The QCRI Recognition System for Handwritten Arabic

Felix Stahlberg(✉) and Stephan Vogel

Qatar Computing Research Institute, HBKU, Doha, Qatar
{fstahlberg,svogel}@qf.org.qa
http://www.qcri.qa/

Abstract. This paper describes our recognition system for handwritten Arabic. We propose novel text line image normalization procedures and a new feature extraction method. Our recognition system is based on the Kaldi recognition toolkit which is widely used in automatic speech recognition (ASR) research. We show that the combination of sophisticated text image normalization and state-of-the art techniques originating from ASR results in a very robust and accurate recognizer. Our system outperforms the best systems in the literature by over 20% relative on the *abcde-s* configuration of the IFN/ENIT database and achieves comparable performance on other configurations. On the KHATT corpus, we report 11% relative improvement compared to the best system in the literature.

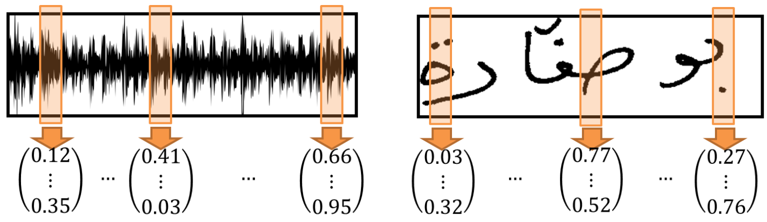
Keywords: Arabic · Handwriting recognition · Text image normalization

1 Introduction

Offline handwriting recognition (HWR) refers to the conversion of handwritings in a scanned image to machine-encoded text. State-of-the-art HWR systems are based on Hidden Markov Models (HMMs) which are statistical models for sequences of feature vectors. The order of the feature vectors within the sequence can represent temporal dependencies. For instance, in automatic speech recognition (ASR), the audio recording is often split into 10-15 ms chunks and a feature vector is extracted for each of these chunks [9] (Fig. 1(a)). In our work, the input for HWR are images of single text lines. Analogously to ASR, we split the image into chunks with three pixel width and extract a feature vector for each of these chunks. The sequential order of the extracted feature vectors is defined by the reading order of the script (e.g. right-to-left for Arabic in Fig. 1(b)).

Realizing the similarities between offline HWR and ASR, many groups apply tools initially develop for ASR to HWR tasks. The HTK Speech Recognition Toolkit [19] from the University of Cambridge is often used in HWR research. The RWTH Aachen University offers the optical character recognition software package *RWTH OCR* [17] which is based on their speech framework *RWTH ASR*.

V. Murino and E. Puppo (Eds.): ICIAP 2015, Part II, LNCS 9280, pp. 276–286, 2015.
DOI: 10.1007/978-3-319-23234-8_26



(a) Speech recognition (simplified). (b) Handwriting recognition.

Fig. 1. Feature extraction in speech recognition and handwriting recognition

The Kaldi speech recognition toolkit [15] is another project intended for use by speech recognition researchers.

In this paper, we describe the Arabic HWR system developed at the Qatar Computing Research Institute (QCRI) based on Kaldi. We adapted the training recipes in Kaldi to make them work with HWR. We show that training procedures and feature transforms popular for ASR are also applicable to HWR. Additionally, we use dedicated *connector* and *space* models as proposed in [1] to capture the distinction between connecting and non-connecting Arabic letters. We also integrated *glyph dependent model lengths* as described in [4] to address the varying complexity of characters. Both approaches are explained in Sec. 4.

The main focus of this paper, however, is the investigation of text line image normalization and feature extraction for Arabic. We demonstrate that the recognition performance can be improved by normalizing the text line image to cope with variations due to different writers and writing styles. We propose the following normalization procedure prior to the feature extraction:

1. The baseline of Arabic handwritings is often not horizontal. Longer text lines even feature curved or discontinuous baselines. This leads to characters translated along the vertical axis which obviously poses problems when using pixel gray values as features. In order to reduce these artefacts, we slice the line image vertically (i.e. along the horizontal axis) and estimate the baseline in each sub image. The slices are translated and rotated separately according the estimated baseline, and then concatenated to obtain an image with a rectified straight baseline at a predefined vertical position.
2. Even with a straight baseline, Arabic handwritings are often italic. The slant varies largely across script images. We estimate the slant of the text and apply a shear transform to remove artefacts due to this variation.
3. The resolution and the size of the letters differ among images. Therefore, we rescale the images to a fixed height while retaining the aspect ratio above the baseline.
4. The thickness of the lines is the last variation we address. The thickness varies due to different pens or the the previous rescaling operation. We apply a line thinning algorithm to obtain the script skeleton, and then increase the thickness to a normalized value with a dilate operation.

The proposed normalization steps lead to a very robust system which can cope with a wide variety of writing styles. We report 11.5% word error rate on

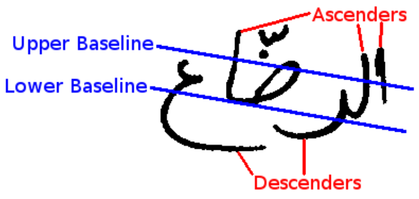


Fig. 2. Upper and lower baselines in Arabic script

the *abcde-s* configuration of the IFN/ENIT database [14] with a simple feature extraction method based on pixel grayscale intensity values. We thereby outperform the best system in the literature [1] by 23% relative. On the KHATT [11] corpus we achieve 11% relative improvement compared to the best system in the literature [8]. Additionally, we propose a novel segment-based approach to feature extraction which conveys enough information to fully recover the written script image from a sequence of low dimensional feature vectors (i.e. the original image is still reconstructible from the feature vector sequence). The resulting features are similar to *Autonomously Normalized Horizontal Differential Features* [6] but are more suitable for using them within the HMM paradigm. Segment-based features lead to comparable or better results in all configurations of the IFN/ENIT database compared to the respective state-of-the art systems of other researchers.

2 Text Line Image Normalization

2.1 Baseline Estimation

We describe our baseline estimation in [18]. Our method assumes that the zone between the lower and the upper baseline (core zone) usually includes a large fraction of all foreground pixels – i.e. is a dense foreground region in the image (Fig. 2). It is justified by the fact that ascenders and descenders usually contribute relatively little to all foreground pixels compared to the core zone. We search for the narrowest stripe in the image that contain a certain fraction of all foreground pixels. We detect the lower baseline at the bottom border of the stripe. Our method outperforms a previous method by 22.4% relative for the task of finding acceptable baselines in Tunisian town names in the IFN/ENIT database [14]. However, if the baseline is curved or discontinuous, our method fails since it tries to fit a straight dense stripe to the image. We propose in [18] to vertically split the image into smaller segments. Our splitting procedure ensures that we do not cut any foreground stroke (Fig. 3). We estimate the baseline for each of the segments separately and rotate/translate them such that the baseline is horizontal at a predefined height. Concatenating all segments results in a normalized image with a rectified straight and horizontal baseline. This procedure has proved effective on the KHATT database [11].

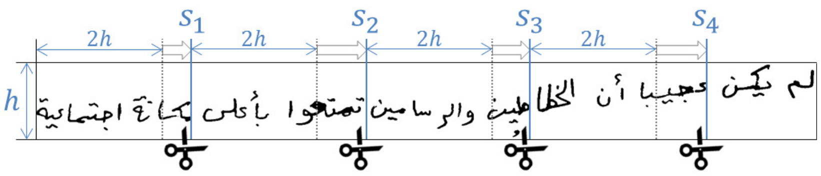
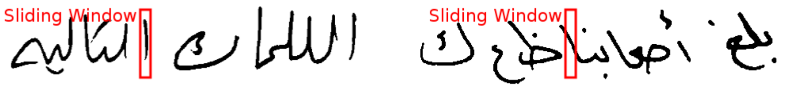


Fig. 3. Image of curved text lines into smaller segments



(a) Text features only slight slant. (b) Text is heavily skewed to the left.

Fig. 4. Different slants in Arabic handwriting.

2.2 Slant Correction

Arabic handwriting is usually italic, i.e. the letters tend to tilt to the left or right instead of being perfectly aligned with the vertical axis. Fig. 4 shows two text images with different slants. The variation in slant heavily affects the feature vectors. For example, the Arabic letter *alif* ("ا") usually stands out from the surrounding letters due to its vertical orientation and its non-connecting characteristic. However, with a high degree of slant, the letter does not fit in a single sliding window and the resulting feature vector is fundamentally different. We reduce these artefacts by applying a shear transform with angle $-\sigma$ to the image. As illustrated in Fig. 5 we estimate the slant angle σ as follows:

1. First, we apply the Hough [5] transform to the text image. In order to avoid quantization errors, we use the modified version of the Hough transform suggested in [18] that represents lines with angle $\Theta + 90°$ and position ρ of its intersection with the x-axis instead of using polar coordinates. We shift Θ by $90°$ such that an orthogonal intersection is represented with $\Theta = 0°$.
2. Straight lines in σ-direction (e.g. caused by the Arabic letters "ا", "ل", "ط", ...) trigger sharp peaks in the projection as shown in Fig. 6(a). In contrast, the projection profile in other directions is blurred (Fig. 6(b)). Therefore, we expect large variations and sudden changes in the Hough transformed image along the ρ-axis in slant direction ($\Theta = \sigma$). Consequently, we take the derivative in ρ-direction of the Hough transformed image using the Sobel filter.
3. Both positive and negative derivatives indicate large and sharp jumps in the projection profile. Therefore, we take the element-wise square.
4. For each angle Θ we sum over all elements in the corresponding row. We set σ to the angle which maximizes this sum.
5. The original text image is corrected with a shear transform with angle $-\sigma$.

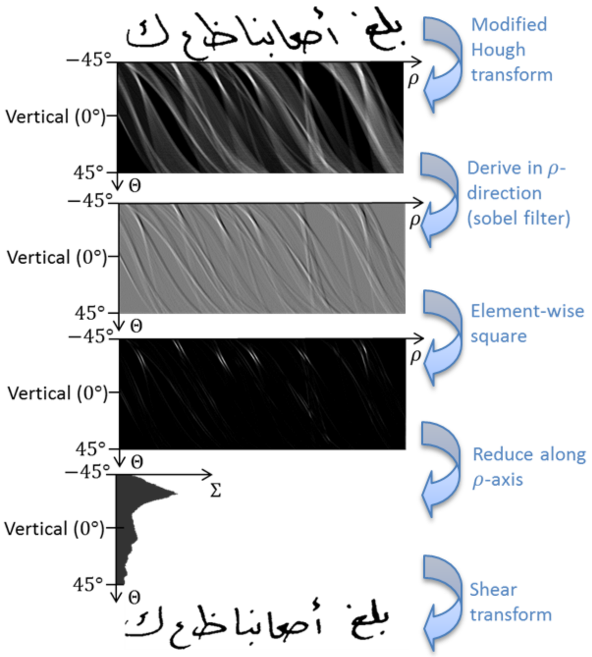
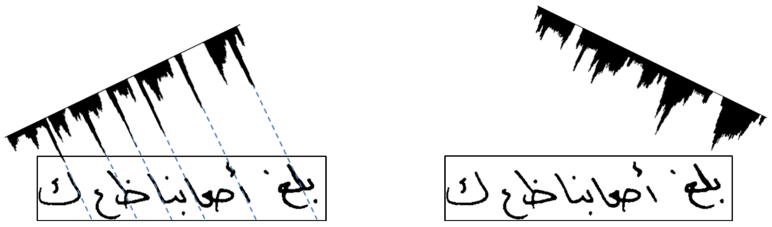


Fig. 5. Slant normalization



(a) Sharp peeks in slant direction. (b) Smoother surface in other directions.

Fig. 6. Slant angle estimation using projection profiles

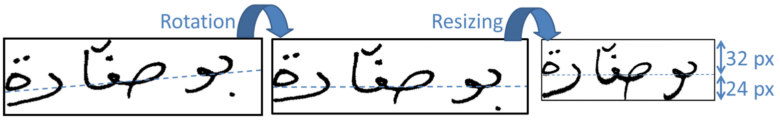


Fig. 7. Image size normalization

2.3 Size Normalization

In order to use pixel values in a sliding window as features, all images need to have the same height because the dimensionality of all feature vectors needs to be constant. Even for segment-based features, image size normalization is beneficial as it reduces the impact of different glyph sizes on the feature vectors. Therefore, we rescale the image to a height of 48 pixels and enforce that the baseline is positioned at a height of 32 pixels: First, we calculate the scaling factor for resizing the partial image over the baseline to a height of 32 pixels with fixed aspect ratio. Then we rescale the entire image with this factor. If the resulting image does not have a height of 48 pixels, we stretch or shrink the area below the baseline accordingly in the vertical direction.

2.4 Pen Size Normalization

Different pen sizes also influence the feature vectors without contributing useful information for the recognition. Additionally, the previous step can introduce undesirable variations in line thickness. Therefore, we apply the line thinning algorithm described in [20] to the image. A dilation operation with ellipse shaped 3x3 kernel followed by a convolution with a 5x5 Gauss filter rethick the lines to a normalized width and blur them to reduce the impact of small variations in line positions. Fig. 8 shows a complete overview of our image normalization steps.

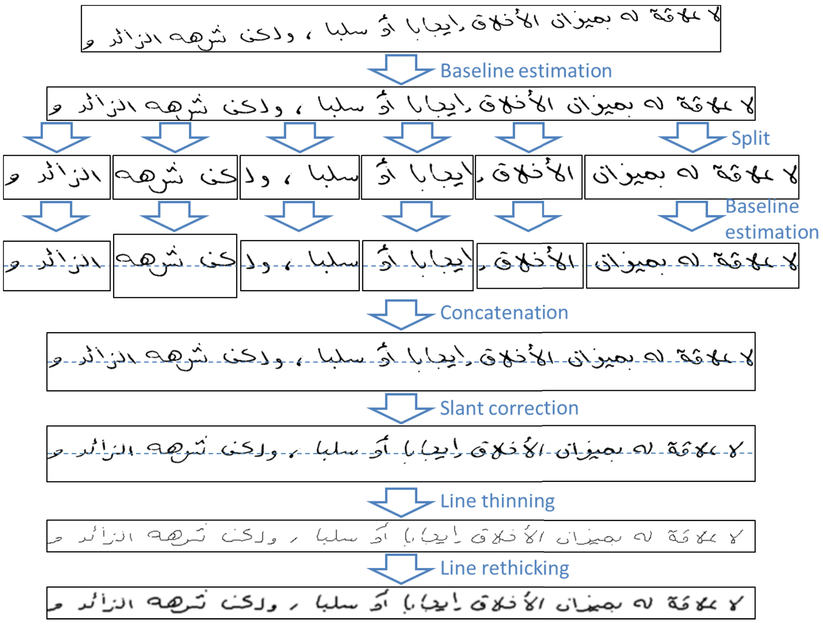


Fig. 8. Overview of all text image normalization steps

3 Feature Extraction

We investigate two different feature extraction strategies: pixel-based and segment-based features. In both methods, feature vectors are derived from a sliding window shifted in reading direction over the normalized text line image (window width: 3 pixels, window shift: 2 pixels). Pixel-based features utilize the raw grayscale intensity values of the pixels within the window. Since the image height is normalized to 48 pixels and the window width is set to 3 pixels, the dimensionality of pixel-based feature vectors is $48 \cdot 3 = 144$.

The high dimensionality of pixel-based features requires rigorous dimensionality reduction with standard techniques like principal component analysis (see Sec. 4). In contrast, the author in [6] proposed a low dimensional feature vector representation which is able to fully recover the original binarized image – i.e. given the feature vector sequence, the original text image can be reconstructed without any loss. In this work, we extend his method with a more natural way to represent connectivity of foreground segments in adjacent windows. As shown in Fig. 9, our segment-based features consist of six centroid features $c_{1...6}$ plus six segment height features $h_{1...6}$ (i.e. 12 dimensions). An area of consecutive foreground pixels within the sliding window is called *segment*. Suppose that there are n segments in the window. If $n = 6$, we set c_i to the y-coordinate of the centroid of the i-th segment, and h_i to height of that segment. In Arabic, the number of segments within a window usually does not exceed six. If the current window contains more than six segments due to binarization or segmentation errors, we discard the lowest $n-6$ segments. If the window contains less than six segments, we distribute the values over the indices 1 to 6 equally. For instance, in case of only two segments, c_1, c_2, and c_3 hold the centroid of the upper segment, and c_4, c_5, c_6 contain the centroid of the lower segment. Analogously, h_1, h_2, and h_3 are the height of the upper segment, and h_4, h_5, and h_6 the height of the lower segment. Windows without any segments are represented with the 0-vector.

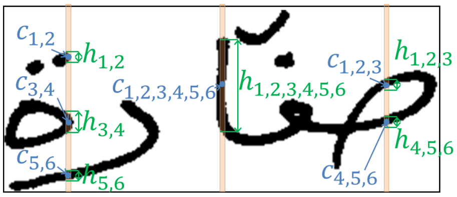


Fig. 9. Extraction of segment-based features

4 Training Procedure

Our recognition system is based on the Kaldi speech recognition toolkit [15]. The Arabic script features 28 letters with two to four contextual forms which we treat separately. As suggested in [1], we insert dedicated *sil* states after each non-connecting character, and *conn* states after each connecting character. We added punctuation symbols and numbers and ended up with a glyph set size of 133 for the KHATT corpus and 126 for the IFN/ENIT database (corresponding to the phoneme set size in speech recognition). We follow a common training scheme

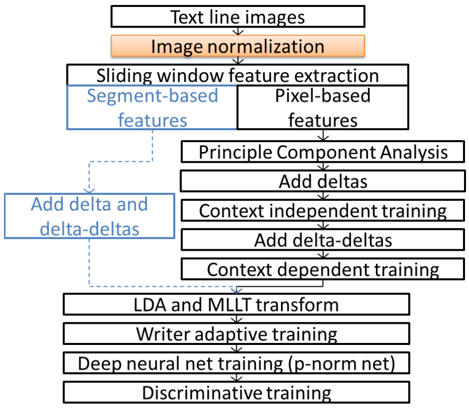

Text line images
Image normalization
Sliding window feature extraction
Segment-based features
Pixel-based features
Principle Component Analysis
Add deltas
Add delta and delta-deltas
Context independent training
Add delta-deltas
Context dependent training
LDA and MLLT transform
Writer adaptive training
Deep neural net training (p-norm net)
Discriminative training

Fig. 10. Overview of the training procedure

specified in Kaldi's example scripts (Fig. 10): First, we normalize the input images (Sec. 2) and extract the 144-dimensional pixel-based feature vectors as described in Sec. 3 (right path in Fig. 10). Note that even if we use segment-based features later on, the early training stages are always based on pixel features. We empirically found out that segment-based features are not useful to start with. A principal component analysis reduces the dimensionality to 8. We add the dynamic delta features and train the context independent system with 40k Gaussians. Then we add the second order dynamic features (delta-deltas) and train a context dependent system with 55k Gaussians in total (context width of 3, i.e. *triphone* in speech recognition terms). We stopped the decision tree splitting at 500 leaves. The tree includes questions regarding the shape of characters [10].

The forced-alignments of the context dependent system are used for deriving linear discriminant analysis (LDA) and maximum likelihood linear transform (MLLT) features [7] with 12 dimensions. At this point, the training differs depending on the selected feature set for the final recognizer (segment-based versus pixel-based). In our experiments with pixel-based features, the LDA and MLLT transforms are estimated on the original pixel features (before applying the principle component analysis). In case of segment-based features, the LDA and MLLT transforms are estimated on the raw segment-based features (left path in Fig. 10 highlighted in blue). On top of the resulting features we apply writer adaptive training (known as *speaker adaptive training* [2] in speech recognition). For deep neural network based optical modelling, we use Kaldi's *nnet2* recipe [16] which supports discriminative training of the network.

Arabic characters differ largely in their complexity. Therefore, authors in [4] use a small number of HMM states for simple characters, and model complex characters with more HMM states. First, we train a context-dependent system with a 3-state left-to-right HMM topology. Then, we estimate the best number of states for each glyph using the forced-alignments of this initial system. Lastly, the final system is trained from scratch with the adjusted HMM topology. This method known as *glyph dependent model lengths* is described in detail in [4].

5 Experiments

5.1 Data Description

We evaluate our recognition system on two different Arabic handwriting recognition tasks. The IFN/ENIT database [14] is a freely available collection of handwritten Tunisian town names and was used in a number of competitions [13]. Images contain one of 937 different town names (i.e. no out-of-vocabulary words and no language model). The corpus is divided into the subsets a, b, c, d, e, f, s (Tab. 1). The s set has been collected in a different region with different writing styles and posses the most challenges to the recognizer.

Table 1. Data statistics for the IFN/ENIT database

	Set a	Set b	Set c	Set d	Set e	Set f	Set s	$\sum$
Number of words	6,537	6,710	6,477	6,735	6,033	8,671	1,573	42,736

The KHATT corpus [11] consists of line images extracted from text areas in forms filled out by a large variety of writers with different origin, educational background, age, and handedness. For the KHATT corpus, we use a trigram language model trained on the KHATT training set. Tab. 2 contains information about the corpus size. The out-of-vocabulary rate on the test set is 11%.

Table 2. Data statistics for the KHATT corpus

	Train set	Dev set	Test set	$\sum$
Number of lines	9,462	1,899	1,996	13,357
Number of word tokens	131,716	26,635	26,921	185,272

5.2 Results

Tab. 3 and 4 compare our recognition system with the best results reported so far on the respective data sets. The best word error rate (WER) for each configuration is written in bold font. Tab. 3 shows four different *train set* - *test set* configurations for the IFN/ENIT database. Our recognizer performance is comparable to state-of-the-art systems on the *abc-d*, *abcd-e*, *abcde-f* configurations with both pixel-based and segment-based features. Discriminative training usually improves the WER slightly. However, training on sets a, b, c, d, e and testing on the set s (*abcde-s* configuration) leads to our largest gains compared to the best system in the literature. Using pixel-based features and discriminative training results in 11.5% WER which constitutes a relative gain of 23%, but it is

Table 3. Word error rate on the IFN/ENIT database (in %)

	abc-d	*abcd-e*	*abcde-f*	*abcde-s*
UPV-PRHLT [12]	4.8	**6.1**	7.8	15.4
Azeem and Ahmed [3]	**2.3**	6.6	6.9	15.2
Ahmad et al. [1]	2.8	6.5	7.8	14.9
This work				
Pixel-based features	2.7	6.9	7.3	12.3
Segment-based features	2.5	6.3	6.9	12.5
Pixel-based features + discriminative training	2.9	6.6	7.0	**11.5**
Segment-based features + discriminative training	2.4	**6.1**	**6.8**	11.9

Table 4. Word error rate on the KHATT corpus (in %)

	Dev set	Test set
Hamdani et al. (baseline for constrained task) [8]	33.6	34.1
This work		
Pixel-based features	**29.4**	**30.5**
Segment-based features	29.5	30.9
Pixel-based features + discriminative training	30.3	31.6
Segment-based features + discriminative training	29.9	30.9

not optimal for the other configurations. Segment-based features with discriminative training still outperform the best system so far by 20% (11.9% WER) on the *abcde-s* configuration, but also achieve state-of-the-art performance on the other configurations.

To the best of our knowledge, the best handwritten text recognizer for the KHATT corpus is described in [8]. In contrast to [8] we do not focus on language modelling. Therefore, we compare our system to the baseline of the restricted task in [8]. Pixel-based features work slightly better than segment-based features and lead to our best WER on the test set of 30.5% (11% relative gain). Discriminative training does not improve recognition accuracy in this case.

6 Conclusion

In this work, we described our recognition system for handwritten Arabic developed at the Qatar Computing Research Institute. We achieve a high degree of robustness with intensive text image normalization. Feature extraction was done either using the raw pixel grayscale intensity values, or a novel method based on foreground segments (segment-based). The recognizer was developed with the Kaldi toolkit and used discriminatively trained deep neural networks for optical modelling. We outperform the best system in the literature by 23% relative on the *abcde-s* configuration of the IFN/ENIT database. On the KHATT corpus, we report a relative gain of 11% compared to the state-of-the-art.

References

1. Ahmad, I., Fink, G.A., Mahmoud, S.A.: Improvements in sub-character HMM model based arabic text recognition. In: ICFHR (2014)
2. Anastasakos, T., McDonough, J., Schwartz, R., Makhoul, J.: A compact model for speaker-adaptive training. In: ICSL. IEEE (1996)
3. Azeem, S.A., Ahmed, H.: Effective technique for the recognition of offline Arabic handwritten words using hidden Markov models. IJDAR **16**(4), 399–412 (2013)
4. Dreuw, P., Rybach, D., Gollan, C., Ney, H.: Writer adaptive training and writing variant model refinement for offline arabic handwriting recognition. In: ICDAR. IEEE (2009)
5. Duda, R.O., Hart, P.E.: Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM **15**(1) (1972)
6. El-Mahallawy, M.S.M.: A Large Scale HMM-Based Omni Font-Written OCR System for Cursive Scripts. Ph.D. thesis, Faculty of Engineering, Cairo University Giza, Egypt (2008)
7. Gales, M.: Semi-tied covariance matrices for hidden Markov models. Transactions on Speech and Audio Processing **7**(3), 272–281 (1999)
8. Hamdani, M., Mousa, A.D., Ney, H.: Open vocabulary arabic handwriting recognition using morphological decomposition. In: ICDAR. IEEE (2013)
9. Huang, X., Acero, A., Hon, H.W., R., R.: Spoken language processing: a guide to theory, algorithm, and system development. Prentice Hall PTR (2001)
10. Likforman-Sulem, L., Mohammad, R.A.H., Mokbel, C., Menasri, F., Bianne-Bernard, A., Kermorvant, C.: Features for HMM-based arabic handwritten word recognition systems. In: Guide to OCR for Arabic Scripts. Springer (2012)
11. Mahmoud, S.A., Ahmad, I., Alshayeb, M., Al-Khatib, W.G., Parvez, M.T., Fink, G.A., Märgner, V., Abed, H.E.: KHATT: arabic offline handwritten text database. In: ICFHR (2012)
12. Margner, V., Abed, H.E.: ICFHR 2010-arabic handwriting recognition competition. In: ICFHR. IEEE (2010)
13. Märgner, V., El Abed, H.: Arabic handwriting recognition competitions. In: Guide to OCR for Arabic Scripts, pp. 395–422. Springer (2012)
14. Pechwitz, M., Maddouri, S.S., Märgner, V., Ellouze, N., Amiri, H., et al.: IFN/ENIT-database of handwritten arabic words. In: CIFED (2002)
15. Povey, D., Ghoshal, A., Boulianne, G., Burget, L., Glembek, O., Goel, N., Hannemann, M., Motlicek, P., Qian, Y., Schwarz, P., Silovsky, J., Stemmer, G., Vesely, K.: The kaldi speech recognition toolkit. In: ASRU (2011)
16. Povey, D., Zhang, X., Khudanpur, S.: Parallel training of Deep Neural Networks with Natural Gradient and Parameter Averaging. CoRR (2014)
17. Rybach, D., Gollan, C., Heigold, G., Hoffmeister, B., Lööf, J., Schlüter, R., Ney, H.: The RWTH Aachen university open source speech recognition system. In: Interspeech (2009)
18. Stahlberg, F., Vogel, S.: Detecting dense foreground stripes in arabic handwriting for accurate baseline positioning. In: ICDAR. IEEE (2015) (to be published)
19. Young, S., Woodland, P., Evermann, G., Gales, M.: The HTK Toolkit 3.4. 1 (2013)
20. Zhang, T.Y., Suen, C.Y.: A fast parallel algorithm for thinning digital patterns. Communications of the ACM **27**(3), 236–239 (1984)