Article

Synthetic-based Validation of Segmentation of Handwritten Arabic Words

Laslo Dinges, Ayoub Al-Hamadi, Moftah Elzobi, and Sherif El-Etriby | Magdeburg – Shebeen El-Kom

Abstract

Suitable and comprehensive databases are crucial for training and validation of various document analysis methods. However, those databases are limited, and their ground truth is often not sufficient for methods like word segmentation. In this article, we propose an active-shape model-based approach for Arabic handwriting synthesis that enables parameterization of several real-world Arabic handwriting distortions, e.g., skew, slant, variations in letter size inside a word, and length of their connections. Furthermore, we develop a technique to render such handwriting so that it reflects the features of real writing instruments. Finally, we test and validate a segmentation method on a large dataset of synthetic samples as well as real-world samples. The initial results are promising, suggesting the synthetic approach will be useful in various handwriting recognition areas.

1. Introduction

Document analysis methods, including automatic document classification, layout analysis, segmentation or text recognition, can be very helpful when dealing with huge amounts of data such as the Ottoman Archives. To test and compare these methods, comprehensive databases are essential. However, the problem of lacking satisfactory handwriting databases is very obvious for Arabic handwriting. Two main word databases that are free are available: the IFN/ENIT (Pechwitz et al. 2002), which contains the names of Tunisian towns, and our IESK-ArDB (Elzobi, et al. 2012), which contains international town names and common terms. Since the most expensive part of Ground Truthing databases is due to the information required for validating segmentation methods, the number of available samples that include such information is still limited. To bypass this problem, it would be very helpful if databases that are adapted to general or specific needs of a concrete document analysis task could be produced automatically. This includes precise manipulation of the degree of challenge, which is hardly possible with traditional databases.

Approaches to text synthesis can be classified into two main categories. The first category includes all research that generates synthetic text by perturbing real text samples, which can be complete units such as text lines or words, but also glyphs, e.g., groups of letters (Guyon 1996), Varga and Bunke 2003; Thomas, Rusu, and Govindaraju 2009; Miyao and Maruyama 2006). The second category encompasses all approaches that are based on deformable models in synthesizing text. Words are composed by randomly permuting and linking glyphs to obtain an unlimited number of synthetic samples (Cheung et al. 1998; Al-Zubi 2004; Shi, Gunn, and Damper 2003). Only a few approaches are known for Arabic. The first approach to Arabic handwriting synthesis uses character images from two writers in the IFN/ENIT database and composes them into words (Elarian, Al-Muhsateb, and Ghouti 2011). To achieve smooth word shapes, only letters are connected that have a similar angle and width (at their juncture), but this limits the number of different samples that can be generated for each word. Shortly thereafter, a second approach was proposed, which composes letter trajectories that are acquired by tablets (Saabni and El-Sana 2012). This approach allows multiple kinds of handwriting of a given Piece of Arabic Word (PAW) to be generated, but without any diacritical marks like dots, which are, however, distinctive features of Arabic letters.

In order to enable intuitive generation of synthetic Arabic text databases, we have developed a user interface (UI) that allows specific features to be manipulated, such as the slant, how well-written the characters are and the size of the *kashida*, which is the connection between two letters. Unlike connections in Latin-based scripts, a *kashida* is often wider than a letter itself, but sometimes vanishes entirely.

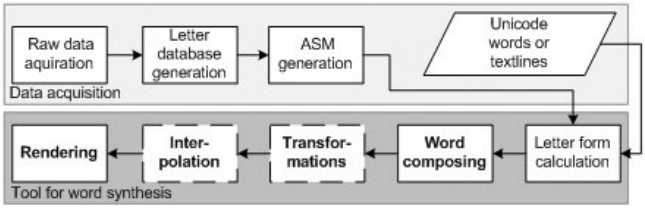


Fig. 1: Overview of the proposed synthesis approach (stippled boxes indicate optional processes).

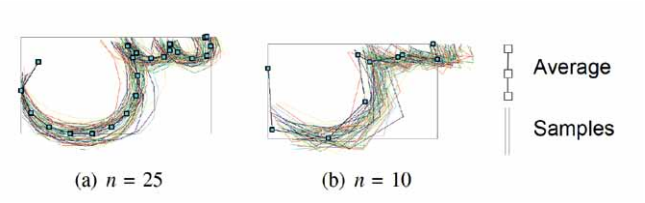


Fig. 2: Normalized samples and average shape of the letter *sin* in an isolated form using a) 25 landmarks (sufficient for all classes) and b) 10 landmarks (insufficient for most classes).

In the preview section of the UI, one can enter Arabic Unicode or type the Latin names of letters. Thereafter, a few synthesis examples can be generated in order to examine the effect of the current settings. Moreover, interpolation settings as well as predefined pigmentation textures (that are used to render the synthetic images) can be employed to achieve the desired outcome.

The UI enables quick generation of databases that focus on specific features. In that way, it is possible to investigate how robust methods are against different writings styles, carefully or roughly written text, and primitive features such as the slant and skew of the writing. We use these options in section 3 to validate a segmentation technique that we proposed in Elzobi et al. 2012.

2. Methodology

In this section, we describe our approach to synthesizing Arabic handwriting from Unicode. Note that the synthesized samples have either an image (png, bmp) or vector graphic format (pdf, eps or svg), but vector graphics will only be used for illustration here.

Fig. 1 shows the main modules of our synthesis approach. The first module includes the concept of data acquisition, mainly generating and processing raw data. The second module uses the acquired data as well as requests from the user to synthesize Arabic handwriting. This includes word composition, directed manipulation of features (like word slant) and the rendering technique.

The basic idea of handwritten Arabic word synthesis from Unicode sequences is to select polygonal samples with correct context-sensitive shapes and subsequently connect them to obtain Pieces of Arabic Words (PAWs), words and complete sentences. These are then modified by affine transformations, smoothed by B-spline interpolation and composed to text, as shown in fig. 3. Finally, the writing is rendered and saved, including ground truth (xml files). In this way, our system produces offline pseudo-handwritten samples with variations in shape and texture. At the end of this section, we briefly discuss word segmentation as an example of image-processing methods that can be validated using synthetic handwriting.

2.1 Data acquisition

Even though the synthesis module is built to create new letter and word samples automatically, suitable and sufficient samples of handwritten Arabic letters are needed in the first place. These samples are crucial to achieve a natural and comprehensive outcome. A total of 28,046 online samples from multiple writers were created to compute Active Shape Models (ASMs) for over one hundred letter classes, since experiments with offline samples led to poor results for most letters that have complex shapes (Dinges, Al-Hamadi, and Elzobi 2013). ASMs are statistically deformable models. To build an ASM, we defined a fixed number of *n* landmarks for all samples in the same class (see fig. 2).

The ASMs contain information about the position of these landmarks and how these positions statistically vary in relation to themselves and all other landmarks. ASMs reflect the main characteristics of the input samples used, which can be specifically manipulated by their eigenvectors. ASMs are used to generate unique letter representations for each synthesis. Since we intended to synthesize offline handwriting, we chose to use online pens instead of tablets, since the former can be used as ordinary biros and do not distort the handwriting style.

2.2 Letter-form calculation

In this and the following section, we describe how to build the trajectory of an Arabic handwriting sample from Unicode. Unicode strings represent every letter as a plain number. Although there are special Unicode signatures for the isolated, end, middle and beginning form (final, medial and initial), only the general form is used for normal Arabic text. Letter forms have a strong influence on letter shape, however, so they have to be determined first. Six letters (ر ا

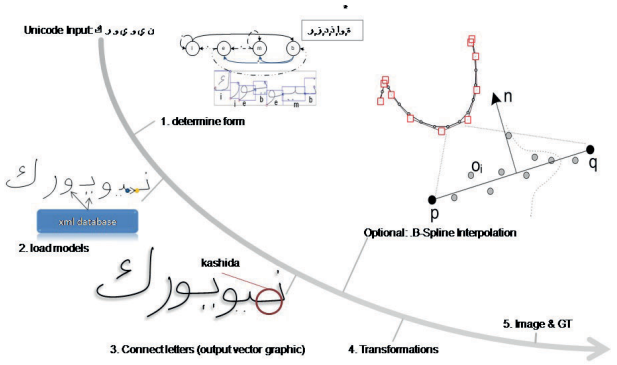


Fig. 3: Scheme shows how Unicode input is turned into a synthetic handwriting image.

د ز ذ و) can only assume an isolated or end form, while all the others also have a middle or initial form — take the letter *ayn*, for example (علعلع ع). Letters that have only two forms split an Arabic word into PAWs which consist of one or more letters. If the first letter comes from the set (ا ر د ز ذ و), it has an isolated form or an initial form. The forms of all the other letters are computed as in fig. 3. If any letter is followed by a space token, it initiates a new word and must assume an isolated or end form.

2.3 Word composition

After all the letter classes have been defined by their names and forms, the corresponding ASMs are loaded. These are used to generate unique shapes for all occurrences of a letter class in order to avoid piecemeal identical syntheses (original samples can be used optionally, though). In order to compose words from these letter shapes, each letter has to be connected to its predecessor in an end or medial form, as illustrated in fig. 3.

Since our samples are currently limited to the 28 regular letters and *tamarbuta* (ة), we substituted special characters, such as *alif* with *hamza* above (أ), with their regular form, *alif* (ا), before starting the synthesis.

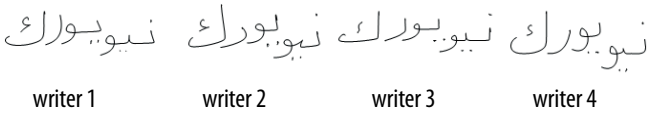


Fig. 4: Syntheses of the Arabic word for 'New York' using ASMs of different writers (vector graphic format).

The height of a PAW in relation to the lower baseline depends on the letter classes the PAW is composed of. Thus, we extracted the average distance (and standard deviation) between the baseline and the center of a letter from a manually created ground truth of our (real) word database IESK-arDB (Elzobi, et al. 2012). We corrected the y-position of all PAWs according to the average heights of all the letters within a PAW.

Finally, the horizontal gap between a PAW and its predecessor has to be defined. Therefore, a user-dependent parameter is applied that may be negative in order to simulate overlapping PAWs. 'Overlapping' means that a letter is above or beneath another one, but its trajectory should not intersect it; if two overlapping PAWs do intersect, the gap is increased by 25% of the average letter width until the intersection is solved. Some examples of such results are shown in fig. 4.

2.4 Transformations

ASMs already contain variations in slant, width and connection size. Nevertheless, these variations are limited by the samples used. In order to increase and control these variations, affine transformations are used that allow optional manipulations of letter and PAW shapes (see fig. 5).

The user interface (UI) of our synthesis tool allows the average μ and standard deviation σ of a Gaussian distribution to be set for all affine transformations. Global variations, in particular, can be achieved this way (e.g., slant or skew). The affine transformations used are scaling, translation, sharing and rotation. The influence these have on the resulting word

stretching	slant	size	skew	position	connection

Fig. 5: Effects of affine transformations and *kashida* cutting (image format).

Fig. 6: Example of syntheses of the PAW في using different *kashida* lengths. The example on the left shows an extremely reduced *kashida* (actually a 'negative' *kashida* that blends two letters). The middle example is created using a moderately reduced *kashida*. The *kashida* almost reaches its maximum size in the right-hand example.

image is shown in fig. 5. Stretching is performed by scaling the components of each letter point. The word slant can be set by sharing the word, the skew of a PAW by rotation. The size of the complete word or single letters can be adjusted by equal scaling, which can be used to control the resolution of the synthesized word images or to increase variation in the letter size.

As we found letter connections of the acquired samples of certain writers to be excessively long, we allow up to 25% of the letter points to be deleted to simulate stretched or missed connections, which often occur in real Arabic handwriting.

In Arabic handwriting, some characters, such as *ya* (ي), are written beneath rather than beside their predecessors. As a result, they resemble a single character, which impedes segmentation and recognition tasks. This effect can be simulated by a strong reduction of the *kashida*, as shown in fig. 6. However, this feature has not yet been completely implemented, since a list of all pairs of letters that typically show this behavior needs to be created first.

2.5 Rendering technique

As described in Dinges, Al-Hamadi, and Elzobi 2013, we compute the coordinates of all the pixels that will be influenced by a virtual pen and then apply an artificial texture to them. To prepare high-resolution syntheses, B-spline interpolation can be used before painting, as shown in fig. 7 (a–b).

To allow a more accurate simulation of writing instruments, such as biros, pencils, fountains pens or quills, we extended our rendering technique. This mainly meant implementing two features: writing speed and the shape of the top of the writing instrument, subsequently called Pen Shape.

Large lines or bows, as with the left part of *sin* (س), are usually written faster than more complex structures. A high writing speed often causes a lack of pigmentation that leads to brighter or dappled lines. However, there is no need to reconstruct writing speed, for the online letter samples we employed already contain this information. We used the normalized writing speed to reduce the pigmentation of our syntheses up to 20% of the average value, as shown in fig. 7 (c).

If Pen Shape is modeled as ellipsoid, the line width depends not only on Pen Shape size but also on the Pen Shape and line angle. We used an angle of 45° for Pen Shape, changing it continuously with a maximum deviation of ±15°. According to the features of Pen Shape, the contact with the paper and

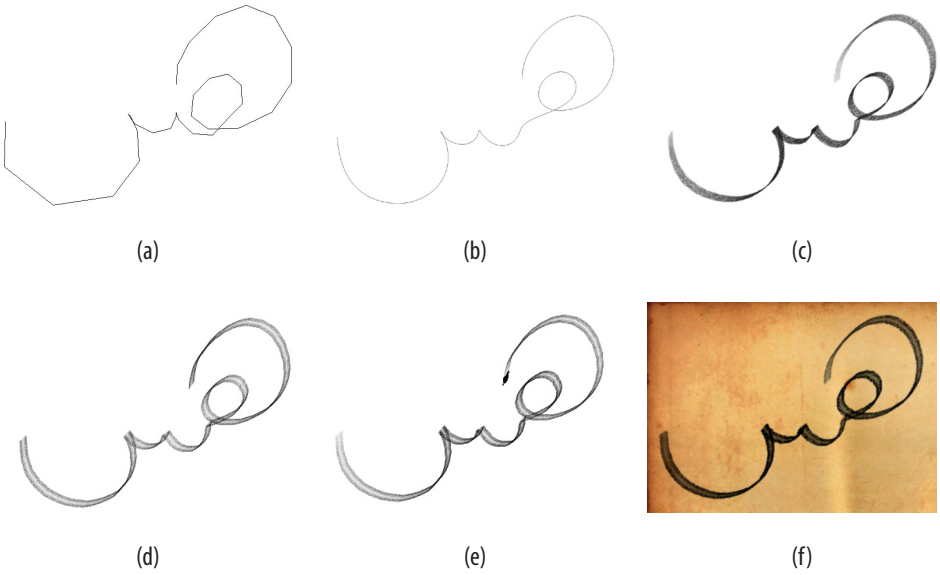
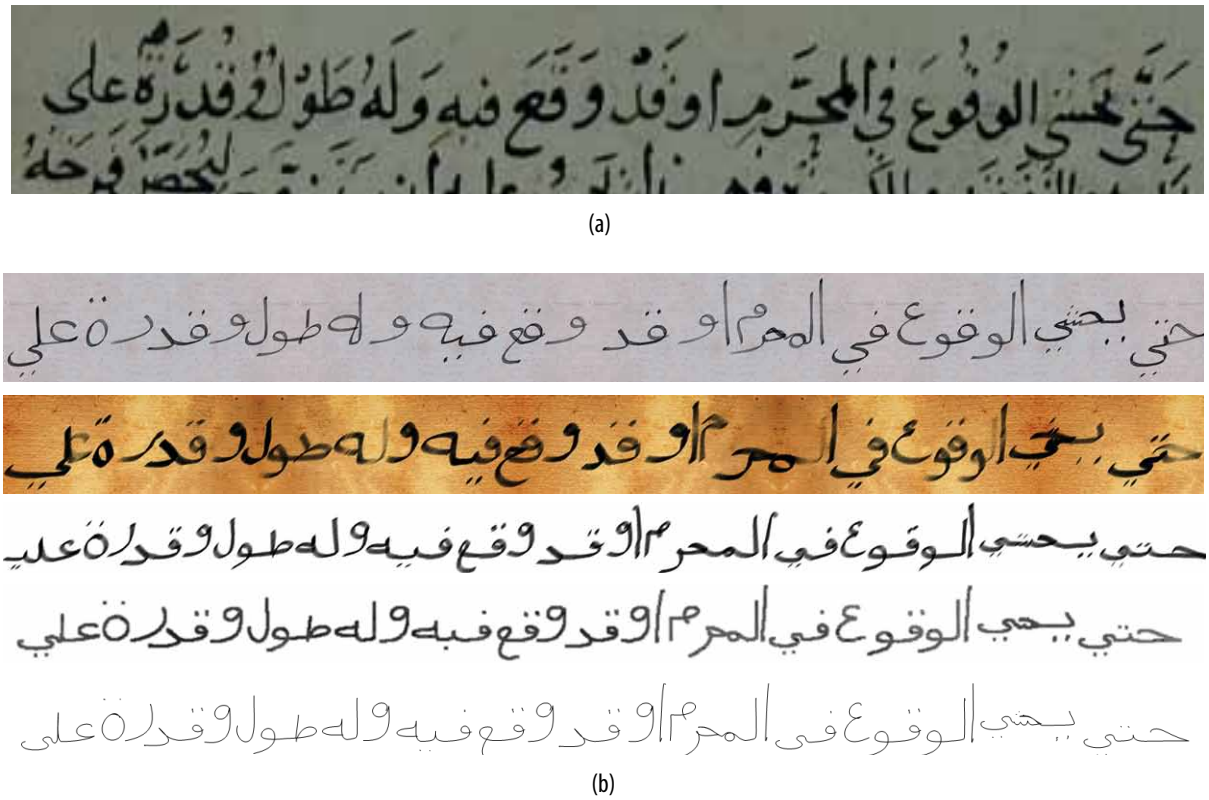


Fig. 7: From polygons to images of synthetic handwriting: (a) simple lines, (b) result after applying B-spline interpolation. Images (c) to (f) show the proposed painting technique using an ellipsoid Pen Shape to simulate a quill-like writing instrument: (c) influence of pen speed, (d) influence of Pen Shape, (e) combination of both, (f) result of (e) combined with an artificial ink texture and transparently drawn over a texture of old paper.



(a)

(b)

Fig. 8: a) First line 'حتي يحشي الوقوع في المحرم او قد وقع فيه وله طول وقدرة علي' of a historical document that is taken from the IESK-arDB manuscript database (file im5_1/gt5_1); (b) five syntheses. The first three images are rendered using an improved rendering technique, where the first two are drawn on a degraded background but the third is on a perfect white background. The fourth image was created using the technique described in Dinges, Al-Hamadi and Elzobi 2013. The last one is a vector graphic (PDF).

the resulting pigmentation can vary. This is simulated by a function that defines the pigmentation potential for each position of Pen Shape. An emphasized example is shown in fig. 7 (d), while fig. 7 (e) is also influenced by writing speed. We created fig. 7 (f) by combining our original rendering technique with the one based on Pen Shape and writing speed as well as using a non-uniform background. We therefore applied a transparent texture to all the pixels in fig. 7 (e) that do not belong to the background texture. Small, random irregularity is simulated this way.

In the examples shown, a black color is used, as most documents are written with dark ink. Pigmentation intensity is implemented as transparency, which allows simulating pigment accumulation at crossing lines or for a textured background. However, opaque or semi-opaque ink can also be simulated. This could be interesting in the context of historical documents, since important passages are often highlighted using red ink. Another important feature of historical documents is degradation of the material on which they have been written. Currently, we simply use images of real paper or parchment as background textures, which are scaled or tiled if they are smaller than the synthesized document.

2.6 Sample method to test applicability: segmentation of Arabic words

Since the data synthesis module is built to ease the development and validation of methods that are related to document analysis, it is not only of interest whether syntheses look realistic or not. In fact, it is crucial how image-processing methods behave when fed with synthetic instead of real data. This is investigated in the following section, where a method that segments handwritten Arabic words into letters is validated using such data. We chose word segmentation as the example due to its sensitivity to character shapes as well as global features, such as overlapping PAWs and varying *kashida* length.

One of the earliest segmentation-based approaches suggested for the recognition of handwritten Arabic text is the one proposed by Almuallim and Yamaguchi 1987, but no segmentation results were reported. Xiu et al. proposed

a probabilistic segmentation model in which tentative, contour-based over-segmentation was first performed on the text image. As a result, a set of what they called graphemes was produced (Xiu et al. 2006). The approach differentiated between three types of graphemes. The confidence of each character was calculated according to the probabilistic model while respecting other factors, such as recognition output, geometric confidence and logical constraint. The authors experimented with the proposed methodology on five different test sets, achieving a success rate of 59.2%.

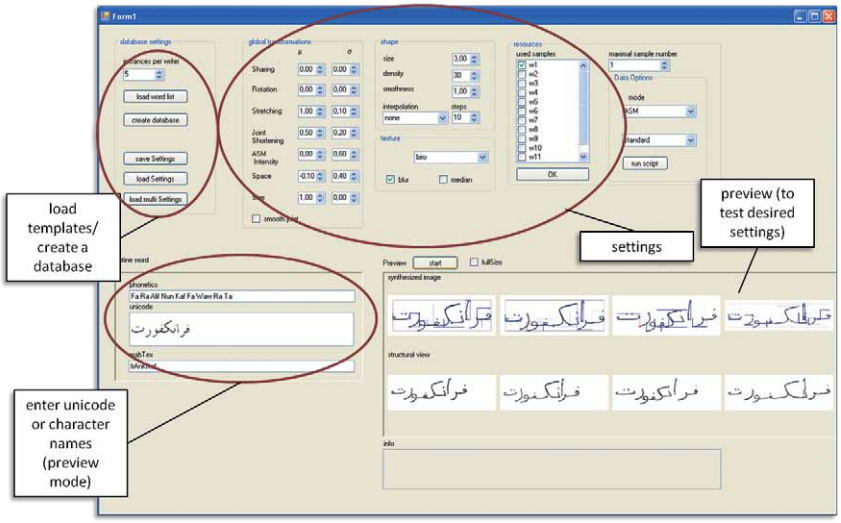


Fig. 9: a) Screen shot of the user interface, which enables handwriting to be synthesized; b) ground truth that is used to validate segmentation.

The segmentation method used for the following experiments is described in Elzobi et al. 2012. It is based on topological features and a set of rules that reduce all candidates to a final set of points that divide two neighboring letters. Unlike other approaches, candidates are not minima that indicate the middle of a *kashida*, but typically the following branch point.

3. Experiments using synthetic databases

The synthetic samples, which are created by the proposed approach, are meant to be used as training or testing data for different document analysis methods. To investigate whether these syntheses can be used instead of real samples, we created synthetic samples (png images + ground truth) of all words in our IESK-arDB database to build IESK-arDB-Syn. Then, we applied a segmentation technique on both the original and the synthesized database. We found that the detected error rates are comparable, as shown in Table 1. This proves at least that the proposed synthesis method is capable of reflecting those features of real Arabic words that are critical for segmentation. Furthermore, we investigated robustness of the segmentation method against the influence of particular features such as slant or skew. Modified versions of IESK-arDB-Syn were created for each experiment, where only the investigated feature differed for samples by the same writer (except the last experiment, which was done using the original IESK-arDB-Syn). We used cross-validation for all the following experiments.

Main experiment: Since ASMs for letters are currently only available from four writers (due to limited resources), multiple ways of writing a word were created for each writer to get sufficient samples. In order to achieve variations close to those of the 12 writers of the IESK-arDB, some optional features, such as slant or *kashida* length modification, have

Table 1: Experimental results.

	IESK arDB		IESKarDB-Syn	
Measure	μ	σ	μ	σ
Error per word	1.67	0.13	1.74	0.024
Error per letter	0.35	0.026	0.34	0.0045
Over-segmentation (per word)	0.79	0.097	0.86	0.0066
Under-segmentation (per word)	0.87	0.071	0.88	0.019
Perfect segmentation (per word)	0.17	0.0019	0.13	0.0067

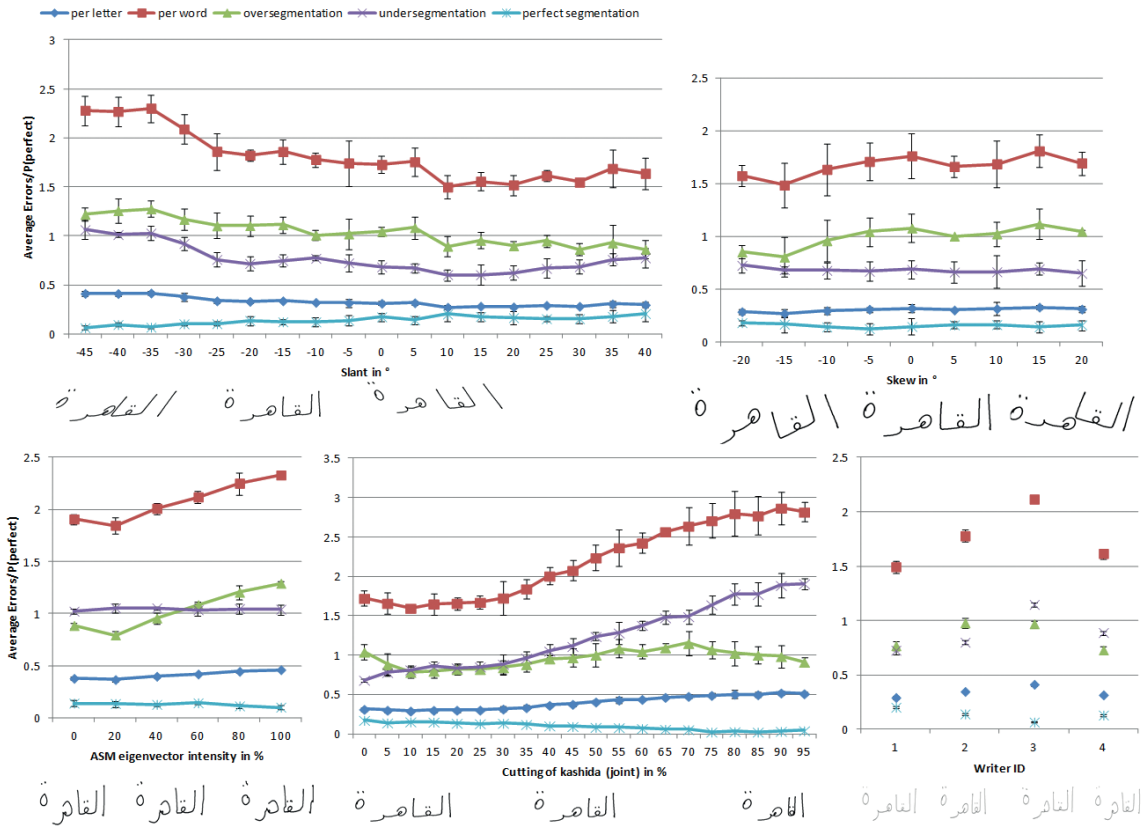


Fig. 10: Influence of different handwriting features on accuracy of the segmentation method.

been parameterized by the user interface. This is shown in fig. 9.

As the main experiment, we segment all samples of the IESK-arDB database (2540) and the IESK-arDB-Syn database (9000) into letters and compare the results. However, a manual validation of the segmentation results of thousands of words is barely feasible. Hence, the ground truths of both databases are used to perform automatic validation. Each point where two letters are connected is defined as a dividing point. We experimentally found a suitable range that defines the area around each dividing point in which exactly one automatic segmentation proposal is expected (shown in fig. 9 b). The validation method measures *under-segmentation* within a word for each area without any segmentation proposal and *over-segmentation* for each additional segmentation proposal and proposals outside the defined areas.

According to the automatic cross-validation, the segmentation method used causes an average of 1.7 errors per word, and the standard deviation σ is 0.13 when applied to the IESK-arDB. Nearly the same accuracy is measured for IESK-arDB-Syn, which shows that validations based on synthetic samples indeed enable one to predict how well a method may perform in real-world situations. This also enables automatic optimization of parameters and comparison of different methods. Nevertheless, the measured perfect segmentation rate is less for IESK-arDB-Syn and the *over-segmentation* is 7% higher. This could be caused by both well and badly written samples occurring in the IEKS-arDB, a property that is not yet mimicked sufficiently by our synthesis approach. Furthermore, the simulation of letter joints has to be improved for some cases to avoid accidental structures that are excessively hard to segment.

The results of the following experiments are shown in fig. 10.

Slant: Since the segmentation method only uses segments at rows with exactly one foreground pixel, extreme slants can cause segmentation errors due to overlapping ascenders. The experiment shows that even a slant of about 20° improves the segmentation results, especially if the *kashidas* between the letters are relatively long. Hence slant correction is a useful preprocessing step for our segmentation method.

Skew: Although skew correction is a common preprocessing step, this experiment shows that this segmentation method is robust against a skew of +/- 20°.

Kashida length: In Arabic handwriting, the length of *kashidas* is highly variable. The experiment shows that a valid segmentation is especially difficult for very small *kashidas*, since most structures that indicate a potential dividing point are hidden or have vanished in such cases. In contrast to slant and skew variation, this problem cannot be solved using a simple preprocessing technique.

ASM eigenvector intensity: The intensity of the used eigenvectors defines the similarity of a computed letter shape to the average shape, where maximal intensity often causes deformed shapes that are hard to classify. The experiment confirms that eigenvector intensity is also proportional to segmentation error. However, even strongly deformed letter shapes are not as critical as the reduction of *kashida* length.

Writer: As one can see, the segmentation method is more sensitive to the writer-dependent style than to the precision of the writer. The best performance was achieved for writer #1, whose letters are well defined and have long *kashidas*.

4. Conclusion

We have presented an efficient approach for generating pseudo-handwritten Arabic words or text lines from Unicode (including diacritical marks). Online Sample and Active Shape model-based glyphs from multiple writers as well as affine transformations allow the researcher to generate various images for a given Unicode string to cover the variability of human handwriting. Features such as the slant of a letter can be controlled manually if desired. To increase variability and realism, a rendering technique that simulates the physical attributes of various writing tools has been developed.

In order to validate how accurately the syntheses reflect features of real handwriting, we compared the results of a segmentation method applied to real and synthetic samples. In addition, the use of specifically manipulable features was investigated. Therefore, we experimentally detected the influence of slant, skew and writing style on segmentation accuracy. Due to the comprehensive ground truth, special preprocessing methods, such as detection and classification of diacritical marks, can be trained, optimized and validated. Such methods can be used to improve different approaches to word recognition, segmentation and spotting.

In our future work, we intend to reduce the amount of synthesized words by means of clustering techniques, such as affinity propagation, to avoid redundant samples. Furthermore, samples by additional writers must be acquired and additional character classes, such as digits or common ligatures, need to be included to improve the quality of syntheses. We will also extend our approach to allow the generation of whole pages of text. This will help synthesize compact but representative databases for training and testing of preprocessing, segmentation and recognition methods for document analysis tasks.

The handwriting synthesis and segmentation approaches proposed here are just modules that are part of a larger project. There are huge archives written in Arabic. We plan to build a system that can segment these documents into characters. Then feature extraction, a priori information as well as a vocabulary of valid Arabic words will be used to recognize the handwriting. The vocabulary should be dynamically defined by the user to check if a document contains keywords from various fields (religion, history, medicine, etc.) or names of specific people or regions. Using explicit segmentation, the recognition process can be separated into a static and a dynamic part that uses contextual information provided by the user. All image-processing methods are assigned to the static part, which only needs to be done once. The proposed synthesis approach will be used to validate this system.

ACKNOWLEDGMENTS

Research project funded by King Abdulaziz City for Science and Technology, (KACST). Project code: 13-INF604-10.

REFERENCES

Almuallim, H., and S. Yamaguchi (1987), 'A method of recognition of Arabic cursive handwriting.' *IEEE Trans. Pattern Anal. Mach. Intell.* (IEEE Computer Society), 9: 715–722.

Al-Zubi, Stephan (2004), 'Active Shape Structural Model', in *Tech. rep*. *Otto-von-Guericke University of Magdeburg*.

Cheung, Kwok-Wai (Student Member), Chin, Roland T., Dit-Yan Yeung, and Chin, T. (1998), 'A Bayesian Framework for Deformable Pattern Recognition With Application to Handwritten Character Recognition', *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 20: 1382–1388.

Dinges, Laslo, Al-Hamadi, Ayoub, and Elzobi, Moftah (2013), 'An Approach for Arabic Handwriting Synthesis based on Active Shape Models', in *12th International Conference on Document Analysis and Recognition (ICDAR),* 1260–1264.

Elarian, Y. S., Al-Muhsateb, H. A., and Ghouti, L. M., 'Arabic Handwriting Synthesis', *International Journal of Signal Processing, Image Processing and Pattern Recognition*, 4.4: 131–143.

Elzobi, Moftah, Ayoub Al-Hamadi, Zaher Al Aghbari, and Dinges, Laslo (2012), 'IESK-ArDB: a database for handwritten Arabic and optimized topological segmentation approach.' In *International Journal on Document Analysis and Recognition*, 16.3: 295–308.

Guyon, Isabelle (1996), 'Handwriting Synthesis From Handwritten Glyphs', *Proceedings of the Fifth International Workshop on Frontiers of Handwriting Recognition*, 309–312.

Miyao, Hidetoshi, and Maruyama, Minoru (2006), 'Virtual Example Synthesis Based on PCA for Off-Line Handwritten Character Recognition', in *Document Analysis Systems*, 96–105.

Pechwitz, Mario, Maddouri, Samia S., Märgner, Volker, Ellouze, Nouredine, and Amiri, Hamid (2002), 'IFN/ENIT – database of handwritten Arabic words', *Proc. of CIFED 2002*, 129–136.

Saabni, R. M., and El-Sana, J. A. (2012), 'Comprehensive synthetic Arabic database for on/off-line script recognition research', *International Journal on Document Analysis and Recognition,* 16:285–294.

Shi, Daming, Gunn, Steve R., and Damper, Robert I. (2003), 'Handwritten Chinese Radical Recognition Using Nonlinear Active Shape Models', *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 25: 277–280.

Thomas, Achint Oommen, Rusu, Amalia, and Govindaraju, Venu (2009), 'Synthetic handwritten CAPTCHAs', *Pattern Recognition* (Elsevier Science Inc.), 42: 3365–3373.

Varga, T., and Bunke, H. (2003), 'Generation of synthetic training data for an HMM-based handwriting recognition system', *Seventh International Conference on Document Analysis and Recognition, 2003. Proceedings,* vol. 1, 618–622.

Xiu, Pingping, Peng, Liangrui, Ding, Xiaoqing, and Wang, Hua (2006), 'Offline Handwritten Arabic Character Segmentation with Probabilistic Model', in *Document Analysis Systems VII 7th International Workshop*, ed. by H.orst Bunke and A. Lawrence Spitz (Springer), 402–412.