Assessing Handwitten Digit Segmentation Algorithms

Felipe Ribas[†], Luiz S. Oliveira[‡], Alceu S. Britto Jr.[‡], and Robert Sabourin[+]

[†]Pontifícia Universidade Católica do Paraná (PUCPR) - Brazil
{felipe,alceu}@ppgia.pucpr.br

[‡]Universidade Federal do Paraná (UFPR) - Brazil
lesoliveira@inf.ufpr.br

[+]Ecole de Technologie Superieure (ETS) - Canada
robert.sabourin@etsmtl.ca

ABSTRACT

This work compares different segmentation algorithms for handwritten digits based on explicit segmentation. For this purpose, algorithms based on different concepts were implemented and evaluated under the same conditions. The algorithms were used to segment 2,369 pairs of touching digits of the NIST_SD19 database and were evaluated in terms of correct segmentation and computational time. We also discuss the complementarity of the segmentation algorithms. We have observed that independently of the individual performance of the algorithms, each method is able to segment samples that can not be segmented by any other method. Based on this observation, we conclude that the combination of different segmentation algorithms may be an interesting strategy to improve the correct segmentation rate.

Categories and Subject Descriptors

H.4 [**Pattern Recognition**]: Miscellaneous; D.2.8 [**Doc. Engineering**]: Handwriting Segmentation—*document analysis*

Keywords

Handwriting Segmentation

1. INTRODUCTION

In spite of the efforts made during last two decades, the recognition of handwritten digit strings still is an open problem. One of the main bottlenecks in this kind of system is the segmentation module, which consists in reading a string of characters and segmenting it into isolated characters. The main problem is the lack of context, i.e., usually we do not know the number of characters in the string, hence, the optimal boundary between them is unknown.

SAC'12 March 26-30, 2012, Riva del Garda, Italy.

Figure 1: Variability of connections.

Segmentation algorithms can be divided into two classes: segmentation-then 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. The literature shows that this kind of approach produces good results, but it is computationally expensive since all hypotheses generated must be evaluated. Segmentation can also be classified into explicit or implicit. In the explicit methods, segmentation is performed prior to recognition producing candidate characters for the recognizer. On the other hand, in the implicit methods segmentation is embedded in the recognition process and it is performed simultaneously with recognition.

In the case of explicit segmentation several algorithms have been proposed during the last years. They normally take into consideration a set of heuristics and information of the foreground [6, 9, 11, 20, 15], background [4, 12, 14], or a combination of them [3, 13] in order to generate potential segmentation cuts. The main drawbacks of most of these algorithms are the elevated number of cuts, which must be evaluated by the recognition algorithm, and the number of heuristics that must be set. An alternative to reduce the number of segmentation cuts was proposed by Vellasques et al [19].

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]. 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 the high sensibility to slanted images, which makes it obligatory the use of a pre-processing step to correct the slant.

Reviewing the literature, one can find different algorithms for handwritten digit segmentation. A direct comparison, though, is not a trivial task. The main obstacle in this case is the database used during the experiments, which can range

from specific forms, bank cheques of different countries, subsets of public databases such as NIST_SD19, CEDAR, and so on. Besides, the amount of data used can range from few hundreds to thousands. Another point that is very often neglected is the computational cost, which can be expressed by the number of segmentation hypotheses produced by the algorithm. In some cases, especially for real time applications, this is a very important issue that can determine the success or fail of an handwriting recognition system.

In this work we compare several different segmentation algorithms based on explicit segmentation. In order to avoid implementing a huge number of algorithms we selected those we deemed most different regarding the features and ideas used to produce the segmentation cuts. Then, these algorithms are assessed in terms of performance, number of segmentation hypotheses produced, and processing time. The algorithms were tested on 2,369 pairs of touching digits of the NIST_SD19 database.

The second aspect we investigate here regards the 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.

2. SEGMENTATION ALGORITHMS

In this section we review several segmentation algorithms proposed in the literature in the last few years. More aspects of the algorithms are reported in Table 1. We believe they provide a good coverage of the main underlying algorithmic approaches.

Fujisawa et al [8] propose a recognition-based algorithm that detects all the connected components (CC) of the image and classifies each of them into isolated digit or a string of digits. This classification is based on the horizontal length of the CC. To segment touching digits, the algorithm first splits the contour information into upper and lower contours. Then it computes an approximated measure of vertical width and assigns potential segmentation points to those locations where this measure exceeds a given threshold h_t. This algorithm treats differently those images that have touching loops, 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 multiple touching were not used. The authors reported a correct segmentation rate of 95%.

The method proposed by Shi and Govindaraju [16] is a segmentation-then recognition based on the observation that touching points and ligatures between two digits reveals that at each touching point, the chain code contour makes significant right turns. The segmentation cuts are then defined by the most significant right turn points together 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 composed of 1,966 images of touching digits from the CEDAR database. The authors reported a correct segmentation rate of 78%.

Fenrich and Krishnamoorthy [7] used a simple recognition-based algorithm based on two primitives, namely, vertical histogram projection and contour information (peaks and valleys). These primitives inspired other authors [10, 11, 13, 17]. The algorithm first tries 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 them 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 for segmenting touching digits was presented by Chen and Wang in [3]. The algorithm described by them uses a segmentation-then recognition approach that combines background and foreground analyses to segment single- or multiple-touching handwritten numeral strings. Thinning of both foreground and background regions are first processed on the image of connected numeral strings 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 geometric properties of each possible segmentation paths are determined and these parameters are analyzed by the mixture Gaussian probability function to decide the best segmentation path or reject it. Similar approaches can be found in [12, 4]. The authors used 4,178 images from the NIST_SD19 database and other 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 reported a correct segmentation rate of 96% on 4,500 images of NIST Database.

Yu and Yan [20] present a segmentation-then recognition algorithm based on morphological structural technique to segment strings of digits. The pre-processing step involves smoothing, linearization, and detection of structural points of 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, a separation process of a string which consists of more than two numerals is reduced to that of a string which consists of two numerals. After a separation, the separated string is processed in 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 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 [9].

Similar to the previous methods, the one proposed by Pal et al [14] first classifies the image into isolated or touching digits. It also takes a segmentation-then recognition strategy. Since they used 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 "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, thus reducing the search area. Firstly, the positions and sizes of the reservoirs are analyzed and a reservoir is detected where touching is made. Considering the type (top or bottom reservoir) and its features, the touching position (top, middle, or bottom touching) is decided. 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 rank point) is noted. Finally, based on touching position, closed loop positions and morphological structure of the touching region the cutting path is generated. As result, the algorithm provides one segmentation cut for a touching pair. The algorithm was evaluated on a database of 2,250 images extracted from French bank cheques and the performance reported was 94.8% of correct segmentation.

Elnagar and Alhajjb [5] designed a segmentation-then recognition algorithm to split pairs of digits which takes a binary image as input and then applies normalization, preprocessing, and thinning processes before 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 is necessary to filter 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 result, the algorithm produces a single segmentation cut. The author tested their algorithm on images from 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 [18] proposed a segmentation-then recognition algorithm to segment string 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 valleys and peaks 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 same algorithm described in [5]. 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, which achieved a correct segmentation rate of 88.7%.

Sadri et al [15] described recognition-based algorithm along with a genetic algorithm. After generating different segmentation hypotheses, the search algorithm tries to identify the one most suitable according to a pre-defined fitness function. Before detecting the segmentation points, the algorithm classifies the connected components into three classes: part of digit, isolated digit, or pair 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 foreground and 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 reported 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 (regards all the tools used before segmentation, such as smoothing, thinning, normalization, slant correction, etc.), Pre-classification (uses some kind of preclassification prior segmentation), ≥ 2 (indicates if the authors report results for strings of digits with more than two digits), Over-segmentation (algorithm that produces several segmentation hypotheses), Approach (Recognition-based or Segmentation-then recognition), Data (Database considered), Size (Number of images used in the experiments), Perf (Correct segmentation rate reported)

3. EVALUATION METHODOLOGY

The ideal comparative study would be having access to the source code of all algorithms. We have tried that with no success. Some algorithms were developed long time ago and others were developed by companies who have no interest, for evident reasons, in sharing their source code. Instead of comparing all the algorithms found in the literature, we have 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 [8], Shi and Govindaraju [16], Fenrich and Krishnamoorthy [7], Chen and Wang [3], Pal et al [14], and Elnagar and Alhajajj [5]. It is worth of mentioning that the algorithms were implemented based on the information provided by the authors in their respective publications.

To properly assess the results of segmentation we performed a visual inspection. In this analysis we have checked both Type I and II errors. In the first case we checked when the classifier gives the wrong recognition for the correct segmentation while in the second case we checked when the classifier gives the correct recognition for the wrong segmentation. Independently of the number of segmentation cuts produced by the algorithms, we are interested in knowing if the good ones are among them. For the algorithms based on the "Segmentation-then-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 among the hypotheses we find two digits (using classification) corresponding to the ground truth, we consider that the segmentation was successful.

Regarding the cost, the metric used was the computational time. Since we have implemented all the algorithms using the same coding standard and the tests were per-

Table 1: Comparison of different segmentation algorithms.

Ref.	Primitives	Ligatures	Pre-Proc	Pre-Class	≥ 2	OverSeg	Approach	Data	Size	Perf.
[7]	Contour	No	No	No	Yes	No	Rec-Based	Zip Codes	450	94.9%
[8]	Contour	No	No	Yes	No	Yes	Rec-Based	Proprietary	920	95.0%
[4]	Skeleton	No	No	Yes	No	Yes	Seg-Then-Rec	Proprietary	120	80.8%
[17]	Contour	No	Yes	No	No	No	Seg-Then-Rec	USPS	212	94.2%
[16]	Contour	Yes	No	No	No	Yes	Seg-Then-Rec	CEDAR	1966	80.8%
[12]	Skeleton	No	No	No	No	No	Seg-then-Rec	NIST,Proprietary	3355	92.5%
[3]	Skeleton	Yes	Yes	No	No	No	Seg-then-Rec	NIST,Proprietary	4500	96.0%
[20]	Contour, Concavities	No	Yes	No	Yes	No	Seg-then-Rec	NIST SD19	3287	94.8%
[13]	Contour Profile	No	No	No	Yes	Yes	Rec-Based	Brazilian Bank Cheques	900	98.5%
[9]	Contour, Concavities	No	Yes	Yes	No	No	Rec-based	NIST SD19	3500	92.5%
[14]	Reservoir	No	No	Yes	No	No	Seg-then-Rec	French Bank Cheques	2250	94.8%
[5]	Skeleton, Contour	Yes	Yes	No	No	No	Seg-then-Rec	NIST, CEDAR, Proprietary	-	96.0%
[10]	Contour	No	Yes	Yes	Yes	Yes	Rec-Based	NIST SD19	3359	97.72%
[18]	Skeleton	Yes	Yes	No	No	No	Seg-then-Rec	NIST SD19	2000	88.7%
[15]	Skeleton	Yes	Yes	Yes	Yes	Yes	Rec-Based	NIST SD19	5000	96.5%

formed on the same hardware, we believe this is a valid metric that can provide a good insight about the complexity of each algorithm. Even not assessing all the segmentation cuts, in the case of over-segmentation, we also provide the number of segmentation cuts produced by those algorithms, which is an important information for recognition-based systems.

4. EXPERIMENTS AND DISCUSSION

All the algorithms were implemented in C++ and the experiments performed on a PC equipped with an Intel Core 2 Duo, 1.6Ghz, 2Gb RAM and Ubuntu Linux 8.04. Regarding 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. In all experiments the number of digits contained in each test data is assumed to be known.

The method presented by Fujisawa et al [8] 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 have used $H_x = 17$. The overall performance of this algorithm was 88.9%. It generates in average 3.9 ($\pm$ 0.6) segmentation cuts on 0.39ms, which makes it one of fastest algorithms implemented.

The worst overall performance, 62,3%, was achieved by the algorithm proposed by Shi and Govindaraju [16]. In this case, the authors use a threshold (THR) to determined significant right turn, hence, potential segmentation cuts. In our experiments, THR = 75 was used. The average time to produce the segmentation cut is 1.31ms.

The algorithm described by Fenrich and Krishnamoorthy [7] produces 3 ($\pm$ 0.5) segmentation cuts in 4.7ms in average. In spite of its simplicity, this algorithm reaches the best overall performance, 96.9%.

The segmentation algorithm proposed by Chen and Wang [3] is the most expensive in terms of computational time and number of segmentation hypotheses. It generates in average 45.4 ($\pm$ 24) segmentation cuts per image in about 74.8ms. Besides, this method makes strong use of skeletonization process (background and foreground) which collaborate to the high computational cost. In order to select the best segmentation hypothesis, the authors trained a mixture of Gaussians using 823 images. Such a filtering process was not implemented in our experiments. This algorithm reaches 96.83% of correct segmentation rate. Similar to Fenrich and Krishnamoorthy [7] approach, but with a higher computational cost.

The most innovative set of features developed recently was proposed by Pal et al [14]. The authors use the concept of reservoir. The method is quite fast finding the optimal segmentation point in about 0.7ms in average. Like the method presented by Fujisawa et al [8], this method also classifies the image into isolated or touching digits before segmentation. Like before, this process was not considered. The average performance of this algorithm was 82.3%.

The method proposed by Elnagar and Alhajajj [5] tries to find the segmentation points using the skeleton of the image. To do so, they have used 32 different configurations of a masks which have a considerable impact on the computational time (7.5ms to segment an image). The authors argue that this process can be parallelized, though. The overall performance was 72.3%.

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, i.e., the combination of the segmentation algorithms. By analyzing the segmentation results we have observed that some images that are not segmented by the best algorithms very often are correctly segmented by other algorithms. Considering all the algorithms together, only one image (Figure 2) was not correctly segmented.

Table 2: Summary of the segmentation algorithms.

Method	Performance	Seg. Cuts	Time (ms)
Fusijawa et al [8]	88.9	3.9	0.39
Shi [16]	62.3	1	1.2
Fenrich [7]	96.9	4.07	3.9
Chen and Wang [3]	96.8	1	74.8
Pal et al [14]	82.3	1	0.7
Elnagar [5]	72.3	1	7.5

Figure 2: Image not segmented.

This combination can be seen like 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. It can be static by cascading all the algorithms or even dynamic, by using the information about the nature of the touching to select the best algorithm. Of course this is not a trivial task due to the huge variability, however this could be an interesting way to create more reliable and efficient segmentation modules.

5. CONCLUSION

In this work we have compared different segmentation algorithms based on explicit segmentation. We selected those algorithms we deemed most different regarding the features and ideas used to produce the segmentation cuts. Then, these algorithms were assessed in terms of performance and cost by using the same experimental protocol. During the evaluation, we have observed that independently of the overall performance, each method is able to segment some samples that can not be segmented by any other method. It corroborates to the argument that even a method with low overall performance can contribute to build a more reliable segmentation system. As we have demonstrated this kind of analysis is also an useful contribution to identify complementarity among the segmentation algorithms, which can be used to develop more intelligent segmentation algorithms.

6. REFERENCES

[1] A. S. Britto, R. Sabourin, F. Bortolozzi, and C. Y. Suen. The recognition of handwritten numeral strings using a two-stage HMM-based method. *IJDAR*, 5:102–117, 2003.

[2] R. Casey and E. Lecolinet. A survey of methods and strategies in character segmentation. *IEEE Trans. on PAMI*, 18(7):690–706, 1996.

[3] Y. K. Chen and J. F. Wang. Segmentation of single- or multiple-touching handwritten numeral string using background and foreground analysis. *IEEE Trans. on PAMI*, 22(11):1304–1317, 2000.

[4] M. Cheriet, Y. S. Huang, and C. Y. Suen. Background region based algorithm for the segmentation of connected digits. In *11^{th} International Conference on Pattern Recognition*, pages 619–622, 1992.

[5] A. Elnagar and R. Alhajj. Segmentation of connected handwritten numeral strings. *Pattern Recognition*, 36(3):625–634, 2003.

[6] R. Fenrich. Segmentation of automatically located handwritten words. In *4^{th} IWFHR*, pages 33–44, 1991.

[7] R. Fenrich and S. Krishnamoorthy. Segmenting diverse quality handwritten digit strings in near real-time. In *5^{nd} USPS Advanced Technology Conference*, pages 523–537, 1990.

[8] H. Fujisawa, Y. Nakano, and K. Kurino. Segmentation methods for character recognition: from segmentation to document structure analysis. *Proc. of IEEE*, 80:1079–1092, 1992.

[9] K. K. Kim, J. H. Kim, and C. Y. Suen. Segmentation-based recognition of handwritten touching pairs of digits using structural features. *Pattern Recognition Letters*, 23(1):13–21, 2002.

[10] Y. Lei, C.S. Liu, X. Q. Ding, and Q. Fu. A recognition based system for segmentation of touching handwritten numeral strings. In *9^{th} Int. Workshop on Frontiers of Handwriting Recognition*, 2004.

[11] E. Lethelier, M. Leroux, and M. Gilloux. An automatic reading system for handwritten numeral amounts on french checks. In *3^{rd} ICDAR*, pages 92–97, 1995.

[12] Z. Lu, Z. Chi, W. Siu, and P. Shi. A background-thinning-based approach for separating and recognizing connected handwritten digit strings. *Pattern Recogniton*, 32:921–933, 1999.

[13] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Y. Suen. Automatic recognition of handwritten numerical strings: A recognition and verification strategy. *IEEE Trans. on Pattern Analysis on Machine Intelligence*, 24(11):1438–1454, 2002.

[14] U. Pal, A. Belaid, and Ch. Choisy. Touching numeral segmentation using water reservoir concept. *Pattern Recognition Letters*, 24:261–272, 2003.

[15] J. Sadri, C. Y. Suen, and T. D. Bui. A genetic framework using contextual knowledge for segmentation and recognition of handwritten numeral strings. *Pattern Recognition*, 40:898–919, 2007.

[16] Z. Shi and V. Govindaraju. Segmentation and recognition of connected handwritten numeral strings. *Pattern Recognition*, 30(9):1501–1504, 1997.

[17] N. W. Strathy. A method for segmentation of touching handwritten numerals. Master's thesis, Concordia University, Montreal - Canada, Setember 1993.

[18] M. Suwa and S. Naoi. Segmentation of handwritten numerals by graph representation. In *9^{th} Int. Workshop on Frontiers of Handwriting Recognition*, 2004.

[19] E. Vellasques, L. S. Oliveira, A. S. Britto Jr, A. Koerich, and R. Sabourin. Filtering segmentation cuts for digit string recognition. *Pattern Recognition*, 41(10):3044–3053, 2008.

[20] D. Yu and H. Yan. Separation of touching handwritten multi-numeral strings based on morphological structural features. *Pattern Recognition*, 34(3):587–598, 2001.