Real-time Segmentation of On-line Handwritten Arabic Script

George Kour
Faculty of Engineering
Tel-Aviv University
Tel-Aviv Jaffa, Israel
Email: georgeko@post.tau.ac.il

Raid Saabne
Department of Computer Science
Tel Aviv-Yaffo Academic Collage, Israel
Triangle R&D Center, Kafr Qara, Israel
Email: saabni@cs.bgu.ac.il

***Abstract*—Real-time performance is necessary in applications involving on-line handwriting recognition. However, conventional approaches usually wait until the entire curve is traced out before starting the analysis, inevitably causing delays in the recognition process. In regards to the Arabic script, the postponed analysis may be attributed to the cursive and unconstrained nature of the Arabic writing system, in both printed and handwritten forms. Nevertheless, this paper proposes a real-time recognition-based segmentation technique of on-line Arabic script. It demonstrate the feasibility of carrying out the most time consuming tasks, required for the segmentation process, during the course of writing. The system has been designed and tested using the ADAB Database, and promising results were obtained.**

***Keywords*-Arabic script segmentation; handwriting recognition; on-line text segmentation;**

I. Introduction

Handwriting remains the most commonly used mean of communication and recording of information in the daily life, therefore, a growing interest in the handwriting character recognition field has emerged in recent years. Handwriting recognition can be categorized into two main areas: off-line and on-line. In the off-line case, a digital image containing text is fed to the computer, and the system attempts to convert the spatial representation of the letters into digital symbols [1]. In contrast, the process of on-line handwriting recognition is done on a digital representation of the text written on a special digitizer, tablet or smart-phone device, where sensors pick up the pen-tip movements.

Research in this field has established two main approaches; the analytic approach, which involves segmentation and classification of each part of the text [2], [3], [4], and the holistic approach, which considers the global properties of the written text and recognizes the input word shape as a whole [5], [6]. While having many advantages, the holistic approach requires the classifier to be trained over the entire dictionary, which is impractical for large dictionaries (containing more than 20,000 words) [7].

The cursiveness of the Arabic script, prima facie, requires delaying the launch of the recognition process until the completion of the word scribing. However, in this paper, we question the necessity of this requirement by demonstrating the feasibility of approximating the position of the *segmentation points* (SPs) while the stroke is being written; and by doing so, accelerating the segmentation and, consequently, the recognition process. The resulted segmentation and letters classification information can be used to significantly reduce the potential dictionary size and accelerate a later holistic recognition process.

In Section II, we mention related studies. The proposed approach is described in detail in Section III. Section IV provides information on the ADAB database and details the segmentation validation method used in this work. Experimental results and analysis is given in Section V.

II. Related Work

The research on on-line handwriting recognition started in the 1960's and has been receiving a great interest from the 1980's [8]. One of the earliest studies on On-line Arabic script recognition was carried out by El-Wakil and Shoukry in 1989 [9].

Randa et al. [7] proposed a two stage on-line Arabic handwritten text segmentation system based on Hidden Markov Model (HMM). In the first stage, SPs were nominated, and then, in a second stage, the nominated points were validated using a rules-based engine. The system was tested using a self-collected database named OHASD.

Digness et al. [4] used a segmentation-based recognition approach based on dividing the word into smaller pieces, which were afterwards segmented into candidate letters, and then classified into letter classes, using statistical and structural features. A k nearest neighbor (k-NN) classifier was used to obtain the final recognition.

A segmentation-based recognition method that operates on the stroke level for on-line Arabic handwritten words recognition was proposed by Daifallah et al. [10]. SPs were nominated and then selected by locating semi-horizontal lines moving from right to left. A portion of the SPs is filtered out by applying a certain set of rules. Then, HMM is used to classify the sub-strokes to letters using the Hu feature. The letters candidates and their scoring were used to determine the best set of SPs.

A recent survey done by Tagougui et al. [8] reviews the status of research in the on-line Arabic handwriting

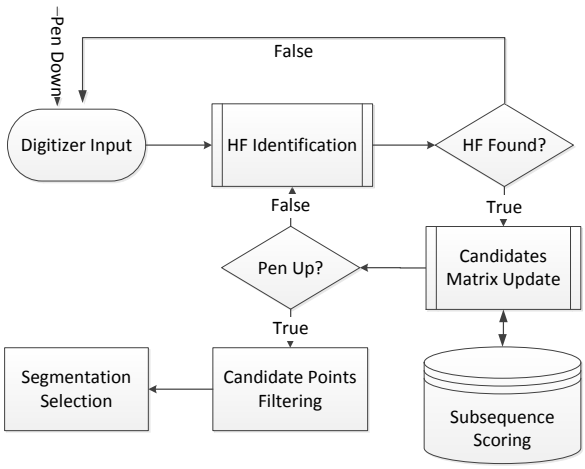

Figure 1: High level visualization of the system flow.

recognition field.

III. Our Approach

Complexity measure: this value indicates the curvature degree of a given 2-D trajectory. As we shall see, it will be found useful in several stages through the segmentation flow. In order to avoid this measure from being affected by small hand vibrations and data imperfection, the trajectory undergone a simplification process using the Douglas-Peucker algorithm [11]. The complexity measure is calculated by summing the parameters α_k computed for each inner point p_k in the simplified version, $T = \{p_i\}_{i=1}^{n}$, of the original trajectory. Namely, $CM(T) = \sum_{k=2}^{n-1} \alpha_k$, where $\alpha_k = C(1 - \frac{\phi_k}{\pi})$, $\phi_k = \angle(\overline{p_{k-1}p_k}, \overline{p_k p_{k+1}})$, and C is a constant which was experimentally set to 6.

The proposed approach goes through three stages. In the first stage, *points of interest* (POIs), are continuously nominated while the stroke is being scribed. The sub-strokes imposed by these POIs are scored by a letters classifier. In the second stage, once the entire stroke is available, a rules-based process is used to refine the set of POIs and re-score the sub-strokes. Eventually, the system heuristically determines the final set of SPs based on the sub-strokes scoring. The system flow is demonstrated in Figure 1.

A. First Stage: POIs Nomination and Sub-strokes Scoring

Horizontal fragment identification: in this stage, the system attempts to identify *horizontal fragments* (HFs) that join pairs of connected letters in the Arabic script. These handlers are usually horizontal, directed right to left, and located near the baseline. In this stage, a smoothed version of the trajectory is used in order to ignore undesired small horizontal regions frequently caused by the digitizer. The smoothed version is obtained using the splines interpolation method. A point p_i is defined as a "horizontal point" if the slope of the line $\overline{p_{i-1}p_i}$ is less than a preset value δ, which was empirically tuned to 0.6. The same exact value for this parameter was found independently in [10].

HFs are continuously identified using the following process. The first detected horizontal point is set as an "HF starting point". All subsequent horizontal points are ignored until a non-horizontal point is detected, indicating the end of an HF sequence. This point is identified as an "HF ending point" and the medial point of an HF is marked as POI. POIs are potential SPs, thus, this process yields an over-segmentation of the stroke. While false positive SPs can be easily removed, missed real SPs cannot be easily recovered. Therefore, in order to minimize the miss-rate, this process is delicate and allows false HFs to be detected.

In a post processing step, fractions of the same horizontal segment, which were identified as multiple HFs, are rejoined to a single HF. The merge decision is made by evaluating the complexity measure between the two consequent HFs.

Sub-strokes scoring: Let $S = \{p_i\}_{i=1}^{n}$ be a sequence representing a handwritten stroke in which L POIs were detected. Let $KP = \{KP_i\}_{i=0}^{L+1}$ (Key points) be the ordered set of POIs, in addition to the first and the last points of the stroke positioned as the first and the last items in the set as demonstrated in Figure 2. Formally, we define:

$$KP_i = \begin{cases} 1, & \text{if } i = 0 \\ POI_i, & \text{if } 1 \leq i \leq L \\ n, & \text{if } i = L+1 \end{cases} \tag{1}$$

A sub-stroke S_i^j is a sub-sequence of the stroke S that starts at KP_i and ends at KP_j, formally:

$$S_i^j = \{p_k\}_{k=KP_i}^{KP_j}; i < j \tag{2}$$

We generate an upper triangular scoring matrix $D \in \mathbb{R}^{(L+1)\times(L+1)}$ where each cell $D_{i,j}$ contains the information returned by the scoring system for the sub-strokes S_i^j. The matrix D is generated dynamically, adding a row and a column for each new detected POI. Given a band B, which impose a locality constraint above the main diagonal of the matrix D, by fixing $D_{i,j} = \infty$ if $j \leq i$ or $j - i > B$, improved the efficiency of the process and the segmentation accuracy. We argue that sub-strokes representing a letter will achieve, in most cases, better scoring, i.e., lower $D_{i,j}$ value than other sub-strokes.

The classifier contains four databases, one for each letter position (Ini, Mid, Fin and Iso). It receives a sequence of points representing the sub-stroke and a letter position, and outputs a list of sample candidates and their scoring which indicates the similarity measure between the sequence and the candidate. The real-time nature of the segmentation technique strictly requires the classifier to be extremely fast. The compliance with the high performance demand

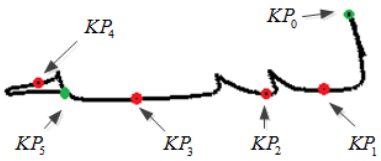

Figure 2: KPs of the word لبيه (Lbyh). POIs are colored in red. The first and last KPs are colored in green.

Table I: A mapping between the subsequence types and the possible letter positions. S denotes a stroke containing L POIs. S_i^j is defined in Equation 2, $m > 0$ and $k < L + 1$.

Notation	Subsequence	Letter Position
α	S_0^k	Ini or Mid
β	S_m^k	Mid
χ	S_m^{L+1}	Mid or Fin
δ	S_0^{L+1}	All

was made possible by using a state-of-the-art method for fast computation of the approximate k-NN, based on the technique proposed in [12].

The relative location of the subsequences in the stroke, was used to avoid accessing all the four databases. As detailed in Table I, we differentiate between four types of subsequences. For each type, we indicate the set of databases that need to be examined, i.e., the possible letter positions the sub-stroke may represent.

Determining the sub-stroke type is performed while the stroke is being written according to the mapping given in matrix D_p (Equation 3). The last column and row are added on the "pen up" event.

$$D_p = \begin{pmatrix} \infty & \alpha & \alpha & \alpha & \cdots & \alpha & \delta \\ \infty & \infty & \beta & \beta & \cdots & \beta & \chi \\ \infty & \infty & \infty & \beta & \cdots & \beta & \chi \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots \\ \infty & \infty & \infty & \infty & \cdots & \beta & \chi \\ \infty & \infty & \infty & \infty & \cdots & \infty & \chi \\ \infty & \infty & \infty & \infty & \cdots & \infty & \infty \end{pmatrix} \quad (3)$$

For a given input (sequence and position), the recognition system returns the K nearest neighbors with different labeling. In our implementation, we set $K = 3$.

B. Second Stage: Candidate points filtering and scoring correction

In this stage we re-score subsequences and eliminate redundant POIs based on the following rules:

1) Inner SPs should lie close to the baseline.
2) SPs do not reside in loops.
3) The sub-stroke length should be proportional to the length of the containing stroke.

	0	1	2	3	4	5
0	N/A				N/A	N/A
1	N/A	N/A				N/A
2	N/A	N/A	N/A			
3	N/A	N/A	N/A	N/A		
4	N/A	N/A	N/A	N/A	N/A	
5	N/A	N/A	N/A	N/A	N/A	N/A

Figure 3: A tabular representation of the D matrix that corresponds the KPs showed in Figure 2. Each cell contains the corresponding sub-stroke colored in red.

Baseline detection: The system determines the baseline by calculating the vertical density histogram of the re-sampled and normalized sub-stroke. The height of the stroke (y-axis) is partitioned into ten equi-length intervals. A POI is filtered out if it does not satisfy the following condition:

$$|POI_y - I_{max}| \leq 2 * max(|I|, 0.15) \quad (4)$$

where POI_y represents the y-coordinate of the POI, $|I|$ is the interval length, and I_{max} denotes the center of the most inhabited interval. In order to reliably determine the baseline, the algorithm is activated only after the fourth POI is detected. This process has proven to be very effective in eliminating challenging false POIs that reside in valleys that appear in several final Arabic letters, such as ق, س and ن. An example can be seen in the letter ـن in Figure 4. The baseline is needed only to verify that the POIs are in a reasonable distance from it, therefore, imprecise valuation of the position or the direction of the baseline is tolerable.

The third rule is used to penalize unreasonably low scoring given to small sub-strokes, which are unlikely to represent a letter. This is done by calculating the ratio of the sub-stroke length proportional to the entire stroke length. For instance, it is common to add a small, hook-like, extension to the suffix of the letter د that may look very similar to the letter ا during the stroke scribing; and thus under-scored by the recognition system, and eventually, result in over-segmenting the letter د. Despite the fact the we use a smoothed version of the stroke, in some cases, POIs are in-correctly nominated on non-horizontal areas. This is caused due to noises in the data and the fact that the nomination is done while the word is being scribed. Our filtering algorithm should be corrected, in a future work, to handle this case.

C. Third Stage: Segmentation Selection

The goal of this phase is to select the *final segmentation points* (FSP) set among the POIs. This is performed by finding the *segmentation path* in D with the best scoring possible. A segmentation path π is an ordered subset of the KPs having KP_0 and KP_{L+1} as the first and the last points in the path. Π denotes the scoring of the segmentation path π; it is calculated by summing up the sub-sequences scoring in the segmentation path divided by the path length; this is done in order to avoid favoring paths that represent under-segmentation.

One can model the scoring matrix D as a directed, edge-weighted graph $G = (V, E)$, for which a path from vertex KP_0 to vertex KP_{L+1} defines a possible segmentation. It can be experimentally validated that finding the shortest path in G (from KP_0 to KP_{L+1}) does not necessarily obtain the optimal segmentation, and in some cases, produces under-segmentation of the stroke. It is because the shortest path is a global property, which may prefer a highly weighted shortcut path over a path that consists of several low weighted fragments; in cases where the accumulative weight of the fragmented path is larger than the shortcut path. However, greedily selecting the outgoing edge with the minimal weight will mostly return a better segmentation.

Several *segmentation selection algorithms* (SSAs) for finding the best segmentation path are proposed in this work. The *forward segmentation selection* (FSS) algorithm starts with the first point, KP_0, advancing toward the end of the stroke. See pseudo-code in Algorithm 1. In each iteration, FSS tries to find the next best KP by selecting the adjacent subsequence S_i^j with the best scoring (as can be seen in line 5). Symmetrically, the *backward segmentation selection* (BSS) algorithm starts from the last point and advances toward the beginning of the stroke. The main drawback is the FSS tendency to under-segment the suffix of the stroke and the BSS tendency to under-segment the stroke's prefix.

In an attempt to overcome the aforementioned drawback, the *backward-forward segmentation selection* (BFSS) is proposed. As can be seen in Algorithm 2, it combines both FSS and BSS. It operates from the sides of the stroke toward the center, i.e., in each iteration it selects two POIs to include to the segmentation path.

The last algorithm operates differently and was given the name *greedy segmentation selection* (GSS). As described in Algorithm 3, in every iteration, the cell $D_{s,e}$ with the lowest (best) scoring is selected and since it represents the sub-stroke S_s^e, both KP_s and KP_e are added to the FSP. Every cell corresponding to a subpart of the sub-stroke S_s^e is removed by setting its corresponding scoring value in the matrix D to ∞ in order to avoid those subpart to be selected in a later iteration. This is done by invoking the function $UpdateMatrix(D, s, e)$. In Algorithm 3, the notation $[\infty]^{(L+1)\times(L+1)}$ indicates a matrix that all it's cells are set to ∞.

Performance evaluation of the mentioned SSA is provided in Section V-B.

1 $\pi = \{0\}$;
2 $i = 0$;
3 $sum = 0$;
4 **while** $i < L + 1$ **do**
5 $\quad j = \arg\min_k (D(i, k))$;
6 $\quad \pi = \pi \cup \{j\}$;
7 $\quad sum = sum + D(i, j)$;
8 $\quad i = j$;
9 **end**

Algorithm 1: Forward Segmentation Selection.

1 $\pi = \{0, L + 1\}$;
2 $kp_a = 0$;
3 $kp_b = L + 1$;
4 **while** $kp_a < kp_b$ **do**
5 $\quad kp_{a,next} = \arg\min_k(D(kp_a, k))$;
6 $\quad \pi = \pi \cup \{kp_{a,next}\}$;
7 $\quad kp_a = kp_{a,next}$;
8 $\quad kp_{b,next} = \arg\min_k(D(k, kp_{b,next}))$;
9 $\quad \pi = \pi \cup \{kp_{b,next}\}$;
10 $\quad kp_b = kp_{b,next}$;
11 **end**

Algorithm 2: Backward-Forward Segmentation Selection.

1 $\pi = \{0, L + 1\}$;
2 **while** $D \neq [\infty]^{(L+1)\times(L+1)}$ **do**
3 $\quad s, e = \arg\min(D)$;
4 $\quad \pi = \pi \cup \{s, e\}$;
5 $\quad sum = sum + D(s, e)$;
6 $\quad UpdateMatrix(D, s, e)$;
7 **end**

Algorithm 3: Greedy Segmentation Selection.

IV. Data Collection and Segmentation Validation

The ADAB database, described in [13], is a de-facto standard in the field of on-line Arabic handwriting recognition. It is freely available and consists of more than 20k Arabic handwritten words (937 Tunisian town/village names) scribed by more than 170 different writers.

Unfortunately, the ADAB only provides data on the strokes for a given city name. No segmentation into letters or *word parts* (WPs) is provided by the database, thus, extra work was needed to add this information in order to use its letter samples and as a ground truth for the segmentation

system. To provide this information, we employed the skills of a human expert to segment the strokes into letters, and to determine the words and the WP boundaries for each sample. This additional information was saved in an XML file. Delayed strokes were not included in the processed information since it is not considered by the segmentation process. We have manually segmented more than 7k samples which consisted about 20k strokes.

Related researches usually use a human expert to validate the accuracy of the SPs. However, in this work, we applied an automatic validation process using the ground truth information provided by the database. We discriminate between three types of final SPs. A final SP is classified as true positive if the complexity measure between the identified point and a true SP is less than a preset threshold; otherwise, it is classified as false positive. A false negative (miss), is the case when the system failed to identify a true SP. The validation process was tested on several sets and found to be highly reliable.

V. Experimental Results

The system was implemented using Matlab environment. Comparing the performance of the proposed approach to results obtained by related researches could be difficult due to the different experimental settings, databases and methodology; not to mention the different measures used to present the results. The usage of the ADAB database, instead of a self-collected database, standardize and reinforces our results. In Table II, we provide basic statistics of our sample set. Table III summarizes the system's performance.

A. Analysis

In this section we discuss common cases of incorrect segmentation.

1) Over-segmentation:

- Several Arabic letters contain a horizontal region in their initial form which does not accommodate a SP, see Figure 4. We overcame this problem by adding the following rule: a POI is nominated only if the sub-stroke that spans from the beginning of the stroke to the POI has a high complexity measure.
- Over-segmentation can also be caused by typing a letter in unusual form where it is spanned over several strokes. It happens mostly in the letter هـ, م and in rare cases in the letter ـحـ. This issue will be addressed in a future work.

2) Under-segmentation:

- Letter pairs that are not separated by HFs cause the system to miss POIs. This was partially solved by extending the notion of a letter to include such pairs. For example the pair لم and لح.
- In some cases, HFs were identified correctly but the corresponding POI were not selected in the third stage.

We have noticed that the absolute majority of the false negative (missed) SPs were actually identified as POIs in the first stage, but were not selected by the SSA. In addition, differentiating between the main body of the letter ـسـ and the main body of two consecutive ـبـ letters is possible only when considering the additional strokes, thus both cases were considered to be correct.

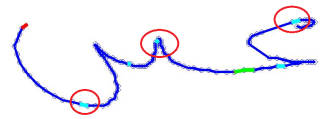

Figure 4: The main body of Arabic word عين (AIN). POIs are colored in cyan. The green areas indicate merge between two subsequent HFs. Three types of false POIs can be seen: 1. A POI at the beginning of a stroke. 2. A POI caused by a bad HF. 3. A POI inside a letter's valley.

Table II: General Statistics

Number of test samples (city name)	319
Number of WPs	1148
Number of Strokes	1237

Table III: Results

Strokes segmentation rate (SR)	83%
Strokes recognition rate (RR)	78%
Total number of true SPs	1081
Valid SPs (True Positive)	85.3%
Missing SPs (False Negative)	14.7%
Invalid SPs (False Positive)	119 (11%)
SPs Precision	88.6%
SPs Recall	85.3%

B. Segmentation Selection Algorithms Performance

It is apparent from the data in Table IV, that the SSA has a crucial effect on the system's performance. We tested several combinations of two SSAs in which the FSP is found by executing both SSAs independently and selecting the segmentation path with the smallest scoring.

In Table IV, a combination of two algorithms is denoted by $\oplus$.

Table IV: SSAs Performance

SSA	WP SR	WP RR	SP Precision	SP Recall
FSS	76%	70%	85%	78%
BSS	79%	73%	84%	81%
BFSS	78%	72%	84%	80%
GSS	80%	74%	81%	**94%**
FSS$\oplus$BSS	**82%**	**76%**	**89%**	82%
GSS$\oplus$BFSS	81%	75%	83%	90%

Figure 5: The impact of increasing the maximal number of samples per class on the segmentation and recognition rates.

C. Sample set size and distribution

The letters is our training set are extracted from a database with a limited words diversity, thus, the distribution of the samples between the different classes is imbalanced. On one hand, it can be regarded as an advantage; since, the training set distribution reflects the a-priory probability of a letter appearance in the test set. On the other hand, a highly imbalanced training set is known to negatively affect many classification algorithms. In the following experiment, we measure the effect of a large and imbalanced training set on the WP segmentation and recognition rates. It is done by gradually increasing the maximal allowed number of samples per class (letter and position).

The graph in Figure 5 shows convergence of the system's performance when the maximal number of samples is larger than 200 per class. Nevertheless, a miniature degradation is apparent, which is caused, probably, due to the increasing imbalance in the distribution of the training set. In addition, it is evident that the recognition rate (RR) is more sensitive to small training set than the segmentation rate (SR).

VI. Summary and Future Work

In this paper, we proposed a novel real-time approach for segmenting open-dictionary, Arabic handwritten script. The segmentation is performed in the strokes level. Morphological features are employed to nominate potential SPs. The sub-strokes induced by the SPs are classified using a k-NN letters classifier, and the classification information is saved in a scoring matrix. The final segmentation points are then selected by finding the best-scored segmentation path in the scoring matrix. This is done using several proposed algorithms. The system has demonstrated promising results.

In future work, we will expand the process by adding a later holistic WP recognizer that exploits the proposed segmentation and the scoring information to inspect a limited number of potential WPs.

Acknowledgment

The authors would like to thank Professor Dana Ron, from the Tel-Aviv University, for her invaluable help in this research and her insightful comments on this paper. This work has been supported in part by the German Research Foundation under grant no. FI 1494/3-2.

References

[1] M. Al-Ammar, R. Al-Majed, and H. Aboalsamh, "Online handwriting recognition for the arabic letter set," *Recent Researches in Communications and IT*, 2011.

[2] S. Abdulla, A. Al-Nassiri, and R. A. Salam, "Off-line arabic handwritten word segmentation using rotational invariant segments features." *Int. Arab J. Inf. Technol.*, vol. 5, no. 2, pp. 200–208, 2008.

[3] T. Sari, L. Souici, and M. Sellami, "Off-line handwritten arabic character segmentation algorithm: Acsa," in *Frontiers in Handwriting Recognition, 2002. Proceedings. Eighth International Workshop on*. IEEE, 2002, pp. 452–457.

[4] L. Dinges, A. Al-Hamadi, and M. Elzobi, "Offline Automatic Segmentation based Recognition of Handwritten Arabic Words," *sersc.org*, pp. 131–144, 2011. [Online]. Available: http://www.sersc.org/journals/IJSIP/vol4_no4/11.pdf

[5] F. Biadsy, R. Saabni, and J. El-Sana, "Segmentation-free online arabic handwriting recognition," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 25, no. 07, pp. 1009–1033, 2011.

[6] R. Saabni and J. El-Sana, "Hierarchical on-line arabic handwriting recognition," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*. IEEE, 2009, pp. 867–871.

[7] R. I. Elanwar, M. Rashwan, and S. Mashali, "Unconstrained arabic online handwritten words segmentation using new hmm state design," in *Proceedings of World Academy of Science, Engineering and Technology*, no. 64. World Academy of Science, Engineering and Technology, 2012.

[8] N. Tagougui, M. Kherallah, and A. M. Alimi, "Online arabic handwriting recognition: a survey," *International Journal on Document Analysis and Recognition (IJDAR)*, vol. 16, no. 3, pp. 209–226, 2013.

[9] M. S. El-Wakil and A. A. Shoukry, "On-line recognition of handwritten isolated arabic characters," *Pattern Recognition*, vol. 22, no. 2, pp. 97–105, 1989.

[10] K. Daifallah, N. Zarka, and H. Jamous, "Recognition-based segmentation algorithm for on-line arabic handwriting," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*. IEEE, 2009, pp. 886–890.

[11] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

[12] R. Saabni, "Efficient word image retrieval using earth movers distance embedded to wavelets coefficients domain," in *Document Analysis and Recognition (ICDAR), 2013 12th International Conference on*. IEEE, 2013, pp. 314–318.

[13] H. El Abed, V. Margner, M. Kherallah, and A. M. Alimi, "Icdar 2009 online arabic handwriting recognition competition," in *Document Analysis and Recognition, 2009. ICDAR'09. 10th International Conference on*. IEEE, 2009, pp. 1388–1392.