AUTOMATIC RECOGNITION OF COMMON ARABIC HANDWRITTEN WORDS BASED ON OCR AND N-GRAMS

Laslo Dinges[†] *Ayoub Al-Hamadi*[†] *Moftah Elzobi*[†] *Andreas Nürnberger*[⋆]

Institute for Information Technology and Communications (IIKT)[†],
Department for Technical & Business Information Systems (ITI)[⋆],
Otto-von-Guericke-University Magdeburg, D-39016 Magdeburg, Germany;

ABSTRACT

Comprehensive databases are vital for training and validation of word recognition systems. To overcome the lack of offline databases of Arabic handwritten words, especially regarding the generality of the underlying vocabulary, we used a synthesis system to generate a database of common Arabic handwritings. Subsequently, we validate a new word recognition system on these synthetic handwritings, to analyze the performance of its segmentation, character recognition, and error correction module. We found, that a dynamic character classifier, that is capable to adapted to the variations that are caused by the segmentation, clearly improves word recognition accuracy. For error detection and correction, n-grams as well as the Levenstein distance to a vocabulary of up to 50,000 valid words have been used.

Index Terms— OCR, Arabic Handwriting, n-gram, Active Shape Models

1. INTRODUCTION

Today, due to the development of cheaper and more flexible devices as tablets, traditional documents are increasingly exchanged by their digital counterpart. This is why also digitalization of existing handwritten text becomes more and more important. Besides methods that allow to acquire handwritings efficiently – and, in case of historical documents, non invasive – automatic word recognition is crucial in this regard. Thus, in this paper, we propose a new segmentation based system for offline word recognition, which is optimized for Arabic handwritings. In contrast to online approaches, offline samples (images) do not contain features as the trajectory or speed of the handwriting, which complicates the recognition process. On the other hand, offline methods have a wider application area and can even be used in case of historical documents.

The offline word recognition system, that we propose in this paper, segments a word into single characters, which are the 28 Arabic letters. These letters are then recognized by Support Vector Machines (SVMs) and Active Shape Models (ASMs). Using a Levenstein comparison with an active vocabulary of 5,000 words, about 50% of the handwritings could be recognized correctly. Applying the n-grams – trained on 50,000 common Arabic words – about 54% of all words with segmentation or recognition errors can be detected. To ensure the availability of sufficient handwriting samples for validation, we use the synthesis system that we proposed in [1]. The rest of the paper is organize as shown in Fig. 1.

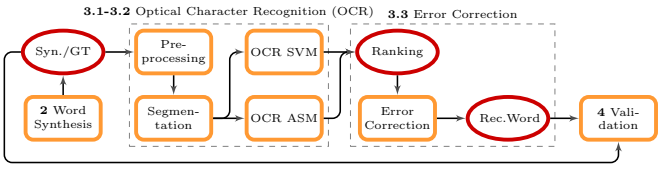

Fig. 1: Overview of the proposed approach.

1.1. Related Works

There are only a few systems that synthesize Arabic handwriting. Elerian et al. concatenate images of Arabic letters, ensuring that only samples with similar angle and line width (of their joint) are connected [2]. Another approach has been published by R. Saabni and J. El-Sana [3]. Online letter samples (without diacritics) are acquired using tablets and then concatenated to PAWs.

Word recognition was one of the earliest applications of image processing. Nevertheless, the automatic recognition of handwritten words is still challenging, especially in the case of Arabic. Beside holistic word recognition approaches, that require multiple training samples for each word of a used vocabulary, there are approaches based on implicit or explicit segmentation [4]. Implicit approaches are typically based on Hidden Markov Models or Neural Networks [5], extracting sequences of features from sliding windows (or the word trajectory in case of online approaches). Segmentation is then a byproduct of letter recognition. Explicit segmentation is based on topological features, and performed separately before letter recognition, which often involves the extraction of statistical features from normalized, segmented letter images. These features are then interpreted by classifiers, as the well

known Support Vector Machines [6]. Also key feature detectors as SIFT or SURF can be used to detect letters, or to extract features from sub-images for classification [7].

2. SYNTHETIC ARABIC WORD DATABASE

We use a synthesis system that is capable of generating images of unique Arabic handwritten words or text pages. For the experiments in section 4, a database called IESK-arDB$_{\mathsf{SynWords}}$ has been created, which contains 5,000 word samples reflecting common contemporary Arabic texts.

The used word synthesis system connects unique glyphs which are generated by Active Shape Models (ASMs). These glyphs are build from normalized online letter sample trajectories $\mathbf{x}$, where $|\mathbf{x}| = 25$. Every ASM allows generation of unlimited glyph trajectories $\mathbf{u}$, as shown in Fig. 2a. The basic synthesis processes are visualized in Fig. 2. A detailed description of our synthesis system is given in [8, 1].

(a) $\mathbf{u}$ (b) $\breve{\mathbf{u}}$ (B-Spline) (c) Binary (d) Feather

Fig. 2: Visualization of the main synthesis stages using the example هو.

IESK-arDB$_{\mathsf{SynWords}}$: Since Arabic letter shapes strongly depend on their position within a word, not only handwriting variation but also the word itself (letter permutation) has a strong influence on handwritings. Hence, databases for training and testing would optimally contain words which are expected in various real-world applications. Thus, to built IESK-arDB$_{\mathsf{SynWords}}$, we used a vocabulary $\mathcal{V}_{\mathsf{50k}}$ of the 50,000 most common Arabic words, computed from modern Texts. We found, that $\mathcal{V}_{\mathsf{50k}}$ also covers in average $62.4 \pm 7.0\%$ and the subset $\mathcal{V}_{\mathsf{5k}}$ at least $44.6 \pm 6.2\%$ of the words of the historical documents (14^{th} century) of the IESK-arDB.

For IESK-arDB$_{\mathsf{SynWords}}$, we randomly choose a word from vocabulary $\mathcal{V} \in \{\mathcal{V}_{50}, \mathcal{V}_{500}, \mathcal{V}_{\mathsf{5k}}, \mathcal{V}_{\mathsf{50k}}\}$ according to its probability of occurrence (unigram) within a text. Some samples of IESK-arDB$_{\mathsf{SynWords}}$ are shown in Fig. 3.

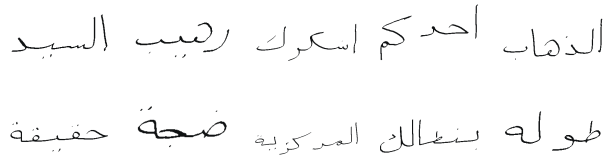

Fig. 3: Samples of synthesised Arabic Words.

3. AUTOMATIC RECOGNITION OF ARABIC HANDWRITINGS

To recognize Arabic words, we integrate segmentation, letter recognition and error correction methods in one system.

3.1. Word Segmentation

We proposed a novel method of segmenting handwritten Arabic word images into their letters in [9]. Based on vertical projections, candidate points, where letters can be divided, are found. These candidates are then reduced by topologically based rules until the final set of points is found and the bounds of all letters can be returned as the segmentation result. The proposed segmentation method was designed carefully and tested successfully on IESK-arDB [9]. However – due to the variations in handwritings – misplaced, missing or additional segmentations can also be caused. Especially the dynamic behavior of Arabic letter shapes and width increases the challenge compared to Latin. This is also true for the synthesized words, which are based on real handwritten letter samples.

3.2. Character Recognition

The segmented letter images are classified using two different methods, Support Vector Machines (SVMs) and Active Shape Model (ASM) based classifiers, that we proposed in [10]. SVMs are well known, fast classifiers, which can be trained on any statistical features. ASMs based classification is slower, since it requires an iterative matching of the sample and the ASMs of all candidate classes. However, better word recognition rate can be achieved due to a dynamic modification of the ASMs. Samples, which are used to train the SVMs and build the ASMs, are always from different writers than test samples, to ensure a writer independent recognition system.

We train separate SVMs for all four letter positions within a word (isolated, beginning, middle, ending). But in order to achieve results compatible with n-grams, no pre-classification (see below) is used. Instead, the number of dots and loops are used as additional features (achieving the same recognition results $\pm 1\%$). The prediction method of the used SVMs returns a ranking inclusive the likelihoods for all 28 letter classes, which is useful for the error detection and correction step. Features are extracted from normalized, thinned letter images, as shown in Fig. 4. We use the features proposed in [11] and [12], which are based on the neighborhood of pixels and are computed for all sub-images. The SVMs have Radial Basic Function kernels, where the parameters C and γ for each SVM are optimized using grid search.

In case of the ASMs, we firstly perform a pre-classification by assigning a letter to a reduced sets of candidate classes. Therefore, a decision tree interprets the number of dots (diacritics) and loops, since many Arabic letters differ only due to these discrete features. Hence, only the main bodies are used for actual classification, which also reduces the high costs of ASM matching. ASMs are built for all letter classes and positions. But in contrast to the ASMs of the synthesis system [1], samples of multiple writers are used for each ASM which increases classification accuracy and lowers compu-

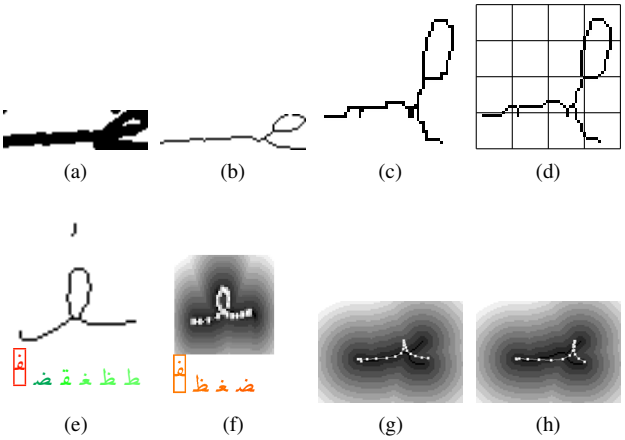

Fig. 4: Letter normalization process for (a-e) SVMs, (f,g) ASMs. (a) Segmented letter غ. (b) Polygon (contour) computed from thinned letter (ignoring diacritics). (c) 50×50 image, resulting from normalized polygon. (d) After second thinning, features can be extracted from sub-images. (e) Example for SVM based ranking (best 6 of 28 letters) without pre-classification. (f) ASM based ranking (after pre-classification). (g) 1^{st} iteration ($\mathbf{u} = \bar{\mathbf{x}}$), adapting an ASM (ر) to the chamfer transformed target sample. (h) 6^{th} iteration, initial Kashida length was reduced.

tational costs. We apply chamfer distance transformation to the segmented target letter, as shown in Fig. 4 (f-g). Then, measures of the correlation between the ASM and the target can be computed using the intensities of the chamfer image at the ASM coordinates. To adopt the ASM to the target, gradient descent is used, which starts with the average $\bar{\mathbf{u}}$. Thereafter, the ASM is improved by iteratively proceeding along the gradient of a feature space with optimized speed. The features are the intensities of the used eigenvectors (here 6 of maximal 25) and additional parameters as affine transformation, here just the Kashida length. To achieve smaller changes of the Kashida length, the new positions of the outer ASM coordinates are interpolated. Stronger adjustments are performed by deletion or insertion. Additional features are extracted from the ASM after the last iteration. This has to be done for the ASMs of all candidate classes, which are then ranked to find the letter class with the highest similarity to the target.

3.3. Error Detection and Correction

For every segmented letter $\mathbf{l}_i$, the classification proposes a ranking $\mathbf{r}_i$ of candidate letters and their likelihoods (see fig. 4 (e)). A low likelihood of the first rank might indicate a probable confusion. The average rank of the correct letter is 0.84±2.6 for SVMs (typically 0,1,2 or 3 but outliers of 10+ are possible) and 0.18±0.73 for ASMs (rank <= 5). If the correct letter has a rank > 0, the letter has been confused. Confusions can be predicted by the SVM likelihoods or using n-grams. We use trigrams on letter level, computing for each the probability $p(l_i|l_{i-1}, l_{i-2})$, that the current letter l_i is of a specific class if the classes of l_{i-1} and l_{i-2} are known. Additionally, unigrams and bigrams are calculated. Using $\mathcal{V}_{50k}$, 82.7% of the bigrams and 33.9% of the trigrams are valid ($p \neq 0$). Any word that contains one or more n-grams of $p = 0$ is flagged as misclassified. To solve letter confusions, we built a graph where the ranked letters are nodes and the n-grams edges. Therefore, the unigrams are used to calculate the edges $1 - p(l_0)$ from the (empty) start node, and the bigrams and trigrams for the remaining edges, as shown in Fig. 5. Then, a modified Dijkstra shortest path algorithms is applied. In contrast to original Dijkstra, the weight of an edge from the visited node l_{i-1} to the next node l_i depends also on the predecessor l_{i-2}. We calculate $\arg\min_{l_{i-2}} 1 - p(l_i|l_{i-1}, l_{i-2})$ to get the weight for l_i and set l_{i-2} as the current predecessor of l_{i-1}. Furthermore, the cost of l_i is a result of the incoming edge weight and the likelihood of the node. This way, the optimized path – that results in

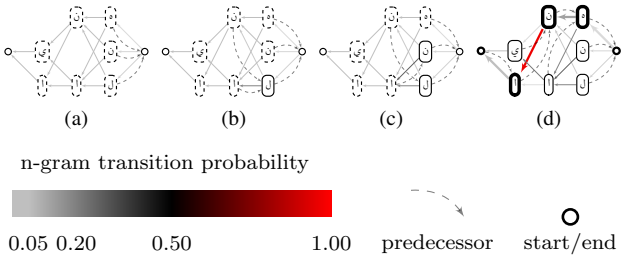

Fig. 5: Schematic example of modified Dijkstra algorithm for trigrams (all likelihoods are 0). (a) Step 1. (b) Step 2, "ا ل=لَا" is a very frequent bigram as well as the word "no". (d) The best path ه→ن→ا delivers the word "هنَا" ("here").

a sequence $\mathbf{w}_r$ of Unicode characters – can also contain letters of rank>0. If $\mathbf{w}_r$ is identical with the true word $\mathbf{w}_t$, the word has been recognized correctly. Otherwise, the number of errors is measured by the Levenstein distance

$$\delta_e = \delta(\mathbf{w}_r, \mathbf{w}_t). \quad (1)$$

To detect and correct such errors that might not be handled by the n-gram based method, we compare $\mathbf{w}_r$ with all entries $\mathbf{v}_i \in \mathcal{V}$ of the used vocabulary $\mathcal{V}$. Then, all $\mathbf{v}_i$ are ranked increasingly according to $\delta(\mathbf{w}_r, \mathbf{v}_i)$, where $\mathbf{v}_{\text{rank}=0}$ is the vocable with the highest similarity to the recognized word. Hence, $\mathbf{v}_{\text{rank}=0}$ is chosen as corrected word $\mathbf{w}'_r$ and the recognition is successful if $\mathbf{w}'_r = \mathbf{w}_t$. If there is more than one $\mathbf{v}_i$ with the same δ, we increase δ by the unigram probability on word level $P(\mathbf{v}_i)$. A validation follows in the next section.

4. EXPERIMENTS

Segmentation: Segmentation of Arabic handwritings is known as a crucial but error-prone step of word recognition.

Often words, that contain the letters س ش ص ض, are excluded from the testing database for segmentation methods, since they typically cannot be segmented correctly. However, since these words represent 25% of $\mathcal{V}_{50k}$, we include them to the test set. We found, that the detected segmentation faults of real and synthetic samples are comparable, using the vocabulary of the IESK-arDB word database [1]. Due to the fact common texts contain mainly small words, a higher rate (35%) of perfectly segmented words was measured using $\mathcal{V}_{5k}$. Segmentation faults might be misplaced segmentations, which could be compensated by letter recognition or n-gram based error correction, and additional or missing segmentations, which require the comparison with the vocabulary.

Table 1: SVM-based and ASM writer independent character recognition.

form	i	e	m	b	∅SVM*	∅ASM**
Precision	96.4	87.8	94.5	96	93.6±0.4	79.5
Recall	96.90	87.18	94.50	96.11	93.50±0.96	79.2
F_1-score	96.67	87.45	94.47	96.02	93.55±0.51	79.3

*Crossvalidation on sets that equally contain all classes are used to compute average accuracy

**Results for the ASM approach include errors from classification and pre-classification, which fails in 9.9% of all cases (SVM results are achieved without pre-classification).

Character Recognition: We trained all SVMs writer independent using the features described in [11] and the aspect ratio of the original letter image as well as the number of letter pixels. The SVMs are tested on letter samples of the remaining writers. As shown in table 1, the results depend on the letter position. Furthermore, many confusions of letters belong to the fact that their shapes only differ by their diacritics or loops, which are, however, not distinctive or misplaced in case of handwritings. SVMs for letter groups based on such features (taken from GT), that guarantee different letter shapes of all candidate classes, achieving much better results.

ASMs achieve similar results as SVMs. However, the SVMs precisions decrease to 64.7% when tested on segmented letter samples, while ASMs behave more robust. The main reason is the bias in Kashida length caused by the segmentation method, that can not be reflected by a straight forward letter database, but adapted by the ASM at runtime. Hence, ASMs appear to be the more dynamic classifiers, that can also be used to determine the critical features of unknown handwritings, as variation of Kashida length, rotation or slant.

Word Recognition and Error Detection: We validate word recognition using ASMs and SVMs with automatic and ground truth based segmentation. The percentage A'_w of correctly recognized words $\mathbf{w}_r = \mathbf{w}_t$ measures how accurate the system is using segmentation, letter recognition and optional n-gram based error correction. A_w is the percentage of words, that is recognized correctly or that could be corrected using the vocabulary $\mathcal{V}_{5k}$. As shown in table 2, the word recognition is best using ASM-based letter recognition. Despite segmentation and classification faults, A_w of about 50% could be achieved (29.45% using $\mathcal{V}_{50k}$). As shown in table 3, most of the wrongly recognized words can be detected using n-grams. This rate can be further increased when also trigrams with $P(A,B,C) > 0$ are considered. In future works, we plan to use error detection in order to choose between results of different segmentation proposals.

Table 2: Word recognition accuracy using vocabulary $\mathcal{V}_{5k}$ – before (A'_w) and after comparison with $\mathcal{V}_{5k}$ (A_w). The average Levenstein distance of the recognized word and the most similar vocable $\mathbf{v}_i \in \mathcal{V}_{5k}$ is given by $\bar{\delta_e}$. GT means that ground truth are used for segmentation.

	ASM	**ASM**$_{Gt}$	**SVM**	**SVM**$_{Gt}$
$\bar{\delta_e}$	1.6 ± 0.1	0.7 ± 0.04	2.±0.16 (2.1±0.1)*	0.36±0.01
A_w	49±0.6	74±1.7	32±0.4 (27.5±1)*	84.50±1.70
A'_w	27±0.3	54±0.1	15.±1.5 (22.3±0.2)*	73.54±1.62

*using Trigram based error correction

The measured computation time (on a machine with Intel(R) Core(TM)2 Quad CPU with 2.66 GHz and 4GB RAM using Windows 7 64 Bit) is $0.83 \pm 0.57s$ per word using SVMs and $1.31 \pm 0.75s$ using ASMs in case that the MATLAB based segmentation method is used. Using segmentation ground truth, we get $0.06 \pm 0.01s$ for SVMs and $0.72 \pm 0.28s$ for ASMs.

Table 3: Trigram based error detection rates in %.

	confusions[1]	1	2	3
Unicode:	$\mathcal{V}_{50k}$	44.18 ± 0.01	58.28 ± 0.01	66.3 ± 0.01
	$\mathcal{V}_{50k}^{random\ 2}$	32.0 ± 0.01	39.27 ± 0.12	42.52 ± 0.08
OCR:	**SVM**$_{Gt}$**:**	44.0 ±0.48	**SVM:**	54.7 ± 0.5

[1] Number of characters that have been confused with a randomly class.

[2] Choosing each time a word $\in \mathcal{V}_{50k}$ according to its a priori probability (with replacement).

5. CONCLUSION

We extended the IESK-arDB database by IESK-arDB$_{SynWords}$, a database of common Arabic words that contains synthesized handwritten word images and detailed ground truth. Then, a word recognition system based on a segmentation, character recognition, and error correction modules has been validated on IESK-arDB$_{SynWords}$. It turns out, that the proposed Active-Shape-Model-based classification approach outperforms Support Vector Machines if integrated into the word recognition module. Furthermore, error correction and detection clearly improves the word recognition results. In future works we are going to use Deep-Learning on character and word level, for comparison and combination with the current approaches.

6. REFERENCES

[1] Laslo Dinges, Ayoub Al-Hamadi, Moftah Elzobi, Sherif El etriby, and Ahmed Ghoneim, "Asm based synthesis of handwritten arabic text pages," *Scientific World Journal*, 2015.

[2] Yousef Elarian, Irfan Ahmad, Sameh Awaida, Wasfi G. Al-Khatib, and Abdelmalek Zidouri, "An arabic handwriting synthesis system," *Pattern Recognition*, , no. 0, pp. –, 2014.

[3] Rais M. Saabni and Jihad A. El-Sana, "Comprehensive synthetic arabic database for on/off-line script recognition research," Springer-Verlag, 2012.

[4] Leila Chergui, Mamar Kef, and Salim Chikhi, "New hybrid arabic handwriting recognizer," in *6th International Conference on Sciences of Electronics, Technologies of Information and Telecommunications (SETIT)*, 2012.

[5] Khaoula Jayech, Mohamed Ali Mahjoub, and Najoua Essoukri Ben Amara, "Arabic handwriting recognition based on synchronous multi-stream HMM without explicit segmentation," in *Hybrid Artificial Intelligent Systems - 10th International Conference, HAIS 2015, Bilbao, Spain, June 22-24, 2015, Proceedings*, 2015.

[6] Lara del Val, Alberto Izquierdo-Fuente, Juan J. Villacorta, and Mariano Raboso, "Acoustic biometric system based on preprocessing techniques and linear support vector machines," *Sensors*, vol. 15, no. 6, pp. 14241, 2015.

[7] Leila Chergui and Kef Maamar, "Sift descriptors for arabic handwriting recognition.," *International Journal of Computational Vision and Robotics 5.4*, pp. 441–461, 2015.

[8] Laslo Dinges, Ayoub Al-Hamadi, and Moftah Elzobi, "An approach for arabic handwriting synthesis based on active shape models," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2013, pp. 1292–1296.

[9] Moftah Elzobi, Ayoub Al-Hamadi, Zaher Al Aghbari, and Laslo Dinges, "Iesk-ardb: a database for handwritten arabic and optimized topological segmentation approach," in *International Journal on Document Analysis and Recognition*. Springer, 2012.

[10] Laslo Dinges, Ayoub Al-Hamadi, and Moftah Elzobi, "An active shape model based approach for arabic handwritten character recognition," in *The 11th International Conference on Signal Processing (ICSP'2012)*, 2012, pp. 1194–1198, accepted by The 11th International Conference on Signal Processing (ICSP'2012).

[11] A. D. Parkins and A. K. Nandi, "Simplifying hand written digit recognition using a genetic algorithm," 2002.

[12] Jamshid Shanbehzadeh, Hamed Pezashki, and Abdolhossein Sarrafzadeh, "Features extraction from farsi hand written letters," in *Proceedings of Image and Vision Computing*, New Zealand,, 2007, Proceedings of Image and Vision Computing, pp. 35–40.