 13:260–267, 1967.