Pattern Recognition 42 (2009) 3383 – 3393

Contents lists available at ScienceDirect

Pattern Recognition

journal homepage: www.elsevier.com/locate/pr

Novel script line identification method for script normalization and feature extraction in on-line handwritten whiteboard note recognition

Joachim Schenk*, Johannes Lenz, Gerhard Rigoll

Technische Universität München, Theresienstraße 90, 80333 Munich, Germany

ARTICLE INFO

Article history:
Received 13 August 2008
Received in revised form 28 November 2008
Accepted 21 December 2008

Keywords:
On-line handwriting recognition
Whiteboard note recognition
Line-member
Script line identification
Preprocessing
Script normalization

ABSTRACT

When writing on a whiteboard, the writer stands rather than sits and the writing arm does not rest. Due to these adverse conditions when writing on a whiteboard, the script lines within the handwritten text suffer from high variations, i.e. they cannot be approximated by polynomials of low order. In this paper, we propose a novel method for identifying script lines in handwritten whiteboard notes by assigning the sample points of the script trajectory to the script lines. The optimal assignment is then found by the Viterbi algorithm. We present two ways to use the script line characterization. First, the script lines are used to normalize the skew and size of the text lines. In a second approach, the feature vector of a standard recognition system is augmented by the explicit script line membership of each sample point, aiming at reducing the confusions between characters differing in size rather than in shape (like "s" and "S" or "e" and "l"). As experiments show, a relative improvement of $r=3.3\%$ in character-level and $r=3.4\%$ in word-level accuracy compared to a baseline system can be achieved with the proposed script line identification method. In addition, the written character confusion as described above can be reduced. Finally, the proposed utilizations are examined and discussed in further detail.

1. Introduction

For the past 35 years, research has been conducted on recognizing handwritten notes, words, and sentences [1]. Handwriting recognition can be divided into *off-line* and *on-line* recognition [2–4]. In case of off-line recognition, a static image of the handwritten script serves as recognizer input, while in on-line recognition the movement of the pen is captured as trajectory [5].

Off-line and on-line recognizers suffer from errors introduced by wrongly presegmented words or characters [6,7], which creates a dependency between segmentation and recognition. This phenomenon is sometimes referred to as "Syres-Paradoxon" [8]. Hidden Markov models (HMMs), first introduced for automatic speech recognition (ASR) [9], enable a combined segmentation and recognition avoiding any error-prone pre-segmentation and have therefore become quite popular in both on-line and off-line handwriting recognition [4]. In handwriting recognition, each symbol (e.g. character) is represented by one single HMM. Words are recognized by combining several character HMMs using a dictionary [10,11]. While high recognition rates are reported for on-line handwriting recognition of isolated words (see for example [12]), recognition performance considerably drops when it comes to unconstrained handwritten sentence recognition.

1.1. Whiteboard note recognition

The recognition of handwritten text and symbols written on whiteboards has been increasingly studied [13,14], as whiteboards play an important role in the so-called "smart meeting room" scenarios (see for example [15–17]). However, in a whiteboard scenario the writer stands rather than sits and the writing arm does not rest, introducing additional variation. The script lines in the text cannot be approximated by a simple polynomial of low order. Furthermore, it has been observed that the size and width of characters and words vary to a higher degree on whiteboards than on tablets. These conditions contribute to the characterization of the problem of on-line whiteboard note recognition as "difficult."

The first attempts at recognizing a limited set of symbols written on a whiteboard are presented in [18]. Whole text lines written on a whiteboard were first recognized using HMMs in [7,13]. In both approaches listed above, the handwritten data on the whiteboard was captured by a camera, which limits these approaches to off-line recognition but allows the use of a standard whiteboard and a normal pen. Hence, the recording of the whiteboard notes is enabled at low cost.

* Corresponding author.
E-mail addresses: joa@mmk.ei.tum.de (J. Schenk), lej@mmk.ei.tum.de (J. Lenz), ri@mmk.ei.tum.de (G. Rigoll).

0031-3203/$ - see front matter © 2009 Elsevier Ltd. All rights reserved.
doi:10.1016/j.patcog.2008.12.015

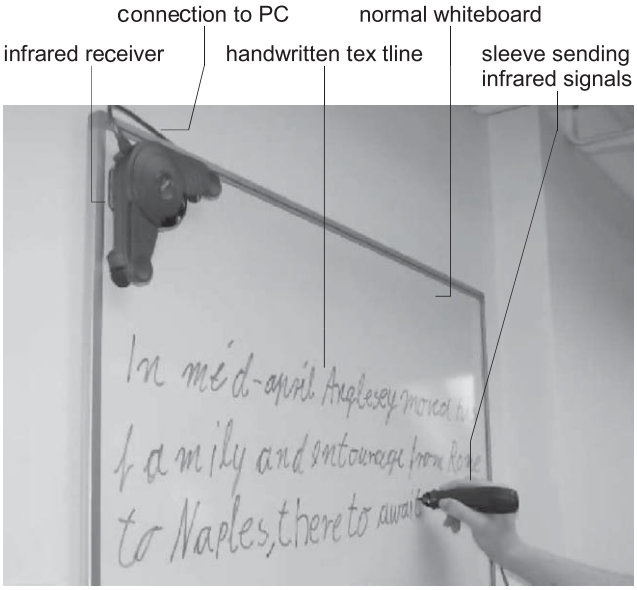

Fig. 1. Exemplary setup for recording handwritten whiteboard notes with the *eBeam*-system. Illustration adopted and altered from [25].

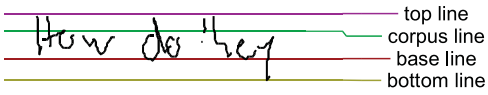

Fig. 2. Illustration of script lines within a (fraction of a) line of text as defined in e.g. [27,28]. Script sample has been taken from [25].

A number of commercial solutions exist for recording whiteboard notes both in an off-line and in an on-line manner. The handwritten as well as additionally attached notes are scanned by a large scanner mounted in front of the whiteboard with the Panasonic PANABOARD [19]. On-line recording of the script trajectory is enabled by the PolyVision WEBSTER [20] and the SMART Technology SMART BOARD [21] in which a sensitive surface captures the position of the pen. The trajectory is then either projected on the surface [21] or displayed on a screen [22].

More recently, new systems for recording whiteboard notes in an on-line fashion written on a standard whiteboard have been developed. The MIMIO INTERACTIVE-system by Mimio [23] and the EBEAM-system by Lucida [24] consist of two devices: a receiver, mounted to any corner of a standard whiteboard and a stylus pen sending either ultrasound or infrared signals to the receiver. While a special pen is necessary in the MIMIO INTERACTIVE-system, in the EBEAM-system a normal pen is put into a special sleeve and used for writing. These systems offer on-line information of the handwritten whiteboard notes at relatively low cost. The EBEAM-system is used in the IDIAP smart meeting room [15] and its recordings (e.g. provided by the IAM-onDB [25], see Section 5) are widely used for research in handwritten whiteboard note recognition [2,14]. Recognition of recordings gathered from the EBEAM-system was performed for the first time in [14,26] with an off-line approach and then in [2] with an on-line approach. In this paper, the recordings of the EBEAM-system as described in [25] are used. Fig. 1 shows an exemplary setup for recording handwritten whiteboard notes with the EBEAM-system.

1.2. Paper outline

Handwritten text can be separated into certain script lines (see for example [27,28]), 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 [28].

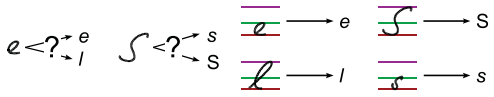

Fig. 3. Possible character confusions of characters that differ in size rather than in shape without script lines (left) and relative size description with script lines (right).

In this paper, the identification of the script lines in a line of text is addressed. Different approaches for identifying these script lines in a handwritten line of text have been published. Base lines and corpus lines are described by a linear regression approximating local minima and local maxima of the trajectory in [29]. In [30,31] the script lines are found by analyzing the profile of the y-projection of the handwritten script. All four script lines are approximated as parameterized curves of a second order polynomial in [27,28]. Their parameters are found by fitting a geometrical model to the trajectory by applying the expectation–maximization (EM) algorithm [28,32]. While these approaches work fine for normal handwriting, enhanced algorithms are needed for lines of text written on a whiteboard. To cope with the observed variations, in [2] a line of text is heuristically segmented into subparts in which the script lines are identified separately.

In this paper, we propose a novel approach for script line identification in text lines written on a whiteboard. First, sample points are found which are potential candidates for *defining* the script lines. Then, a trellis is built holding all possible sample point to script line assignments. The least costly path through that trellis is found by applying the Viterbi algorithm [33]. Finally, we iteratively refine the sample point assignment and obtain script lines, which may have any characteristic (i.e. any bend). The identified script lines are utilized in two ways. In the first approach, the script line-assignment is used for skew correction and script size normalization by “equalizing” the script lines: the script lines are forced to run both horizontally and straight and the handwritten script is morphed accordingly.

As illustrated in Fig. 3 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, these characters are distinguishable from each other. Following this reasoning, the second utilization of the script lines presented in this paper is the definition of a novel “line-member” feature, describing the script line association of certain sample points to the corresponding script lines. This aims to reduce the confusion between characters of similar shape and different size by adding additional discriminance.

1.3. Structure of the paper

The next section gives a brief overview of our baseline system, including the preprocessing used as initialization for the novel script line identification, the set of standard features, and the HMM-based recognizer. In Section 3 the novel script line identification is described. Section 4 introduces the utilization of the script lines for skew and size normalization, and as a novel “line-member” feature, whereby the line-member feature describes the assignment between the sample points and the script lines. In an experimental section (Section 5) the influence of both script line utilizations is examined. Conclusions are drawn and an outlook to further work is given in Section 6.

2. System overview

This section gives a brief overview of the recognition system, including the preprocessing, the feature extraction, and the HMM-based recognizer, used for the experiments in Section 5.

2.1. Preprocessing

The handwritten whiteboard data are first recorded using the eBeam-system as described in the Introduction. Fig. 1 shows the experimental setup. After recording, the handwritten data are represented by the sample points $s(t) = (x(t), y(t), p(t))^{\mathrm{T}}$, where $x(t)$ denotes the x-coordinate, $y(t)$ the y-coordinate, $p(t)$ the pressure of the pen at time instance t and $(\cdot)^{\mathrm{T}}$ the transpose of a matrix or a vector. Afterwards, the recorded data are heuristically segmented into lines [2]. Due to the recording system's limitations the sampling is neither space nor time equidistant. 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 [34]. Next, all text lines are normalized to meet a distance of "one" between the corpus and the baseline followed by a final resampling of the data. For the initial script line estimation, the corpus and baseline are assumed parallel and horizontal. The position of both lines is estimated using a histogram-based projection approach similar to [30]. This normalization serves as initialization for the novel script line identification as introduced in Section 3.

2.2. Standard feature extraction

Following the preprocessing, 24 features are extracted from the three-dimensional sample points in order to describe the handwritten data more thoroughly, and the feature vector $\mathbf{f}(t) = (f_1(t), \ldots, f_{24}(t))$ is derived. This 24-dimensional feature vector is enhanced by a 25th feature ($f_{n,25}(t)$) later in Section 4.2, which describes the "line membership" of each sample point.

The 24 state-of-the-art *on-line* and *off-line* features for handwriting recognition [2,27] used in this paper are briefly explained below. As on-line features we extract

$f_1(t)$indicating the pen "pressure", i.e.

$$f_1(t) = \begin{cases} 1 & \text{pen tip on whiteboard,} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

$f_2(t)$velocity equivalent, which is computed *before* resampling and later interpolated;
$f_3(t)$$x$-coordinate after resampling, initial script normalization, and subtraction of the moving average;
$f_4(t)$$y$-coordinate after resampling and initial script normalization;
$f_{5,6}(t)$angle α of consecutive strokes (coded as $\sin\alpha$ and $\cos\alpha$, the "writing direction");
$f_{7,8}(t)$difference of consecutive angles $\Delta\alpha = \alpha(t) - \alpha(t-1)$ (coded as $\sin\Delta\alpha$ and $\cos\Delta\alpha$, the "curvature").

In addition, certain on-line features describing the relation of the sample point $\mathbf{s}_t$ to its neighbors $\mathbf{s}(t-\tau), \mathbf{s}(t-\tau+1), \ldots, \mathbf{s}(t-1)$ were adopted and altered from those described in [2,27]:

$f_9(t)$ the aspect of the trajectory between the points $\mathbf{s}(t-\tau)$ and $\mathbf{s}(t)$ (referred to as "vicinity aspect"),

$$\nu = \left(\frac{\Delta y - \Delta x}{\Delta y + \Delta x}\right), \quad \begin{array}{l} \Delta x = x(t) - x(t-\tau), \\ \Delta y = y(t) - y(t-\tau). \end{array} \tag{2}$$

As ν tends to peak for small values of $\Delta y + \Delta x$, we narrow its range by

$$f_9(t) = \mathrm{sign}(\nu) \cdot \log(1 + |\nu|). \tag{3}$$

Hence, a logarithmic transform of the vicinity aspect is used as feature.

$f_{10,11}(t)$ angle φ between the line $\overline{\mathbf{s}(t-\tau)\mathbf{s}(t)}$ and the horizontal line (coded as $\sin\varphi$ and $\cos\varphi$, the "vicinity slope");
$f_{12}(t)$ length of the trajectory, normalized by the length $\max(|\Delta x|; |\Delta y|)$ ("vicinity curliness");
$f_{13}(t)$ average square distance to each point in the trajectory and the line $\overline{\mathbf{s}(t-\tau)\mathbf{s}(t)}$.

The second class of features extracted for the baseline system, the *off-line* features, are as follows:

$f_{14-22}(t)$ 3×3 subsampled bitmap slid along the pen's trajectory ("context map") to incorporate a 30×30 fraction of the currently written letter's actual image;
$f_{23-24}(t)$ number of pixels above, or, respectively, beneath the current sample point $\mathbf{s}_t$ (the "ascenders" and "descenders").

2.3. Recognizer

After the extraction of the features, the handwritten data are recognized by using a recognizer based on continuous HMMs [9]: each symbol (in this paper: characters) is modeled by one HMM. For comparability, the HMM topology is mainly adopted from [2]. However, in this paper use $N_{\mathrm{Gauss}} = 32$ Gaussian mixtures for approximating the output probabilities, instead of $N_{\mathrm{Gauss}} = 36$ as are used in [2]. Training of the HMMs is performed by the EM algorithm [32]. Using the Viterbi algorithm the handwritten data are recognized and segmented [9,33].

3. Script line identification

In this section we describe our novel approach for script line identification in on-line handwritten whiteboard note recognition. First, all sample point to script line assignments are modeled by a finite state machine and a reasonable reduction of sample point candidates for the script line description is made. Then a trellis-representation of the sample point to script line assignment, a suitable metric, and the identification of the optimal assignment are described. Finally, an iterative refinement of the sample point to script line assignment is given. Throughout this section, it is assumed that the handwritten data are preprocessed by the basic steps as summarized in Section 2.1.

3.1. Finite state machine modeling

As explained in the Introduction, the four script lines are defined by certain sample points. However, it is not always clear which sample point lies on a specific script line, making the sample point to script line assignment unknown. If the association between sample points and script lines is known, the characteristics of the script lines can be derived. This is the basic principle underlying our approach: Identifying the script line characteristics by finding the corresponding sample points lying on the specific line.

This can be modeled by the finite state machine as depicted in the upper part of Fig. 4: for each sample point $\mathbf{s}(t)$, the script line assignment $l(t) = i$ is represented by one state s_i in the finite state machine, with $i = 0$ when the sample point is assigned to *no* line and $i = \{1, \ldots, 4\}$ for the top, corpus, base, and bottom line, respectively. A transition between two states s_i and s_j is made, if the sample point $\mathbf{s}(t-1)$ is assigned to line i and the sample point $\mathbf{s}(t)$ is assigned to

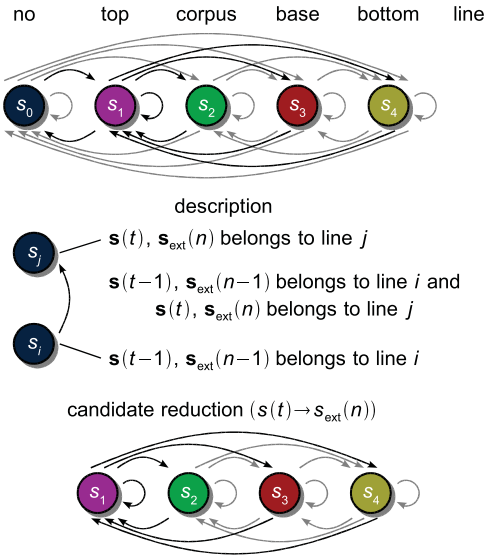

Fig. 4. Finite state machine for modeling all possible sample point to script line assignments (upper part), description (middle) and reduced finite state machine (lower part) for assigning extreme points ($\mathbf{s}_{\text{ext}}$) to the script lines. For the purposes of clarity, some transitions are grayed.

line j, hence $l(t-1)=i$ and $l(t)=j$. As all states are interconnected, each sample point $\mathbf{s}(t)$ of the T sample points in $\mathbf{S}=\{s_1,\ldots,s_T\}$ may be assigned to any of the $N_{\text{l}}=4$ script lines regardless of the assignment $l(t-1)$ of the previous sample point $\mathbf{s}(t-1)$. This leads to $N_{\text{tot}}=(N_{\text{l}}+1)^T$ different mappings, out of which one describes the "correct" sample point to script line assignment. If $T\approx 100$ is assumed (this assumption is valid for the database used for the experiments in Section 5), $N_{\text{tot}}\approx 7.8\times 10^{69}$ different mappings have to be investigated.

3.2. Candidate reduction

In order to reduce the number of possible sample point to script line assignments, first the number of potential candidates defining the script lines is reduced. Meeting the script line definitions given in the introduction we use *spatial* extreme points[1] $\mathbf{s}_{\text{ext}}(n)\in\mathscr{S}_{\text{ext}}$. For the extreme points, local minima and local maxima are extracted from the text line $\mathbf{S}$ according to

$$\begin{aligned}\mathscr{S}_{\min}&=\{\mathbf{s}(t)|y(t)<y(t-1)\wedge y(t)<y(t+1)\},\\ \mathscr{S}_{\max}&=\{\mathbf{s}(t)|y(t)>y(t-1)\wedge y(t)>y(t+1)\},\end{aligned}\tag{4}$$

$2\leqslant t\leqslant T-1$. The extreme points are then derived to $\mathscr{S}_{\text{ext}}=\mathscr{S}_{\min}\cup\mathscr{S}_{\max}$, with

$$\begin{aligned}&\mathbf{s}_{\min}(n)\in\mathscr{S}_{\min},\quad 1\leqslant n\leqslant N_{\min}=|\mathscr{S}_{\min}|,\\ &\mathbf{s}_{\max}(n)\in\mathscr{S}_{\max},\quad 1\leqslant n\leqslant N_{\max}=|\mathscr{S}_{\max}|,\end{aligned}\tag{5}$$

and $N_{\text{ext}}=N_{\min}+N_{\max}$ being the total number of extreme points. Hence, in the following the script lines are defined by the extreme points.

As each extreme point $\mathbf{s}_{\text{ext}}(n)$ is assumed to lie on a script line, in a second step the finite state machine defining the script line assignment described in Section 3.1 can be reduced by the first state yielding the reduced state machine as displayed in the lower part of Fig. 4. However, even with the reduced number of sample points and

[1] Some authors recommend an estimation of extreme points in the velocity domain rather than in the spatial domain (see for example [35]). However, as mentioned in Section 2 the sample rate of the recording system differs in a wide range, inhibiting a robust estimation of extreme points in the velocity domain.

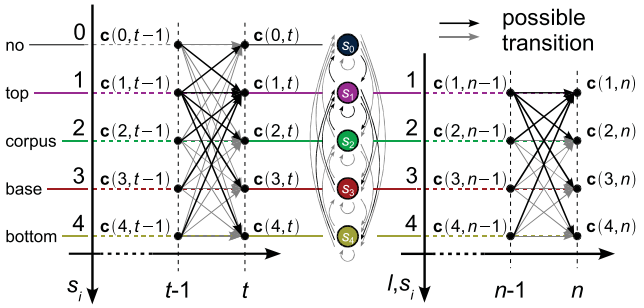

Fig. 5. Trellis representation of the finite state machine as shown in the upper part of Fig. 4 (left) and the reduced finite state machine as shown in the lower part of Fig. 4 (right). For the purposes of clarity, some transitions are grayed.

states in the finite state machine $N_{\text{tot}}=(N_{\text{l}})^{N_{\text{ext}}}$ (in case of $N_{\text{ext}}\approx 30$, $N_{\text{tot}}\approx 1.2\times 10^{18}$) different extreme point to script line assignments can be found, which is computationally infeasible. Therefore a further reduction of the number of assignments is necessary.

3.3. Trellis representation

Although reduced, all possible script line to extreme point assignments cannot be searched for the optimal assignment by an exhaustive search. The question therefore arises as to how the best extreme point to script line assignment can be found with lower cost. For a temporal interpretation of the finite state machines displayed in Fig. 4, they are evoluted in a trellis diagram as shown in Fig. 5 for a transition from $t-1$ to t and $n-1$ to n, respectively. The assignment of the extreme point $\mathbf{s}_{\text{ext}}(n)$ to script line l is represented by the trellis node $\mathbf{c}(l,n)$ and is called assignment "hypothesis" in the following. Additionally, all possible transitions between states are displayed in Fig. 5 and marked by arrows. Each path through the trellis represents one certain mapping of all extreme points to the script lines.

The extreme points which are assigned to the script line l are contained in the set $\mathscr{L}_l$, where

$$\mathbf{s}_{\text{ext}}(n)\in\mathscr{L}_l \text{ if } \mathbf{s}_{\text{ext}}(n) \text{ is assigned to script line } l.\tag{6}$$

The script line l is therefore characterized by the consecutive extreme points $\mathbf{s}_{\text{ext}}(n)\in\mathscr{L}_l$. For each script line l an initial y-position $c_l(0)$, $1\leqslant l\leqslant N_{\text{l}}$, where each $c_l(0)$ belongs to $\mathbf{c}(0)=(c_1(0),\ldots,c_{N_{\text{l}}}(0))^{\text{T}}$, is defined as

$$\mathbf{c}(0)=\left(\max_{1\leqslant n\leqslant N_{\text{ext}}} y_{\text{ext}}(n),1,0,\min_{1\leqslant n\leqslant N_{\text{ext}}} y_{\text{ext}}(n)\right)^{\text{T}},\tag{7}$$

as the line of text is normalized to a corpus–base line distance of "one" during preprocessing (see Section 2.1). The absolute y-distance between the current extreme point $\mathbf{s}_{\text{ext}}(n)$ and the script line l is

$$m(l,n)=|y_{\text{ext}}(n)-c_l(0)|.\tag{8}$$

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

$$\hat{l}(n)=\operatorname*{argmin}_{1\leqslant l\leqslant N_{\text{l}}} m(l,n)\ \Rightarrow\ \mathbf{s}_{\text{ext}}(n)\in\mathscr{L}_{\hat{l}(n)},\tag{9}$$

where $\hat{l}(n)$ is the script line assignment hypothesis of extreme point $\mathbf{s}_{\text{ext}}(n)$. However, the simple assignment described in Eq. (9) is not valid for handwritten whiteboard notes—as pointed out in [2], 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

extreme 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 extreme point, whereas the following extreme points may decide whether this assignment is valid. In the next section we therefore enhance the trellis nodes and define the trellis-based script line-assignment using the mapping hypothesis.

3.4. Line assignment

As explained in Section 3.1, for each extreme point $\mathbf{s}_{\text{ext}}(n)$, $\mathbf{s}_{\text{ext}}(n) \in \mathscr{S}_{\text{ext}}$, $1 \leqslant n \leqslant N_{\text{ext}}$ an assignment to all script lines l $(1 \leqslant l \leqslant N_{\text{l}})$ is assumed. This hypothesis is represented in the trellis nodes $\mathbf{c}(l,n) = (c_1(l,n), \ldots, c_{N_{\text{l}}}(l,n))^{\text{T}}$, $1 \leqslant l \leqslant N_{\text{l}}$, $1 \leqslant n \leqslant N_{\text{ext}}$ of an $N_{\text{l}} \times N_{\text{ext}}$ trellis (see Fig. 5), where $c_i(l,n)$ holds the current absolute y-position of script line i if $\mathbf{s}_{\text{ext}}(n)$ is assigned to the script line l. The variation in the characteristics of the script line l (expressed as the absolute value of the change in its y-position) introduced by the association of the current extreme point $s_{\text{ext}}(n)$ is captured by the metric $M_n(l)$ (which is more formally defined in Eqs. (10) and (11)). A high value of $M_n(l)$ indicates a high variation of the text lines, whereas a small value describes horizontally running script lines.

As transitions in the trellis are made with respect to the assignment of the previous extreme point, e.g. if $\mathbf{c}(2,n-1)$ is followed by $\mathbf{c}(4,n)$, $\mathbf{s}_{\text{ext}}(n-1)$ is assigned to the corpus line *and* $\mathbf{s}_{\text{ext}}(n)$ is assigned to the bottom line (see Section 3.1), the current line-assignment depends on the assignment of the subsequent extreme point $\mathbf{s}_{\text{ext}}(n+1)$.

In order to find an, in some sense, optimal assignment between the extreme points and the script lines, the transitions in the trellis are made with respect to minimizing the metric $M_n(l)$ for each trellis node $\mathbf{c}(l,n)$ which describes the accumulated absolute variation of each script line. The final extreme point to script line assignment is derived from the path through the trellis yielding the smallest metric $M_{N_{\text{ext}}}(l)$. A path variable $\psi_n(l)$ is therefore used to store the preceding trellis node. Hence, the metric $M_n(l)$, the path variable $\psi_n(l)$ and the trellis nodes $\mathbf{c}(l,n)$ are derived as follows: first the metric $M_1(l)$, the path variable $\psi_1(l)$, and the trellis nodes $\mathbf{c}(l,1)$ corresponding to the first extreme point $\mathbf{s}_{\text{ext}}(1)$ are initialized using Eq. (7):

$$\begin{aligned} &M_1(l) = |c_l(0) - y_{\text{ext}}(1)|, \\ &\psi_1(l) = 0, \\ &c_i(l,1) = \begin{cases} y_{\text{ext}}(1) & \text{if } l = i, \\ c_i(0) & \text{otherwise.} \end{cases} \end{aligned} \qquad 1 \leqslant i,l \leqslant N_{\text{l}}. \tag{10}$$

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

$$\begin{aligned} &M_n(l) = \min_{1 \leqslant i \leqslant N_{\text{l}}} (M_{n-1}(i) + |c_l(i,n-1) - y_{\text{ext}}(n)|), \\ &\psi_n(l) = \operatorname*{argmin}_{1 \leqslant i \leqslant N_{\text{l}}} (M_{n-1}(i) + |c_l(i,n-1) - y_{\text{ext}}(n)|), \\ &c_i(l,n) = \begin{cases} y_{\text{ext}}(n) & \text{if } l = i, \\ c_i(\psi_n(l), n-1) & \text{otherwise,} \end{cases} \end{aligned} \tag{11}$$

with $2 \leqslant n \leqslant N_{\text{ext}}$ and $1 \leqslant i,l \leqslant N_{\text{l}}$. If the current extreme point $\mathbf{s}_{\text{ext}}(n)$ is assigned to script line l the characteristics of script line l are changed such that it runs through the coordinates of that point. This is expressed by $c_l(l,n) = y_{\text{ext}}(n)$. However, the characteristics of the remaining script lines i, $1 \leqslant i \leqslant N_{\text{l}}$, $i \neq l$ are not affected by this assignment and the position information of these lines is taken from the trellis node of the preceding extreme point $\mathbf{s}_{\text{ext}}(n-1)$ yielding the lowest metric $M_{n-1}(l)$, and deriving $c_i(l,n) = c_i(\psi_n(l), n-1)$. The updating defined by Eq. (11) is illustrated in Fig. 6 for an exemplary transition from $\mathbf{c}(2,n-1)$ to $\mathbf{c}(4,n)$ resulting in a minimum weight, and hence $\mathbf{s}_{\text{ext}}(n-1)$ is assigned to the corpus line and $\mathbf{s}_{\text{ext}}(n)$ is assigned to the bottom line.

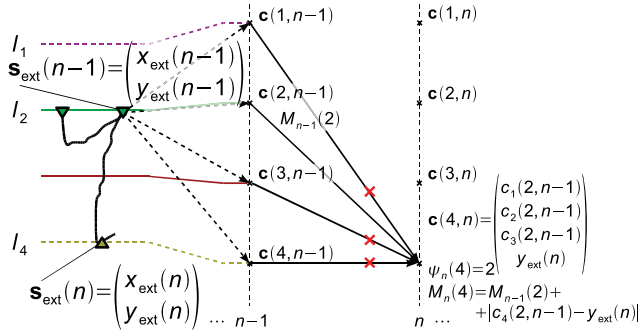

Fig. 6. The information of the script line positions stored in trellis node $\mathbf{c}(4,n)$ of script line $l=4$ is updated by the y-position $y_{\text{ext}}(n)$ of the current extreme point $\mathbf{s}_{\text{ext}}(n)$ $(l=4)$ and the position information $(c_1(2,n-1)$, $c_2(2,n-1)$ and $c_1(3,n-1))$ of the preceding trellis node $\mathbf{c}(2,n-1)$ in case that a transition from $\mathbf{c}(2,n-1)$ to $\mathbf{c}(4,n)$ leads to the smallest metric $M_n(4)$ (assigning $\mathbf{s}_{\text{ext}}(n-1)$ to the corpus line and $\mathbf{s}_{\text{ext}}(n)$ to the bottom line). For the purposes of clarity, some transitions have been omitted. The script trajectory has been slightly interpolated for improved display.

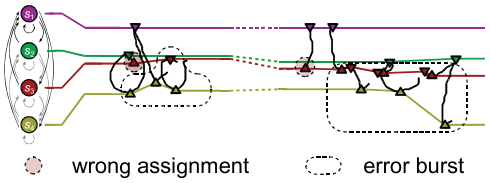

Fig. 7. Burst error in the assignment of successive extreme points after one previous extreme point has been assigned to the wrong script line. Script sample taken from [25].

The final script line assignment $\hat{l}(n)$ for each extreme point $\mathbf{s}_{\text{ext}}(n)$ is found via backtracking:

$$\begin{aligned} &\hat{l}(N_{\text{ext}}) = \operatorname*{argmin}_{1 \leqslant l \leqslant N_{\text{l}}} M_{N_{\text{ext}}}(l) \;\Rightarrow\; \mathbf{s}_{\text{ext}}(n) \in \mathscr{L}_{\hat{l}(n)}, \\ &\hat{l}(n) = \psi_{n+1}(\hat{l}(n+1)), \quad n = N_{\text{ext}}-1, \ldots, 1, \end{aligned} \tag{12}$$

yielding the explicit assignments $\mathbf{s}_{\text{ext}}(n) \in \mathscr{L}_{\hat{l}(n)}$. This implements the well-known Viterbi algorithm [9,33].

It shall be noted that the script line-assignment as performed by adequately applying Eqs. (10)–(12) allows for the incorporation of constraints on the characteristics of the script lines, e.g. that script lines may not cross or lie on each other. This is expressed by the condition

$$c_1(l,n) > \cdots > c_{N_{\text{l}}}(l,n), \quad \begin{matrix} 1 \leqslant l \leqslant N_{\text{l}}, \\ 1 \leqslant n \leqslant N_{\text{ext}}. \end{matrix} \tag{13}$$

All paths through the trellis in which at least one extreme point to script line assignment violates Eq. (13) are omitted.

3.5. Refinement

The basic trellis approach as introduced in the above section and explicitly expressed by Eqs. (10)–(12) 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 mischaracterized 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. 7.

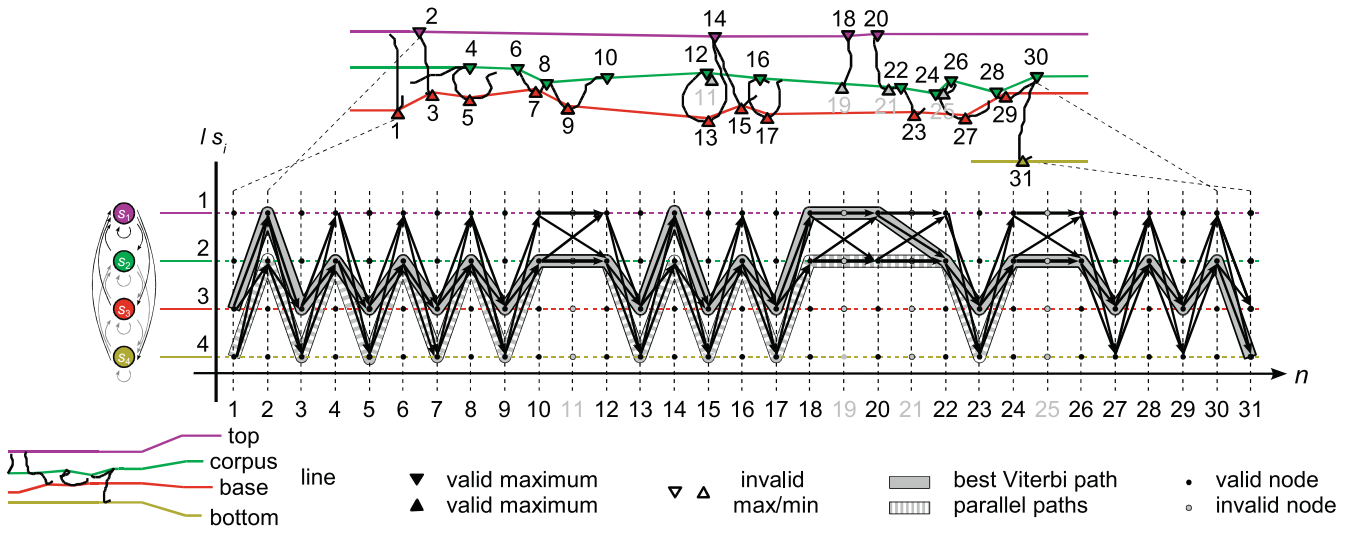

Fig. 8. Assignment between certain extreme points within a fraction of a text line and the script line is found by applying the Viterbi algorithm in order to find the least costly path through a trellis built by the algorithms as presented in the Sections 3.4 and 3.5. Script sample taken from [25]. For the purposes of clarity, some transitions have been omitted.

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

1. the extreme points contained in $\mathscr{S}_{\min}$ belong to the bottom and base line, whereas $\mathscr{S}_{\max}$ consists of extreme 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 extreme points. The base line and the corpus line are assumed to be the main line for $\mathscr{S}_{\min}$ and $\mathscr{S}_{\max}$, respectively.

The first assumption is motivated by the definition of the script lines as given in the introduction and is commonly used, see [2,27,28]. 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 $\mathscr{S}_{\min}$ and $\mathscr{S}_{\max}$ as defined in Eq. (5), 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 extreme points N_{main} assigned to the main line is counted. After this first assignment, both sets are iteratively reduced by one extreme point and the assignment is repeated on the reduced sets. In cases where the number of extreme points assigned to the main lines is higher than for the initial assignment, the omitted extreme 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.

Fig. 8 shows the exemplary trellis that results if the basic algorithm presented in Section 3.4 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 extreme point, a final decision on the script line-assignment of all *preceding* extreme points is made.

4. Normalization and "line-member" feature

In this section we suggest two methods of how to utilize the script lines found by the algorithms described in Section 3: first, they are used for normalizing the skew and size of the text line as presented in [36]. A second approach uses the line assignment of the extreme points to augment the feature by describing the line membership of each sample point yielding a 25th feature $f_{n,25}(t)$ [37].

Algorithm. Extended script line-assignment.

Data: $\mathscr{S}_{\min}, \mathscr{S}_{\max}$
Result: script line-assignment
forall the $\mathscr{S} \in \{\mathscr{S}_{\min}, \mathscr{S}_{\min}\}$ **do**
  $\mathscr{S}_{\text{proc}} = \mathscr{S} = \{\mathbf{s}(1), \ldots, \mathbf{s}(|\mathscr{S}_{\text{proc}}|)\}$;
  imp = 1;
  while imp = =1 **do**
    $N_{\text{main}}(0) =$ Anzahl Extrempunkte auf Hauptlinie($\mathscr{S}_{\text{proc}}$); nach Gleichungen (10)–(12)
    forall the $i \in \{1, \ldots, |\mathscr{S}_{\text{proc}}|\}$ **do**
      $N_{\text{main}}(i) =$ Anzahl Extrempunkte auf Hauptlinie($\{\mathscr{S}_{\text{proc}} \backslash \mathbf{s}(i)\}$); nach Gleichungen (10)–(12)
    if $\hat{i} = \underset{0 \leqslant i \leqslant |\mathscr{S}_{\text{proc}}|}{\operatorname{argmax}} N_{\text{main}}(i) = =0$ **then**
      imp = 0
    else
      $\mathscr{S}_{\text{proc}} = \{\mathscr{S}_{\text{proc}} \backslash \mathbf{s}(\hat{i})\}$

4.1. Script line normalization

After assigning all extreme points to the script lines according to Eq. (6), by properly applying the methods as described in Section 3, the script lines can be equalized, i.e. the particular assigned points $\mathbf{s}_{\text{ext}}(n) \in \mathscr{L}_l$ of each script line are shifted in order to lie on a horizontal line and meet the same y-position for all text lines. The target heights r_l, $1 \leqslant l \leqslant N_l$ of the script line l are set to $\mathbf{r} = (2, 1, 0, -1)^{\mathrm{T}}$. By warping each sample point $\mathbf{s}(t) = (x(t), y(t))^{\mathrm{T}}$ of the script trajectory, which is limited by the script lines according to the shifts of the supporting points, the script trajectory is normalized both in skew and in size.

To perform the mentioned warping of the script trajectory, the script lines are interpolated between the supporting points. In our paper this is done by a linear interpolation

$$\hat{y}_l(x(t)) = y_l(n-1) + \frac{y_l(n) - y_l(n-1)}{x_l(n) - x_l(n-1)} \cdot (x(t) - x_l(n-1)), \tag{14}$$

where $\hat{y}_l(x(t))$ is the linearly interpolated y-position of script line l at the x-position $x(t)$; $x_l(n-1)$, $x_l(n)$ and $y_l(n-1)$, $y_l(n)$ denote the x- and y-position of the supporting points $\mathbf{s}_{\text{ext}}(n-1), \mathbf{s}_{\text{ext}}(n) \in \mathscr{L}_l$ lying closest to the sample point $\mathbf{s}(t)$. The warped y-position $\tilde{y}(t)$ of each sample point $\mathbf{s}(t)$ of the script trajectory $\mathbf{S}$ is given by

$$\tilde{y}(t) = r_{l2} + \frac{y(t) - \hat{y}_{l2}(x(t))}{\hat{y}_{l1}(x(t)) - \hat{y}_{l2}(x(t))} \cdot (r_{l1} - r_{l2}), \tag{15}$$

where l_1 and l_2 ($l_1 < l_2$) are the two script lines in between which $\mathbf{s}(t)$ lies. To cope with horizontal distortions due to the vertical warping,

$$s = \frac{1}{T} \sum_{t=1}^{T} \frac{\hat{y}_{l2}(x(t)) - \hat{y}_{l1}(x(t))}{r_{l2} - r_{l1}} \tag{16}$$

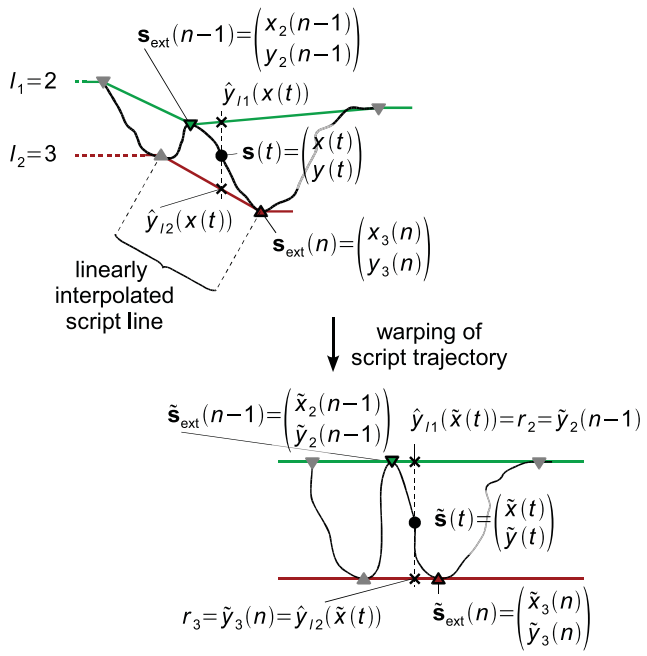

Fig. 9. Script trajectory before novel normalization (upper part) and normalized script trajectory (lower part) after shifting the extreme points $\mathbf{s}(n-1)$ and $\mathbf{s}(n)$ and the sample point $\mathbf{s}(t)$ in order to achieve parallel and horizontally running script lines. The script trajectory has been slightly interpolated for better illustration.

is derived and horizontal scaling is performed by

$$\tilde{x}(t) = s \cdot x(t), \quad 1 \leqslant t \leqslant T, \tag{17}$$

with $\tilde{x}(t)$ the warped x-position of the sample point $\mathbf{s}(t)$. As a result the warped sample point $\tilde{\mathbf{s}}(t) = (\tilde{x}(t), \tilde{y}(t))^{\mathrm{T}}$ is obtained. The warping procedure is illustrated in Fig. 9. The extreme points $\mathbf{s}_{\text{ext}}(n)$ assigned to the script lines are shifted according to Eqs. (15) and (17), resulting in the shifted extreme points $\tilde{\mathbf{s}} = (\tilde{x}_l(n), \tilde{y}_l(n))^{\mathrm{T}}$. All extreme points $\mathbf{s}_{\text{ext}}(n) \in \mathscr{L}_l$ defining the script line l share the same y-coordinate $\tilde{y}_l(n) = r_l$ after warping as indicated in the lower part of Fig. 9.

The result of the normalization of whole text line using the warping as expressed in Eqs. (15) and (17) is shown in the lower part of Fig. 10. After normalization of the script trajectories according to Eqs. (17) and (15), resampling is performed in order to achieve space equidistantly distributed sample points (see Section 2.1), and subsequently the 24 standard features are extracted as explained in Section 2.2.

4.2. Line-member

As explained in the Introduction, characters that 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 $f_{n,25}(t)$ is given by

$$f_{n,25}(t) = \begin{cases} 0 & \text{if } \mathbf{s}(t) \text{ lies on } \textit{no} \text{ line,} \\ 1 & \text{if } \mathbf{s}(t) \text{ lies on top line,} \\ 2 & \text{if } \mathbf{s}(t) \text{ lies on corpus line,} \\ 3 & \text{if } \mathbf{s}(t) \text{ lies on base line,} \\ 4 & \text{if } \mathbf{s}(t) \text{ lies on bottom line.} \end{cases} \tag{18}$$

After reducing the number of line-member candidates to the spatial extreme points in the script trajectory, the line assignment (see Section 3.4), and the subsequent refinement (see Section 3.5) the final assignment between the extreme points $\mathbf{s}_{\text{ext}}(n)$, $1 \leqslant n \leqslant N_{\text{ext}}$ and the script line l is given by the sets $\mathscr{S}_l$. Hence, the value assignment of the line-member feature as defined in Eq. (18) derives to

$$f_{n,25}(t) = \begin{cases} l & \text{if } \mathbf{s}(t) \in \mathscr{S}_l, 1 \leqslant l \leqslant N_{\text{l}}, \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

The former feature vector $\mathbf{f}(t)$ is augmented by the feature defined by Eqs. (18) and (19), yielding the 25-dimensional feature vector $\mathbf{f}_n(t) = (\mathbf{f}^{\mathrm{T}}(t), f_{n,25}(t))^{\mathrm{T}}$. In contrast to the novel normalization scheme as explained in Section 4.1, the position of the sample points is not changed and a further resampling is therefore not necessary.

5. Experimental results

The experiments presented in this section are conducted on a database containing $N_{\text{tot}} = 13\,049$ handwritten and heuristically

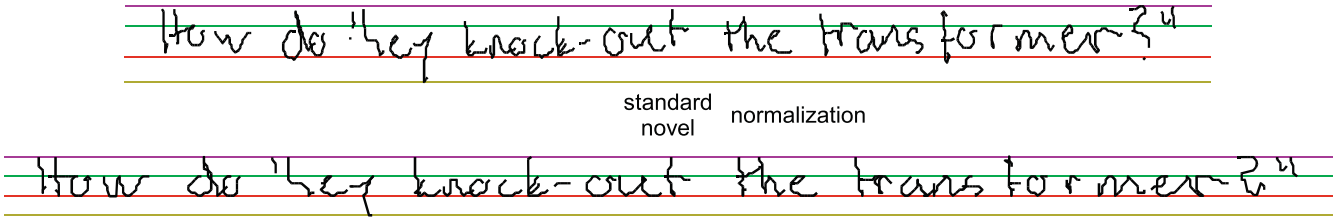

Fig. 10. Script trajectory after standard normalization (upper part) and after applying the novel preprocessing (lower part) presented in this paper. Script sample taken from [25].

Table 1
Summary of the results obtained by the experiments conducted in Section 5 given as character-level accuracy measured on the validation set and word-level accuracy gained on the test set of the IAM-OnDB-t1 benchmark [38].

	Baseline (*Experiment* 1)	Novel normalization (*Experiment* 2)	Line-member feature (*Experiment* 3)	Reference [25] (*Ref. system*)
Char. ACC (on validation set) relative improvement $\Delta r(p_N)$	$a_b = 61.2\%$	$a_{norm} = 62.2\%$	$a_{lm} = 63.3\%$	–
(compared to baseline)	–	1.6% (0.91)	3.3% (> 0.99)	
(compared to a_{norm})		–	1.7% (0.93)	
Word. ACC (on test set)	$A_b = 62.6\%$	$A_{norm} = 63.5\%$	$A_{lm} = 64.8\%$	$A_{[25]} = 65.2\%$
relative improvement $\Delta r(p_N)$				$\Delta r_{base} = 4.0\%$
(compared to baseline)	–	1.4% (0.92)	3.4% (> 0.99)	$\Delta r_{norm} = 2.6\%$
(compared to A_{norm} and/or A_{lm})		–	2.0% (0.98)	$\Delta r_{lm} = 0.6\%$

Additionally the relative changes Δr in recognition performance of the various systems as well as the significance level p_N are given. *Notes to the editor*: In the final paper format a greater font size for this table will be used. This table is designed for being spread over two columns.

line-segmented whiteboard notes (IAM-OnDB[2]). For further information on the IAM-OnDB see [25]. Comparability of the results is provided by using the well-defined settings of the writer-independent IAM-onDB-t1 benchmark, consisting of 56 different characters, and an 11k dictionary. It provides four data-sets: one for training, two for validation, and one for testing. The recognition of the handwritten whiteboard notes is performed independently of the writer. Hence, text lines written by one specific writer appear in just one of the four data-sets. The training set defined by the IAM-onDB-t1 benchmark consists of $N_{train} = 5365$ text lines; the two validation sets consist of $N_{val,1} = 1438$ and $N_{val,2} = 1518$ text lines, and the test set consists of $N_{test} = 3857$ text lines. Since only one validation set is required in this paper, the combination of both validation sets provided by the IAM-OnDB-t1 benchmark serves as validation set, containing $N_{val} = 2956$ lines of text. In sum $N_{t1} = 12\,177$ text lines out of the $N_{tot} = 13\,049$ text lines of the IAM-OnDB are used for the experiments presented in this section.

The parameters of the HMMs used for recognition are estimated from the data in the training set. In order to avoid over-fitting of the models to the training data, after each iteration character-level recognition is performed on the validation set. Training continues, until no improvement on the validation set can be achieved. For measuring the performance of the systems the *accuracy* (ACC) is used:

$$\mathrm{ACC}: a = 100 \cdot \left(1 - \frac{N_{ins} + N_{subs} + N_{del}}{N_{occ}}\right) \ (\%), \tag{20}$$

with N_{ins} the number of insertions, N_{subs} the number of substitutions, and N_{del} the number of deletions of the recognizer's transcriptions compared to the transcriptions of the IAMonDB, referring to a data-set with a total of N_{occ} character and words, respectively.

In all our experiments presented in this section, the recognition accuracy obtained on the validation set is given on the *character*-level (the number of characters in the validation set is $N_{occ} = 71\,193$). The best performing parameters are then used for a final test on the test set. The recognition performance measured on the test set is given as *word*-level accuracy (the number of words in the test set is $N_{occ} = 24\,685$). By applying the same language model as in [38] for the final test, comparability to the results as, for example, presented in [38] is provided. For our experiments, the same HMM topology as in [2] is used in all our systems. However, in contrast to [2], the observation probability of the HMMs is modeled with $N_{Gauss} = 32$ instead of $N_{Gauss} = 36$ Gaussian mixtures. Statistical significance of the results is, whenever possible, proven by the one sided t-test giving the probability p_N of rejecting the hypothesis "both approaches perform equally."

[2] http://www.iam.unibe.ch/~fki/iamnodb/

Fig. 11. Although the correct sample point to script line assignment is found, after applying the novel normalization scheme a wrongly normalized trajectory is the result, causing a reduction in character-level accuracy.

Experiment 1: In the first experiment, the baseline system (*System* 1) is evaluated. It uses the standard preprocessing (see Section 2.1) and the 24 standard features (see Section 2.2). A character-level accuracy of $a_b = 61.2\%$ (obtained on the validation set) and a word accuracy of $A_b = 62.6\%$ for the final test are achieved. Both results are shown in the second column of Table 1. These results form the baseline with which other results are compared.

Experiment 2: The influence of the novel script line normalization is shown in the second experiment. Thereby, in *System* 2 the same 24-standard features as for *System* 2 are extracted from the sample points of the script trajectory which have been normalized as explained in Section 4.1. The results are shown in the third column of Table 1: on the character-level an accuracy of $a_{norm} = 62.2\%$ is obtained which translates to a relative improvement of $\Delta r = 1.6\%$ compared to the performance of *System* 1. This improvement is statistically significant ($p_N = 0.91$). When a word-level recognition of the test-set is conducted, an accuracy of $A_{norm} = 63.5\%$ can be reported, which denotes a relative improvement of $\Delta r = 1.4\%$ when compared to *System* 1. Again, this is a statistically significant improvement ($p_N = 0.92$).

While the novel normalization scheme significantly improves the overall recognition performance, in some cases the novel normalization introduces errors which are not observable without the novel normalization. This is shown in Fig. 11 taking the character "a" as example. Depending on the character's shape, the novel normalization can lead to a script trajectory suffering from higher distortions than before the normalization. As can be seen in Fig. 11, all extreme points are assigned to the correct script line. However, since a part of the character's trajectory runs above an extreme point lying on the corpus line, this part is located above the corpus line after normalization. This leads to the observed additional distortion. In these cases a *local* normalization will be used in future approaches to limit the influence of the novel normalization on already correctly positioned parts of the characters.

Another issue with the proposed normalization technique is illustrated on the left-hand side of Fig. 13, in which a wrong sample point to script line assignment leads to an incorrect script line estimation. Once the text line is normalized according to the script lines, features are extracted from the script trajectory. As both the on-line and the off-line features as introduced in Section 2 are derived from

Table 2
Absolute count of character confusions of selected character pairs without and with the novel line-member feature (f_{25}).

System	e ↔ l	s ↔ S	a ↔ d
Novel feature			
Without	388	392	156
With	39	193	90

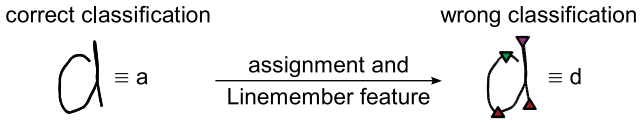

Fig. 12. While a correct classification is possible without the line-member feature (left), a character confusion occurs if a wrong line-member feature value is used for recognition (right).

the script trajectory, a wrong normalization influences *all* features, which introduces additional recognition errors.

However, as the results indicate, the errors caused by the above-mentioned additional distortions introduced by our novel normalization rarely occur. This leads to an improvement of the overall recognition accuracy of our recognition system.

Experiment 3: In the last experiment, the benefit of the line-member feature is examined. Taking *System* 1 as a start, the feature vector is augmented by the explicit information whether the current sample point lies on a script line. In case it does, the information on which of the four script line the sample point lies is coded into the feature value according to Section 4.2. This forms *System* 3. No further preprocessing is performed. Table 1 shows some rather promising results: while a character-level accuracy of $a_{\mathrm{lm}} = 63.3\%$ conducted on the validation set and therefore a relative improvement of $\Delta r = 3.3\%$ compared to *System* 1 can be observed, a slightly higher relative improvement of $\Delta r = 3.4\%$ in word-level accuracy (yielding $A_{\mathrm{lm}} = 64.8\%$) on the test set is achieved. Both improvements are statistically highly significant ($p_N > 0.99$ in both cases).

As explained in the Introduction, the main motivation of the line-member feature is to distinguish between characters that differ in size rather than in shape. The absolute counts of mutual character confusions of selected character pairs for the baseline system are shown in the second line of Table 2. In order to prove the reduction of the character confusions, the same confusion analysis has been performed on the system using the line-member feature. The result is shown in the third line of Table 2. As intended, the absolute count of mutual character confusions drops. However, there are still some confusions.

The remaining, observable confusions are caused by an improper feature extraction: in some cases the sample points are not assigned to the correct script lines. By carefully examining the examples showing character confusions, two impacts of false line membership can be identified. First, an erroneous line-member feature leads to the same confusion as without the line-member feature. Second, the line-member feature introduces novel character confusions compared to the baseline system. This is illustrated in Fig. 12. The left part of Fig. 12 shows a trajectory of the character "a" with an unusual, yet not unnatural ascender. When using the baseline system for recognition, the character is recognized correctly. Due to a wrong extreme point to script line assignment, as depicted on the right-hand side of Fig. 12, the top of the ascender is associated with top line. This leads to a character confusion, as in the novel system which utilizes the line-member feature, the same character is recognized as "d". However, the overall number of mutual character confusions can still be reduced by the line-member feature.

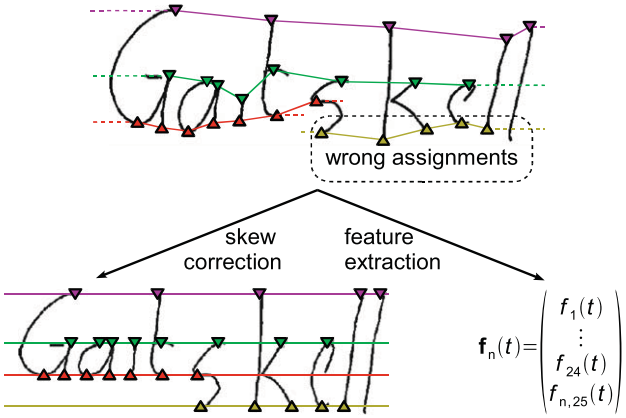

Fig. 13. Influence of wrong extreme point to sample point assignment on the novel normalization (left) and the line member feature (right). While in case of the novel normalization *all* features are influenced by the wrong assignment, the influence is limited *solely* to the line-member feature.

Compared to *System* 2, which also utilizes the sample point to script line assignments (see experiment 2), a significant, relative improvement of $\Delta r = 1.7\%$ ($p_N = 0.93\%$) in character-level accuracy and $\Delta r = 2.0\%$ ($p_N = 0.98$) in word accuracy can be reported. A reasonable explanation for this observation is the influence of misclassified sample points leading to wrongly characterized script lines as mentioned earlier and displayed in Fig. 13, left. After normalization, each feature in the feature vector $\mathbf{f}(t)$ is influenced by the erroneous script lines. However, in case of the line-member feature, the wrong extreme point to script line assignment influences one single feature and hence, delivers a higher improvement in both character- and word-level accuracy.

Reference system: In order to prove the competitiveness of our systems, our results are compared to the recently published result of a state-of-the-art HMM-based recognition system evaluated using the same IAM-OnDB benchmark. The word-level accuracy as published in [38] is shown in the last column of Table 1. As can be seen, all our systems are (slightly) outperformed. This is due to several reasons. First, a slightly altered and reduced standard feature set is used in all our systems. Second, as stated in [39], the performance of continuous HMM-based on-line handwriting recognition systems is influenced by the number of both Gaussian mixtures and training iterations. While [38] $N_{\mathrm{Gauss}} = 36$ Gaussian mixtures are used for modeling the observation, we reduced the number of Gaussians to $N_{\mathrm{Gauss}} = 32$ for computational reasons. Additionally, the exact training process of the system in [38] could not be completely duplicated. This is mainly responsible for the difference in performance. However, the slight drop of $\Delta r_{\mathrm{base}} = 0.6\%$ when comparing our best system to the system presented in [38] shows the competitiveness of our systems. Furthermore, as the parameters through all our experiments stay the same, the positive impact of both the novel normalization and the utilization of the line-member feature is proven.

6. Conclusions and outlook

In this paper we introduced a novel approach for identifying script lines in text handwritten on a whiteboard. By assuming that the script lines in the script trajectory are defined by the sample points lying on them, the principal idea behind the approach is to find an, in some sense, optimal sample point to script line assignment, the "line-members". To that end, first the number of potential line-member candidates is reduced to spatial extreme points (either maxima or minima of the script trajectory). Then, the assignment of each sample point to every script line is hypothesized, which is modeled by a

finite state machine. All states and transitions of the finite state machine given a line of text can be summarized in a trellis diagram. Assignments of all sample points to script lines are then expressed as a path through the trellis. By formulating constraints, e.g. that the derived script lines have minimal bendings or may not cross, the optimal path through that trellis is found by the Viterbi algorithm. Additionally, the system is further refined by omitting certain line-members leading to mischaracterized script lines. As a result, for each script line l a set $\mathscr{S}_l$ of sample points $\mathbf{s}(t) \in \mathscr{S}_l$ is obtained, characterizing the script lines, which may have any bending. This is the main advantage of the proposed method: no restrictions are made on the characteristics of the script lines.

Subsequently, the sample point to script line assignment is used in two different ways in order to improve a state-of-the-art HMM-based handwriting recognition system. In the first approach, the identified script lines are forced to run in parallel and horizontally by shifting the supporting points of each script line. The whole script is warped and thereby normalized in skew and size. The system using the novel normalization is denoted *System* 2. The second approach uses the line membership of the sample points directly as a feature and forms the third system (*System* 3): in case the current sample point lies on a script line, the feature vector is augmented by the information on which script line the sample point lies.

In an experimental section both approaches that utilize the novel method for script line identification have been evaluated. A baseline HMM-based system (*System* 1) has been defined, using standard preprocessing and 24 features for recognition. As the results show, regardless to how the additional script line information is incorporated into the system, a significant or even highly significant improvement of both the character- and the word-level accuracy can be observed. *System* 2 achieves a significant relative improvement of $\Delta r = 1.6\%$ ($p_N = 0.91$) on the character-level and $\Delta r = 1.4\%$ ($p_N = 0.92$) on the word-level accuracy compared to *System* 1. A peak character-level accuracy of $a_{\text{norm}} = 62.2\%$ and word-level accuracy of $A_{\text{norm}} = 63.5\%$ can be reported. *System* 3 shown an even better performance: a relative improvement of $\Delta r = 3.3\%$ ($p_N > 0.99$) compared to *System* 1 and $\Delta r = 1.7\%$ ($p_N = 0.93$) compared to *System* 2 is achieved on the character-level and an even higher relative improvement of $\Delta r = 3.4\%$ ($p_N > 0.99$) compared to *System* 1 and $\Delta r = 2.0\%$ ($p_N = 0.98$) compared to *System* 2 is obtained on the word-level. The peak rates for the character- and the word-level accuracy for *System* 3 are $A_{\text{lm}} = 63.3\%$ and 64.8%, respectively. As can be seen, *System* 3 performs better than *System* 2. This is due to the influence of misclassified script lines. While in the case of the novel normalization all features are affected, in the case of the augmented feature vector the affection is limited to the line-member feature. When performing a character confusion analysis, it has been shown that the use of the line-member feature for recognition leads to a reduction of the confusion of characters that differ in size rather than in shape.

Although they offer the above-mentioned improvements in both character-level and word-level accuracy, there are some issues with the two novel recognition systems. In case of the novel normalizations, correctly placed parts of a character may be additionally distorted by the global normalization (see Fig. 11). Hence, a local normalization is needed, which limits the impact of the normalization on the script trajectory to the sample points neighboring the current extreme point. In the case of the line-member feature, in rare cases additional character confusions have been introduced. However, these can be reduced when association between the extreme point and the script lines becomes more accurate. To that end, in future work, different metrics, such as the ascending slope rather than the absolute y-position of the script lines will be investigated. We also plan to construct a baseline system with hand-annotated sample point to script line assignments for certain sample points. Furthermore, the line-member approach will be extended in order to include all sample points with the distance of each sample point to its closest script line being used to augment the feature vector. Additionally the overall training process of the Gaussians will be optimized according to [40]. In [41] we proposed novel vector quantizing systems for using discrete HMMs for on-line recognition of handwritten whiteboard notes. In future work, the influence of both the novel normalization and the augmented feature vector will be investigated on the systems presented in [41].

Acknowledgments

The authors sincerely thank M. Liwicki for providing the lattice for the final benchmark and P.R. Laws for her crucial comments.

References

[1] H. Bunke, Recognition of cursive roman handwriting—past present and future, in: Proceedings of the International Conference on Document Analysis and Recognition, vol. 1, 2003 pp. 448–459.
[2] M. Liwicki, H. Bunke, HMM-based on-line recognition of handwritten whiteboard notes, in: Proceedings of the International Workshop on Frontiers in Handwriting Recognition, 2006, pp. 595–599.
[3] A. Vinciarelli, A survey on off-line cursive script recognition, Int. J. Pattern Recognition 7 (35) (2002) 1433–1446.
[4] R. Plamondon, S. Srihari, On-line and off-line handwriting recognition: a comprehensive survey, IEEE Trans. Pattern Anal. Mach. Intelligence 22 (1) (2000) 63–84.
[5] P. Lallican, C. Viard-Gaudin, From off-line to on-line handwriting recognition, in: Proceedings of the International Workshop on Frontiers in Handwriting Recognition, 2000, pp. 303–312.
[6] T. Steinherz, E. Rivlin, N. Intrator, Off-line cursive script word recognition—a survey, Int. J. Document Anal. Recognition 2 (2) (1999) 90–110.
[7] M. Winecke, G. Fink, G. Sagerer, Towards automatic video-based whiteboard reading, in: Proceedings of the International Conference on Document Analysis and Recognition, 2003, pp. 87–91.
[8] K.M. Sayre, Machine recognition of handwritten words: a project report, Pattern Recognition 5 (3) (1973) 213–228.
[9] L. Rabiner, A tutorial on hidden Markov models and selected applications in speech recognition, Proc. IEEE 77 (2) (1989) 257–285.
[10] K.S. Nathan, J.R. Bellegarda, D. Nahamoo, E.J. Bellegarda, On-line handwriting recognition using continuous parameter hidden Markov models, in: Proceedings of the International Conference on Acoustics, Speech, and Signal Processing, vol. 5, 1993, pp. 121–124.
[11] K. Takahashi, H. Yasuda, T. Matsumoto, A fast HMM algorithm for on-line handwritten character recognition, in: Proceedings of the International Conference on Document Analysis and Recognition, vol. 1, 1997, pp. 369–375.
[12] J. Schenk, G. Rigoll, Novel hybrid NN/HMM modelling techniques for on-line handwriting recognition, in: Proceedings of the International Workshop on Frontiers in Handwriting Recognition, 2006, pp. 619–623.
[13] M. Winecke, G. Fink, G. Sagerer, Video-based whiteboard reading, in: Document Analysis and Recognition, 2005, pp. 188–200.
[14] M. Liwicki, H. Bunke, Handwriting recognition of whiteboard notes, in: Proceedings of the Conference of the Graphonomics Society, 2005, pp. 116–122.
[15] D. Moore, The IDIAP smart meeting room, Technical Report, 2002.
[16] A. Waibel, T. Schultz, M. Bett, I. Rogina, R. Stiefelhagen, J. Yang, SMaRT: the smart meeting room task at ISL, IEEE Int. Conf. Acoust. Speech Signal Process. 4 (2003) 752–755.
[17] S. Reiter, G. Rigoll, Segmentation and classification of meeting events using multiple classifier fusion and dynamic programming, in: Proceedings of the International Conference on Pattern Recognition, 2004, pp. 434–437.
[18] Q. Stafford-Fraser, P. Robinson, BrightBoard: a video-augmented environment, in: Proceedings of the International Conference on Human Factors and Computing Systems, 1996, pp. 134–141.
[19] Panasonic, Electronic board—Panaboard ⟨http://panasonic.net/pcc/eboard⟩.
[20] PolyVision, TS series interactive whiteboard, Datasheet, 2008.
[21] SMART Technologies Inc., SMART Board Interactive Display Frame ⟨http://www.smarttech.com⟩.
[22] SMART Technologies Inc., The truth about interactive whiteboard resolution—Why bigger isn't always better ⟨http://www.smarttech.com⟩.
[23] Mimio, Mimio Interactive—specifications ⟨http://www.mimio.com⟩.
[24] Lucida Inc., eBeam whiteboard technical specifications ⟨http://www.e-beam.com⟩.
[25] M. Liwicki, H. Bunke, IAM-OnDB—an on-line English sentence database acquired from handwritten text on a whiteboard, in: Proceedings of the International Conference on Document Analysis and Recognition, vol. 2, 2005, pp. 1159–1162.
[26] T. Plötz, C. Thurau, G.A. Fink, Camera-based whiteboard reading: new approaches to a challenging task, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2008, pp. 385–390.
[27] S. Jaeger, S. Manke, J. Reichert, A. Waibel, The NPen++ recognizer, Int. J. Document Anal. Recognition 3 (2001) 169–180.
[28] Y. Bengio, Y. Cun, Word normalization for on-line handwritten word recognition, in: Proceedings of the International Conference on Pattern Recognition, 1994, pp. 409–413.

[29] T. Caesar, J. Gloger, E. Mandler, Preprocessing and feature extraction for a handwriting recognition system, in: Proceedings of the International Conference on Document Analysis and Recognition, 1993, pp. 408–411.
[30] R. Bozinovic, S. Srihari, Off-line cursive script word recognition, IEEE Trans. Pattern Anal. Mach. Intelligence 11 (1) (1989) 68–83.
[31] H. Bunke, M. Roth, E. Schukat-Talamazzini, Off-line cursive handwriting recognition using hidden Markov models, Int. J. Pattern Recognition 28 (9) (1995) 1399–1413.
[32] A. Dempster, N. Laird, D. Rubin, Maximum likelihood from incomplete data via the EM algorithm, J. R. Stat. Soc. B 39 (1) (1977) 1–38.
[33] A. Viterbi, Error bounds for convolutional codes and an asymptotically optimum decoding algorithm, IEEE Trans. Inf. Theory 13 (1967) 260–267.
[34] E. Kuvallieratou, N. Fakotakis, G. Kokkinakis, New algorithms for skewing correction and slant removal on word-level, in: Proceedings of the International Conference on Electronics, vol. 2, 1999, pp. 1159–1162.
[35] V. Vuori, J. Laaksonen, E. Oja, J. Kangas, Speeding up on-line recognition of handwritten characters by pruning the prototype set, in: Proceedings of the International Conference on Document Analysis and Recognition, 2001, pp. 501–505.
[36] J. Schenk, J. Lenz, G. Rigoll, On-line recognition of handwritten whiteboard notes: a novel approach for script line identification and normalization, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2008, pp. 540–543.
[37] J. Schenk, J. Lenz, G. Rigoll, Line-members—a novel feature in on-line whiteboard note recognition, in: Proceedings of the International Conference on Frontiers in Handwriting Recognition, 2008, pp. 469–474.
[38] M. Liwicki, H. Bunke, Combining on-line and off-line systems for handwriting recognition, in: Proceedings of the International Conference on Document Analysis and Recognition, 2007, pp. 372–376.
[39] S. Günter, H. Bunke, HMM-based handwritten word recognition: on the optimization of the number of states, training iterations and Gaussian components, Int. J. Pattern Recognition 37 (2004) 2069–2079.
[40] Y. Normandin, Optimal splitting of HMM Gaussian mixture components with MMIE training, in: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, 1995, pp. 449–452.
[41] J. Schenk, S. Schwärzler, G. Ruske, G. Rigoll, Novel VQ designs for discrete HMM on-line handwritten whiteboard note recognition, in: Proceedings of the 30th Symposium of DAGM, 2008, pp. 234–243.

About the Author—JOACHIM SCHENK was born in 1981 (Munich, Germany). He received his Dipl.-Ing. degree from Technische Universität München in 2004. Currently he is working as a research assistant at the Institute for Human–Machine Communication, Technische Universität München in order to earn his PhD in the field of handwriting recognition and on-line recognition of handwritten whiteboard notes. He has 13 previous publications, as well as one patent.

About the Author—JOHANNES LENZ was born in 1983 (Munich, Germany). He received his BSc from Technische Universität München in 2007 with a focus on the preprocessing and recognition of handwritten whiteboard notes. Currently, he is a student at Technische Universität München, continuing his studies to receive his MSc. He has two previous publications.

About the Author—GERHARD RIGOLL was born in 1958 (Essen, Germany). He received the Dipl.-Ing. degree in 1982 and the Dr.-Ing. degree in 1986, both from Stuttgart University. After working as a researcher in the USA and Japan for several years, he was appointed full professor of computer science at Gerhard-Mercator-University in Duisburg in 1993. In 2002, he joined Technische Universität München, heading the Institute for Human–Machine Communication. His research interests are human–machine communication; multimedia information processing; speech; handwriting and gesture recognition; face detection and identification; emotion recognition; person tracking; information retrieval; video-indexing; and interactive computer graphics. He is the author and co-author of more than 300 papers in the above-mentioned fields.