

# New baseline correction algorithm for text-line recognition with bidirectional recurrent neural networks

Olivier Morillot

Laurence Likforman-Sulem

Institut Mines-Télécom/Télécom ParisTech

46 rue Barrault, 75013 Paris, France

E-mail: morillot@telecom-paristech.fr

Emmanuèle Grosicki

DGA Ingénierie des Projets

7-9 rue des Mathurins, 92220 Bagneux, France

---

**Abstract.** Many preprocessing techniques have been proposed for isolated word recognition. However, recently, recognition systems have dealt with text blocks and their compound text lines. In this paper, we propose a new preprocessing approach to efficiently correct baseline skew and fluctuations. Our approach is based on a sliding window within which the vertical position of the baseline is estimated. Segmentation of text lines into subparts is, thus, avoided. Experiments conducted on a large publicly available database (Rimes), with a BLSTM (bidirectional long short-term memory) recurrent neural network recognition system, show that our baseline correction approach highly improves performance. © 2013 SPIE and IS&T [DOI: 10.1117/1.JEI.22.2.023028]

---

## 1 Introduction

Character and word recognition have been studied for more than 50 years<sup>1</sup> motivated by the urge of automatic processing of bank checks, postal envelopes, and forms. The last two decades have shown an increased activity in the field of cursive handwriting recognition.<sup>2–4</sup>

Within this period, the scope of recognition systems has been enlarged from word recognition to block or text-line recognition. Such systems allow companies and administrations to automatically sort, search through, and even answer the large amount of handwritten mail they receive. Beyond modern documents, a huge amount of archive and historical documents are still to be exploited by reading systems.<sup>5</sup>

Text-line recognition is similar to word recognition if text lines are segmented into words prior to recognition. However, recognition systems generally avoid the error-prone text-line segmentation into words.

State-of-the-art handwriting recognition systems are based on hidden Markov models (HMMs) and, more recently, on recurrent neural networks (RNNs). Both HMMs and RNNs take as inputs, sequences of frames provided by sliding windows. The sliding window approach consists of shifting a

narrow window from left to right on the word or text-line image. In each window, a set of features is extracted, thus, resulting in a frame. Features often rely on the position of the baseline since its position is useful for detecting ascenders and descenders. Therefore, a correct estimation of the baseline has a strong impact on recognition performance.

In previous works, we have defined sets of statistical and geometric features for HMM-based recognition systems.<sup>6,7</sup> Such features have proven to be powerful for Latin and Arabic word recognition.<sup>8,9</sup> In another work, we have extended the HMM-based word recognition approach to text lines by introducing adapted language models.<sup>10</sup>

Text lines are provided by a segmentation process which extracts isolated lines from a document image. Such segmented text lines are generally included in handwriting databases. The segmentation process is relatively easy when text lines are straight, sufficiently spaced, and oriented in the same direction. However, when ascenders and descenders in the inter-line space are present, or when skew is different from one line to another, fragments from neighboring lines may appear in the background of text-line images. They are considered as noise. Handwritten text lines are also characterized by fluctuations. Due to writer movement, changes in baseline position occur along the text line. The baseline, the fictitious line which follows and joins the lower part of the character bodies, may be straight, straight by segments, or curved.<sup>11</sup>

For the reasons mentioned above, most recognition systems start with preprocessing steps. Their objective is to remove noise, correct slant, and estimate baselines.<sup>12</sup> Other preprocessings may consist of reducing the variability of grey level pixel intensities and normalizing the size of ascenders and descenders.<sup>13</sup> This paper presents a new approach for processing free-style handwritten text lines. It consists of normalizing the lower baseline to a horizontal line using a sliding window approach. Our approach copes with baselines which are skewed, fluctuating, or both. This approach differs from machine learning approaches<sup>13</sup> which need manual pixel assignments to baselines. We apply our baseline correction approach to RNN-based recognition.

---

Paper 12540 received Dec. 21, 2012; revised manuscript received Mar. 30, 2013; accepted for publication May 7, 2013; published online Jun. 21, 2013.

0091-3286/2013/\$25.00 © 2013 SPIE and IS&T

RNNs differ from classical feedforward networks by the fact that outputs of hidden layers, at time  $t - 1$ , are fed as inputs to hidden layers at time  $t$ . This is the so-called recurrent links within hidden layers. In addition, hidden units, in our recurrent networks, are memory blocks called long short-term memory (LSTM). These blocks which include memory cells which can keep information through long time intervals (more than 1000 time samples) and be reset in an instant. LSTMs are particularly appropriate with tasks with long periods between pertinent information such as handwriting sequences. We couple two RNNs into the so-called bidirectional architecture (BLSTM: bi-directional long short term memory) which has been successfully applied to handwriting recognition.<sup>14,15</sup> In this architecture, two RNNs share the same input and output layers. The first network scans the text line from left to right and the second one from right to left. Thus, both past and future dependencies can be taken into account in the bidirectional architecture.

Our paper is organized as follows. In Sec. 2, we describe the preprocessing and feature extraction steps. Preprocessing includes background cleaning, slant correction, and the proposed sliding window approach for baseline correction. It reduces the variability of feature sequences provided to our RNN classifier, described in Sec. 3. Latin script text-line recognition experiments as well as a description of

the large and publicly available Rimes database are provided in Sec. 4. Conclusions and perspectives are given in Sec. 5.

## 2 Preprocessing Steps

Handwriting in free-style handwritten mails may be slanted, skewed, or fluctuating. This is due to the style of the writer and to the lack of guidelines on paper sheets. In addition, due to imperfect segmentation, text lines may include components from neighboring text lines. Such extra components are considered as noise.

To cope with noisy components, skew, slant and writing fluctuations (Fig. 1), our preprocessing includes the following steps:

- background cleaning;
- baseline correction;
- slant correction.

### 2.1 Background Cleaning

Text lines result from the segmentation of a document image [see Fig. 2(a)] into rectangular snippets corresponding to text lines [see Fig. 2(b)]. Rectangular cropping is a popular way to segment an image. However, extra components from preceding and following lines are often included in text line snippets [see Fig. 3(a)]. The background cleaning preprocessing consists of keeping components of the main text line and removing noisy components from neighboring text lines. This process is followed by whitening the background. Nonetheless, cleaning has to be performed with caution. Indeed, removing text-line components is highly damaging for the training and recognition process. Our approach does not require a training phase such as in machine-learning approaches.<sup>13</sup> It is based on the extraction of the main text line. Then, components distant from the main text line are removed. It may be noted that a component is qualified as close or distant from the main line, through a threshold directly extracted from writing characteristics of the input image. The background cleaning approach includes the following steps:

1. The text-line image is binarized (by Otsu thresholding) and contours are extracted (by a Canny edge extractor). A geometric line is then extracted from a Hough transform on the contour image. This line corresponds to the highest peak in the Hough  $(\rho, \theta)$  table [see continuous line in Fig. 3(a)].

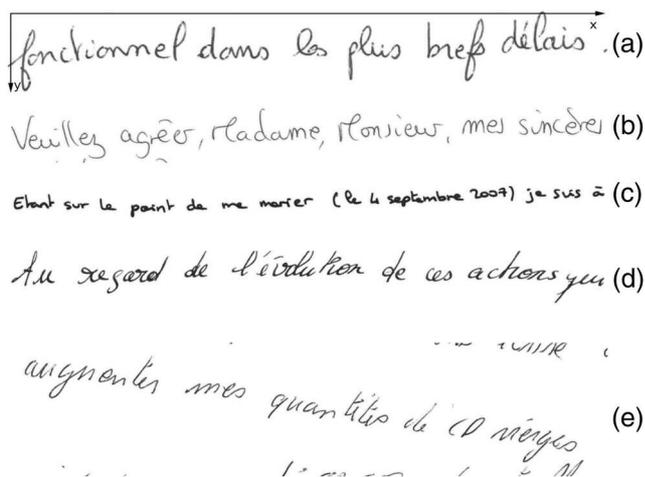


Fig. 1 Text line samples: (a-c) skewed and fluctuating, (d) slanted, and (e) including background noise from neighboring lines.

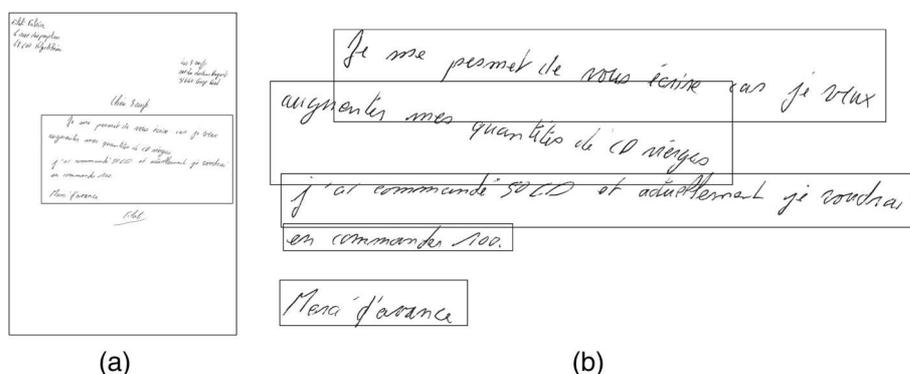
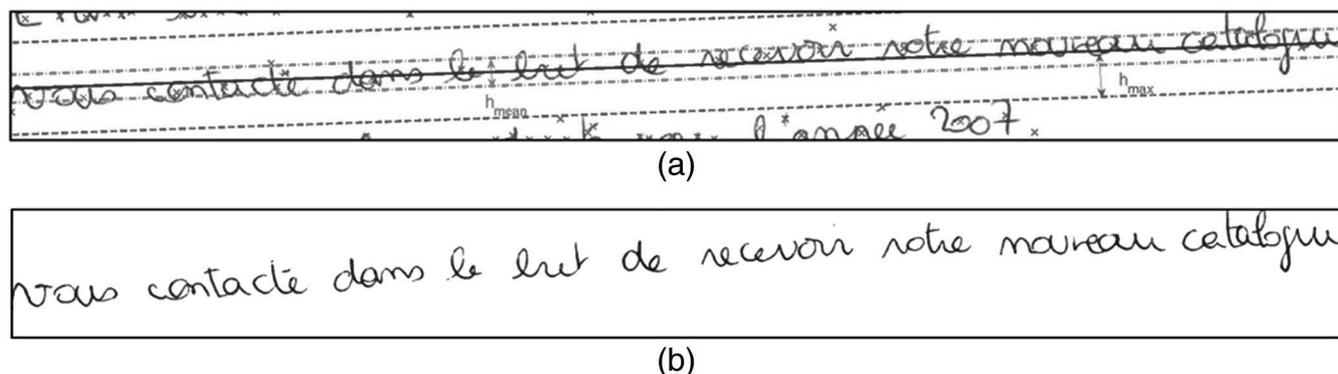


Fig. 2 Sample text block with text-line bounding boxes (b) extracted from original mails (a).



**Fig. 3** Background cleaning: (a) noisy text line including fragments from two neighboring lines. Gravity centers of connected components (crosses), geometric Hough-based line (continuous) and decision lines (dashed) (b) cleaned line (noisy components removed).

2.  $h_{\text{mean}}$  and  $h_{\text{max}}$ , the average and maximum component heights are extracted on connected components which do not reach upper or lower limits of the image.
3. An initial set  $S_c$  of candidate noisy components is composed of connected components which are distant from the main text line. Their distance is greater than  $h_{\text{mean}}/2$  computed in Step 2.
4. From previous set  $S_c$ , we build set  $S_n \subset S_c$  of noisy components to remove. These are  $S_c$  components in contact with the upper and lower edges of the image as well as  $S_c$  components whose gravity center is peripheral: the distance of their gravity center to the main text line is greater than  $h_{\text{max}}$  computed in Step 2.

Step 1 ensures locating the main text line regardless of neighboring line fragments or text-line skew. The maximum peak in the Hough  $(\rho, \theta)$  table always exists and corresponds to the actual text line. Step 2 ensures that the average height of writing components is computed from components which were not cropped by the text-line segmentation process. Step 3 ensures that components that belong to the main text line are not removed even if they do not intersect the geometric Hough-based line. The distance is computed as the nearest distance of the components' pixels to the main text line. Step 4 ensures that peripheral components may be removed even if they do not touch the edge of the image. See, for instance, the isolated components of sequence "2007" in the lower right of Fig. 3.

Components, classified as noisy, are subtracted from the original grayscale image [see Fig. 3(b)]. Residual errors of background cleaning may consist of removing accents from the main text line which are too distant from the geometric line. It may also consist of accents and commas from neighboring text lines which are too distant from these lines and which are not removed. In addition, touching components cannot be separated in this process. They should be processed during the text-line segmentation step which is out of the scope of this paper.

Due to scanning and image compression, variations in background intensities appear on images. Even if it is not disruptive for human reading, it can have an impact on our features' values. To remove such variations, the background is whitened while preserving grey level foreground values. Thus, features based on grey level intensities can still

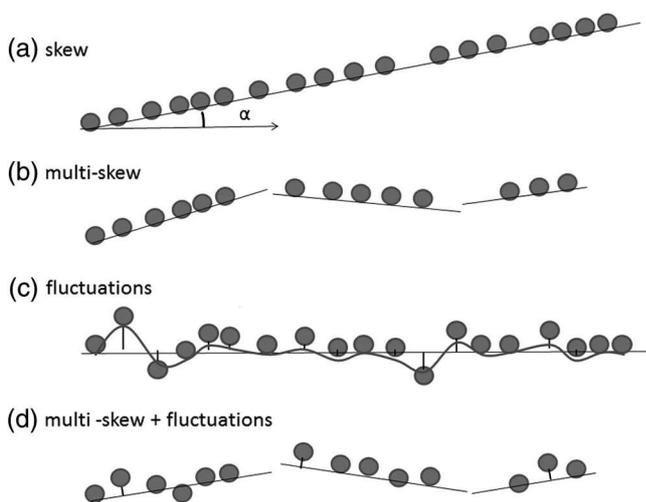
be extracted (see Sec. 3.1). Background pixels are retrieved using the Otsu thresholding method and saturated to value 255.<sup>16</sup>

## 2.2 Baseline Correction

Baseline correction is a major step for robust handwriting recognition. It is a necessary step for normalizing handwriting components such as descenders, ascenders, and text body. It also improves feature extraction when features depend on baseline positions. The novel approach, we propose for baseline correction, copes with both skew and fluctuation. It is based on a sliding-window approach which takes into account both inter-word and intra-word fluctuations.

Baseline correction has been addressed mostly at word level and is tightly linked to deskew. Deskew consists of estimating a global skew angle on the word image and rotating the word according to this skew angle. Thus, word baseline is corrected through the global rotation of the word. Word deskew approaches are based on linear regressions performed on sets of points. These points may be the centers of mass of word components or the lowest points of the word excluding descenders.<sup>17-19</sup> Word-based deskewing approaches can be extended to text-lines when lines are straight along one direction [see Fig. 4(a)], thus, assuming a single skew angle for each line.<sup>20</sup> This assumption is valid for forms or paper sheets including guidelines or for ancient documents when scribes traced marks with lead points. However, this assumption is no longer valid for free-style handwritten documents, since words can be skewed differently alongside text line [see Fig. 4(b)]. Moreover, a handwritten line can be seen quite straight if considered globally, whereas, at word level, baseline position fluctuates greatly [see Fig. 4(c)].

To our knowledge, there is one approach for baseline correction specific to text-line level. It consists of splitting a text line image into vertical strips. A single skew angle is assumed within each strip and a word-based deskewing approach is applied to each strip based on linear regression.<sup>13,21,22</sup> A refined approach consists in selecting the extrema points on which the regression is applied through classification.<sup>13</sup> A trained neural network classifies contour points as belonging to the lower baseline or not. However,



**Fig. 4** (a) Skewed baseline: bottom positions of writing components lay on a line with angle  $\alpha \neq 0$ . (b) Multiskewed baseline. (c) Fluctuating baseline: bottom positions of writing components deviate from a horizontal line. (d) Multiskewed and fluctuating baseline.

this machine-learning approach needs training from labeled samples which is time consuming due to data preparation.

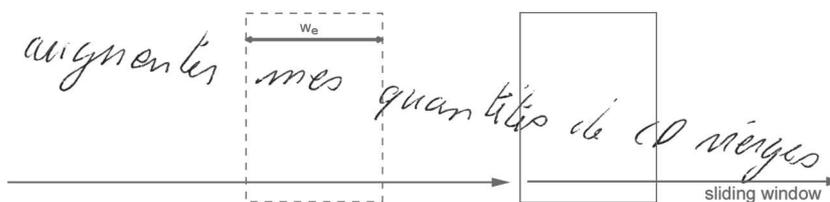
Baseline correction needs a further step since within word fluctuations are not corrected. It is performed in the so-called normalization phase.<sup>13</sup> A second neural network is trained on the deskewed lines in order to classify points as belonging to the lower baseline or not. The lower baseline is then

corrected jointly with writing height normalization. Such machine-learning approach can also correct within-word baseline fluctuations but needs training.

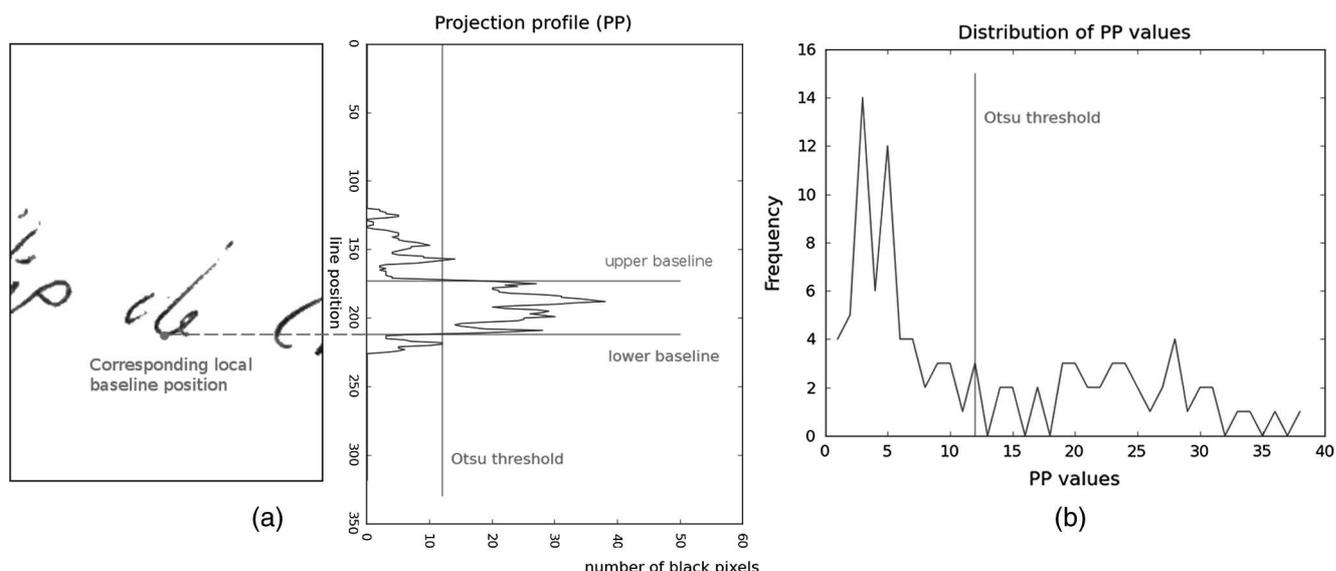
We propose, in this work, an efficient baseline correction approach specific to free-style text-line images. Deskew is performed jointly with baseline correction. Our approach does not require preliminary tasks such as text-line segmentation into words, connected components detection, or run-length analysis.<sup>23</sup> Indeed, text-line segmentation into words, or other subparts, is prone to error and does not guarantee a single skew in each subpart. The principle of our approach is to estimate the lower baseline position for each image column, by a sliding window approach, and correct it by a vertical shift. This approach copes with rapid and slow variations of baseline position.

Our approach starts with a sliding window of size  $w_e$  which scans the text line image from left to right (see Fig. 5). Using an analysis window allows us to cope with descenders and blank spaces. For the central pixel of the sliding window, the local baseline  $y$  position is computed using the projection profile approach<sup>19</sup>:

- First, the vertical projection profile (PP) is created by counting at each  $y$  position the number of foreground pixels along the horizontal direction. [see Fig. 6(a)].
- Second, the distribution (histogram) [see Fig. 6(b)] of these projection values is computed. This histogram is expected to have a principal mode corresponding to the text-line core zone since the core zone is assumed to



**Fig. 5** Sliding window approach: a window is shifted from left to right along the text line. Within each window, local baseline position is estimated.



**Fig. 6** (a) Projection profile (PP) is obtained by a projecting pixels on the vertical axis along horizontal direction. (b) Distribution of PP values and Otsu threshold. Threshold value is then used on PP to extract lower and upper baseline positions.

**Algorithm 1** Pseudo-code for lower baseline extraction

---

```

function FINDLOWERBASELINE(PP, threshold_value)
  inside_region = 0 ▷ Boolean value
  for  $i = 1 \rightarrow \text{length}(PP)$  do
    if  $PP(i) \geq \text{threshold\_value}$  and  $\text{inside\_region} = 0$  then ▷ Entering a core zone
      inside_region ← 1 ▷ Create new core region
      append [i,.] to core_regions ▷ Higher baseline
    else if  $PP(i) \leq \text{threshold\_value}$  and  $\text{inside\_region} = 1$ 
      inside_region ← 0 ▷ Leaving a core zone
      core_regions[length(core_regions)][2] ← i ▷ Lower baseline
    end if
  end for
  if  $\text{length}(\text{core\_regions}[\text{length}(\text{core\_regions}) - 1]) = 1$  then
    append length(PP)-1 to core_regions[length(core_regions)-1]
  end if
  for  $k = 1 \rightarrow \text{length}(\text{core\_regions})$  do
    add core_regions[k][2]-core_regions[k][1] to size_regions ▷ Length of k-th core zone
  end for
  Let i_max the index of max(size_regions) ▷ Index of larger core zone
  return core_regions[i_max][2] ▷ Lower baseline
end function

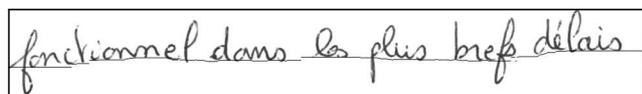
```

---

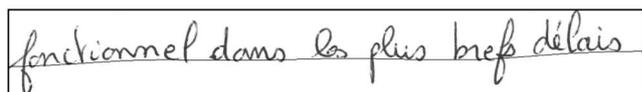
have a higher pixel density than the zones corresponding to ascenders and descenders.

- Third, a thresholding algorithm (Otsu's method) is applied to the above distribution. The text core zone corresponds to the largest continuous core zone above threshold on the PP profile (see Fig. 6(a) and Algorithm 1).

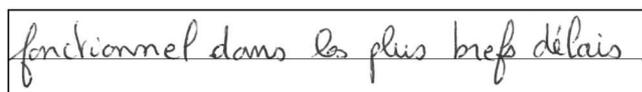
Baseline position curve is then smoothed with a Gaussian filter of width  $w_{\text{smooth}}$  to get rid of discontinuities [see Figs. 7(b) and 8(d)]. Gaussian distribution standard deviation is  $\sigma = w_{\text{smooth}}/(4\sqrt{2})$ . Without this smoothing, shearing artifacts can appear on corrected images. The correction step corresponds to a vertical translation. Pseudo-code of the baseline correction algorithm is provided in Algorithms 1 and 2. Figures 7(a) and 8(c) show sample text lines. The fluctuating text line in Fig. 7(a) is corrected in Fig. 7(c). The skewed and fluctuating text line in Fig. 8(c) is corrected in Fig. 8(e). The resulting text line shows fluctuations which



(a)



(b)



(c)

**Fig. 7** Baseline correction steps: (a) local baseline position estimation, (b) baseline smoothing, and (c) baseline correction.

are now too small to impact the feature extraction process described in Sec. 3.1.

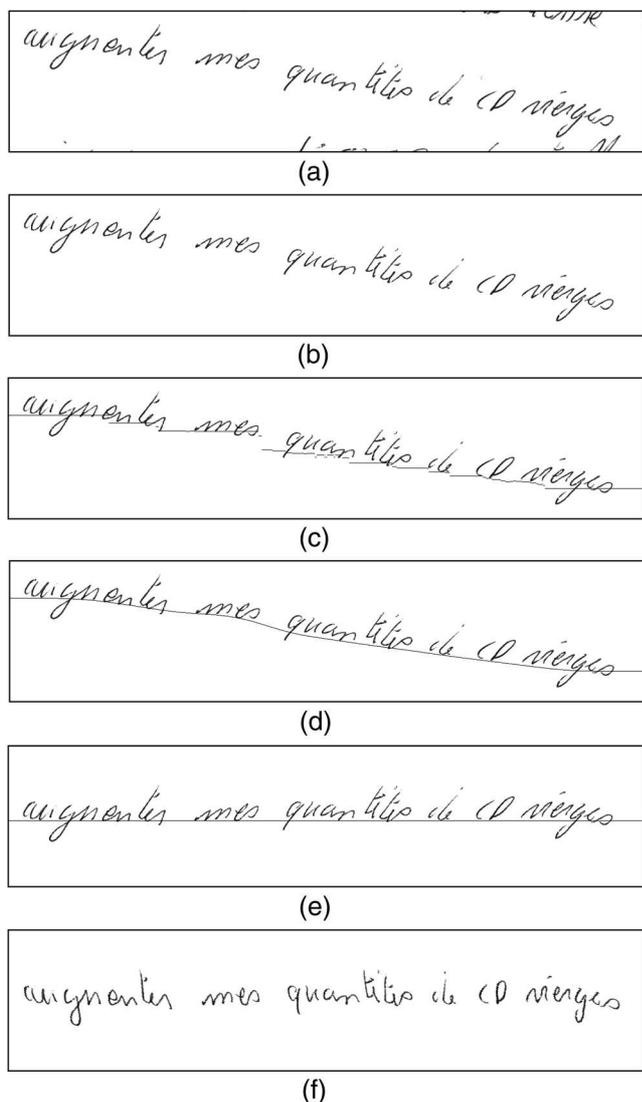
### 2.3 Slant Correction

The last preprocessing consists of deslanting previous baseline-corrected text lines [see Fig. 8(f)]. Deslanting consists in horizontally shifting writing strokes, such as descenders and ascenders, in order they seem vertical. This step is necessary since our feature extraction step relies on a sliding window approach (see Sec. 3.1). Thus, it is desirable that the feature extraction window only includes parts of a single character. However, the feature extraction sliding window is distinct from the window used for preprocessing (Sec. 2.2). They differ both in size and shift. We use the Vinciarelli's approach which relies on histograms of projection values along several angle directions.<sup>19</sup>

A single slant angle is presently assumed for a text line. This could be refined by local approaches such as proposed by Uchida et al.<sup>24</sup> However, we have noticed that skew and baseline fluctuations are more varying than slant in our data.

## 3 Free-Style Text-Line Recognition

BLSTMs are recurrent networks which take as input sequences of frames. These frames are extracted from a sliding window shifted from left to right on the text-line image (Fig. 9). This feature extraction process is quite similar to feature extraction for HMMs. However, in BLSTMs, frames are processed both from left to right and right to left, thanks to the bidirectional architecture. Thus, past and future long-term dependencies can be taken into account. This is the so called *contextual* information relevant for recognizing a piece of handwriting. This differs from HMMs where context-dependent character models take into account the preceding and following characters only.



**Fig. 8** Preprocessing steps: (a) original cropped line, (b) background cleaning, (c) local baseline estimation, (d) baseline smoothing, (e) baseline correction, and (f) after deslant.

### 3.1 Feature Extraction

Frame sequences are extracted from previous preprocessed text-line images by a sliding window of size  $w$  and shift  $\delta$  (Fig. 9). Each frame consists of a set of 28 features which have been successfully applied to Latin and Arabic handwritten word recognition and, recently, to text-line recognition.<sup>7,10,25</sup> According to previous work, each feature of this set has a positive impact on recognition performance.<sup>6,25,26</sup> Indeed, the feature extraction step, in isolation, takes about one second. In order to extract features, each window is divided into a fixed number of cells. This allows feature extraction to cope with different image heights. Experimental setup of parameters  $\delta$  and  $w$  is discussed in Sec. 4.

These 28 features (F1 to F28) are the following:

1. F1: foreground pixel density
2. F2: number of foreground/background transitions between adjacent cells
3. F3: gravity center position difference with following window

4. F4: relative position of gravity center
5. F5: pixel density above upper baseline (ascenders zone)
6. F6: pixel density under lower baseline (descenders zone)
7. F7: number of foreground/background transitions between cells above lower baseline (ascenders+core zone)
8. F8: relative position of gravity center wrt baselines
9. F9 to F20: local convexity features
10. F21 to F28: pixel density for each frame column (considering a window size  $w = 8$ )

Derivative features are obtained by taking the first-order derivatives of the above 28 features. The derivation is computed with a regression which takes into account the two neighboring frames. This is similar to the delta coefficients in speech recognition domain. Thus, the total set of features for each frame consists of 56-features. Feature sequences are provided to the BLSTM recognizer (see Sec. 3.2).

### 3.2 BLSTM Recognizer

The BLSTM recognizer consists of the coupling of two RNNs. Each recurrent network takes as input a frame sequence as described in Sec. 3.1. However, one network (called the *forward network*) takes the original frame sequence as input (from  $t = 1$  to  $T$ ) while the *backward network* takes as input the reversed sequence (from  $t = T$  to 1) (Fig. 10).

A recurrent network consists of an input layer, an output layer and an hidden layer. At each time step  $t$ , the input layer consists of the frame extracted at  $t$ . The output layer consists of 91 units, each unit being associated with a given character (A to Z, a to z, numerals, symbols). The two networks, forward and backward, share the same input and output layers. They differ in their hidden layers. Hence, the value of an output unit at time-step  $t$  is the linear combination of the outputs of the forward and backward hidden layers at this time-step  $t$ . The outputs of the forward hidden layer, at time-step  $t$ , depend on the inputs at  $t$  and the outputs of the hidden layer at  $t - 1$  and, subsequently, depend on all previous frames (1 to  $t - 1$ ). Similarly, the backward hidden layer, at time  $t$ , depends on all following frames ( $t + 1$  to  $T$ ). In theory long-term dependencies can thus be taken into account. However in practice there is an important decay in the error signals during gradient-based learning. This issue is described as “vanishing gradient.”<sup>27</sup> It has been partially addressed by replacing in the hidden layer, classical neural network cells (performing input summation and passing the result through an activation function) with memory blocks. These blocks, called LSTM contain a memory cell (Fig. 11) which either keeps information through long time intervals (more than 1000 time samples) or can be reset in an instant.<sup>15</sup>

An LSTM block includes a memory cell and multiplicative logical gates which are specifically designed to memorize or forget relevant information through time. Those gates can pass or block signals.

- Input gate is placed before the cell unit (at the center of the block in Fig. 11).<sup>15</sup> Its main design is to protect the

**Algorithm 2** Pseudo-code for baseline correction.

```

I(1:m, 1:n)                                     ▷ Text-line image with m lines and n columns
Define  $w_e w_{smooth}$  values

Local baseline estimation
Let  $baseline(j) = 0$  ( $j = 1:n$ )
for  $j = 1 \rightarrow n$  do
     $I_w = I(1:m, j - w_e/2:j + w_e/2)$ 
    PP = projectionprofile( $I_w$ )
    hist = histogram(PP)
    threshold_value = Otsu_method(hist)
     $baseline(j) = FINDLOWERBASELINE(PP, threshold\_value)$ 
end for                                         ▷ Program detailed in Algorithm 1

Baseline smoothing with gaussian filter
Let  $baseline\_smoothed(j) = 0$  ( $j = 1:n$ )
Create gaussian_filter of width  $w_{smooth}$  and standard deviation  $\sigma = w_{smooth}/(4\sqrt{2})$ 
 $baseline\_smoothed = filter(baseline, gaussian\_filter)$ 

Correction
Let  $I_{corr}(1:m, 1:n)$ 
 $baseline\_mean = \frac{1}{n} \sum_{i=1}^n baseline\_smoothed(i)$ 
for  $j = 1 \rightarrow n$  do
    for  $i = 1 \rightarrow m$  do
        if  $I(i, j)$  is foreground pixel then
            Let  $\Delta = baseline\_smoothed(j) - baseline\_mean$ 
            if  $0 < i + \Delta < m$  then
                 $I_{corr}(i + \Delta, j) \leftarrow I(i, j)$ 
            end if
        end if
    end for
end for                                         ▷ Vertical shift for correction
    
```

cell's memory from nonrelevant inputs. Thus, cell can keep relevant information through time via its recurrent loop.

- Output gate is placed after the cell unit.<sup>15</sup> Similarly, output gate protects output layer from current information contained in the cell if it is not yet relevant for outputting.

- Forget gate is a more recent improvement designed to reset the cell's value and to avoid its endless growth.<sup>29</sup> Moreover, forget gate can erase the cell's memory once its content is irrelevant to present inputs.
- Peepholes are connexions between the cell and the different gates which allow those gates to spy on current cell state (dotted lines in Fig. 11).

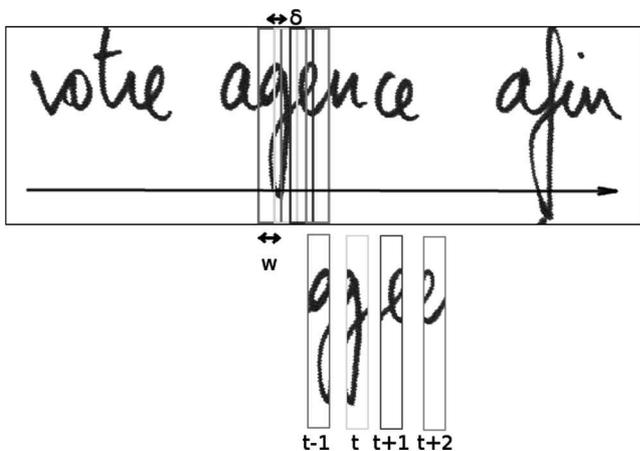


Fig. 9 Feature extraction with a sliding window.

The inputs of the gates are the input data from the input layer at  $t$ , the outputs of all LSTM blocks of the hidden layer at  $t - 1$  as well as the cell state at  $t - 1$ . Inputs of the LSTM block are the input data at  $t$  and the outputs of all LSTM blocks at  $t - 1$ .

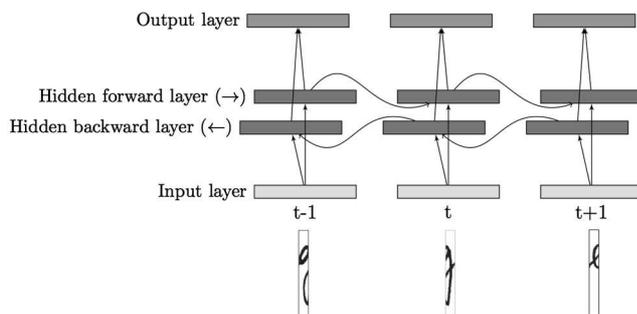
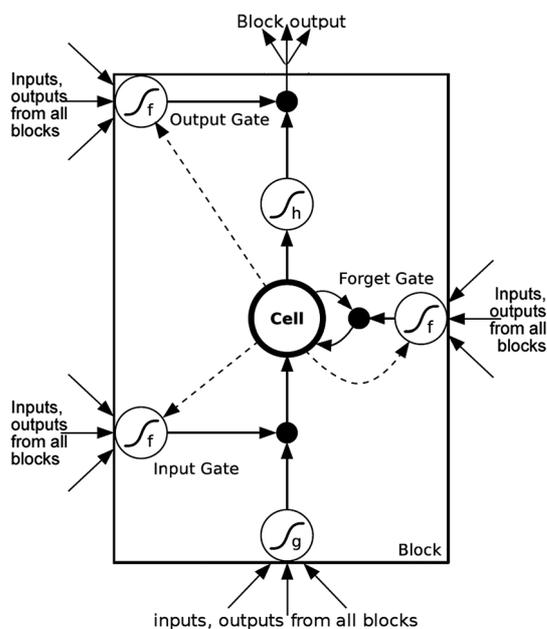


Fig. 10 Bidirectional network unfolded through 3 time steps.



**Fig. 11** LSTM memory block with one cell and three gates: multiplicative units are represented by full black circles,  $f$ ,  $g$ , and  $h$  are activation functions (adapted from Graves).<sup>28</sup>

Inputs at the entrance of gates, or LSTM block, are summed and passed through an activation function. The cell state depends on its previous state (multiplied by the output of the forget gate) and the output of the activation function  $g$  (multiplied by the output of the input gate) (see Fig. 11).

We use a BLSTM with one hidden layer containing 100 blocks as in Ref. 14. The BLSTM recognizer is trained with a gradient-based method, "Back-Propagation Through Time."<sup>30,31</sup> After each training epoch, the recognition error rate is evaluated on a validation set. If error rate does not decrease for twenty epochs, network training is stopped. This strategy avoids data over-fitting.

We use the BLSTM implementation detailed in Ref. 14. The BLSTM computes, for each frame its outputs, each associated to a character class. These outputs are normalized, thus, providing the posterior probability for each character class. Then, a backward-forward token passing algorithm, referred to as connectionist temporal classification (CTC), takes the posteriors as input and provides a sequence of words given the dictionary and the language model. We use the CTC implementation introduced in the works of Graves<sup>14</sup> and Frinken.<sup>14,32</sup> Our dictionary and language model are based on training set transcriptions (see Sec. 4).

## 4 Experiments

### 4.1 Rimes Database

The Rimes database of handwritten letters was created in 2006 with the funding of French Defense and Research Ministries in order to evaluate automatic recognition and indexing systems.<sup>33</sup> The database was collected by asking volunteers to write letters (on white paper, without guidelines) according to nine realistic scenarios which include change of personal information, information request, opening and closing account, modification of contract or order, complaint, payment difficulties, reminder letter, and

**Table 1** Error rates obtained on Rimes 2011 validation word database with various feature extraction window widths  $w$  and shifts  $\delta$  (dictionary size: 5334).

$w$	$\delta$	Error rates (%)
4	4	19.64
8	4	18.77
8	8	24.61
9	4	18.53
9	3	17.74
9	2	<b>15.72</b>
9	1	20.71
10	3	19.35

damage declaration. The volunteers composed a letter with those pieces of information using their own words. Two word recognition competitions and one text block recognition competition took place between 2009 and 2011.<sup>9,34</sup> Thus, Rimes 2011 includes a word database and a text-line database. We use both of them since the word database is useful for setting up parameters (see Sec. 4.2).

- The Rimes 2011 word database is divided into training, validation, and test sets including 51,738, 7484 files and 9880 word images, respectively.
- The Rimes 2011 text-line training database includes 11,329 text-line images. They are extracted from 1500 text blocks and text-line bounding boxes are provided by database creators. Typical size of a text-line image is  $2000 \times 150$  pixels. We divide this database into two sets: 10,329 text lines (from 1370 text blocks) for training and 1000 lines (from 130 text blocks) for validation. The test set is composed of the 778 text lines (100 text blocks) provided for the ICDAR 2011 competition.<sup>34</sup>

**Table 2** Word error rates (WER) obtained on Rimes 2011 test text-line database according to preprocessing steps and various combination of dictionary (dic.) and language model (LM).

Preprocessing	WER (no dic./no LM) (%)	WER (dic./no LM) (%)	WER (dic./LM) (%)
No background cleaning and no baseline correction	64.4	31.6	19.9
Background cleaning and no baseline correction	60.7	27.9	18.0
Background cleaning and baseline correction	56.8	26.0	15.8

The dictionary is built from train text-line transcriptions and contains 6000 entries. A bigram language model is also built from train transcriptions following previous work.<sup>10</sup>

As in ICDAR 2011 competition, all results for text-line recognition are provided through word error rates (WER). This rate includes all word substitutions, insertions, and deletions. Thus, a WER can be greater than 100% if many word insertions occur.

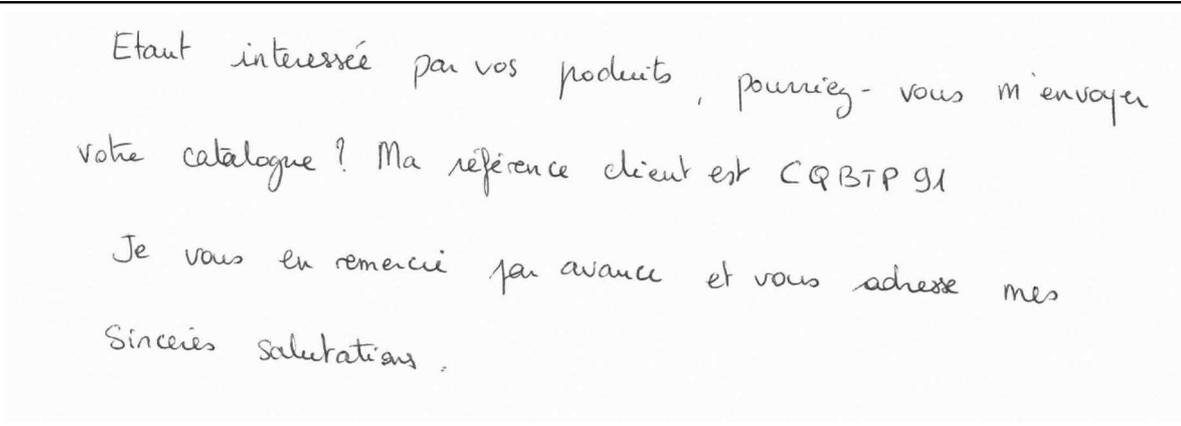
$$\text{WER} = \frac{\text{word.substitutions} + \text{word.insertions} + \text{word.deletions}}{\text{ground truth number of words in text lines}} \quad (1)$$

## 4.2 Parameter Setup

To build our text-line recognizer, four parameters are to be set up. Two parameters deal with baseline correction which are the width of the baseline extraction window  $w_e$  and the width of the smoothing filter of the baseline positions  $w_{\text{smooth}}$ . The other two are related to the feature extraction process which are the width  $w$  and the shift  $\delta$  of the feature extraction window ( $\delta < w$ ). All parameters are set up on a validation database which is distinct from the test database.

Feature extraction parameters  $w$  and  $\delta$  can be optimized on the word database. Indeed, both text lines and words are extracted from the same documents and have identical pixel

**Table 3** Sample text-lines, ground-truth, and BLSTM recognition outputs, according to different dictionary and language model conditions. Bold characters indicate recognition errors.


Ground-truth
Etant intéressée par vos produits, pourriez vous m'envoyer votre catalogue? Ma référence client est CQBTP91 Je vous en remercie par avance et vous adresse mes sincères salutations.
BLSTM (no dictionary + no LM)
Etant intéressée par vos <b>poduits</b> , pourriez vous <b>m'envyer</b> votre <b>catalogue l</b> IMMa référence <b>dclient</b> est CRQHRP 91 <b>e</b> vous en <b>remercié</b> par <b>avauce</b> et vous adresse mes Sinceres <b>ssalutations</b> .
BLSTM (dictionary + no LM)
Etant interesse par vos produits pourriez vous m'envoyer votre catalogue l Ma référence client est <b>CH P 21</b> Je vous en remercie par avance et vous adresse mes Sincères salutations.
BLSTM (dictionary + LM)
Etant intéressée par vos produits, pourriez vous m'envoyer votre catalogue? Ma référence client est <b>CCP 21</b> Je vous en remercie par avance et vous adresse mes Sincères salutations.

resolutions. For our feature extraction optimization, we use the Rimes 2011 word database (see Sec. 4.1).

Table 1 shows that optimal extraction parameters are  $w = 9$  and  $\delta = 2$ . This optimum window width is intrinsic to image resolution and is directly correlated to character width and to the kind of features. On the other hand, an optimal shift value equal to two is more interesting. This yields a great overlap in extraction windows and, thus, creates information redundancy in frame sequences.

The interest of overlapped sequences for BLSTM recognizer was not obvious since those networks are specially engineered to keep information through time due to its recurrent architecture and its LSTM cells. However, the shift value should not be less than two pixels. Overlapping feature extraction has one main drawback which means it leads to longer frame sequences and, thus, to longer training and decoding phases.

Baseline correction parameters,  $w_e$  and  $w_{\text{smooth}}$  (see Sec. 2.2), are both optimized on the validation text-line database: optimal values are  $w_e = 225$  pixels and  $w_{\text{smooth}} = 350$  pixels. Both values are quite large in order to cope with long blank sequences between words. Indeed, window size  $w_e$  is related to image resolution and should be set in order to avoid blank windows. Rimes parameter values  $w_e = 225$ ,  $w_{\text{smooth}} = 350$  pixels have been applied to sample text-lines of IAM and OpenHart databases, thus, yielding good visual results.

### 4.3 Evaluation

To evaluate our baseline correction algorithm, we conduct text-line recognition experiments including various combinations of preprocessing steps. In order to distinguish the improvement brought by language help (dictionary and language model) from the brute BLSTM recognition, we provide all results with and without a dictionary and with or without a language model. Indeed, the recognition system can provide character strings from activation lattices without any dictionary. We will refer to this in Table 2 as the “no dictionary/no LM” case. Recognition can also be constrained by a dictionary. Only character strings from the dictionary can be output: this is the “dictionary/no LM” case. Then, an language model can be added to the dictionary: the bigram transition probabilities between words are taken into account by the CTC token passing algorithm: This is the “dictionary/LM” case. We provide results for all these cases in Table 2.

Our dictionary and language model are built on train and validation text-line databases which include 6000 different words. For each major preprocessing step (background cleaning and baseline correction), we go through the complete process (feature extraction, training, decoding) on the original text-line images and on the preprocessed ones. The effect of our preprocessing(s) is measured through WER (see Sec. 4.1) provided in Table 2.

The WER obtained without preprocessing and language help (no dictionary, no LM) is about 64.4%. Due to our preprocessing steps (background cleaning and baseline correction), WER is reduced by 7.6% in absolute value. The WER reduction brought by our preprocessings using a language model and a dictionary is 4.1% in absolute value which is lower than without language help since language model and dictionary compensate for recognition errors. This

corresponds to a relative 20% reduction in error rate. The baseline correction step accounts for a relative reduction of 11% in error rate after background cleaning. The improvement brought by our baseline correction and background cleaning is, thus, significant. An example of BLSTM recognition is provided in Table 3 for different cases, with and without dictionary or language modeling. In this example, the BLSTM recognizer (case dic./LM) is only challenged by out-of-vocabulary words, here, a code sequence. Even without any language help (no dictionary/no LM), the BLSTM outputs a number of real word sequences. Most errors are missing or doubled letters. These recognition outputs clearly show the discriminative power of BLSTMs.

Our 5.6% WER reduction, for the dictionary/no LM case, can be related to the 8.1% WER reduction observed in a recent study on the preprocessing of Rimes isolated word images.<sup>35</sup> Both WER reductions show that preprocessing brings significant improvement. Our 15.8% WER achieved for text-line recognition is similar to the 16.8% WER obtained in Ref. 35 for word recognition, although text-line recognition is a more complex task.

Our 15.8% WER is a state-of-the-art result very close to the 15.2% WER of the best system of the Rimes text-line recognition competition.<sup>34</sup> This WER was achieved by combining three recognition systems at the decision level while we use a single system.

## 5 Conclusions

Given the recent development of text-line recognizers, it is necessary to develop preprocessing approaches at line level. Considering a single skew value, as for isolated words, is not relevant for free-style handwritten text lines. In this paper, we have proposed an efficient algorithm for baseline correction which copes with a great variety of skew and fluctuating situations. This approach uses a sliding window, thus, avoiding the segmentation of text lines into subparts. We have shown that a state-of-the-art BLSTM recognizer performs significantly better using this preprocessing. Such performance was obtained on the large Rimes database of French handwritten mails. Our approach can easily be extended to other Latin or non-Latin scripts such as Arabic. The letters of such scripts should rest on a baseline.

We have conducted experiments with a BLSTM recognizer. Other recognizers can also benefit from our baseline correction. Popular ones are HMMs whose feature extraction process is quite similar from that of BLSTMs.

### Acknowledgments

This work has been supported by Direction Générale de l'Armement (DGA) under Grant No. 2010-60-045 and the Futur & Ruptures foundation of Institut Mines-Telecom, France. The authors are also grateful to Marcus Liwicki, Andreas Fischer, Volkmar Frinken and Saad Bin Ahmed for providing them with the RNN library and with kind advice and guidance.

### References

1. L. Harmon, “Automatic recognition of print and script,” *Proc. IEEE* **60**(10), 1165–1176 (1972).
2. T. Steinherz, E. Rivlin, and N. Intrator, “Offline cursive script word recognition—a survey,” *Int. J. Doc. Anal. Recogn.* **2**(2), 90–110 (1999).
3. R. Plamondon and S. Srihari, “Online and offline handwriting recognition: a comprehensive survey,” *IEEE Trans. Pattern Anal. Mach. Intell.* **22**(1), 63–84 (2000).

4. A. Vinciarelli, "Online and offline handwriting recognition: a comprehensive survey," *Pattern Recognit.* **35**(7), 1433–1446 (2002).
5. A. Fischer, "Handwriting recognition in historical documents," Ph.D. Thesis, University of Bern (2012).
6. A.-L. Bianne-Bernard et al., "Dynamic and contextual information in HMM modeling for handwritten word recognition," *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(10), 2066–2080 (2011).
7. R. Al-Hajj-Mohamad, L. Likforman-Sulem, and C. Mokbel, "Combining slanted-frame classifiers for improved HMM-based Arabic handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(7), 1165–1177 (2009).
8. V. Märgner, M. Pechwitz, and H. E. Abed, "Arabic handwriting recognition competition," in *Int. Conf. on Document Analysis and Recognition*, pp. 70–74, IEEE Computer Society Press, New York (2005).
9. E. Grosicki and H. El-Abed, "ICDAR 2009 handwriting recognition competition," in *Int. Conf. on Document Analysis and Recognition*, pp. 1398–1402, IEEE Computer Society Press, New York (2009).
10. O. Morillot, L. Likforman-Sulem, and E. Grosicki, "Construction of language models for an handwritten mail reading system," *Proc. SPIE* **8297**, 82970S (2012).
11. L. Likforman-Sulem, A. Zahour, and B. Taconet, "Text line segmentation of historical documents: a survey," *Int. J. Doc. Anal. Recogn.* **9**(2–4), 123–138 (2007).
12. M. Schambach, J. Rotland, and T. Alary, "How to convert a Latin handwriting recognition system to Arabic," in *Int. Conf. Frontiers in Handwriting Recognition*, IEEE Computer Society Press, New York (2008).
13. S. España-Boquera et al., "Improving offline handwritten text recognition with hybrid HMM/ANN models," *IEEE Trans. Pattern Anal. Mach. Intell.* **33**(4), 767–779 (2011).
14. A. Graves et al., "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.* **31**(5), 855–868 (2009).
15. S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.* **9**(8), 1735–1780 (1997).
16. N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Trans. Syst. Man Cyber.* **9**(1), 62–66 (1979).
17. M. M. Blumenstein, C. K. Cheng, and X. Y. Liu, "New preprocessing techniques for handwritten word recognition," in *The 10th IASTED International Conference on Visualization, Imaging, and Image Processing*, pp. 480–484, Acta Press (2002).
18. R. Bozinovic and S. Srihari, "Off-line cursive script word recognition," *IEEE Trans. Pattern Anal. and Mach. Intell.* **11**(1), 68–83 (1989).
19. A. Vinciarelli and J. Luettin, "A new normalization technique for cursive handwritten words," *Pattern Recogn. Lett.* **22**(9), 1043–1050 (2001).
20. U.-V. Marti and H. Bunke, "Handwritten sentence recognition," in *Proc. Int. Conf. on Pattern Recognition*, pp. 467–470, IEEE Computer Society Press, New York (2000).
21. A. Vinciarelli, S. Bengio, and H. Bunke, "Offline recognition of unconstrained handwritten texts using HMMs and statistical language models," *IEEE Trans. Pattern Anal. Mach. Intell.* **26**(6), 709–720 (2004).
22. M. Liwicki et al., "A novel approach to on-line handwriting recognition based on bidirectional long short-term memory networks," in *Proc. 9th Int. Conf. Document Analysis and Recognition*, pp. 367–371, IEEE Computer Society Press, New York (2007).
23. Z. Shi and V. Govindaraju, "Skew detection for complex document images using fuzzy runlength," in *Int. Conf. on Document Analysis and Recognition*, pp. 715–720, IEEE Computer Society Press, New York (2003).
24. S. Uchida, E. Taira, and H. Sakoe, "Nonuniform slant correction using dynamic programming," in *IEEE Int. Conf. Document Analysis and Recognition*, pp. 434–438, IEEE Computer Society Press, New York (2001).
25. A.-L. Bianne-Bernard, "Reconnaissance de mots manuscrits cursifs par modèles de Markov cachés en contexte: application au français, à l'anglais et à l'arabe," Ph.D. Thesis, Télécom ParisTech (2012).
26. C. Oprean, L. Likforman-Sulem, and C. Mokbel, "Handwritten word preprocessing for database adaptation," *Proc. SPIE* **8658**, 865808 (2013).
27. S. Hochreiter et al., "Gradient flow in recurrent nets: the difficulty of learning long-term dependencies," in *A Field Guide to Dynamical Recurrent Neural Networks*, IEEE Press, New York (2001).
28. A. Graves, "Supervised sequence labelling with recurrent neural networks," in *Studies in Computational Intelligence*, Springer, New York (2012).
29. F. A. Gers, J. Schmidhuber, and F. A. Cummins, "Learning to forget: continual prediction with LSTM," *Neural Comput.* **12**(10), 2451–2471 (2000).
30. P. J. Werbos, "Generalization of backpropagation with application to a recurrent gas market model," *Neural Networks* **1**(4), 339–356 (1988).
31. R. J. Williams and D. Zipser, "Gradient-based learning algorithms for recurrent networks, and their computational complexity," Chapter 13 in *Back-Propagation Theory, Architectures, and Applications*, pp. 433–486, Lawrence Erlbaum Associates, Hillsdale, NJ (1995).
32. V. Frinken et al., "A novel word spotting method based on recurrent neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.* **34**(2), 211–224 (2012).
33. E. Grosicki et al., "La campagne d'évaluation rimes pour la reconnaissance de courriers manuscrits," in *Colloque International Francophone sur l'Écrit et le Document*, pp. 61–66 (2006).
34. E. Grosicki and H. El-Abed, "ICDAR 2011: French handwriting recognition competition," in *Int. Conf. Document Analysis and Recognition*, pp. 1459–1463, IEEE Computer Society Press, New York (2011).
35. H. Pesch et al., "Analysis of preprocessing techniques for Latin handwriting recognition," in *Int. Conf. Frontiers in Handwriting Recognition*, pp. 280–284, IEEE Computer Society Press, New York (2012).



IS&TSPiE 24th Annual Symposium on Electronic Imaging) held in Burlingame, California.



the IEEE.



the French Ministry of Defense. She has organized international handwriting recognition competitions hosted during the ICDAR 2009 and 2011 conferences.