IJDAR (2013) 16:127–137
DOI 10.1007/s10032-012-0185-9

ORIGINAL PAPER

Handwritten digit segmentation: a comparative study

F. C. Ribas · L. S. Oliveira · A. S. Britto Jr. · R. Sabourin

Received: 6 July 2010 / Revised: 13 February 2012 / Accepted: 9 March 2012 / Published online: 28 March 2012

Abstract In this work, algorithms for segmenting handwritten digits based on different concepts are compared by evaluating them under the same conditions of implementation. A robust experimental protocol based on a large synthetic database is used to assess each algorithm in terms of correct segmentation and computational time. Results on a real database are also presented. In addition to the overall performance of each algorithm, we show the performance for different types of connections, which provides an interesting categorization of each algorithm. Another contribution of this work concerns the complementarity of the algorithms. We have observed that each method is able to segment samples that cannot be segmented by any other method, and do so independently of their individual performance. Based on this observation, we conclude that combining different segmentation algorithms may be an appropriate strategy for improving the correct segmentation rate.

1 Introduction

In spite of the efforts made over the past two decades, the recognition of handwritten digit strings is still an open problem. One of the main bottlenecks in this kind of system is the segmentation module, which reads a string of characters (usually digits, but sometimes non-digits can appear, in bank check processing systems, for example) and segments them into isolated characters. The main problem is a lack of context, that is, usually we do not know the number of characters in the string and so the optimal boundary between them is unknown.

Segmentation algorithms can be divided into two classes: segmentation–recognition, and recognition-based [2]. In the former, the segmentation module provides a single sequence hypothesis where each sub-sequence should contain an isolated character, which is submitted to the recognizer. In the latter, the algorithm yields a list of segmentation hypotheses and then assesses each of them through the recognition process. The literature shows that this kind of approach produces good results, but is computationally expensive, since all the hypotheses generated must be evaluated. Moreover, the recognition module has to discriminate different patterns, such as fragments, isolated characters, and connected characters. In this strategy, segmentation can be explicit or implicit. In the explicit methods, segmentation is performed prior to recognition, which producing candidate characters for the recognizer. In contrast, in the implicit methods, segmentation is embedded in the recognition process and is performed simultaneously with recognition (Fig. 1).

In recent years, several algorithms have been proposed for explicit segmentation. They normally take into consideration a set of heuristics and information of the foreground [7,10,12,18,23], background [4,13,16], or a combination of these, [3,15] in order to generate potential segmentation cuts. The main drawbacks of most of these algorithms are the large number of cuts, which must be evaluated by the recognition algorithm, and the number of heuristics that must be set. A way to reduce the number of segmentation cuts has been proposed by Vellasques et al. [22].

F. C. Ribas · A. S. Britto Jr.
Pontifical Catholic University of Parana (PUCPR),
R. Imaculada Conceição, 1155, Curitiba, PR 80215-901, Brazil

L. S. Oliveira (✉)
Federal University of Parana (UFPR), Rua Cel. Francisco H. dos Santos, 100, Curitiba, PR 81531-990, Brazil
e-mail: lesoliveira@inf.ufpr.br

R. Sabourin
Ecole de Technologie Superieure, 1100 rue Notre Dame Ouest, Montreal, QC, Canada

Fig. 1 Variability of connections between handwritten digits

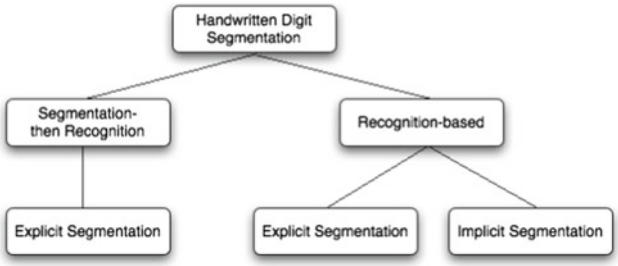

Fig. 2 Methods for segmentation and recognition of digit strings (adapted from [18])

In order to avoid explicit segmentation and the complexity of setting several heuristics, some authors have tried implicit segmentation to recognize strings of digits [1,17]. The literature has shown that explicit segmentation has achieved better results, but implicit segmentation offers very interesting perspectives. The main drawback of implicit segmentation is its high sensitivity to slanted images, which makes the use of a pre-processing step obligatory, in order to correct the slant. Figure 2 depicts the various methods for the segmentation and recognition of digit strings.

In reviewing the literature, we find various algorithms for handwritten digit segmentation. A direct comparison, though, is not a trivial task. On the contrary, it may not even be feasible. The main obstacle in this case is the database used during the experiments, which can range from specific forms, to the bank checks of different countries (Canada, France, Brazil, USA, etc.), to subsets of publicly available databases such as NIST-SD19, CEDAR, and so on. Moreover, the number of data used can range from a few hundred to thousands. Also, the computational cost, which can be expressed by the number of segmentation hypotheses produced by the algorithm, is very often neglected. In some cases, especially in real-time applications, this is a very important issue that can determine the success or failure of an handwriting recognition system.

In this work, we compare various explicit segmentation algorithms. In order to avoid implementing a huge number of algorithms, we selected those we deemed the most different, in terms of the features and ideas used to produce the segmentation cuts. We then assess them for performance, number of segmentation hypotheses produced, and processing time.

Regarding performance, we are interested not only in the global performance, but also the performance of each algorithm on the different types and locations of connections. We believe that this categorization will be a useful contribution to the development of more intelligent segmentation algorithms. For example, if we know a priori the type of connection we are dealing with, we can choose the most suitable segmentation algorithm. To make this kind of investigation possible, we created a database of 79,966 images of touching pairs, and manually labeled them with respect to the type and location of the connection. These images were extracted from the synthetic database proposed in [14], and they represent realistic scenarios of connected handwritten digits where cursive writing is usually involved. To demonstrate the usability of the synthetic data for research purposes, we applied the segmentation algorithms on a real database, composed of 2,369 images of connected pairs extracted from NIST SD19. We show that the results on both real and synthetic data are very similar.

The second aspect we investigate here concerns computational cost. Very often, promising results are reported in the literature, but important details, such as the number of segmentation cuts produced, the number of heuristics, and the amount and complexity of the features used to yield the cuts, are omitted. Besides this comparative study, we also show that the use of different features provide a high degree of complementarity which can be used to build more reliable segmentation systems. The experimental results show that all the algorithms combined can produce a correct segmentation rate of 99.57 % considering the ideal case based on an oracle.

This paper is structured as follows. Section 2 surveys several different segmentation algorithms reported in the literature. Section 3 presents the database in which the algorithms were assessed. Section 4 shows the assessment methodology we have applied. Section 5 reports all the experimental results and discusses them. Finally, Sect. 6 concludes this work.

2 Segmentation algorithms

In this section, we review several segmentation algorithms proposed in the literature in the last few years. We show how the algorithms fit into the taxonomy depicted in Fig. 2 and report the number of images used for testing and their accuracy. More aspects of the algorithms are reported at the end of this section in Table 1. We believe they provide a good coverage of the main underlying algorithmic approaches.

Fujisawa et al. [9] propose a recognition-based algorithm that detects all the connected components (CC) of the image and classifies each of them as an isolated digit or into a string of digits. This classification is based on the horizontal length of the CC. In other words, if the horizontal length of the CC is greater than a threshold T, then it is considered as a string of digits ($\geq$2). To segment touching digits, the algorithm first splits the contour information into upper and lower contours. Then it computes an approximate measure of vertical width and assigns potential segmentation points to those locations where this measure exceeds a given threshold h_t.

Table 1 Comparison of different segmentation algorithms

References	Primitives	Ligatures	Pre-proc	Pre-class	≥2	OverSeg	Approach	Data	Size	Perf. (%)
[8]	Contour	No	No	No	Yes	No	Rec-based	ZIP codes	450	94.9
[9]	Contour	No	No	Yes	No	Yes	Rec-based	Proprietary	920	95.0
[4]	Skeleton (background)	No	No	Yes	No	Yes	Seg-then-Rec	Proprietary	120	80.8
[20]	Contour	No	Yes	No	No	No	Seg-then-Rec	USPS	212	94.2
[19]	Contour	Yes	No	No	No	Yes	Seg-then-Rec	CEDAR	1,966	80.8
[13]	Skeleton (background)	No	No	No	No	No	Seg-then-Rec	NIST, proprietary	3,355	92.5
[3]	Skeleton (background, foreground)	Yes	Yes	No	No	No	Seg-then-Rec	NIST, proprietary	4,500	96.0
[23]	Contour, concavities	No	Yes	No	Yes	No	Seg-then-Rec	NIST SD19	3,287	94.8
[15]	Contour profile	No	No	No	Yes	Yes	Rec-based	Brazilian bank cheques	900	98.5
[10]	Contour, concavities	No	Yes	Yes	No	No	Rec-based	NIST SD19	3,500	92.5
[16]	Reservoir	No	No	Yes	No	No	Seg-then-Rec	French bank cheques	2,250	94.8
[6]	Skeleton, contour	Yes	Yes	No	No	No	Seg-then-Rec	NIST, CEDAR, proprietary	–	96.0
[11]	Contour	No	Yes	Yes	Yes	Yes	Rec-based	NIST SD19	3,359	97.72
[21]	Skeleton	Yes	Yes	No	No	No	Seg-then-Rec	NIST SD19	2,000	88.7
[18]	Skeleton (background, foreground)	Yes	Yes	Yes	Yes	Yes	Rec-based	NIST SD19	5,000	96.5

This algorithm treats images with touching loops differently, such as "0–0", "8–8", and "0–9." It divides the inner loops into two groups (left and right) and computes the distance between them. If this distance is greater than a threshold D, the algorithm produces a segmentation cut.

All segmentation cuts are produced using line segments connecting the segmentation points. Thereafter, all segmentation points are used to build a segmentation graph. The best segmentation hypothesis is the shortest path of the graph, which is computed using some thresholds on the size of the CCs. The authors have tested their algorithm on a proprietary dataset composed of 46 most frequent touching pairs. Twenty samples per class were considered summing up 920 images. Pairs of digits with touching multiple times were not used. The authors reported a correct segmentation rate of 95 %.

The method proposed by Shi and Govindaraju [19] consists of segmentation followed by recognition, based on the observation that touching points and ligatures between two digits reveal that the chaincode contour makes significant right turns at each point of touching. The segmentation cuts are then defined by the most significant right turn points, along with their opposite contour point. The method assumes that the touching pair is free of slant. The final segmentation point is then determined using a vertical histogram and a set of heuristics. This method was evaluated on a database of 1,966 images of pairs of digits from the CEDAR database. The authors reported a correct segmentation rate of 78 %.

Fenrich and Krishnamoorthy [8] used a very simple recognition-based algorithm based on two primitives, namely, vertical histogram projection and contour information (peaks and valleys). These primitives inspired other authors [11,12,15,20]. The algorithm first attempts to segment the string using vertical histogram projection. The column with the minimal value becomes a candidate for a vertical split through the component. If the minimal value is larger than a stroke width threshold or the candidate cut crosses two or more vertical runs, the cut is aborted. If the histogram produces no segmentation cuts, then the second part of the algorithm uses upper and lower contours of the components to define the segmentation cuts. If a peak of the lower contour and a valley of the upper contour can be connected with a line segment that satisfies slope thresholds, then a piecewise linear split is made. If they cannot be connected, then a straight line is used to produce the segmentation cut. The authors reported a correct segmentation rate of 94.9 % on 450 images of ZIP Codes.

An alternative approach to segmenting touching digits has been presented by Chen and Wang in [3]. The algorithm they describe also uses a segmentation/recognition approach, theirs combining background and foreground analysis to

segment handwritten numeral strings that touch single or multiple times. The foreground and background regions on the image of connected numeral strings are first thinned, and the feature points on foreground and background skeletons are extracted. Several possible segmentation paths are then constructed and ligatures are removed. Finally, the parameters of the geometric properties of every possible segmentation path are determined, and these parameters are analyzed by a mixture of Gaussian probability functions to decide on the best segmentation path or rejection of a path. Similar approaches can be found in [4,13]. The authors used 4,178 images from the NIST SD19 database and another 322 proprietary images. The main drawback of this kind of approach is the computational cost, since both background and foreground skeletons should be created. The authors report a correct segmentation rate of 96 % on 4,500 images of the NIST Database.

The segmentation/recognition algorithm presented by Yu and Yan [23] uses the morphological structural technique to segment strings of digits. The preprocessing step involves smoothing, linearization and detection of the structural points of the image contours, which are used to define the segmentation cuts. If a string contains more than two numerals, the region of the left two numerals of this string is determined first. In this way, the process of separating a string consisting of more than two numerals is reduced to that of separating a string of two numerals. After a separation, the separated string is processed with the same method until there is no region left that contains at least two numerals. The algorithm is implemented with a huge set of rules and a considerable number of heuristics. The algorithm was assessed on 3,287 images extracted from the NIST database. The correct segmentation rate ranges from 85 to 95 %, depending on the string length. Another approach using similar features is presented by Kim et al. [10].

The method proposed by Pal et al. [16] is similar to the previous methods, in that, it first classifies the image into isolated or touching digits. It also uses a segmentation/recognition strategy. Since they use background information, no preprocessing is necessary. The rationale behind this algorithm is that, when two digits touch each other, they create a large space, also known as a "reservoir," between the digits. According to the authors, such a space is very important because it concentrates the extraction of cutting points essentially around the reservoir, reducing the search area. First, the positions and sizes of the reservoirs are analyzed and a reservoir is detected where touching occurs. Considering the type (top or bottom reservoir) and its features, the touching position (top, middle, or bottom) is ascertained. Then, based on the touching position and the analysis of the profile of the reservoir, the initial feature points for segmentation are determined. Considering closed loops, reservoir heights, and the distance from the component center, the initial feature points are ranked and the best feature point (highest ranking point) is noted. Finally, based on touching position, closed loop positions and the morphological structure of the touching region, the cutting path is generated. As a result, the algorithm provides one segmentation cut for a touching pair. The algorithm was evaluated on a database of 2,250 images extracted from a French bank cheque database. The performance reported was a 94.8 % rate of correct segmentation.

Elnagar and Alhajjb [6] designed a segmentation/recognition algorithm to split pairs of digits that takes a binary image as input and then applies normalization, preprocessing, and thinning processes prior to segmentation. The authors argue that, although thinning is computationally expensive, it is essential to obtain uniform stroke width that simplifies the detection of feature points. Since all thinning algorithms create spurious points, a noise reduction technique is needed to filter out some of these points. Segmentation is performed using features extracted from the skeleton and contour. A set of heuristics is defined to determine the most probable segmentation cuts. As a result, the algorithm produces a single segmentation cut. The author tested their algorithm on images from the CEDAR and NIST databases, and reported a correct segmentation rate of 96 %. The number of images used in the tests was not mentioned.

Suwa and Naoi [21] proposed a segmentation/recognition algorithm to segment strings of digits, which takes as input the skeleton of the image. From this skeleton, edges and vertices are extracted and the pattern is represented as a connected graph. Potential segmentation points are located based on the peaks and valleys of the upper and lower parts of the skeleton. The segmentation path is computed through graph theory techniques and heuristic rules. The algorithm is designed to detect and remove ligatures using the algorithm described in [6]. According to the authors, the segmented digits have a more natural shape than can be achieved using algorithms that split patterns using straight lines or line segments. Experimental results on 2,000 images from the NIST SD19 database were used in their experiments, and they achieved a correct segmentation rate of 88.7 %.

Sadri et al. [18] describe a combination of a recognition-based algorithm and a genetic algorithm. After generating various segmentation hypotheses, the search algorithm attempts to identify the most suitable one according to a predefined fitness function. Before detecting the segmentation points, the algorithm classifies the connected components into three classes: parts of digits, isolated digits, or pairs of digits. A connected component is considered a touching digit if it is considerably larger than higher. Segmentation cuts are generated based on the skeleton. Both the foreground and the background skeleton are used to construct the segmentation paths. Thereafter, all segmentation hypotheses are combined into a segmentation graph, and a genetic algorithm is used to search among all the possible outputs

of such a graph. The authors report a performance of 96.5 % on 5,000 touching pairs extracted from the NIST SD19 database.

Table 1 summarizes the segmentation algorithms discussed in this section. The following aspects are considered:

- Primitives: main primitives used to build segmentation hypotheses.
- Ligatures: identifies and removes ligatures between digits.
- Pre-processing: involves all the tools used before segmentation, such as smoothing, thinning, normalization, slant correction.
- Pre classification: identifies algorithms that detect if a given image contains a single digit or touching digits. If this step fails, two different kind of error may occur, that is, an attempt was made to segment an isolated digit, or a string of digits lacked segmentation points.
- ≥2: indicates that the authors are reporting results for strings of digits with more than two digits.
- Over segmentation: algorithm that produces several segmentation hypotheses.
- Approach: Recognition or Segmentation/Recognition.
- Data: Database considered.
- Size: Number of images used in the experiments.
- Perf: Correct segmentation rate reported.

3 Database

The database used in this work was first introduced by Oliveira et al. in [14]. It was generated based on 2,000 isolated digits extracted from the hsf_0 series of NIST SD19. The main goal of this database was to provide a common catalog for evaluating segmentation algorithms, but, in the long run, it could be useful for training a segmentation-free system like the one proposed in [5]. It is important to mention that the 2,000 images used to create it were correctly recognized by the classifier described in [15], a multi-layer perceptron that uses a 132-dimensional feature vector based on concavities and contour information. This issue is relevant for assessing the segmentation, and it will be further discussed in subsequent sections. The algorithm responsible for building the synthetic database is very simple, and is based on two rules:

1. It connects only digits produced by one writer. The information about the writer is provided in NIST SD19. Fifty different writers were considered.
2. The reference axis along which the digits slide is the center line.

The aim of these rules is to avoid unreasonable connections (e.g., very small digits connected to very big ones) and

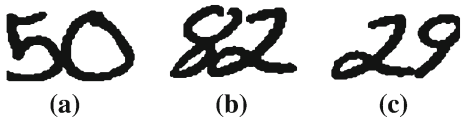

Fig. 3 Samples extracted from the synthetic database

Fig. 4 Example of the ground truth generated by the algorithm. **a** The ground truth information, **b** the starting and ending points

Category	Type	Style of touching	Examples
Single-touching	I		
	II		
	III		
	IV		
Multiple-touching	V		

Fig. 5 Types of connected numeral strings proposed in [3]

Fig. 6 False ligature

make the synthetic data more real. As depicted in Fig. 3, the touching pairs represent realistic scenarios of connected handwritten digits where cursive writing is usually involved.

In addition to the image, the algorithm also produces its ground truth, which contains the label of the image, and the starting and ending coordinates of the optimal segmentation paths ($P = \{p_1, p_2, \ldots, p_n\}$). Figure 4a shows an example of the ground truth file. The first line indicates the label of the image, while the second and third indicate the starting and ending coordinates of the optimal segmentation paths (p_1 and p_2 in this case). Figure 4b shows the points in the corresponding image.

Besides, the image were labeled to identify the type of touching, following the classification technique proposed by Chen and Wang [3] (Fig. 5). However, the algorithm proposed in [14] does not generate samples with ligatures (type IV), since they do not belong to the digit itself.

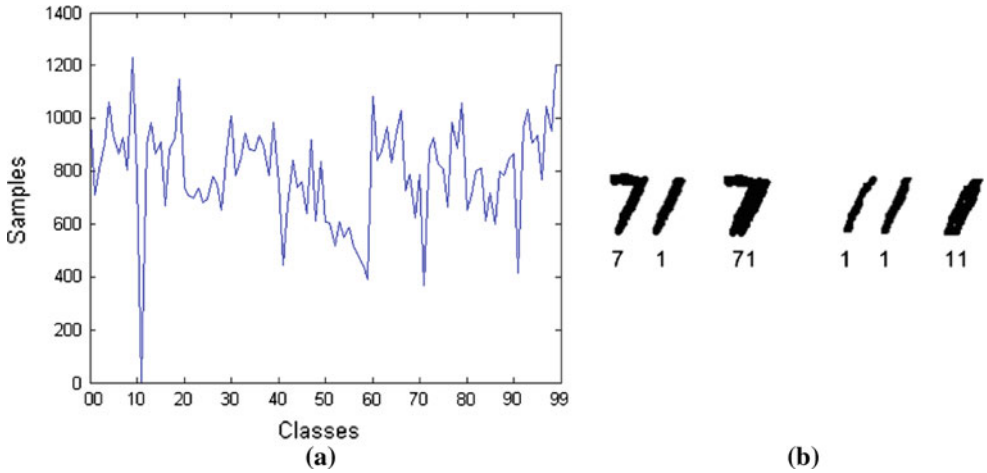

Fig. 7 **a** Database distribution and **b** problems faced when connecting the digit "1"

Table 2 Distribution of the database regarding the type of connection

Connection type	%
I	34.8
II	53.6
III	1.6
V	10.0

Fig. 8 Segmentation points: **a** ground truth with six segmentation points and three segmentation paths and **b** segmentation path created by a segmentation algorithm and both segmented digits

In very few cases, though, do we have anything resembling a ligature, as depicted in Fig. 6. As we can see, this stroke is not really a ligature, since it clearly belongs to the digit "0". After connecting the two digits, it looks like a ligature. The few cases where this was observed were labeled as type 1 connections.

Figure 7a shows how the 79,966 samples in the database are distributed into the 100 classes of touching pairs, which correspond to the possible combinations of two digits. Some of the classes involving the digit 1 still contain fewer samples than other classes. Owing to the American style of handwriting, the digit 1 is very often with the other digit in the pair. Figure 7b illustrates this problem with samples that were manually removed from the database.

Table 2 shows the distribution of the database based on the type of connection. This database is available upon request for research purposes.[1]

4 Evaluation methodology

As reported in Sect. 2, several different segmentation algorithms have been proposed in the literature.

The ideal comparative study would involve having access to the source code of all the algorithms. We tried that without success, however. Some algorithms were developed a long time ago and others were developed by companies with no interest, for obvious reasons, in sharing their source code. Instead of comparing all the algorithms found in the literature, we decided to implement those that use different features or strategies to generate the segmentation points. In this context, the algorithms selected were the ones proposed by Fujisawa et al. [9], Shi and Govindaraju [19], Fenrich and Krishnamoorthy [8], Chen and Wang [3], Pal et al. [16], and Elnagar and Alhajajj [6]. It is worth noting that the algorithms were implemented based on the information provided by the authors in their respective publications. Very often the heuristics used by the algorithms are not discussed in the works, however, and in those cases, we defined them empirically using a validation set composed of 500 images. The remaining 79,466 were used for testing.

4.1 Assessment criteria

As stated before, we are interested in comparing the performance of the algorithms in terms of correct segmentation and also computational cost. Since we have the coordinates of all the optimum segmentation cuts, it seems obvious that we should compare the segmentation points produced by the algorithms with the ground truth. However, depending on the way this comparison is performed, we may face problems.

[1] http://web.inf.ufpr.br/vri/touching-digits.

Consider, for example, the touching pair depicted in Fig. 8a. In this case, the ground truth is composed of six segmentation points (three segmentation paths). Now, a segmentation algorithm can produce a single segmentation path (Fig. 8b) that is able to split the 2-digit string into two isolated digits. In spite of the fact that this segmentation is different from the ground truth, both digits can be classified as the number 3.

It is clear, therefore, that using only the ground truth information is not ideal for assessing the segmentation cuts. To overcome this problem, we used a classifier to recognize the segmented pieces.

To guarantee that the digits are correctly classified when the segmentation algorithm produces the best segmentation path (the closest to the ground truth), we used only isolated digits that are correctly classified by the classifier to build the database of touching digits. In addition, we performed a visual inspection to guarantee that errors caused by the OCR engine are not introduced at this level. In this analysis, we checked for both Type I and Type II errors. In the first case, we looked for instances where the classifier gave the incorrect recognition for the correct segmentation, while in the second case, we looked for cases where the classifier gave the correct recognition for the incorrect segmentation.

We are interested in knowing whether or not the segmentation cuts produced by the algorithms are good ones, independently of their number. For the algorithms based on the segmentation/recognition approach, this task is straightforward, since there is only one hypothesis to be assessed. For those algorithms based on over segmentation, we have to evaluate all the cuts. If we find two digits among the hypotheses (using classification) corresponding to the ground truth, we consider that the segmentation was successful. If we send only the best (the closest to the ground truth) segmentation cuts for classification, that is, the ones closest to the ground truth, we will risk classification issues, since digits produced by a bad segmentation cut can be recognized with a higher probability/score by the classifier than those generated by the correct (ground truth) segmentation cut. We believe that the assessment strategy adopted here is fair, since the deficiencies of the classifier will not penalize the segmentation algorithms.

Regarding cost, the metric used was the computational time. Since we implemented all the algorithms using the same coding standard and the tests were performed on the same hardware, we believe this is a valid metric that can provide good insight into the complexity of each algorithm. Even if we do not assess all the segmentation cuts, in the case of over segmentation, we provide the number of segmentation cuts produced by those algorithms, which is the important information for recognition-based systems.

5 Experiments and discussion

All the algorithms were implemented in C++, and the experiments were performed on a PC with an Intel Core 2 Duo, a speed of 1.6 Ghz, 2 Gb of RAM, and Ubuntu Linux 8.04. With respect to the computational time, in this work, we are interested only on the time spent to produce the segmentation cuts. In the following paragraphs, we present some implementation details, as well as the performance of each algorithm. For all the algorithms, we report the overall performance, as well as the performance for each type of segmentation, remembering that Type IV connections are not considered in the database. In all the experiments, the number of digits contained in each piece of test data is assumed to be known.

The method presented by Fujisawa et al. [9] classifies the image into isolated or touching digits before segmentation. Since we have only touching digits in our database, this step was skipped. The algorithm uses a threshold called H_x to identify touching-region candidates. In our experiments, we used $H_x = 17$ (the value defined empirically on the validation set). The overall performance of this algorithm was 89.85 %, and it achieved 95.45, 91.27, 83.57, and 63.72 % for Type I, II, III, and V connections respectively. It generates 3.66 ($\pm$0.6) segmentation cuts in 0.4 ms, on average, which makes it one of fastest of the algorithms we implemented. Incidentally, the authors argue that this algorithm was not designed to segment Type V connections.

The worst overall performance was achieved by the method proposed by Shi and Govindaraju [19]. In this work, the authors use a threshold (THR) to identify a significant right turn, and hence the potential for segmentation. In our experiments, THR = 75 was used (a value defined empirically on the validation set). The performance for Type I, II, III, and V connections were 68.31, 59.72, 60.35, and 25.44 % respectively. The average performance of this algorithm was 59.30 %. The average time to produce a segmentation cut is 1.2 ms.

The algorithm described by Fenrich and Krishnamoorthy [8] produces 4.07 ($\pm$0.5) segmentation cuts in 3.9 ms, on average. In spite of its simplicity, this algorithm achieves an interesting overall performance of 92.37 %. The performance for Type I, II, III, and V connections was 97.54, 93.79, 99.45, and 65.57 % respectively. However, it is surpassed by other segmentation algorithms for certain connection types.

The segmentation algorithm proposed by Chen and Wang [3] produces the best overall performance, but it is the most expensive in terms of computational time and the number of segmentation hypotheses. It generates, on average, 45.4 ($\pm$24) segmentation cuts per image in about 74.8 ms. Unlike the other algorithms discussed so far, the computational time varies considerably, depending on the type of connection. For Type V, for example, it takes about 100 ms on average. Besides, this method makes extensive use of the

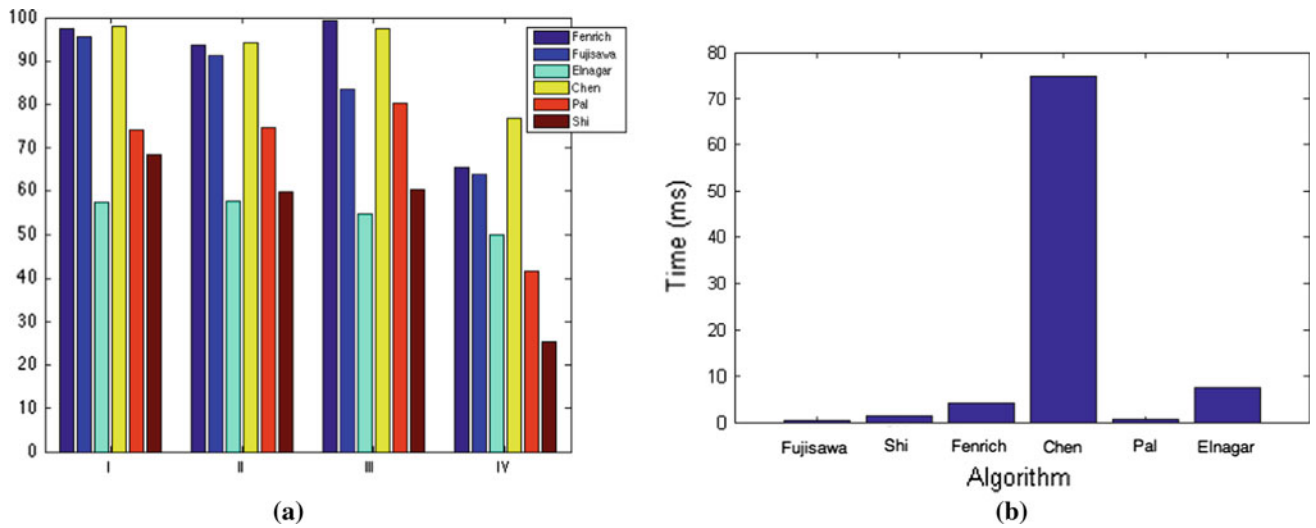

Fig. 9 **a** Performance of the segmentation algorithms on different connection types and **b** average processing time

Table 3 Summary of the segmentation algorithms

Method	Performance (%)	Connection type (%)				Segmentation cuts	Time (ms)
		I	II	III	IV		
Fusijawa et al. [9]	89.85	95.45	91.27	83.57	63.72	3.66	0.4
Shi and Govindaraju [19]	59.30	68.31	59.72	60.35	25.44	1	1.2
Fenrich and Krishnamoorthy [8]	92.37	97.54	93.79	99.45	65.57	4.07	3.9
Chen and Wang [3]	93.80	97.87	94.23	97.55	76.76	45.40	74.8
Pal et al. [16]	71.21	73.96	74.69	80.09	41.52	1	0.7
Elnagar and Alhajajj [6]	67.34	63.88	71.51	56.40	58.73	1	7.5

skeletonization process (background and foreground), which contributes to the high computational cost. In order to select the best segmentation hypothesis, the authors trained a mixture of Gaussians using 823 images, although this filtering process was not implemented in our experiments. The Chen and Wang algorithm achieves the best overall performance of 93.80 %, and its performance for Type I, II, III, and V connections was 97.87, 94.23, 97.55, and 76.76 % respectively.

The most innovative set of features developed recently was proposed by Pal et al. [16]. The authors use the concept of the reservoir. The method is quite fast, finding the optimal segmentation point in about 0.7 ms on average. Like the method presented by Fujisawa et al. [9], this method also classifies the image into isolated or touching digits prior to segmentation. As before, this process was not considered. The average performance of this algorithm was 71.21 %, while the performance for connections of Type I, II, III, and V was 73.96, 74.69, 80.09, and 41.52 % respectively.

The algorithm proposed by Elnagar and Alhajajj [6] achieved an overall performance of 67.54 %. It attempted to find the segmentation points using the skeleton of the image. To do so, they used 32 different configurations of a mask, and this has a considerable impact on the computational time (7.5 ms to segment an image). The authors argue that this process can be parallelized, though. The performance for connections of Type I, II, III, and V was 63.88, 71.51, 56.40, and 58.73 % respectively.

In Fig 9a, the performances of all the algorithms. for each connection type are compared, and in Fig. 9, their average processing times are compared. Table 3 summarizes these algorithms.

As stated before, recognition-based algorithms such as the ones described in [3,8,9] produce better , but their processing time should be multiplied by the number of segmentation hypotheses, which increases exponentially as a function of the number of segmentation cuts. Figure 10 exemplifies this problem: The segmentation algorithm produces four cuts, leading to 15 classifier calls.

For this reason, some authors have investigated the use of filters to reduce the number of segmentation hypotheses [22]. As mentioned before, Chen and Wang [3] trained a mixture of Gaussians to reduce the huge number of segmentation hypotheses created by their algorithm. Of course, there is a cost involved in filtering these points, but it should be considerably less expensive than using the classifier to assess all the segmentation hypotheses.

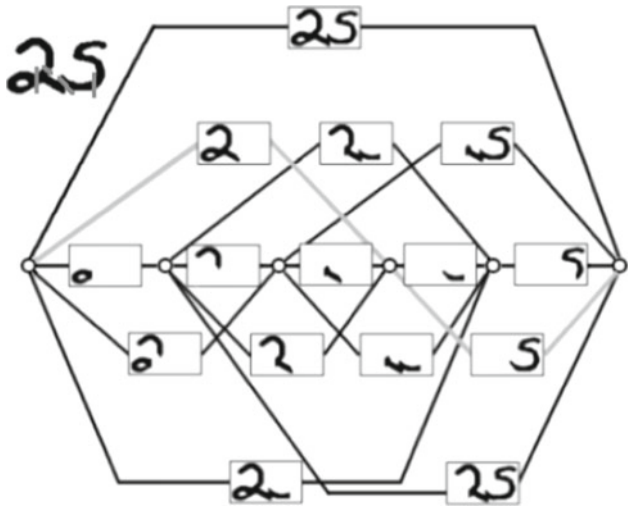

Fig. 10 Segmentation hypotheses created by four segmentation cuts

To put this into context, consider the algorithm proposed by Chen and Wang [3]. Using their filter, which ultimately provides only one segmentation hypothesis, that is, the classifier has to evaluate only two digits. For the sake of simplicity, we are abstracting all the complexity and processing time of such a filter. Consider also the algorithm proposed by Fenrich and Krishnamoorthy [8] which is considerably faster than the one proposed in [3]. However, the former yields about four (3.6) segmentation cuts, which means 15 classifier calls as against two yielded by the latter. Considering the classifier used in our work, which takes about 28 ms (using the hardware specified above) to classify a given image, the final processing time required to segment and classify a connected pair of digits would be 423 ms and 130 ms for Fenrich and Krishnamoorthy and Chen and Wang respectively. Of course, the classification process can be parallelized and the classifier optimized. Nevertheless, this is an important issue that should be considered during specification of a recognition system.

So far we have compared the algorithms in terms of performance and cost. However, there is another facet of these results that can be explored, which is combining the segmentation algorithms. As we can see in Fig. 9a and Table 3, some algorithms are better than others, depending on the connection type.

In this context, Table 4 shows the number of images correctly segmented given a number of algorithms. For example, "2/6" means that 2,976 images are correctly segmented by only two of the six segmentation algorithms used. The label "0/6" means that no algorithm was able to segment 344 images (i.e., 0.43 % of the images). Table 4 presents this analysis for each type of connection.

In the last line of Table 4, the label "Oracle" means that at least one algorithm was able to segment a given image. As we can see, by combining all segmentation algorithms, we can reach a correct segmentation rate of 99.57 %, which is

Table 4 Images correctly segmented given a number of algorithms

	Total	%	Type I	%	Type II	%	Type III	%	Type V	%
0/6	344	0.43	3	0.01	178	0.42	0	0.00	163	2.05
1/6	1,233	1.55	37	0.13	535	1.26	2	0.16	659	8.30
2/6	2,976	3.74	257	0.93	1,173	2.75	24	1.90	1,522	19.18
3/6	7,083	8.91	1,748	6.32	3,281	7.70	105	8.29	1,949	24.56
4/6	16,090	20.25	5,765	20.85	8,050	18.89	327	25.83	1,948	24.55
5/6	27,749	34.92	10,784	38.99	15,190	35.65	483	38.15	1,292	16.28
6/6	23,991	30.19	9,062	32.77	14,202	33.33	325	25.67	402	5.07
Oracle	79,122	99.57	27,653	99.99	42,431	99.58	1,266	100	7,772	97.95

Table 5 Distribution of the images that were correctly segmented by only one algorithm ("1/6")

Algorithm	Total	%	Connection types							
			I	%	II	%	III	%	V	%
Fenrich and Krishnamoorthy [8]	236	19.14	8	21.62	130	24.30	2	100	96	14.57
Chen and Wang [3]	337	27.33	12	32.43	129	24.11	0	0	196	29.74
Fujisawa et al. [9]	269	21.82	14	37.84	82	15.33	0	0	173	26.25
Pal et al. [16]	44	3.57	1	2.70	16	2.99	0	0	27	4.10
Shi and Govindaraju [19]	67	5.43	2	5.41	26	4.86	0	0	39	5.92
Elnagar and Alhajajj [6]	280	22.71	0	0.00	152	28.41	0	0	128	19.42
Total	1,233	100	37	100	535	100	2	100	659	100

Table 6 Distribution of the database regarding the type of connection

Connection type	%	
	Synthetic	Real
I	34.8	57.7
II	53.6	31.7
III	1.6	6.1
IV	–	3.0
V	10.0	1.5

Comparison between synthetic and real data

Table 7 Performance of the segmentation algorithms on the touching pairs extracted from NIST SD19

Algorithm	Correct segmentation (%)
Fujisawa et al. [9]	88.9
Shi and Govindaraju [19]	62.3
Fenrich and Krishnamoorthy [8]	96.9
Chen and Wang [3]	96.8
Pal et al. [16]	82.3
Elnagar and Alhajajj [6]	72.3

considerably better than the best algorithm. This result shows that the algorithms rely on complementary features and therefore can be further combined to yield a more reliable segmentation. Concerning complementarity, Table 5 details the line "1/6" of Table 4 where the images were correctly segmented by only one algorithm.

Table 5 allows us to observe that, independently of their individual performances, each algorithm is responsible for correctly segmenting a specific subset of images of the database. Another interesting point worth noting is that, even though it achieves the worst individual performance, the algorithm proposed by Elnagar and Alhajajj [6] performs well for a subset of images. Of the 1,233 images correctly segmented by only one algorithm, this one is responsible for 22.71 % of them. As in the classification task, where weak classifiers can be used to build more reliable ensembles, the experiments reported here show that combining segmentation algorithms can also contribute to building more efficient handwriting recognition systems.

The results presented in Tables 4 and 5 can be seen as an oracle, in the sense that if a suitable combination scheme is employed, this would be the upper limit for this database using these algorithms. This combination can be made to be static, by cascading all the algorithms, or dynamic, by using the information about the nature of the touching in the pairs to select the best algorithm. Of course, this is not a trivial task owing to the huge variability of the algorithms; however, it could be an interesting way to create more reliable and efficient segmentation modules.

After analyzing the experiments reported so far, one question arises: Do the synthetic data present the same degree of segmentation difficulty that we find in real databases? To answer this question, we designed an experiment where the six algorithms were used to segment 2,369 images of touching pairs extracted from the well-known NIST SD 19 database. Table 6 shows the distribution of the connection types for the real data. For the sake of comparison, we show the numbers presented in Table 2 as well. As stated before, the synthetic data does not have Type IV connection.

The results of the correct segmentation are reported in Table 7. As we can see, the performance achieved in this dataset is similar to the performance reported in Table 3. The slightly better performance achieved on the real dataset by most algorithms can be justified by the fact that it contains more images of Type I, which are easier to segment.

6 Conclusion

In this work, we have compared various segmentation algorithms of the explicit type. We selected those algorithms that we deemed most different, in terms of the features and ideas used to produce the segmentation cuts. Then, these algorithms were assessed for performance and cost using the same experimental protocol on 79,966 synthetic images and 2,369 real images extracted from the NIST_SD19 database. The proposed evaluation criteria provided the global performance of each algorithm, as well as their performance on four different types of connections. The experimental results show that these algorithms achieve similar performances on both databases, which qualifies the synthetic dataset as a viable alternative for benchmarking segmentation algorithms.

During the evaluation, we observed that, independently of the overall performance, each method is able to segment some samples that cannot be segmented by any other method. It corroborates the argument that even a method with low overall performance can contribute to building a more reliable segmentation system.

As we have demonstrated, this kind of analysis also constitutes a useful contribution to identifying complementarity among the segmentation algorithms, which can be used to develop more intelligent systems. The main challenge in building such an intelligent system lies in the correct identification of the connection types, which certainly is not a trivial task.

Acknowledgments This research has been supported by The National Council for Scientific and Technological Development (CNPq) grant 301653/2011-9.

References

1. Britto, A.S., Sabourin, R., Bortolozzi, F., Suen, C.Y.: The recognition of handwritten numeral strings using a two-stage HMM-based method. IJDAR **5**, 102–117 (2003)
2. Casey, R., Lecolinet, E.: A survey of methods and strategies in character segmentation. IEEE Trans. PAMI **18**(7), 690–706 (1996)
3. Chen, Y.K., Wang, J.F.: Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis. IEEE Trans. Pattern Anal. Mach. Intell. **22**(11), 1304–1317 (2000)
4. Cheriet, M., Huang, Y.S., Suen, C.Y.: Background region based algorithm for the segmentation of connected digits. In: Proceedings of the 11th International Conference on Pattern Recognition, pp. 619–622 (1992)
5. Choi, S., Oh, I.: A segmentation-free recognition of two touching numerals using neural networks. In: Proceedings of 5th International Conference on Document Analysis and Recognition (ICDAR), pp. 253–256, Bangalore, India (1999)
6. Elnagar, A., Alhajj, R.: Segmentation of connected handwritten numeral strings. Pattern Recognit. **36**(3), 625–634 (2003)
7. Fenrich, R.: Segmentation of automatically located handwritten words. In: Proceedings of 4th IWFHR, pp. 33–44 (1991)
8. Fenrich, R., Krishnamoorthy, S.: Segmenting diverse quality handwritten digit strings in near real-time. In: Proceedings of 5th USPS Advanced Technology Conference, pp. 523–537 (1990)
9. Fujisawa, H., Nakano, Y., Kurino, K.: Segmentation methods for character recognition: from segmentation to document structure analysis. Proc. IEEE **80**, 1079–1092 (1992)
10. Kim, K.K., Kim, J.H., Suen, C.Y.: Segmentation-based recognition of handwritten touching pairs of digits using structural features. Pattern Recognit. Lett. **23**(1), 13–21 (2002)
11. Lei, Y., Liu, C.S., Ding, X.Q., Fu, Q.: A recognition based system for segmentation of touching handwritten numeral strings. In: Proceedings of 9th International Workshop on Frontiers of Handwriting Recognition (IWFHR), Tokyo, Japan (2004)
12. Lethelier, E., Leroux, M., Gilloux, M.: An automatic reading system for handwritten numeral amounts on french checks. In: Proceedings of 3rd International Conference on Document Analysis and Recognition, pp. 92–97 (1995)
13. Lu, Z., Chi, Z., Siu, W., Shi, P.: A background-thinning-based approach for separating and recognizing connected handwritten digit strings. Pattern Recognit. **32**, 921–933 (1999)
14. Oliveira, L.S., Britto, A.S. Jr., Sabourin, R.: A synthetic database to assess segmentation algorithms. In: Proceedings of 8th International Conference on Document Analysis and Recognition, pp. 207–211 (2005)
15. Oliveira, L.S., Sabourin, R., Bortolozzi, F., Suen, C.Y.: Automatic recognition of handwritten numerical strings: a recognition and verification strategy. IEEE Trans. Pattern Anal. Mach. Intell. **24**(11), 1438–1454 (2002)
16. Pal, U., Belaid, A., Choisy, C.: Touching numeral segmentation using water reservoir concept. Pattern Recognit. Lett. **24**, 261–272 (2003)
17. Procter, S., Elms, A.J.: The recognition of handwritten digit strings of unknown length using hidden markov models. In: Proceedings of 14th International Conference on Pattern Recognition, pp. 1515–1517 (1998)
18. Sadri, J., Suen, C.Y., Bui, T.D.: A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings. Pattern Recognit. **40**, 898–919 (2007)
19. Shi, Z., Govindaraju, V.: Segmentation and recognition of connected handwritten numeral strings. Pattern Recognit. **30**(9), 1501–1504 (1997)
20. Strathy, N.W.: A method for segmentation of touching handwritten numerals. Master's thesis, Concordia University, Montreal, Canada, Sept 1993
21. Suwa, M., Naoi, S.: Segmentation of handwritten numerals by graph representation. In: Proceedings of 9th International Workshop on Frontiers of Handwriting Recognition (IWFHR), Tokyo, Japan (2004)
22. Vellasques, E., Oliveira, L.S., Britto, A.S. Jr., Koerich, A., Sabourin, R.: Filtering segmentation cuts for digit string recognition. Pattern Recognit. **41**(10), 3044–3053 (2008)
23. Yu, D., Yan, H.: Separation of touching handwritten multi-numeral strings based on morphological structural features. Pattern Recognit. **34**(3), 587–598 (2001)