Pattern Anal Applic (2017) 20:1005–1027
DOI 10.1007/s10044-016-0543-1

THEORETICAL ADVANCES

A proposal for touching component segmentation in Arabic manuscripts

Nabil Aouadi[1] · **Afef Kacem**[1]

Received: 8 October 2014 / Accepted: 25 February 2016 / Published online: 15 March 2016

Abstract Text-line segmentation is one of the key factors which affect the performance of handwriting recognition system. Therefore, to make recognition systems more effective and accurate, segmentation of touching text-lines is an important task. One of the problems making this task crucial is the presence of touching components (TCs) representing connections between word letters of consecutive text-lines or those of words of the same text-line. The proposed method aims to segment TCs. It is mainly based on two steps: (1) finding for a localized TC a similar model, stored in a dictionary with its correct segmentation, using shape context descriptor and an interpolation function: the thin plate spline transformation, (2) segmenting the TC based on central point of the found similar model parts. TCs are assumed to be already extracted from Arabic manuscript images. Experiments are carried on a common TC database, using two metrics: Manhattan and Euclidean distances. Obtained results outperform the state of the art, considering the different types, variability and complexity of the TCs data set, and show the effectiveness of the proposed TC segmentation method.

Keywords Arabic manuscript segmentation · Touching components · Shape context descriptor · Thin plate spline transformation · Midpoint

✉ Afef Kacem
afef.kacem@esstt.rnu

Nabil Aouadi
nabil.aouadi@utic.rnu.tn

[1] Present Address: ENSIT, 5, Avenue Taha Hussein, Bab Mnara, BP 56, 1008 Tunis , Tunisia

1 Introduction

The segmentation of manuscripts into text-lines and words is an important process for several document analysis tasks, such as word spotting and text alignment. However, the presence of touching components (TCs), where neighboring letters touch each other (see Fig. 1), is one of the problems making hard manuscript segmentation. These TCs are generally due to presence of noise in images, writing style, low ink quality and aging, narrow spacing interlines (see Fig. 2b) and big jambs in the calligraphy (see Fig. 2c). This problem is common in unconstrained Arabic manuscripts since most of its letters (21 from 28) are ascendant or descendant and contain special marks and dots (see Fig. 2a). When the stated difficulties of handwritten documents are present: touching lines, curvilinear text-lines and horizontally overlapping components, the performance decreases and the accurate segmentation is very difficult. Finer analysis processes must be performed, especially in touch parts. Some segmentation methods use connected components to group the touching parts to the closest text-lines, while other methods are more accurate and analyze touching parts, considering their properties and shapes.

Touching components generally occur between consecutive text-lines (vertically TCs) or between words of the same text-line or letters of the same word (horizontally TCs). According to the number of connected letters and connection nature, TCs can be classified into three categories: (1) simple touching (see Fig. 2d), (2) multi-touching where the connection is between more than two letters (see Fig. 2e) and (3) cross-touching (between text-lines or words, see Fig 2f, g).

As noted by Kang and Doermann [1], there are two main approaches to segment touching components: recognition-

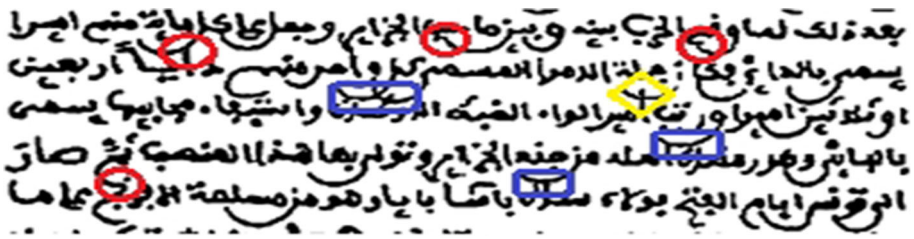

Fig. 1 Example of Arabic manuscript having touching components

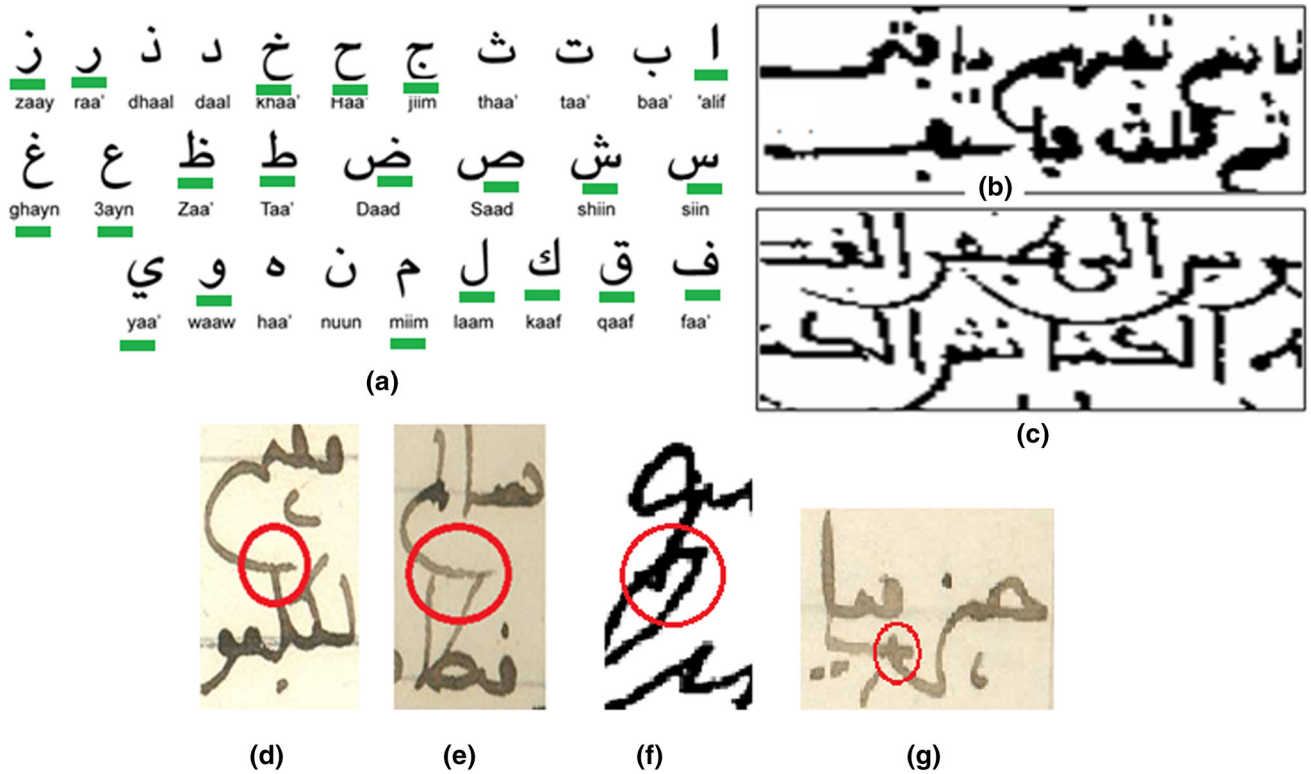

Fig. 2 Samples of TCs in Arabic and Latin manuscripts: **a** Arabic letters (ascenders and descenders are *underlined*), **b** text with narrow inter-lines spacing, **c** text with big jambs, **d** simple touching component, **e** multi-touching component, **f** cross-touching component between two text-lines, **g** cross-touching component between two words

free and recognition-based. Recognition-free segmentation methods use contour, skeleton, projection profile analysis and structural information of the TCs. Thus, a TC can be divided into segments by rules without recognition. These methods may efficiently address some touching problems but may not be robust enough to deal with a lot of variation or uncertainty. Recognition-based segmentation methods generate multiple candidate segmentation hypotheses and choose the optimal one based on recognition or other evaluation functions. But, these methods succeed only when the correct segmentation is in one of the candidates. In this work, we propose a recognition-based segmentation method for solving the problems of vertically and horizontally TCs, in unconstrained Arabic handwritten text, which succeeds to correctly segment TCs, even if the correct segmentation is not among the candidates. We assume that TCs are already and perfectly localized in manuscript images (knowing that many works such as [2, 3] are able to extract these components). The main idea of our proposed method is to separate the TC by approximation to models stored in a dictionary (see Fig. 3) with their known correct segmentation. Thus, there are two fundamental steps before TC can be segmented: (1) the recognition step to find the most similar model stored in a dictionary and (2) the approximation step to estimate the transformation aligning the selected model to the TC. Finally, we use the midpoints of the model parts to segment the TC.

The rest of the paper is organized as follows. Section 2 describes various approaches and methods, undertaken by researchers during the last decades, and compares them in tabular format. Section 3 presents the proposed method

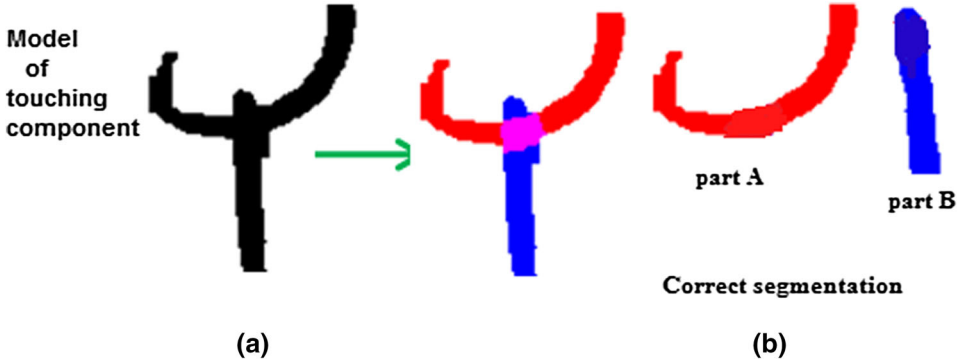

Fig. 3 An entry of dictionary: **a** TC's model, **b** model correct segmentation

where a first subsection handles with the recognition and the transform steps to find the most similar segmented model and the second subsection is dedicated to TC segmentation using midpoints. Experimental results are reported in Sect. 4 and some conclusions are drawn in Sect. 5.

2 TC localization and segmentation: state of the art

Among recognition-free segmentation methods, we can quote that of Likforman et al. [4]. In their work, TC is localized when a conflict occurred between two alignments since a TC would be assigned to only one. The TC segmentation method is recognition-free. It is based on the component projection profile analysis.

As shown in Fig. 4, if the projection profile contained two peaks, a horizontal cut would be done middle way from them, else the component would be cut into two equal parts. But, we think that by this way, separated letters of the TC risk to be broken and do not conserve their morphology.

In Vassilis's work [5], the document was divided into non-overlapping equi-width vertical narrow zones. A TC has been assigned to a text-line if its intersection with the area, defined by the boundaries of the text-line, exceeds a certain threshold R (fixed to 75 % of TC's height). Assuming a TC in the mth vertical zone ran into text-lines j and $(j+1)$, the method extended the mth zone horizontally on either side up to the point where the new area included a reasonable number of foreground pixels that have been assigned either to line j or to line $(j+1)$. In the extend area, N_j^b and N_{j+1}^b are the number of foreground pixels assigned to text-lines j and $(j+1)$, respectively. Similarly, considering N_j^a and N_{j+1}^a as the number of foreground pixels assigned to the text-lines j and $(j+1)$ that lie at the same horizontal line with the pixels of the TC under consideration. The ratio $r_k = \frac{N_k^a}{N_k^b}$ (for $k = j, j+1$) has been considered as a measurement of attraction of the TC to either text-line. Many cases have been considered:

- If both ratios are below a threshold (0.4), the TC will be assigned to the text-line with the greatest overlap. It is the case of a stress mark or a segment of a broken letter.
- If only one ratio is over the threshold, the TC will be assigned to the text-line with the highest ratio. It is the case of TC that involves ascenders and/or descenders.
- If both ratios are over the threshold, the TC under consideration will run along two text-lines (see Fig. 5a, d) and will need to be split. The method extracted the TC skeleton, found the junction points that lie near the separator (the middle line) between the jth and $(j+1)$th text-lines by excluding the junction points whose distance from the separator is greater than a threshold. Starting from the nearest junction point to the separator, the method converted this point and its neighbors into black-ground pixels. Hence, the skeleton was segmented into two or more parts. For each part the method estimated the ratios r_k as mentioned above. If all parts could be assigned to text-lines by applying the

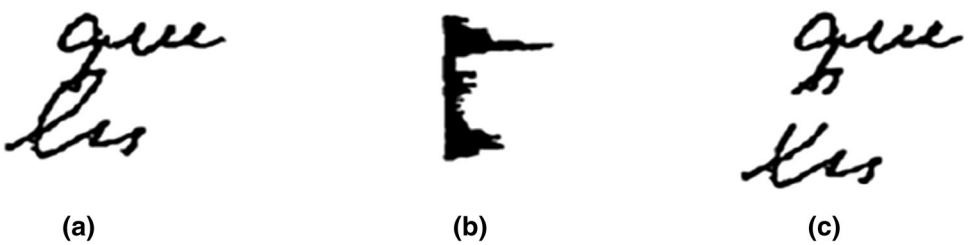

Fig. 4 **a** Touching component, **b** TC horizontal projection profile, **c** separation into two parts [4]

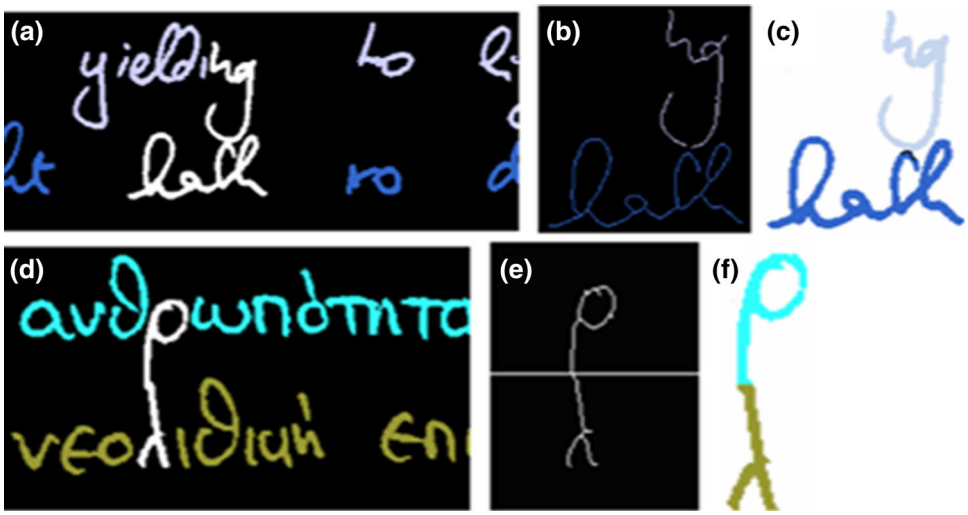

Fig. 5 Segmentation of TCs running along two text-lines: *a* extension of the zone containing the TC, **b** TC's skeleton is segmented into three parts, **c** assignment of segments of the TC to text-lines, **d** extension of the zone containing the TC, **e** TC's skeleton and the corresponding text-line separator. No junction points can be found, **f** TC is separated at the intersection point with the separator [5]

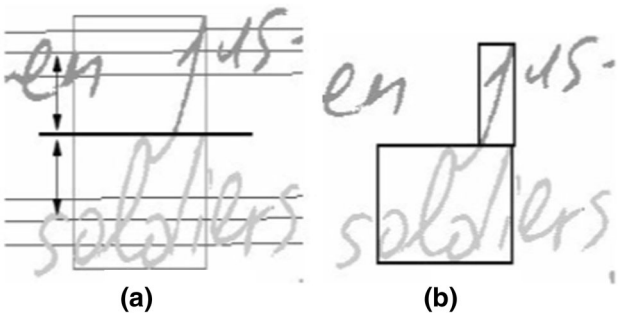

Fig. 6 TC segmentation: **a** touching component, **b** segmentation result [6]

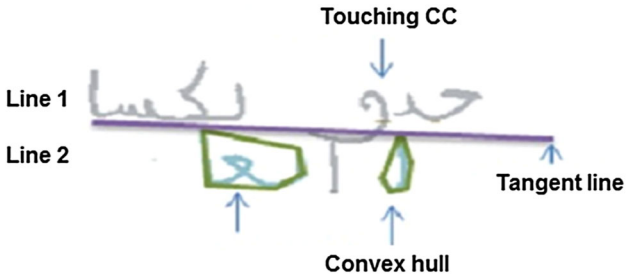

Fig. 7 Touching component segmentation [3]

foreground mentioned constraint, this point will be considered as a separator and each pixel of the TC will be assigned to the text-line of the nearest skeleton point (see Fig. 5a–c). Otherwise, this process will be repeated with the next junction point. If no suitable junction point can be found, the TC will be split at the point of intersection with the separator (see Fig. 5d–f). We think that this segmentation method greatly depends on thresholds and in case no suitable junction point is found, the TC might be incorrectly segmented.

In [6], Lemaitre et al. studied the position of TCs relative to text-line baseline. They used a global perception of baselines to define the point where the TC should be segmented. This point corresponds to the middle between the upper and the lower baselines (see Fig. 6). Yet as it can be seen in Fig. 6b, the TC is incorrectly segmented as the letter *j* has been broken and may not be fully restored.

In [3], to localize TCs, Kumar et al. found the common upper and lower tangent of the convex hull of consecutive components in neighboring text-lines. As mentioned by the authors, the upper and lower tangent to convex-hull of the components gave a good approximation of the text-line's extent locally in that region. A component was considered as a TC if the ratio of its length, below or above this tangent, to its total height exceeded a threshold. The TC segmentation was performed by a cutting at the nearest junction point on the tangent line to its centroid (see Fig. 7). But, this method might have problem when two letters completely overlap with each other since some parts of the ascendant (respectively descendant) letter will be higher (respectively lower) than the tangent line.

In [7], Louloudis et al. considered that a connected component exceeding the average height of all ones should be segmented. For TC segmentation, authors computed line y_i values of the intersection of the detected line i and the TC's bounding box $(i = 1...n)$ (see Fig. 8A(a), B(a)). For each line $i, i = 1...n-1$, zones Z_i assumed containing TCs were identified (see Fig. 8A(b), B(b)). The skeleton parts of every zone Z_i that intersected with line i was flagged with color_1, while all other parts were flagged with color_2. Finally, the TC segmentation was performed by assigning to each pixel the color of the closest skeleton pixel (see Fig. 8A(c), B(c)).

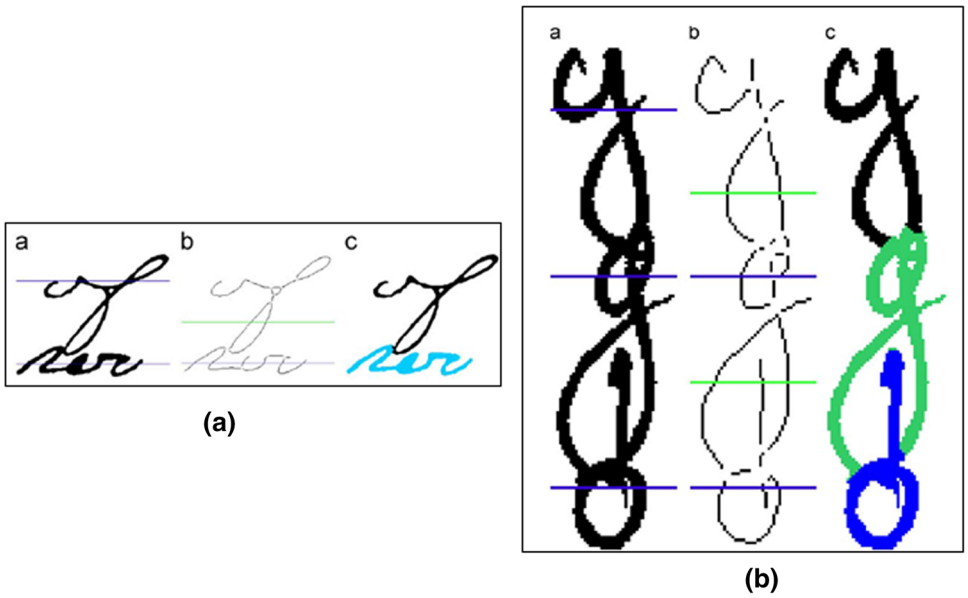

Fig. 8 Segmentation method steps: figure *A*: sample of a TC segmented into two parts, **a** line y_i (in *blue*), **b** zone Z_i (begins from *green line until blue line*), **c** final segmentation ; figure *B*: sample of a TC segmented in three parts, **a** lines y_i (in *blue*), **b** zone Z_i (begins from *green line until blue line*), **c** final segmentation [7]

Takru et al. [8] thought that if the TC's histogram projection profile contains a valley between two relatively large peaks (see Fig 9a), it should be separated into two parts following the valley horizontally. An analysis area (see Fig 9b, c) is determined according to the distance between the cut-line and the two components' gravity centers. Pixel assignment, to the upper or lower part, is performed depending on the writing style. Tests were performed on 20 images of Latin handwritten documents including several forms of TCs (see some examples in Fig 9d) where an accuracy about 77 % was achieved.

To segment a TC, the following operations were performed by [9]:

- Thinning was applied on the extracted TC to capture the structural knowledge (see Fig. 10a)
- The run-length algorithm was applied on the thinned image, using transition from foreground pixel to background pixel and vice versa (see Fig. 10b). Thus, one of the components is extracted (see Fig. 10c) and the second component is deduced by subtraction (see Fig. 10d).

Ouwayed and Belaïd in [2] detected a TC when it belongs to two adjacent text-lines. To disconnect this latter into two parts, the TC's skeleton is first extracted and a junction point S_p, linking the two letters, is identified. Authors looked for the starting point of the ligature B_p which is the highest point near the baseline of the top line. From this point, the descending letter is followed from the left or from the right as shown in Fig. 12. The tracking continues beyond the intersection point, respecting an angular variation corresponding to the curvature of the descending character (see an example in Fig. 11). Due to the symmetry of the curve branches, the value of the orientation angle must always be positive. Figure 13 shows the details of the separation process of a cross-touching component. Experiments were performed on 256 TCs between text-lines. 94 % of the TCs were correctly segmented. Note that only four types of vertically TCs, whose segmentation can be automatically solved, were treated. For the segmentation of others TC types, text comprehension is required as notified by [10].

Fernandez-Mota et al. in [11] used valleys in background image's skeleton to select the best paths for text-line segmentation. Paths were transformed into graphs whose particular pixels (endpoints or intersection) are vertices and ordinary pixels constituted edges connecting these vertices. TCs were identified when a discontinuity was observed in the skeleton background image. TC segmentation was performed by constructing virtual edges connecting two end points.

In [12], Kang et al. proposed a contour-based shape decomposition approach that provides local segmentation of touching letters. The shape contour is linearized into edgelets (see Fig. 14a using Ramer–Douglas–Peucker algorithm), which is merged into boundary fragments (see Fig. 14b). The connection cost between boundary fragments is obtained by considering local smoothness, connection length and a stroke-level property, called the Same Stroke Rate. The global configuration returns polygons that reflect how the shape contour is divided (see Fig. 14c). Samples of connections among boundary fragments are randomly generated and the one with the minimum global cost is selected to produce the final segmentation of the shape. To obtain a bipartite segmentation, an iterative search of the parameters is performed to have two

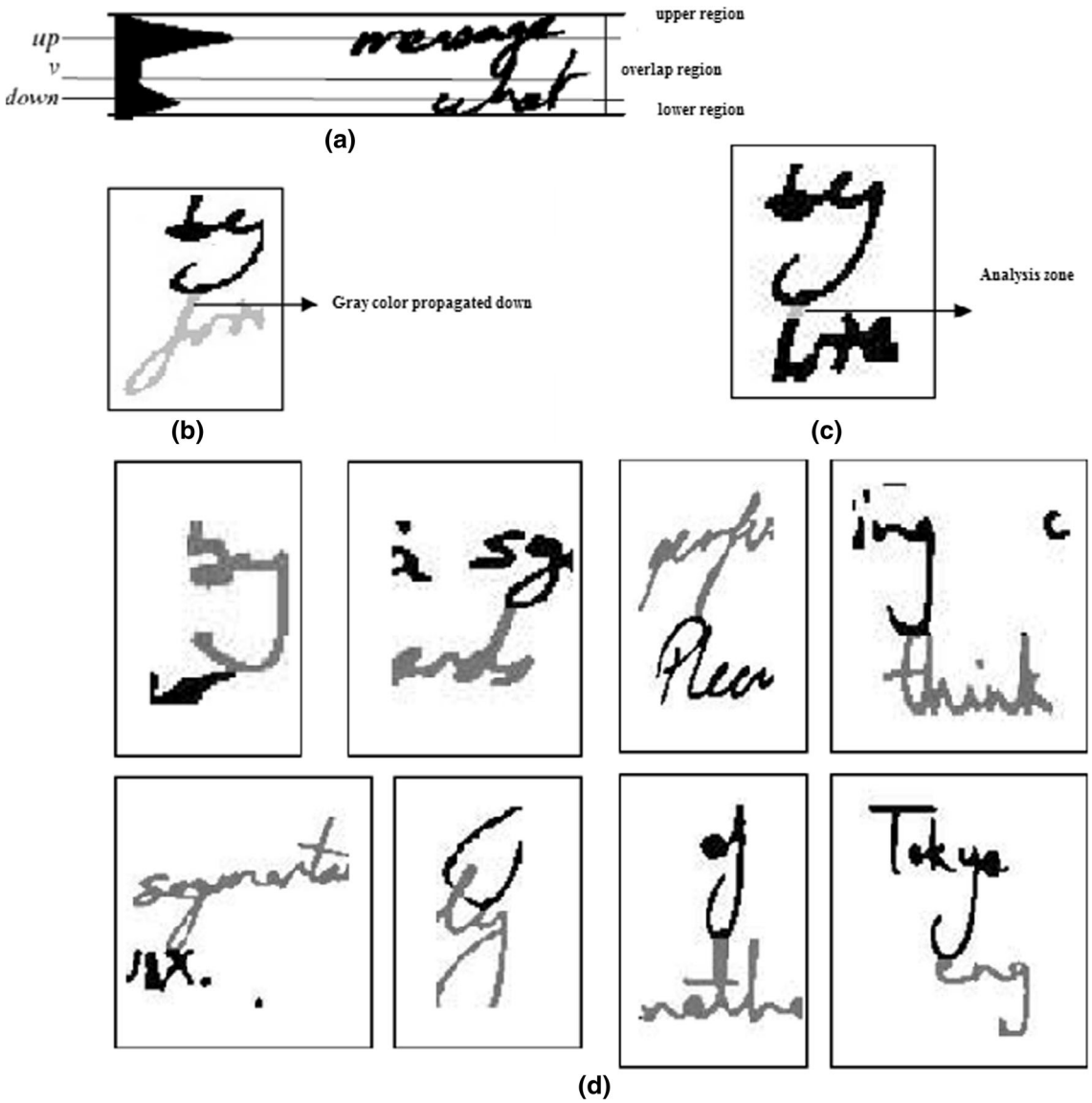

Fig. 9 **a** TC detection based on horizontal projection profile analysis, **b** TC, **c** analysis zone, **d** segmentation results of some Latin TCs [8]

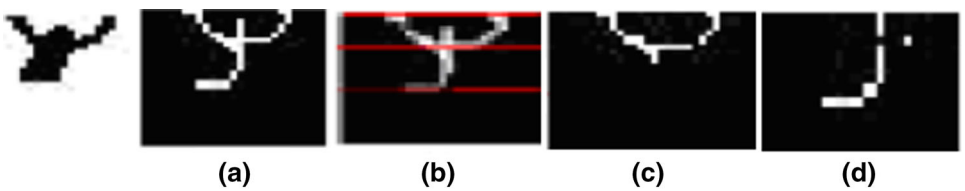

Fig. 10 **a** TC, **b** calculating run-length, **c** and **d** segmentation results [9]

components on a shape (see Fig. 14d). The segmentation accuracy is about 71.2 %.

In Boukerma et al. work [13], TCs are localized by identifying a junction point below the baseline in text-lines' skeleton images. To segment them, a vertical, a diagonal or a horizontal cut, according to the frequent direction, along with image contour, smoothing should be applied (see Fig. 15a, b).

In accordance with Alaei et al. [14], TCs are localized when the separating line (the line which separates two successive text-lines) passes through a black pixel and TCs are extracted based on their height. TCs are then segmented in the thinnest width region in Z_1 as shown in Fig. 15c–e. The method was tested on 20 samples of Latin handwritten text-pages (Handwriting Segmentation Competition, of ICDAR 2007) where the number of TCs was 68. Out of this, 56 TCs (82.4 %) were

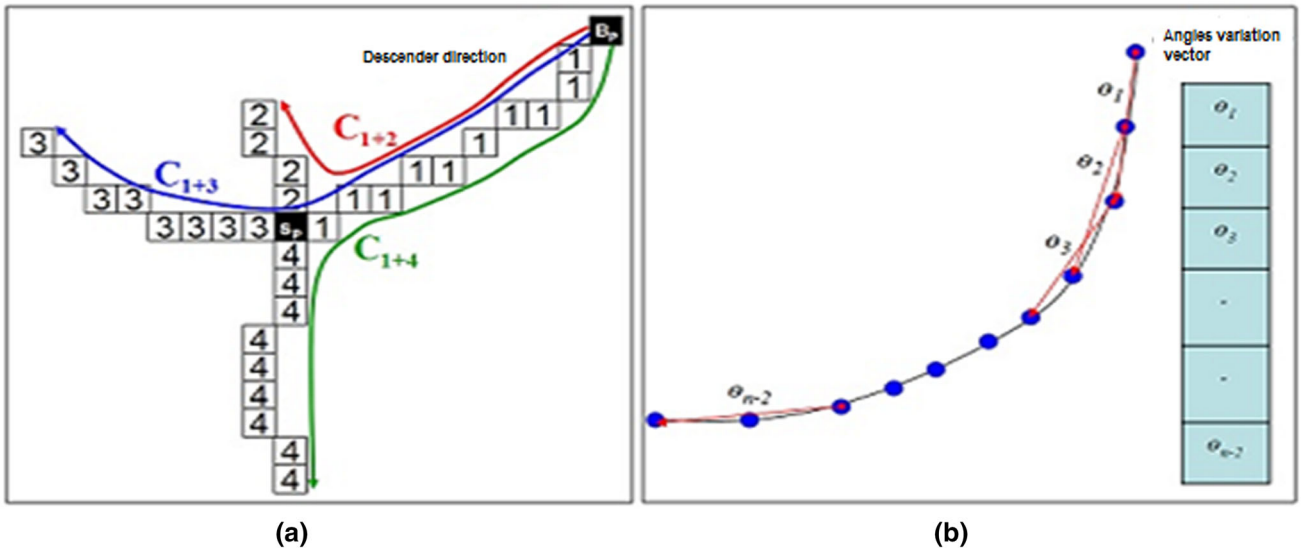

Fig. 11 **a** Example of an Arabic cross-touching component TC ("ر " crosses "ل"), B_p is the highest point, **b** angles variation vector [2]

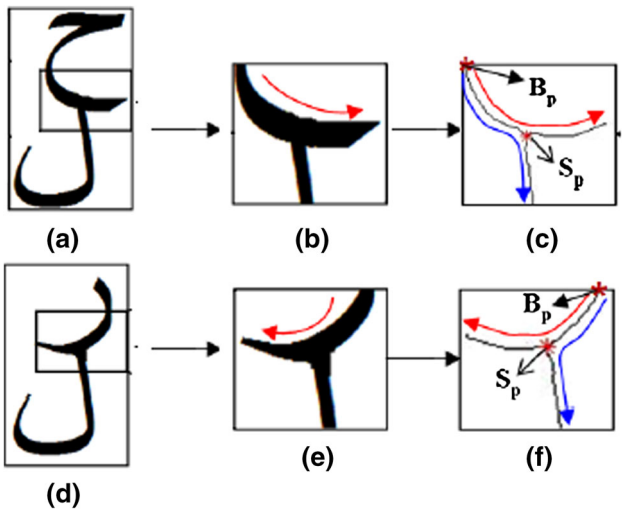

Fig. 12 TCs and descenders direction: **a** and **d** TC localization, **b** and **e** TC extraction, **c** descender direction from *left* to *right*, **f** descender direction from *right* to *left* [2]

correctly segmented. The same method was also tested on Farsi handwritten documents [15], as shown in Fig. 15f–h. But, sometimes a component may be incorrectly considered as a TC, because of strokes' stretching and characters' size variation. Misclassification mostly occurs when localizing TCs in Persian documents. We believe that segmenting a TC on region with minimum stroke's width could not be usually suitable, especially for character recognition, and in case of cross-touching or multi-touching connection.

Among recognition-based segmentation methods, we can mainly report the Kang and Doermann's method where a dictionary is created, consisting of TC models together with their correct segmentation [1]. To select the most similar model, stored in dictionary, for an input TC to be segmented, authors used Belongie's matching method [16]. This method is based on the shape context descriptor, to solve correspondence problem between TCs and models and the Thin Plate Spline (TPS) function [17] to estimate the transformation that best aligns them. Using affinity propagation [18], models were

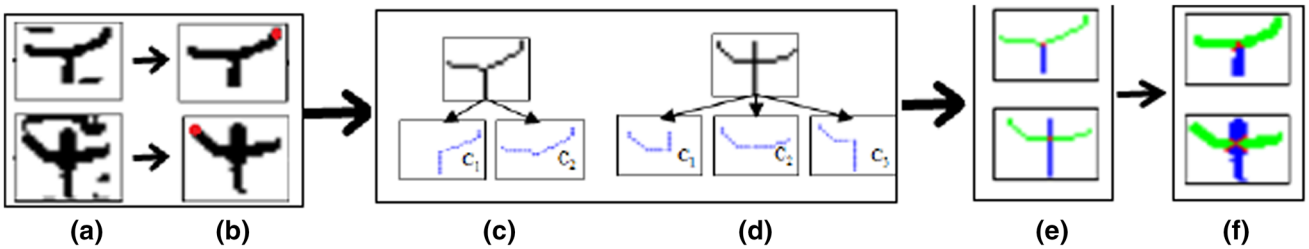

Fig. 13 **a** Two TCs with spurious little components, **b** TCs after eliminating spurious components, (B_p in *red*), **c** higher TC divided into two parts according to S_p: the minimum angular variance corresponds to the *curve* C_2, **d** lower TC divided into three parts according to S_p: the minimum angular variance corresponds to the *curve* C_2, **e** and **f** final segmentation [2]

organized into clusters, each one with its representative. Authors looked for the cluster's representative having the highest similarity with the TC, and then, search in the corresponding cluster to find the most similar template. Once found, the TC will be segmented according to the two model parts. First, its two contour's parts will be deformed with the same TPS parameters (used to find the best matching between TC and model), then overlaid on the input TC. Finally, TC's pixels will be split in two segments as they are closer to the first or second part of the model. Segmentation process is detailed in Fig. 16. Experiments were carried on a training set and a testing set, each having 744 samples. The rate of correct segmentation is about 71.4 %. As underlined by Kang and Doermann, four reasons were related to bad matchscore (matchscore <0.80): (1) when there is no closest model for an input TC, (2) when recognition step fails to choose the best one, (3) when input TC includes disturbing components (noise) and (4) when the TPS transform cannot mark corresponding components very accurately. Our work is inspired from the Kang and Doerman's method and it is an attempt to address these problems, as it will be explained later.

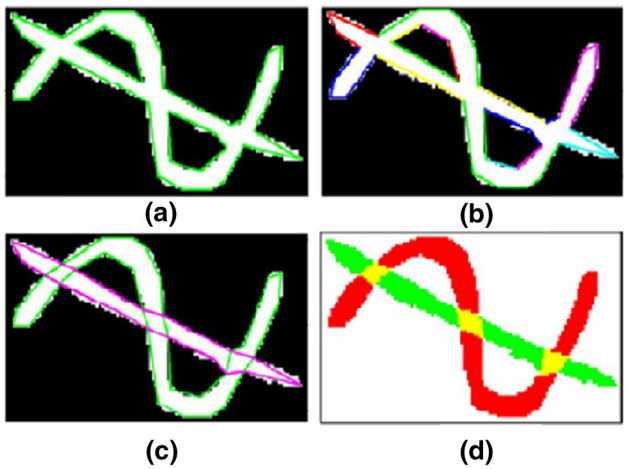

Fig. 14 **a** Touching component edgelets, **b** boundary fragments in *different colors*, **c** obtained polygons, **d** segmentation results (*yellow* for cross-touching) [12]

Piquin et al. in [19] localized TC when it escapes the text-line edge bounding box. The Binary TC's skeleton was symbolically represented by a decision tree. When a corresponding graph is found in a dictionary, built with the most representative cases, the TC will be segmented according to that graph. Else, TC segmentation will be done by analyzing the skeleton's arc tangent value which determines whether an arc is in the extension of another. Figure 17a–c shows the initial TC, the TC skeleton and the decision tree which indicates that the cut have to be done in S_4 vertex. 20 cases of cross-touching components have been treated but only 18 were correctly segmented.

Under the same approach, Ikeda et al. [20] localized TCs according to their sizes and ratios. To search the cutoff point, the TC is divided into columns. The resulting area of merging from left to right columns is studied. If the analysis of the combination of two characters generated has no sense, method will look for other break points on a graph segmentation hypothesis, produced in advance. From 152

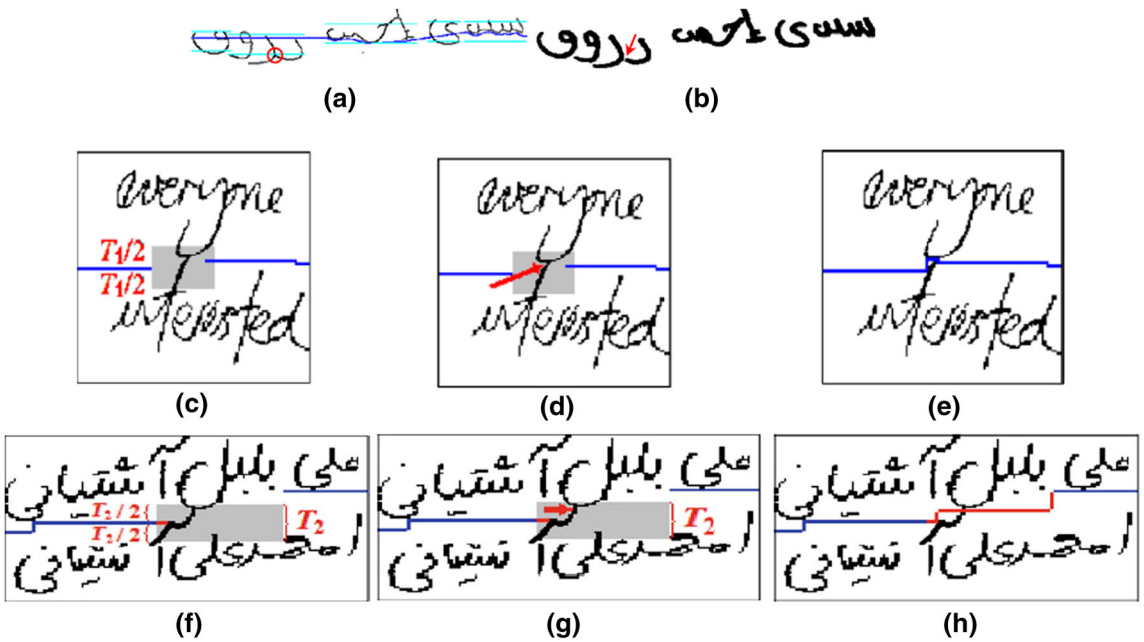

Fig. 15 Samples of TCs segmented in the low-density regions for Arabic TCs [13], for Latin TCs [14] and for Persian [15]: **a** detection of junction point in skeleton image, **b** result of segmentation, **c** and **f** location of connection zone (in *gray*), **d** and **g** minimum width position, **e** and **h** TC segmentation

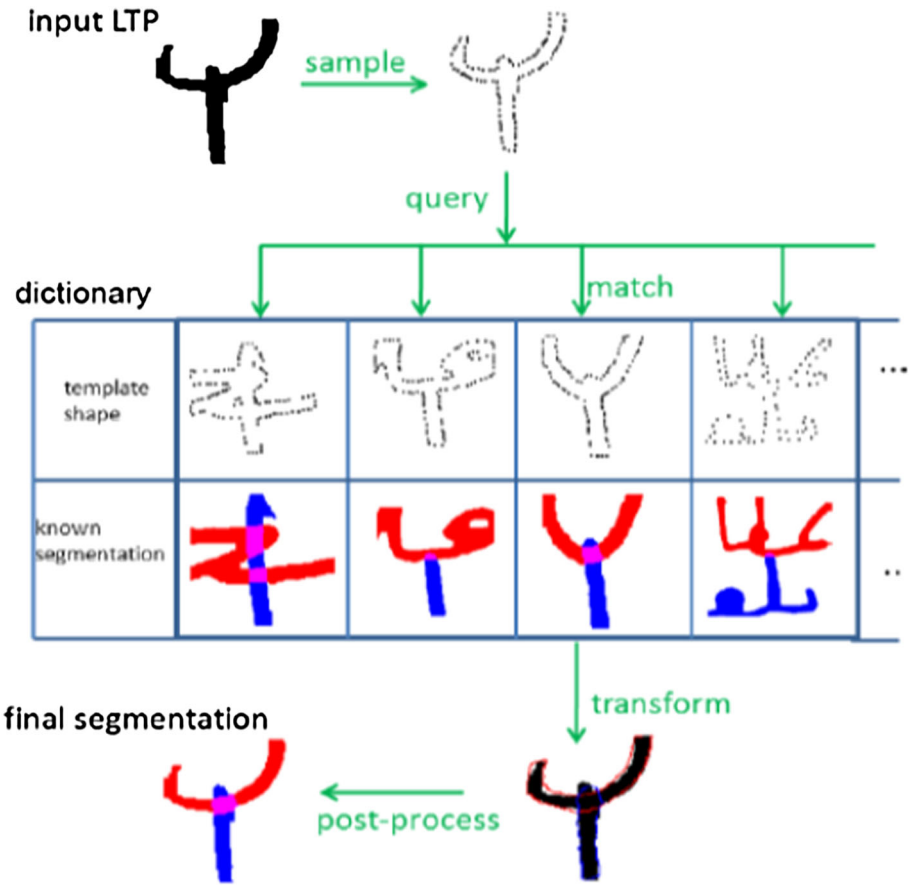

Fig. 16 TC segmentation process [1]

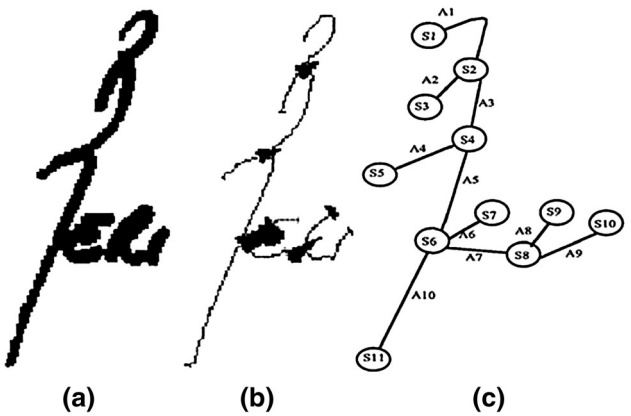

Fig. 17 **a** Example of TC, **b** TC pseudo-skeleton, **c** TC decision tree [19]

address lines with cross-touching characters, only 88.2 % of cases were correctly segmented.

In accordance with the previous work, Tseng and Lee in [21] detected text-lines using segmentation graphs where nodes are candidate nonlinear paths and edges represent a connection cost between two nodes. The Viterbi algorithm revealed cross-touching Chinese characters when the candidate segmentation path passes through a sufficient number (threshold) of black pixels. This threshold is determined, based on stroke width of a text-line image. The validity of a candidate path is verified by feeding extracted character images into an optical character recognition engine.

A neural network has been used by You and Kim in [22] to detect potential segmentation cut-points in digit string. 80 images were used for training the neural network and segmentation points were marked manually. Authors displayed a segmentation rate of 89.1 %. Note that, probabilistic models can be used in TC segmentation by providing only valid segmentation paths and then reducing in computational cost. According to authors, the failure to localize correct segmentation point is due to the wrong clustering of the primitives.

In [23], to segment touching digits database, Vellasques and al. used a Support Vector Machine and a Multi-layer Perception to classify a possible cut-points' list. TC is segmented based on recognition using its contour and analyzing its multilevel concavity.

Table 1 shows the type of documents (oriented, cursive, binary or grey level), the used data set, the obtained results and if the method treats all kinds of TCs or if it is limited to only some ones. Table 2 classifies the TC segmentation methods according the adopted approaches and presents the techniques used for the TC localization and

Table 1 Proposed methods in literature for TC segmentation

References	Script	Nature	All TC treated	Ori	Cur	Data set	Results (%)
[15]	Persian, Latin, Greek, Bengali	S, C	No	+	+	52 TCs from Persian handwritten dataset, scanned with 300dpi	–
[14]	Latin	S, C	No	+	−	20 binary images with 68 TCs	82.4
[13]	Arabic	S	No	+	+	Binary images from IFN-ENIT	–
[8]	Latin	S, C	No	+	−	20 binary images with TCs	77
[4]	Latin	S, C	No	+	+	Gray level images of postal addresses with TCs	–
[6]	Latin	S	No	+	−	ICDAR'09, 100 images for training and 200 for test	–
[3]	Arabic	S	No	+	+	125 binary Arabic text with TCs	–
[24]	Bengali	S	No	+	+	1560 binary text-lines images with TCs	–
[25]	Arabic	S, C	No	+	+	Ancient Arabic documents with TCs	–
[26]	Latin	S, C, M	No	−	−	30 gray level images from CEDAR database	95.3
[12]	Arabic	S, C, M	Yes	+	+	Gray level images with 744 TCs	71.2
[27]	Latin	S	No	+	+	100 color images scanned with 200dpi	–
[7]	Latin	S, C	No	−	−	100 images from ICDAR'07 handwriting segmentation context	–
[5]	Latin	S	No	+	+	100 images from ICDAR'07 handwriting segmentation context	–
[2]	Arabic	S, C	No	+	+	Binary images with 256 TCs	93.72
[11]	Latin	S, C	No	+	+	Images from ICDAR'09, ICDAR'13, UMD, GW, Barcelona database Marriage	–
[9]	Latin	S, C	No	−	−	200 images from IAM database	–
[20]	Japanese	S	No	+	+	152 binary images with TCs	88.2
[21]	Chinese	S	No	+	−	125 text-lines and 1132 words scanned with 300dpi	95.58
[22]	Latin touching string digit	S	No	+	−	303 TCs for test and 1284 for training scanned with 200dpi	82.8
[23]	Latin touching string digit	S	No	+	+	10,000 samples of two-digit single touching numerical strings	SVM:98.9, MLP:99.4
[1]	Arabic	S, C, M	Yes	+	+	744 binary TCs	74.4
[28]	Arabic	S, C, M	Yes	+	+	1500 binary TCs	94
[19]	Latin	S, C, M	Yes	+	+	20 binary TCs	90

S simple touching, *C* cross-touching, *M* multi-touching, *Ori* TCs identified in a skewed manuscripts, *Cur* TCs identified in cursive handwriting
No/yes means if all cases of TCs are treated or not

segmentation and precise the segmentation context. As it can be noted from these tables, most of segmentation methods are recognition-free which exploit structural information to identify the TC composing parts. The majority of the methods are also document dependent and focus on some TC cases, leaving the field open to treat other TC cases. Methods that segment TCs by cutting horizontally have difficulty in localizing the best cut position. The cut position is chosen according to the structure analysis. Results are sometimes unsatisfactory because connection between characters is generally more complicated than simple horizontal touch. Contour-based methods seem to be more efficient, since they consider the TC irregularity. Different methods used are generally related to the writing type and based on the writing morphology. Others methods greatly depend on thresholds which limit them to a reduced set of writing styles. Methods, based on the identification of junction point in the skeleton image, are also prone to errors because they assume the uniqueness of the junction point which is not usually true. In case of a complex connection (which contains loops for example) or when the skeletonization method provides more than one junction points, the junction point uniqueness assumption becomes false. In

Table 2 Taxonomy of touching component segmentation methods

Approach class	TC localization	TC segmentation	Segmentation context	References
Recognition-free	When a connected component crosses a separating line	In the low-density region	Into text-lines.	[14, 15]
	Detection of junction pixel in skeleton image under baseline	In the low-density region	Into parts of words	[13]
	Height of the connected component followed by the TC's histogram projection profile analysis	Horizontal histogram projection profile analysis	Into text-line and words	[8]
	Component belongs into two text-lines	Horizontal projection profile analysis	Into text-lines	[4]
	Absence of two separating lines in two adjacent regions	Horizontal histogram projection profile analysis	Into text-lines, words and characters	[24]
	Connected component's histogram projection profile analysis	Independent component analysis algorithm (ICA)	Into characters	[26]
	When a connected component intersects two consecutive text-lines	In the middle of the two text-line baselines	Into text-lines	[6]
	Analysis of the shortest path distance between two text-lines components	Following the tangent line between convex-hulls of components on either side of the TC	Into text-lines	[3]
	Connected component classification by *K*-means	Inter-space classification by *K*-means	Into text-lines	[25]
	Assumption that TCs are already localized	Contour decomposition	Into characters	[12]
	When a component intersects with two consecutive text-lines	Contour tracking	Into text-lines	[27]
	When a component intersects with two consecutive text-lines	Skeletonization	Into text-lines	[5]
	Height of the connected component	Skeletonization.	Into text-lines	[7]
	Detection of junction pixel in skeleton image	Skeletonization	Into text-lines	[2]
	Discontinuity in background skeleton image	Skeletonization	Into text-lines	[11]
	Intersection with two baselines	Skeletonization	Into text-lines	[9]
Based recognition	Viterbi algorithm	Recognition of segmentation result by OCR	Into characters	[21]
	Assumption that TCs are already localized	Overlaying of segmentation's template and the input TC	Into characters	[1]
	Height of the connected component	Merging parts followed by recognition step	Into characters	[20]
	Assumption that TCs are already localized	Recognition based on the multilevel concavity analysis	Into digits	[23]
	Assumption that TCs are already localized	Recognition based on Neural network and run-length coding scheme	Into digits	[22]
	Height of the connected component	Dictionary and skeleton analysis	Into text-lines	[19]

addition, the thinning process is known by its loss of information, which makes these methods inappropriate, especially for ancient documents. Hereafter, we discuss our proposed TC segmentation method based on three main steps: recognition, transformation and segmentation.

3 Proposed method

We propose a novel method for TCs segmentation which is an enhancement of our previously works [28–30]. The proposed method concerns segmentation of TCs not only between text-lines, but also TCs between words in the same text-line. It consists of three steps: (1) a recognition step to select from different models, stored in dictionary with their known correct segmentation, the most similar one that approximates the TC to be segmented, (2) a transformation step to adjust parts of the found model to the TC and (3) a segmentation step to split TC pixels into two segments according their position from adjusted model parts' midpoints. The proposed method has the merit to be robust since it is able to correctly segment TCs even if its most similar model has not been identified. It is thanks to the segmentation step which exploits well the associated model part's pixel distribution and their mid-points as it will be shown later. We first describe how the model dictionary is built. We then explain how the most similar model is selected. We finally clarify how the TC is segmented using its most similar model.

3.1 Model dictionary construction

We constructed the model dictionary from TCs and their correct segmentations. The correct segmentation of a TC consists of the two separated strokes or parts: A and B which cover only the part of connected components shared by two disjoint letters. First, we used TC models proposed by Kang and Doermann [1]. Then, we enriched the dictionary by extracting TCs and their parts from ancient arabic manuscripts (with size 120×120). We also created new TCs by reducing manuscript interlines. Finally, we added others TCs by randomly switching, rotating and warping parts of TCs.

To reduce the time for the most similar model selection, TC models are organized into clusters with their representatives (exemplar or centroid), using affinity propagation algorithm [18], as done by Kang and Doermann in [1]. The affinity propagation has also the advantage to let free cluster's number and not to force a new TC to belong to a non-homogenous cluster, contrarily to most clustering algorithms.

3.2 Most similar model selection

To select the most similar model for a TC, we compare this latter to only model cluster representatives, stored in the dictionary, using thread library [31]. Note that the dictionary can be dynamically incremented by new TCs for which we have not found similar models in the dictionary. That is the distance (in Eq. 8) which separates them exceeds a threshold (set to 0.22). These TCs will be stored in a draft then added later in the dictionary.

To compute similarity, we first make correspondences between a TC and a model using shape context descriptor. We then compute the distance between them as the sum of matching errors, between corresponding points and the energy of the aligning transform.

3.2.1 Correspondences with shape context descriptor

Finding correspondences between model and a TC consists on searching for each point p_i of TC's contour, the best matching point q_j on the model's contour by comparing their edge point's shape context histogram, as illustrated in Fig. 18. The Canny edge detector [32] followed by a sampling process is applied to get a fixed number of points, which should reduce the problem complexity.

Then, for each couple of points (p_i, q_j) from the two shapes (TC and the model), our system computes the matching cost by the χ^2 statistic test (see Eq. 1) and returns an $N \times N$ cost matrix C where $h_i(k)$ and $h_j(k)$ denote the K-bin normalized histograms at p_i and q_j. K is fixed to 60 (we used 5 bins for $\log(r)$ and 12 bins for θ).

$$C_{ij} = C(p_i, q_j) = \frac{1}{2}\sum_{k=1}^{K}\frac{[h_i(k) - h_j(k)]^2}{[h_i(k) + h_j(k)]} \quad (1)$$

Finding the best correspondences between the shape's points becomes an assignment problem. The input is the cost matrix C with entries C_{ij} and the objective is to find a permutation π minimizing the total cost of matching C (see Eq. 2). This is an instance of the square assignment (or weighted bipartite matching) problem which can be solved in $O(N^3)$ time using the Hungarian method. In our experiments, we have used the Jonker and Volgenant's algorithm [33] which seems to be efficient.

$$C = \sum_i C(p_i, q_{\pi(i)}) \quad (2)$$

Note that we used the shape context descriptor for many reasons:

- To demonstrate its power for pattern recognition, in our case to find the most similar model for an input TC.
- To summarize global shape in a rich and local descriptor.
- To greatly simplify recovery of correspondences between points of two given shapes.
- To have a robust score for measuring shape similarity, once shapes are aligned.
- It is tolerant to all common shape deformations. As a key advantage no special landmarks or key-points are necessary.

But, even if the shape context is a rich descriptor, the presence of outliers is a problem that can affect the matching result (see Eq. 2) especially when handling with small and finical shapes such as TCs.

Outlier pixels Note that outlier pixels problem has been raised by Chui [34] and Belongie [16] in two ways: (1) when two shapes have different number of contour points, which results in adding fictitious nodes with a constant matching cost, (2) when the cost of point-to-point matching exceeds a certain value, according to the remaining point-to-point matching costs. These pairs of points significantly affect the quality of match and therefore the similarity distance. To remedy this problem, we propose a region-to-region matching (instead of Belongie point-to-point matching) which imposes a constraint to discard the outlier pixels and to highlight similarities between parts of the two shapes.

Region-to-region matching Let us consider Matched Points (MP) the set of N matched couples of points, using the Eq. (2) proposed by [16]. Let g be a bijective function that associates to each point p_i from TC, its matched point q_j from the model verifying Eq. (2). The Well-Matched Points (WMP) is a subset of MP,

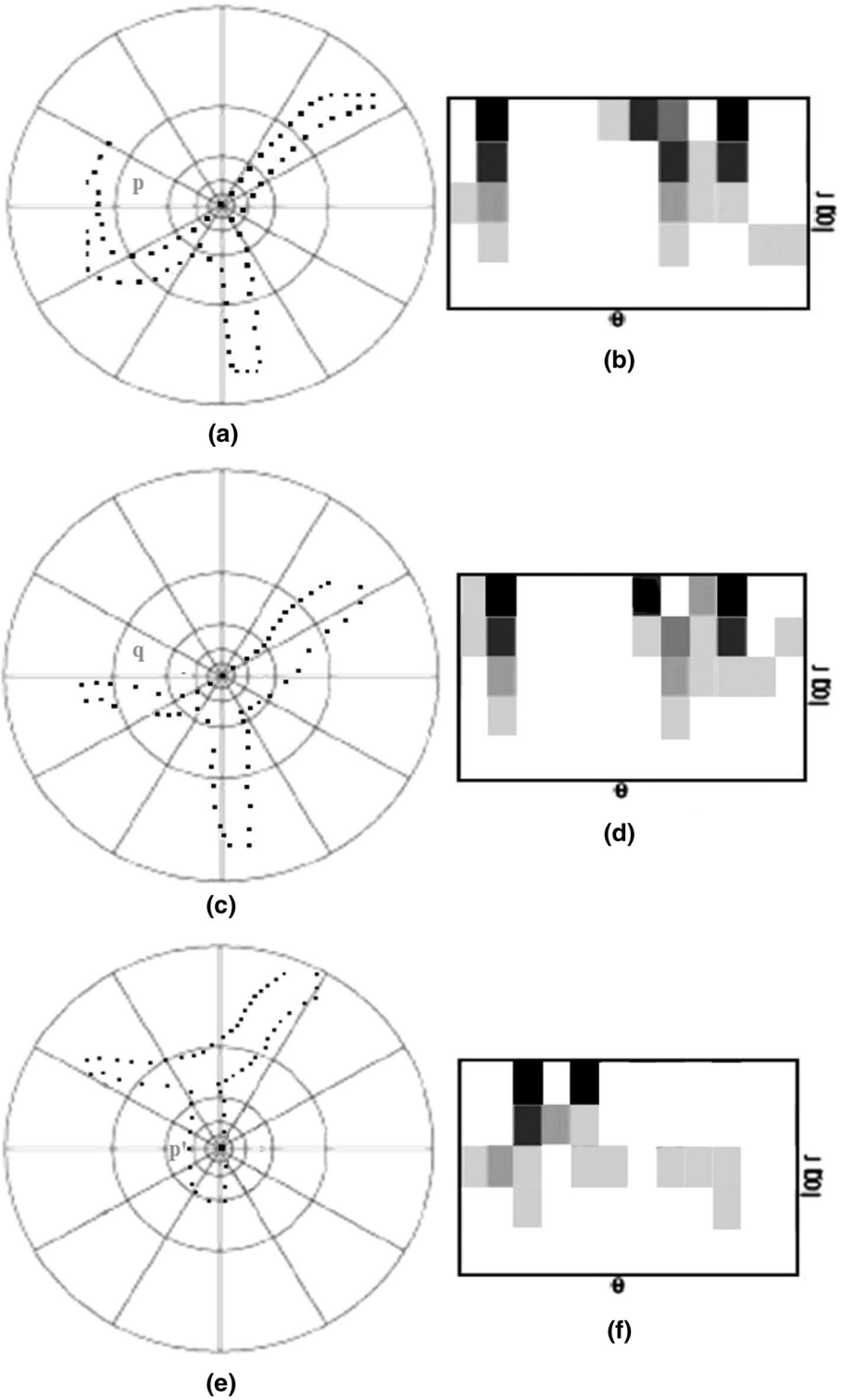

Fig. 18 Matching with SC: **a** SC of p in TC_1, **b** log-polar histogram for p, **c** SC of q in TC_2, **d** log-polar histogram for q which is similar to that in **b**, but different from p' in **f**. The best matching is between p in TC_1 and q in TC_2. *Black bins* correspond to a higher number of pixels in that bin, *gray bins* contain fewer pixels than the *black cells*. Log-polar histogram similarity is according to the χ^2 distance

composed of selected couples (p_i, q_j) based on their neighborhood. For each matched couple (p_i, q_j) from MP, we consider their corresponding k nearest neighbors k-NN: $N_{\mathrm{TC}}(p_i, k)$ and $N_{\mathrm{Model}}(q_j, k)$, using Euclidian distance. A couple (p_i, q_j) belongs to WMP if just a subset of their corresponding neighbors (at least n from k neighbors) are also matched together. Then, WMP can be defined as follows:

Table 3 Parameters setting for region-to-region matching

Parameters	Values	Interpretation
N	200	Number of contour pixels in the TC and in the model
k	30	Number of the nearest neighbors of p_i and q_j
n	10	Minimum number of matched couples in neighboring regions of p_i and q_j required to be among WMP

$$N_{\mathrm{TC}}(p_i,k)\times N_{\mathrm{Model}}(q_j,k)\subseteq \mathrm{MP} \text{ with } 1\leq \mathrm{i}\leq \mathrm{N},\ 1\leq \mathrm{j}\leq \mathrm{N}$$
$$\text{and}$$
$$\mathrm{Card}\{\mathrm{g}^{-1}\{\mathrm{N_{TC}}(\mathrm{q_j},\mathrm{k})\}\bigcap \mathrm{N_{Model}}(\mathrm{p_i},\mathrm{k})\}\geq \mathrm{n}$$
$$\text{where}$$
$$N_{\mathrm{H}}(h,k)=\{h'\neq h\in k\text{-NN}(h)\} \tag{3}$$

Table 3 displays parameters used to select the most similar model.

Two reasons were behind the choice of 200 for N, instead of 100 as set by Belongie et al. [16]: (1) this value save the TC structure and (2) the number of WMP remains meaningful when it decreases using Eq. (3). For the choice of k, we found that by comparing several numbers of neighbors, we concluded that below 30, WMP tends to zero and beyond 30, WMP tends to MP. n is set to 10, because under this value, the concept of region matching is no longer significant. But, if n increases, WMP might be approach zero.

Thus, contrarily to the matching process, proposed by Belongie in [16], which considers all pairs (p, q) minimizing the cost C defined (see Eq. 2), we just take into account the well-matched points WMP (see Eq. 3) according to neighborhood relation. We then introduce a constraint to filter the matching results and avoid the outlier points. Therefore, couples (p, q), qualified as outliers, will be discarded in the aligning transform.

3.2.2 Similarity computing

To measure the distance between two shapes, we have used three parameters:

- D_{sc}: The shape context cost (see Eq. 4),
- N_{match}: The number of WMP divided by the total number of contour points N (see Eq. 9),
- D_{be}: The bending energy which corresponds to the amount of transformation necessary to align the model on the TC (see Eq. 7).

$$D_{\mathrm{sc}}=\frac{1}{n}\sum_{p\in P}\mathrm{argmin}_{q\in Q}C(P,T(q))+\frac{1}{m}\sum_{q\in Q}\mathrm{argmin}_{p\in P}C(P,T(q)) \tag{4}$$

where $T(.)$ denotes the estimated TPS shape transformation.

Recall that the similarity metric defined by Belongie in [16] is a weighted sum of three terms (see Eq. 5) where D_{ac} is the image appearance distance, D_{sc} is the shape context distance and D_{be} is the bending energy.

$$\text{Similarity}=-\mathrm{D}\quad\text{and}\quad \mathrm{D}=1.6\mathrm{D_{ac}}+\mathrm{D_{sc}}+0.3\mathrm{D_{be}} \tag{5}$$

We will explain how to use the set of correspondences (see Eq. 3), to estimate an aligning transform and then compute D_{sc} and D_{be}.

Estimation of the aligning transform Several technical transformations exist, among them the affine transform which is the most known and the Thin Plate Spline (TPS) which is widely used for flexible coordinate transformation. In our case, we used the TPS transformation [17] as an interpolation technique. Hence, we compute, from two sets of points, having a correspondence relation (WMP), the function f which is defined everywhere in the $\Re$ space by:

$$f(x,y)=a_1+a_xx+a_yy+\sum_{i=1}^{N}w_iU(||P_i-(x,y)||) \tag{6}$$

where $U(r)=r^2\log r^2$ and $\sum_{i=1}^{N}w_i=\sum_{i=1}^{N}w_ix_i=\sum_{i=1}^{N}w_iy_i=0$. This TPS interpolating $f(x, y)$ minimizes the bending energy I_f (see example in Fig. 19).

$$I_f=\int\int_{\Re}\left(\left(\frac{\partial^2 f}{\partial x^2}\right)+2\left(\frac{\partial^2 f}{\partial x\partial y}\right)+\left(\frac{\partial^2 f}{\partial y^2}\right)\right)d_xd_y \tag{7}$$

The estimation of the aligning transform between the model and the TC accordingly to the matching result (WMP) is necessary for two reasons: (1) to compute D_{sc} (see Eq. 4) and D_{be} (see Eq. 7) and (2) to transform the separated parts of

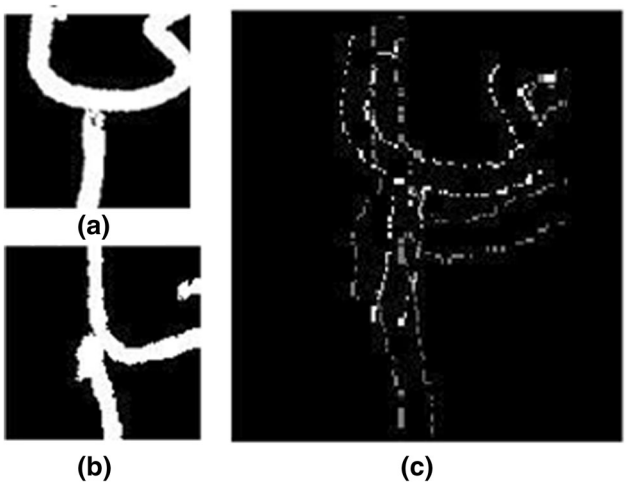

Fig. 19 **a** Input TC, **b** most similar model, **c** interpolation of the input TC and the most similar model after deformation with TPS transformation to compute D_{be}

Table 4 Comparison results to select the most similar model

	Eq. (5) (Belongie's method)		Most similar model	Eq. (8) (our method)		Most similar model
	Model 1	Model 2		Model 1	Model 2	
MP number	200	200	Model 1	91	155	Model 2
Similarity	−0.087	−0.096		0.004	0.019	

the most similar model to approximately coincide with the input TC's parts. More details are given in Sect. 3.3.

Similarity computing To compute similarity between a TC and a model, we adjusted the initial similarity metric, proposed by Belongie, as defined in Eq. (5), to consider the number of WMP. We do not used D_{ac} as we handle with binary images. Instead, we considered the percentage of the valid matched points N_{match} with an empirically fixed coefficient Coeff, set to 0.2 and used to adjust similarity computing. Thus, the proposed similarity metric is defined as follows:

$$\text{Similarity} = -\text{D} \quad \text{and} \quad \text{D} = \text{D}_{sc} + 0.3\text{D}_{be} - \text{coeff} * \text{N}_{match} \tag{8}$$

$$N_{match} = \frac{\text{WMP}}{N} \tag{9}$$

N_{match} is a reliable indicator of the matching quality and the term $N_{match} * \text{coeff}$ makes more accurate the distance computing. In fact, if this ratio (N_{match}) is large, it means that the model approaches the TC and therefore the distance between them will decrease and even when the number of WMP is small, the interpolation function and the bending energy are not very significant since they will act on a reduced set of couples. So, the equation should be balanced with the term $N_{match} * \text{coeff}$. The most similar model is the one having the highest similarity with the input TC.

Some examples are given to demonstrate that similarity between TCs and some similar models is higher, using our proposed similarity metric (see Eq. 8), compared to that obtained by Belongie (see Eq. 5) as displayed in Fig. 21.

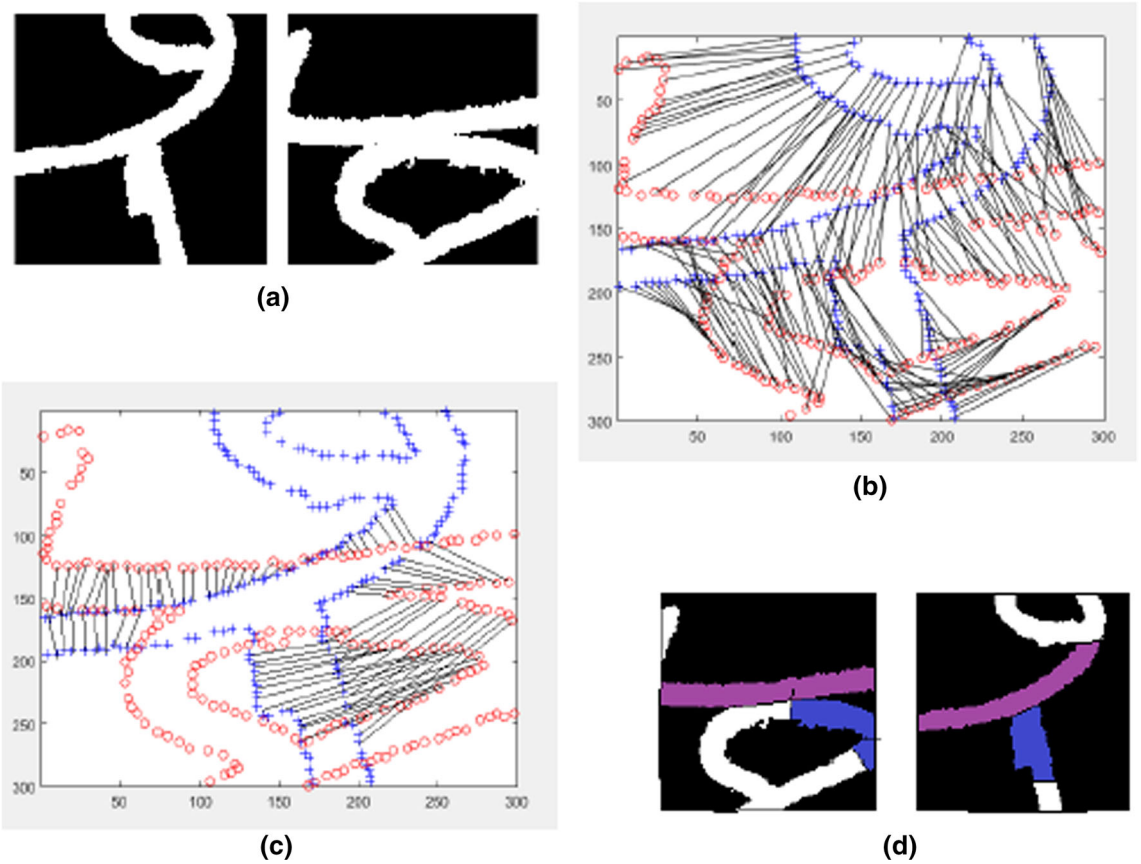

Fig. 20 **a** TC and its most similar model, **b** point-to-point matching, **c** region-to-region matching, **d** illustration of region matching

An other example is given where Eq. (5) fails to identify the most similar model, whereas Eq. (8) managed to do so (see Fig. 22; Table 4). Figure 20 compares results from point-to-point matching, using Belongie's method, and region-to-region as we proposed.

3.3 Segmentation

Once the most similar model for an input TC is identified, we used its known segmentation (the two model parts: A and B) and the estimated TPS parameters that warps the selected model into the TC for the TC segmentation. The warped model parts A and B, using the same TPS parameters, should be aligned with TC's segments.

Let (P_a, P_b) be the contours of the two transformed model's parts and (C_a, C_b) be their respective midpoints. Recall that for a given point set P to which we associate the metric D (Euclidean or Manhattan distances for example), the midpoint M is the most accessible point (the one that minimizes the sum of the distances of all the set). These midpoints are approximately those of the parts composing the input TC. Thus, to segment TC and reconstitute its parts, we have to associate pixels to the closest midpoint. However, the assignment can be ambiguous for pixels in the shared zone which can be assigned to the wrong part (see Fig. 24a). For that, we propose to adjust midpoints before being used. The intersection points (I_a, I_b)

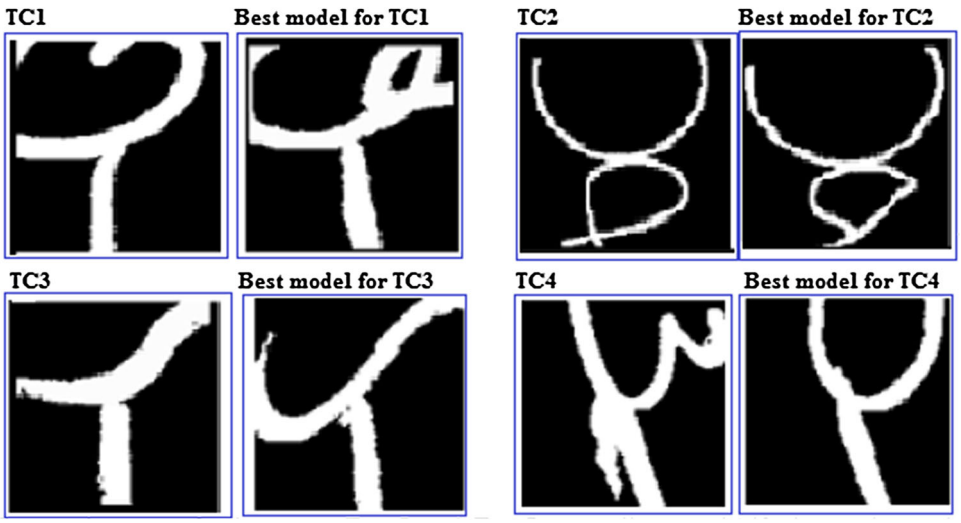

Fig. 21 Comparison results between Eqs. (5) and (8) according to similarity and matched points. (TC_1, best model for TC_1): MP = 200, similarity = −0.1509, WMP = 113, similarity = −0.0379, (TC_2, best model for TC_2): MP = 200, similarity = −0.1232, WMP = 165, similarity = +0.0097, (TC_3, best model for TC_3): MP = 200, similarity = −0.0923, WMP = 110, similarity = +0.0176, (TC_4, best model for TC_4): MP = 200, similarity = −0.1045, WMP = 160, similarity = +0.0554

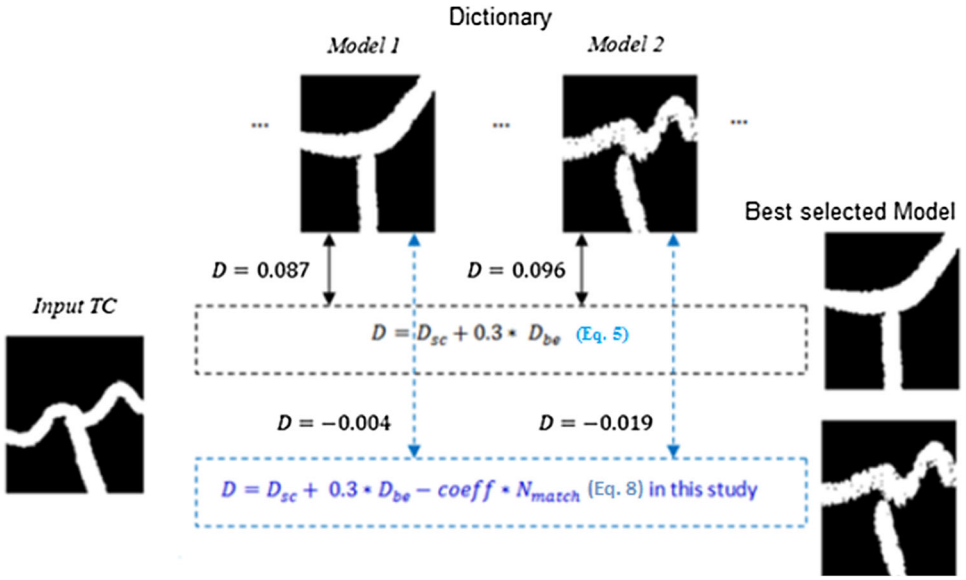

Fig. 22 Comparison results (Eq. 5 versus Eq. 8) when selecting the most similar model

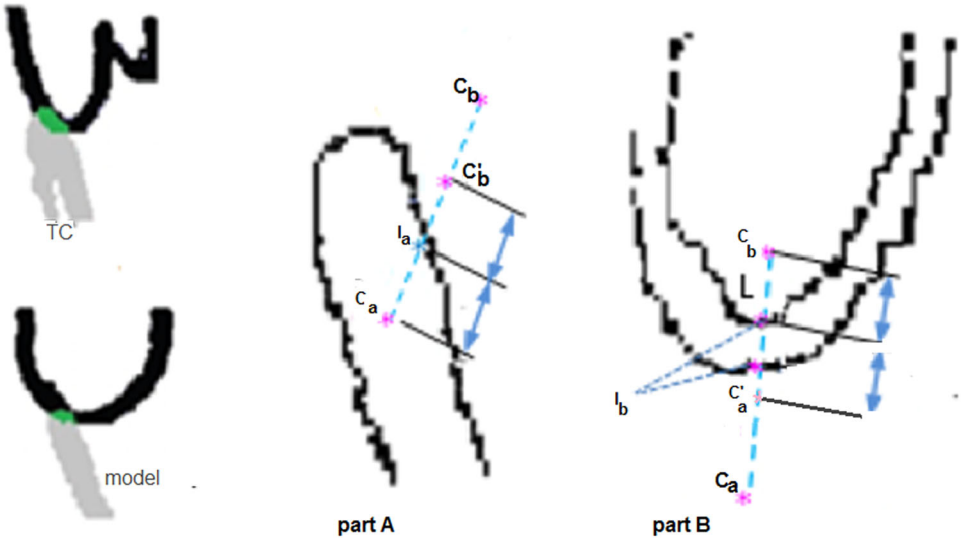

Fig. 23 **a** Contour of the model's part A after TPS transformation, **b** contour of the model's part B after TPS transformation

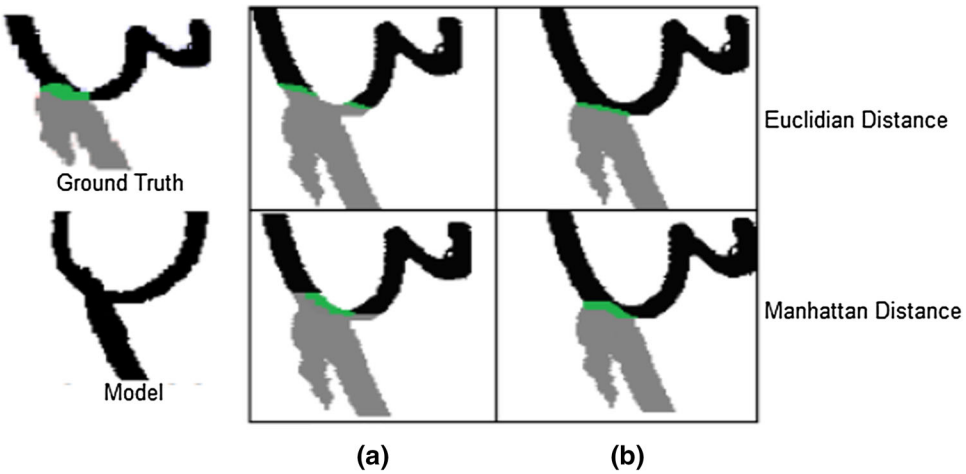

Fig. 24 Segmentation results: **a** without centers transform, **b** with centers transform

between the segment L, connecting the two initial midpoints and (P_a, P_b) respectively is localized as indicated in Eq. (10). Using each intersection point, we set an adjusted midpoint as the symmetric of the nearest one to the intersection (see C'_b in Fig. 23a or C'_a in Fig. 23b) and based on that, the pixel attribution will be performed (see Fig. 24b).

$$I_a = L \cap P_a \quad \text{and} \quad I_b = L \cap P_b \tag{10}$$

Extracting the shared zone for the two parts is important for the character recognition. For that, we use the intersection point I_b of the segment connecting the initial midpoints and the transformed model's part P_b to compute and identify the part A of the TC and vice versa for the other part. More precisely, if two parts are connected, so to separate them and affect pixels to each part according to the closest center, a natural judgment says that we must measure the degree of emergency of each part in the other. To determine part A of the TC, we compare $d(C_a, I_b)$ to $d(C_b, I_b)$. If the first term is the highest, C_a changes to be C'_a as the symmetric of C_b according to I_b on L. Else, C_b moves to C'_b as the symmetric of C_a according to I_b on L. These new midpoints are putted on the TC and then, TC's part A will include all closest pixels nearest C_a or C'_a compared to C_b or C'_b (d is Euclidean or Manhattan distance). The same treatment will be done on part B comparing distances between $d(C_b, I_a)$ and $d(C_a, I_a)$. Finally, the TC's shared zone represents the intersection between part A and part B. This segmentation step is summarized in Algorithm 1.

Algorithm 1 TC segmentation

Input: the input TC, the *MSM* with its correct segmentation ($part_A$ and $part_B$) , Ψ: the estimated parameters that warps the *MSM* into the TC.
Output: TC's $part_A$, TC's $part_B$, TC's *sharedzone*.
Begin:
1:$P_a \leftarrow$ TPS(Ψ, $part_A$'s contour) ; $P_b \leftarrow$ TPS(Ψ, $part_B$'s contour)
2:$C_a \leftarrow$ Midpoint(P_a) ; $C_b \leftarrow$ Midpoint(P_b)
3:$L \leftarrow$ Line(C_a,C_b): Line connecting C_a to C_b
4:$I_a \leftarrow L \bigcap P_a$; $I_b \leftarrow L \bigcap P_b$
5: **for each** pair of intersection point (I_a,I_b), choose the nearest one
endfor
Initialization
6:$C'_a \leftarrow C_a$; $C'_b \leftarrow C_b$; TC's $part_A \leftarrow \emptyset$; TC's $part_B \leftarrow \emptyset$; TC's *Sharedzone* $\leftarrow \emptyset$
Extraction of TC's $part_A$
7: **if** Distance(C'_a, I_b)) < (Distance(C'_b, I_b)) **then**
$C'_b \leftarrow$ **symmetric** (C'_a,L,I_b)
else
$C'_a \leftarrow$ **symmetric** (C'_b,L,I_b)
endif
8: Put C'_a and C'_b on the TC
9: TC's $part_A \leftarrow$ {pixels near C'_a compared to C'_b }
Extraction of TC's $part_B$
10: **if** Distance(C_b, I_a) < Distance(C_a, I_a) **then**
$C''_a \leftarrow$ **symmetric** (C_b,L,I_a)
else
$C''_b \leftarrow$ **symmetric** (C_a,L,I_a)
endif
11: Put C''_a and C''_b on the TC
12:TC's $part_B \leftarrow$ {pixels near C''_b compared to C''_a }
Extraction of the TC's *sharedzone*
13:TC's *sharedzone* $\leftarrow$ TC's $part_A \bigcap$ TC's $part_B$
End

The advantages of this segmentation scheme are various. First, it is script independent. Thus, we do not need to study the connections relative to any script. We only have to adjust the training set for the given scrip. Second, we do not have to find a very similar model for the input TC or require a perfect warping, as done by the Kang and Doermann in [1]. It is sufficient that TC and model's shapes are approached to have similar geometrical features.

In Fig. 25, we summarize all the followed steps to segment an example of TC.

4 Experiments and discussion

To evaluate the performance of the proposed segmentation method, we form a testing set having 820 varied samples whose distribution accordingly to the connection's type is 220 for inter-word and 620 for inter-lines. For training, the model dictionary construction contained 500 samples (the most representative ones) which are clustered using the affinity propagation (AP). Recall that AP computational complexity is expected to scale like $O(N^2)$. It involves the matrix S of pair distances. But, fixing some parameters related to AP as number of clusters, number maximum of iterations and the damping factor, the run-time gets around to 33 s for a 500×500 similarity matrix. Note that the run-time is proportional to the number of clusters and the clustering with AP is done only once.

To match a TC and a model, we used the Jonker and Volgenant algorithm [33], which takes almost 10 ms for a 200×200 costs C_{ij} matrix. Table 5 displays run-time tests.

In the recognition and transformation steps, we used a multi-threading approach, which reduces the computational time to almost the half.

To evaluate the segmentation results, we computed the Matchscore [35] for each obtained segment. It is based on a comparison of the segmentation of the input TC to its ground truth. Let (G_A, G_B) be the sets of all points inside the ground truth segments, (R_A, R_B) the set of all points inside the result segments and $T(s)$ a function that counts the elements of set s. MS_i represents the matching results of the i ground truth region and the i result region.

$$\text{MatchScore} = \frac{2 * \text{MS}_\text{A} * \text{MS}_\text{B}}{\text{MS}_\text{A} + \text{MS}_\text{B}} \tag{11}$$

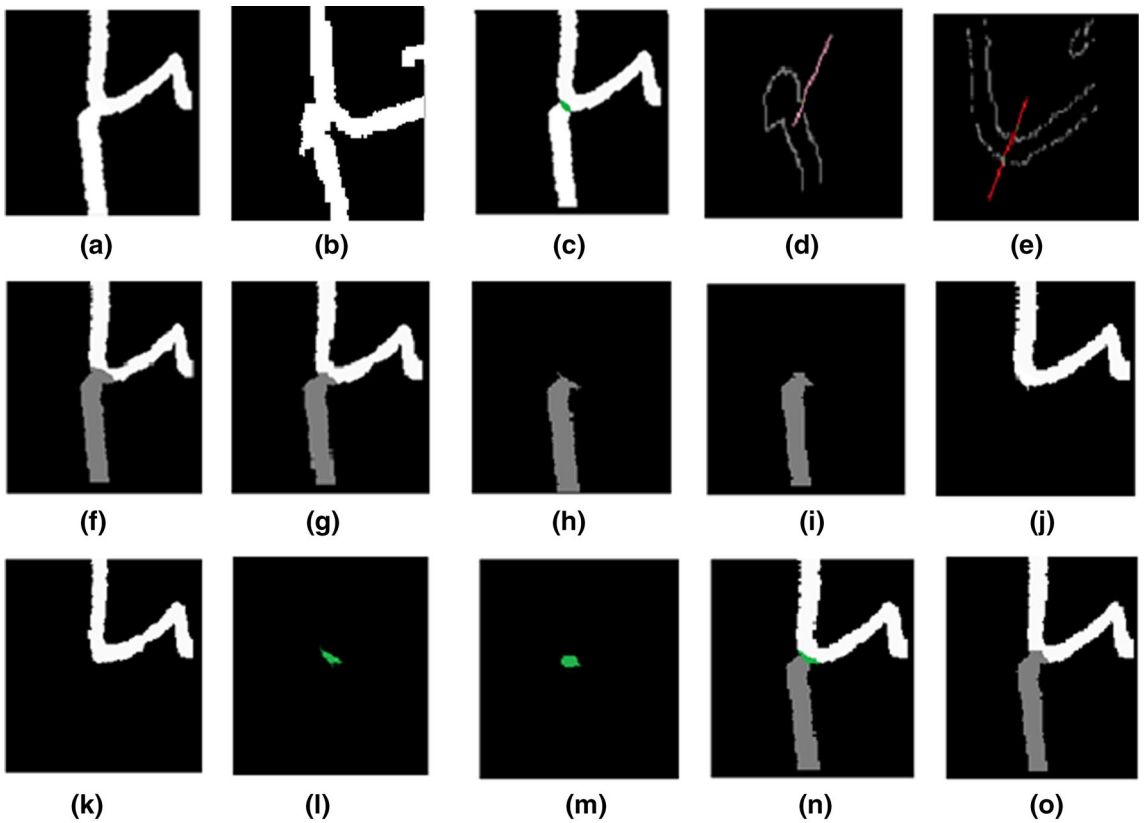

Fig. 25 Illustration of the segmentation method applied to **a** tested on both distances (Euclidean with gravity center and Manhattan with midpoint): **a** input connection, **b** most similar model: similarity = 0.26, WMP = 148, **c** ground truth, **d** model's part A after transformation with TPS, **e** model's part B after transformation with TPS, **f** result of segmentation without center correction using Euclidean distance and gravity center: MS = 0.9172, **g** result of segmentation without center correction using Manhattan distance and midpoint: MS = 0.9498, **h** result of extraction of the part A of the input connection using Euclidean distance after correction of gravity center C_a (60, 75), new center C'_b(76, 47), **i** result extraction of the part A of the input connection using Manhattan distance after correction of midpoints: C_a (60, 75), new center C'_b(76, 47), **j** result of extraction of the part B of the input connection using Euclidean distance after correction of gravity center: C_a (60, 75), new center C'_b(72, 59), **k** result of extraction of the part B of the input connection using Manhattan distance after correction of midpoints: C_a (61, 75), new center C'_b (71, 59), **l** extraction of the shared zone after correction of gravity center, **m** extraction of the shared zone after correction of midpoint, **n** final result of segmentation using Euclidean distance after correction of gravity centers: MS = 0.98, **o** final result of segmentation using Manhattan distance after correction of midpoints: MS = 0.98. C_a (60, 75), C_b (80, 42), I_a (66, 67), I_b (68, 61) are respectively centers of part A, part B, intersection of part A with L and intersection of part B with L according to Euclidean distance. C_a(61, 75),C_b(75, 45.5), I_a (66, 67), I_b (66, 67) are respectively centers of part A, part B, intersection of part A with L and intersection of part B with L according to Manhattan distance

$$\mathrm{MS}_i = \frac{T(G_i \cap R_i)}{T(G_i \cup R_i)} \tag{12}$$

The results, displayed in Table 7, are the rates of accepted segmentation for a matching score upper than 0.8.

For TC segmentation, we tested with two metrics: Euclidean and Manhattan distances and two center points: gravity center and midpoint. The obtained results are satisfactory. The findings that strongly discriminate between these two metrics are: (1) the shapes of the segments, using Manhattan distance and midpoint, are more similar to their ground truth and (2) the TC segmentation, using the Euclidean distance, is like cutting according to the orthogonal on the line connecting the two gravity points of the model's parts. Therefore, the Manhattan distance seems to be more appropriate for this task (see Fig. 24).

Notice that comparison with other segmentation methods has no sense, especially when most of them are dedicated to a particular script or specific TC types. In fact, some methods seem to be efficient to segment TCs between text-lines, but cannot be applied on TCs between words of the same text-line. Moreover, not all of proposed methods evaluate the segmentation rate by the Matchscore method.

Let us notice that few are works on TC segmentation in Arabic manuscripts. Among the most related ones which

Table 5 Test results

Samples	Time needed to compute matrix similarity (min)	Time for clustering with AP (min)
100	0.50	1.38
150	2.03	2.43
200	3.26	4.12
500	52.00	52.46

deal in depth with this problem is that of Kang and Doermann [1]. There are also Ouwayed and Belaïd's [2] and Boukerma and Farah's [13] works. To compare with these methods, we implemented them. Experiments are carried on a common TC database. The obtained results for a matching score greater than 0.8 demonstrate the efficiency of our proposed segmentation method as shown in Table 6.

When testing on TCs that occur between successive words of the same text-line, as done by Boukerma and Farah [13], we find that our method achieves a superior accuracy: 93.7 % in comparison to 60 %. In addition, our method has the merit to resolve problems raised by Kang and Doermann [1] as shown in Fig. 26. Recall that these problems are due to the steps prior to the TC segmentation, notably the TC extraction, recognition and transformation are due to (1) "bad template" or ambiguity, (2) disturbing components and (3) when the TPS cannot mark corresponding components very accurately.

For problem of "bad template", we proposed a segmentation method that exploits pixel distribution of the most similar model parts and their central points well. Ambiguity and transform deviation problems are almost solved and few are the recorded errors (see Fig. 26, column d, obtained results) thanks to the use of region-to-region matching which extends the comparison by looking for similarities between parts of two shapes.

Figure 27 shows examples for TCs where Kang and Doermann's method [1] reached very good score followed by our results. An example failure case of the proposed method is shown in Fig. 27 (line 4: our results). Although the segmentation ratio exceeds 85 %, the TC is badly segmented. The letter in below س (sin), seems to be closer to م (mim).

Table 7 Test results with MatchScore threshold 0.80

Metric	Euclidean (%)	Manhattan (%)
Accuracy	92.6	94

As our work is inspired by that of Kang and Doermann [1], we summarized their similarities and differences as follows:

- Similarities:
  - Like Kang and Doermann method, we proposed a TC segmentation method based on recognition approach, using the shape context descriptor (first proposed by Belongie [16] for object recognition) to find the most similar model for an input TC.
  - Like Kang and Doermann method, we built a dictionary that includes models and their correct segmentation.
- Differences:
  - Contrarily to Kang and Doermann method, we proposed in the recognition step:
    - A new matching method: a region-to-region matching (instead of point-to-point matching, as done by Belongie in [16] then used by Kang and Doermann in [1]), to avoid the problem of outlier pixels. By ignoring outliers, the recognition results, especially when matching, transforming and computing distance between input TCs and models, get performed.
    - A new similarity metric (versus the one proposed by Belongie [16], then used by Kang and Doermann to select the most similar model), considering the proposed region-to-region matching. The proposed similarity metric seems

Table 6 Comparison segmentation methods with MatchScore threshold 0.80

References	Approach	Segmentation method	Accuracy (%)	Errors
[2]	Recognition-free	Detection of junction point and computation of angular variation curves in skeleton image	67.6	When missing junction point or when dealing with complicated TCs
[1]	Recognition-based	Selection of the nearest model and overlaying of known segmentation on the TC	71.4	(1) When no adequate template can be found for a TC, (2) bad template: when the most similar model may not be the right one, (3) transform deviation: when the TPS transform cannot mark corresponding components very accurately
Our method	Recognition-based	Selection of the nearest model and adjust centers of model's parts	94	Some errors due to bad template and transform deviation

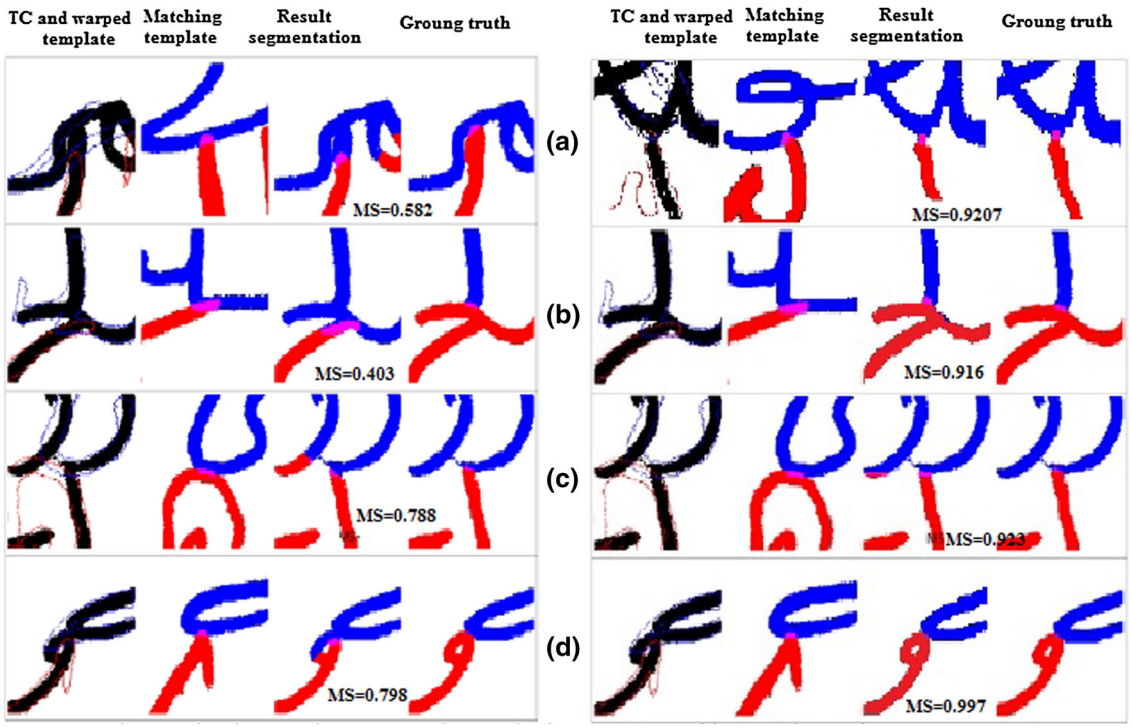

Fig. 26 The four kinds of problems related to method in [1] followed by our obtained results

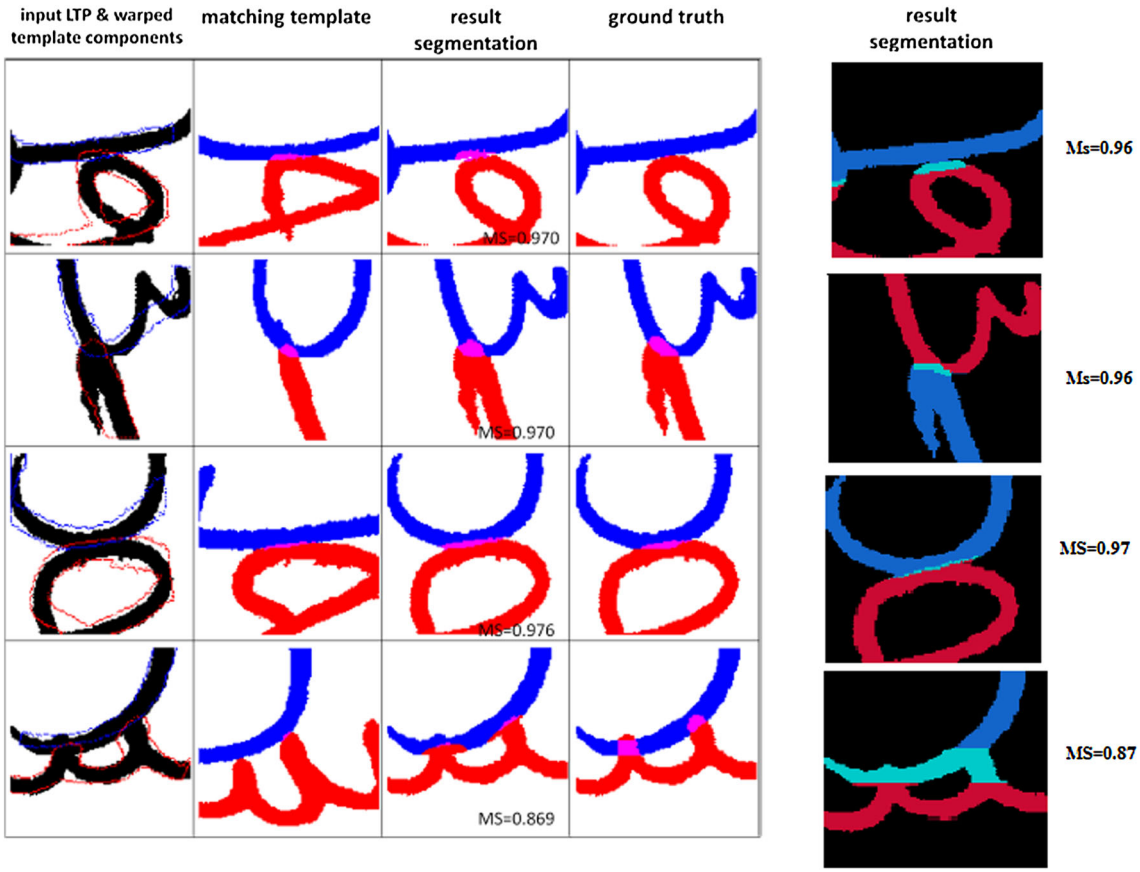

Fig. 27 **a** Examples of TCs with MatchScore ≥ 0.8 segmented with Kang's method, **b** results with our method

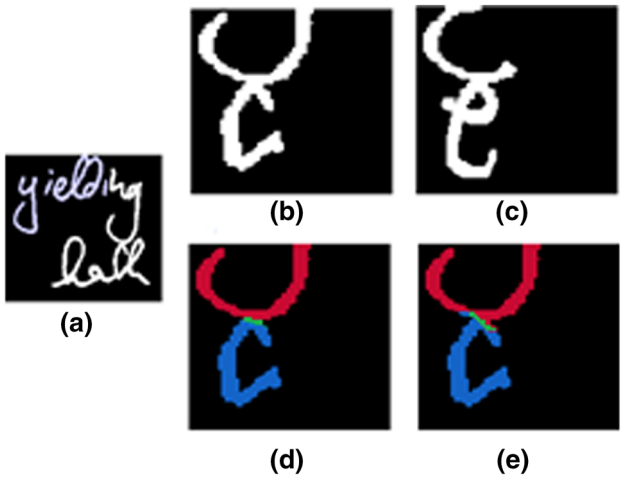

Fig. 28 Segmentation results on a Latin TC: **a** manuscript used in [7], **b** TC, **c** most similar model, **d** and **e** segmentation results with Euclidean (Matchscore = 0.98) and Manhattan (Matchscore = 0.95) distances

to be more accurate in selecting the most similar model compared to that used by Kang and Doermann. It is worth to note that the use of the proposed similarity metric can be generalized for any pattern recognition applications.

- To find the most similar model, we compare the TC to only the cluster centroid (or examplar) in the dictionary, because we do not need a very closest model, as required in Kang and Doermann method which needs to compare to all the cluster models. Consequently, our run-time gets better.
- In our method, the dictionary can be dynamically incremented by new TCs for which we have not found similar models in the dictionary. That is the distance (in Eq. 8) which separates them exceeds a threshold (set to 0.22). These TCs will be stored in a draft then added later in the dictionary.

– Contrarily to Kang and Doermann method, we proposed in the segmentation step:

  – A different method. The Kang and Doermann method is just an overlaying between the most similar model and the input TC. But when the model and the TC are not well overlaid, the method generally fails to provide an accurate TC segmentation result. Our method is able to correctly segment the input TC even if the most similar model does not perfectly look like the TC thanks to the proposed mid-point adjustment. The main idea is that the two parts, composing the most similar model, are approximately the same as those of the TC after transformation. Likewise for their corresponding midpoints, they would be almost the same. This is ensured by the proposed region-to-region matching because with this matching, we aimed that TPS transform clearly marks the corresponding components. TC segmentation can then be performed by assigning pixels to their nearest midpoint. For pixels in the TC shared zone, we have to adjust the two model parts mid-points before being assigned.
  – Unlike Kang and Doermann method (which has been only tested on TCs between successive text-lines), our method has the benefit to segment connections between successive text-lines, words or letters.
  – Unlike Kang and Doermann method (which has been only tested on handwritten Arabic manuscripts), our method is script independent. We tested it on handwritten Arabic and Latin manuscripts (see Fig. 28).

Our method is also an attempt to resolve some problems, mainly "bad template", ambiguity and transform deviation, raised by Kang and Doermann as explained before.

5 Conclusion

In this work, we proposed a novel recognition-based segmentation method of TCs in Arabic manuscripts. It mainly consists of three steps: recognition, transformation and segmentation. Hereafter is a summarization of our method's key points:

– to represent TC and models, we used the shape contexts which is able to summarize global shape in a rich and local descriptor, tolerant to all common shape deformations.
– To select the most similar model, we proposed a region-to-region matching instead of point-to-point matching to avoid outlier pixels problem and better identify the most similar model,
– To find the most similar model, we compared the TC to only cluster representative, so the run-time gets better. The run-time is further reduced by parallel research.
– To not require a perfect model (a very similar model), because even if not the best model is chosen, our method succeeds to correctly segment the TC as shown in Fig. 26a (our obtained results),
– To precisely decide where TC should be segmented at the shared zone, we proposed to adjust the model part's center points,

– To check if our method is script independent, we tested on some Latin TCs (200 TCs), as shown in Fig. 28, and the obtained results are encouraging.

Add to that, the proposed method has the benefit (1) to segment connections between successive text-lines as well as between words of the same text-line, (2) to be general since it is writer and script independent and (3) it is able to cope with skew and warping disturb. Experiments confirm effectiveness of the proposed method. In the future, we plan to handle with multi-touching components where difficulty is very important.

References

1. Kang L, Doermann D (2011) Template based segmentation of touching components in handwritten text-lines. Proceedings of the ICDAR, Beijining
2. Ouwayed N, Belaïd A (2009) Separation of overlapping and touching lines within handwritten Arabic documents. In: Proceedings of the 13th international conference on computer analysis of images and patterns, Münster (North Rhine-Westphalia), pp 237–244
3. Kumar J, Kang L, Doermann DS, Abd-Almageed W (2011) Segmentation of handwritten text-lines in presence of touching components. In: Proceedings of the ICDAR, pp 109–113
4. Likforman-Sulem L, Faure C (1995) Une méthode de résolution des conflits d'alignements pour la segmentation des documents manuscrits. Traitement Signal 12(6):541–549
5. Vassilis P, Themos S, Vassilis K, George C (2010) Handwritten document image segmentation into text lines and words. Pattern Recognit 43(1):369–377
6. Lemaitre A, Camillerapp J, Cousnon B (2011) A perceptive method for handwritten text segmentation. Proceedings of the DRR, San Francisco
7. Ouloudis GL, Gatos B, Pratikakis I, Halatsis C (2009) Text-line and word segmentation of handwritten documents. Pattern Recognit 42(12):3169–3183
8. Takru K, Leedham G (2002) Separation of touching and overlapping words in adjacent lines of handwritten text. Proceedings of the IWFHR, Ontario
9. Rohini S, Uma Devi RS, Mohanavel S (2012) Segmentation of touching, overlapping, skewed and short handwritten text lines. Int J Comput Appl 49(19):24–27
10. Ouwayed N (2010) Segmentation en lignes de documents anciens: application aux documents arabes. PhD thesis, Nancy University
11. Farnandez-Mota D, Lldos J, Fornes A (2014) Graph based approach for segmenting touching lines in historical handwritten documents. In: IJDAR
12. Kang L, Doermann DS, Cao H , Prasad R, Natarajan P (2012) Local segmentation of touching characters using contour based shape decomposition. In: Proceedings of the DAS, Gold Coast, pp 460–464
13. Boukerma H, Farah N (2012) PAW-IFN/ENIT: une nouvelle base de pseudo-mots arabes pour une approche de reconnaissance pseudo analytique, vol 1, ARIMA, pp 1–9
14. Alaei A, Pal U, Nagabhushan P (2011) A new scheme for unconstrained handwritten text-line segmentation. Pattern Recognit 44(4):917–928
15. Alaei A, Nagabhushan P, Pal U (2011) Piece-wise painting technique for line segmentation of unconstrained handwritten text: a specific study with Persian text documents. Pattern Anal Appl 14:381–394
16. Belongie S, Malik J, Puzicha J (2002) Shape matching and object recognition using shape context. In: IEEE transactions on pattern analysis and machine intelligence, pp 509–522
17. Bookstein FL (1989) Principal warps: thin-plane spline and the decomposition of deformations. In: IEEE transactions on pattern analysis and machine intelligence
18. Schaefer SE (2007) Graph clustering survey. Elsevier, New York
19. Piquin P, Viard-Gaudin C, Barba D (1994) Coopération des outils de segmentation et de binarisation de documents. In: Proceedings of the colloque national sur l'Ecrit et le document, Rouen
20. Ikeda H, Ogawa Y, Koga M, Nishimura H, Sako H, Fujisawa H (1999) A recognition method for touching Japanese handwritten characters. In: Proceedings of the ICDAR 99, pp 641–644
21. Tseng YH, Lee HJ (1999) Recognition-based handwritten Chinese character segmentation using a probabilistic Viterbi algorithm. Pattern Recognit Lett 20(8):791–806
22. You D, Kim G (2003) An approach for locating segmentation points of handwritten digit strings using a neural network. In: Proceedings of the ICDAR, pp 142–146
23. Vellasques E, Oliveira LS, Sabourin R, Britto AS, Koerich AL (2006) Modeling segmentation cuts using support vector machines. In: Proceedings of the IWFHR, pp 41–46
24. Pal US, Datta S (2003) Segmentation of Bangla unconstrained handwritten text. In: Proceedings of the ICDAR, pp 1128–1132
25. Zahour A, Taconet B, Ramdane S (2004) Contribution la segmentation de textes manuscrits anciens. In: Proceedings of the CIFED 2004, La Rochelle
26. Chen YA, Leedham G (2005) Independent component analysis segmentation algorithm. In: Proceedings of the ICDAR, pp 680–684
27. Bruzzone E, Coffetti MC (1999) An algorithm for extracting cursive text lines. In: Proceedings of the ICDAR, 20–22 September 1999, pp 749–752
28. Aouadi N, Kacem A, Belaïd A (2014) Segmentation of touching component in Arabic manuscripts. In: Proceedings of the ICFHR, 1–4 September 2014, pp 452–457
29. Aouadi N, Amiri S, Kacem A (2013) Segmentation of connected component in Arabic handwritten documents. In: Proceedings of the CIMTA, vol 10. Elsevier, New York, pp 738–746
30. Aouadi N, Amiri S, Kacem A (2013) Segmentation of touching components in Arabic handwritten documents. In: Proceedings of the CIMTA, 27–28 September 2013, pp 738–746
31. The Java Tutorial: Thread Pools. http://www.math.unihamburg.de/doc/java/tutorial/essential/threads/group.html. Accessed 14 Feb 2005
32. Canny J (1986) A computational approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–698
33. Jonker R, Volgenant A (1987) A shortest augmentating path algorithm for dense and sparse linear assignment problem. J Comput 38(4):325–340
34. Chui H (2001) Non-rigid point matching: algorithms, extensions and applications. PhD dissertation, Yale University
35. Gatos B, Stamatopoulos N, Louloudis G (2009) ICDAR2009 handwriting segmentation contest. In: Proceedings of the ICDAR, 26–29 July 2009, 1393–1397