# A Synthetic Database to Assess Segmentation Algorithms

Luiz S. Oliveira, Alceu S. Britto Jr., and Robert Sabourin
Pontifícia Universidade Católica do Paraná, Curitiba, Brazil
Ecole de Technologie Supérieure - Montreal, Canada
soares@ppgia.pucpr.br

## Abstract

*In this paper we describe a synthetic database composed of 273,452 handwritten touching digits pairs to assess segmentation algorithms. It contains several different kinds of touching and it was generated by connecting 2,000 images of isolated digits extracted from the NIST SD19. In order to get a better insight on the proposed database and establish some parameters for further comparisons, we carried out experiments using four state-of-the-art segmentation algorithms.*

**Keywords:** Handwriting Recognition, Segmentation, Synthetic Data.

## 1 Introduction

Automatic reading of numerical fields has been successfully attempted in several application areas due to the availability of relatively inexpensive CPU power, and the possibility to reduce the manual effort involved in this task. This kind of system usually is composed of several steps, such as image acquisition, preprocessing, segmentation, representation, and recognition. One of the main bottlenecks is the segmentation step, which consists in reading a string of characters (usually digits, but sometimes non-digits can appear, e.g., systems to read bank cheques) and segmenting it into isolated characters. The main problem lies in the fact that usually we do not know the number of characters in the string, hence, the optimal boundary between them is unknown.

Some authors have devoted efforts in order to build more robust segmentation algorithms, which can be divided into two classes: segmentation-then-recognition (sometimes called segmentation-free) and segmentation-based recognition. 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. This technique shows its limits rapidly when the correct segmentation does not fit with the pre-defined rules of the segmenter. Very often, contextual information is used during the segmentation process to improve the robustness of the system. In the latter, the algorithm yields a list of segmentation hypotheses and then assess 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. Moreover, the recognition module has to discriminate various configurations such as fragments, isolated characters and connected characters. In this strategy, segmentation can be explicit when based on cut rules or implicit when each pixel column is a potential cut location.

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, 7, 8, 13], background [4, 9, 11], or a combination of them [3, 10] 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. 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.

In spite of all efforts made in this direction, it is difficult to effectively assess the performance of the algorithms. The main reason lies in the fact that in order to evaluate new algorithms, authors use different databases or even different subsets of the same database, which can have different levels of complexity. We can cite for example the NIST SD19, which

contains strings ranging from 2- to 10-digits. This database does not provide a directory of touching digit strings, so that each author must extract them to build his own database. Therefore, if we scan the literature from 1992 until nowadays, we can find several works using NIST database reporting the most different levels of accuracy.

Another problem found in works describing new segmentation algorithms is the lack of information about the number of segmentation cuts generated. It is not difficult to find algorithms working fine under several different conditions, but the cost for that (strong set of heuristics, great number of segmentation cuts, etc.) is very often omitted. Such an information is vital, especially when dealing with a segmentation-based recognition.

In this paper we describe a synthetic database of touching digits to be used in the assessment of segmentation algorithms. It was generated by connecting isolated handwritten digits of NIST SD19 database. It is composed of 273,452 images of 2-digit strings and contains several different kinds of touching. In addition, we carried out experiments using four different state-of-the-art algorithms on this dataset.

## 2 The Database

In a recent paper, Bunke [2] discussed the past, present, and future of handwriting recognition, and pointed out some recent trends that have emerged recently. According to him, and the authors agree on that, one of the trends in handwriting recognition concerns the generation of synthetic data for training and testing systems. Most of the works in this vein regard the generation synthetic data through distortions and shape variations in order to increase training or testing sets. In this work we address a different use for synthetic data, i.e., algorithm evaluation. To the best of our knowledge, no similar data have been built so far. Choi and Oh [5] did some efforts towards this, but their goal was to generate samples of touching pairs to train a hundred classifiers (one for each possible touching pair) in order to avoid explicit segmentation.

The database introduced here contains 273,452 images of touching pairs (300 dpi, bi-tonal) and it was generated based on 2,000 isolated digits of NIST SD19 extracted from hsf_0 series. Wang et al [12] presented statistics indicating that 85% of the touching occur between two consecutive numerals while the remaining occurs among three or more consecutive numerals. Besides, most of the algorithms in the literature deal with the problem of 2-digit strings. For this reason, we decide to focus in a first moment in building a database of touching pairs. Regarding the style of touching, 89% of images contains single-touching pairs, while the remaining 11% contains multiple-touching pairs. Figure 1 presents some images extracted from the database.

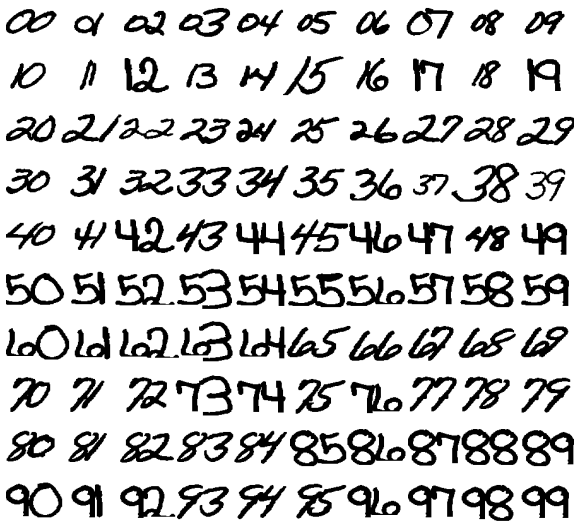

**Figure 1. Some images extracted from the synthetic database.**

In most of the existing databases such as NIST, CEDAR, and CENPARMI, the occurrence frequencies are severely unbalanced. We have noticed and Choi and Oh [5] corroborate that samples from some classes such as 89, 20, 44, 56, and 69 are abundant while the ones from classes 10, 11, 47, and 81 are rare. This could be a problem when one wants to learn something from the data. The idea of generating this database is to assess segmentation algorithms, but in the long run it could be useful to train a segmentation-free system like in [5]. For this reason we have considered a uniform distribution to generate the data. However some of the classes involving the digit 1 still contain less samples than other classes. Due to the US handwriting style, very often the digit 1 is fused into the other digit. Figure 2 illustrates this problem, which were manually removed from the database.


**Figure 2. Problems faced when connecting the digit "1".**

The algorithm responsible for building the synthetic database is very simple and it is based on two rules:

1. It connects just the digits coming from the same writer. The information about the writer is provided in the NIST SD19. Fifty different writers were considered.

2. The reference axis along which they slide is the center line.

The aim of those rules is to avoid unreasonable connections (e.g., very small digits connected to very big ones) and make the synthetic data more real. Figure 3a shows some examples of the original data extracted from NIST SD19, while Figure 3b depicts the different styles of touching available in the synthetic database.

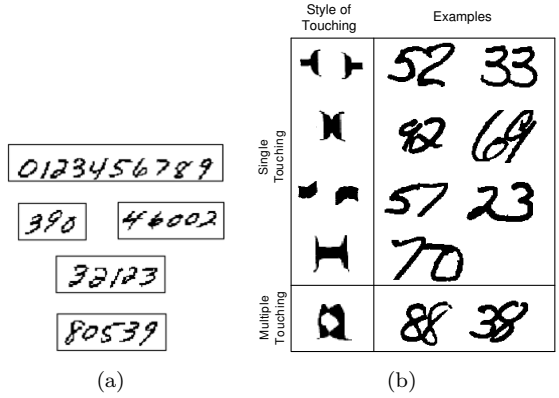


**Figure 3. Examples of NIST SD19 fields (a) Original fields and (b) Touching pairs generated by the algorithm.**

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, and the remaining indicates 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.

A common way of verifying the performance of a given segmentation algorithm consists in using a classifier to evaluate the segmented pieces. However, it is difficult to measure, especially in a huge database, whether the error is due to the segmentation failure or due to a mistraining of the classifier. For a given image $I$, most of the segmentation algorithms generated

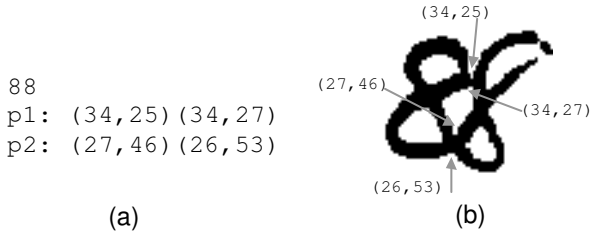


**Figure 4. Example of the ground truth generated by the algorithm (a) The ground truth information (b) The starting and ending points**

a set of potential cut $S = \{s_1, s_2, \ldots, s_n\}$. Theoretically, an image $I$ could be considered correctly segmented if $S \subset P$. However, a perfect match is very difficult since different algorithms can produce slightly different segmentation cuts. Figure 5 exemplifies this problem where the segmentation cut is different from the optimal one, but still segments correctly the image.

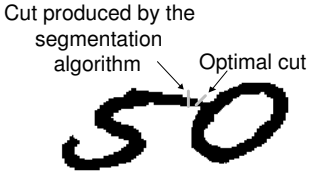


**Figure 5. The ground truth region**

In order to avoid such problems and be able to use a classifier to assess the segmentation, the 2,000 isolated digit images were selected so that they would be correctly classified by a neural network recognizer [?]In this way, if after the segmentation process the digits are not recognized anymore, we can conclude that the segmentation is not correct. Using this approach we can surpass the problem depicted in Figure 5, and still focus just on segmentation.

## 3 Segmentation Algorithms

In order to get a better insight on the proposed database and establish some parameters for further comparisons, we have used four state-of-the-art algorithms to segment these data. The choice of the algorithms was based on the features they use to generate the segmentation cuts. The first algorithm takes into account just the points of the contour and profile to produce the cuts. This algorithm was first proposed

by Fenrich [6] and successfully used by Lethelier et al into system to recognize numerical amount of French bank cheques [8]. The second algorithm, proposed by Oliveira et al [10], is complementary to the first one in the sense that it adds some features extracted from the skeleton in order to improve the quality of the segmentation cuts. Differently to the first two, the third algorithm considers both foreground and background of the image to generate the segmentation cuts [3]. The last algorithm considered in this study is based on the concept of water reservoir, and it was proposed by Pal et al [11].

## 4 Experiments

Table 1 reports the performance of the aforementioned algorithms on the proposed database. Since 89% of the database is composed of single-touching pairs, Table 1 also reports the performance on single- and multiple-touching pairs separately. The segmentation is deemed correct if the recognizer classifies correctly both digits of the image. Since our focus in this study is the segmentation, we evaluated just the best segmentation cut produced by the algorithms, i.e., the cut closest to the ground truth. We want to assess if the algorithms can produce segmentation cuts close to the optimal ones. It is a matter of fact that sometimes a classifier can produce high scores for wrong segmentation hypotheses, but this is out of the scope of this work.

**Table 1. Performance of the segmentation algorithm.**

| Algorithm | Correct Segmentation (%) | | |
|---|---|---|---|
| | Global | Single | Multiple |
| 1 | 87.0 | 95.0 | 27.0 |
| 2 | 89.5 | 97.5 | 27.0 |
| 3 | 88.0 | 92.0 | 60.0 |
| 4 | 61.0 | 65.0 | 29.8 |

The best performance for single-touching images was achieved by the algorithm #2, while the best performance for multiple touching was achieved by the algorithm #3. However, the global performance of the algorithms #1, #2, and #3 is similar. Among the four algorithms, the #4 is the only one that tries to identify whether the image contains connected numerals or not. It first classifies the image into connected, confused, or isolated and then it tries to segment just those classified as connected. The performance reported by the authors in [11] is very good and it was computed in a database of 1189 touching pairs extracted from French cheques. It seems that the thresholds presented in [11] do not work well in the proposed database since a considerable part of the database was classified as isolated or confused, hence, the segmentation process was skipped. We have realized that it occurs very often when the this algorithm faces a touching pair with the digit "1". In such cases, the image is very often classified as isolated.

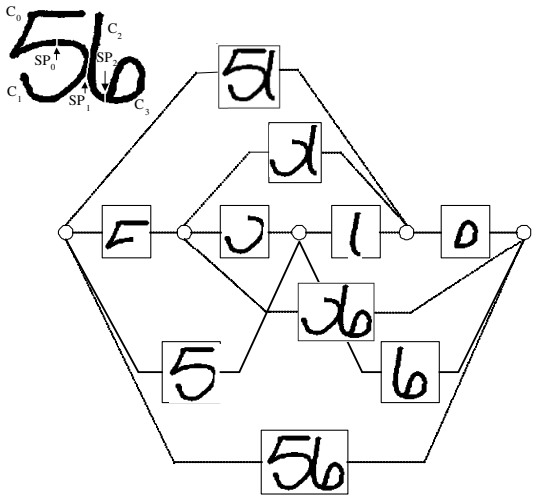


**Figure 6. Example of a segmentation graph.**

All the algorithms discussed here, except algorithm #4, can be classified as heuristic over-segmentation, where the basic idea is to segment the image as much as necessary to produce the optimal segmentation cuts. As stated before, this approach gives good results, but it is computationally expensive since all hypotheses generated must be evaluated. This is a very important issue that has been ignored very often in the literature. Figure 6 shows an example of an image segmented by the algorithm #1 and its corresponding segmentation graph. We can see that the algorithm produced three segmentation cuts ($SP_0$, $SP_1$, and $SP_2$) and four sub-images ($C_0$, $C_1$, $C_2$, and $C_3$). In the case of segmentation-based recognition, the classifier must be invoked ten times (one for each sub-image of the graph) in order to produce a score for each sub-image and then the path that maximizes the score is supposed to represent the optimal segmentation cut.

The algorithm #4 produces just one segmentation cut, i.e., a single sequence hypothesis where each subsequence should contain an isolated character. This strategy reduces the cost involved in calling the recognizer, but its limits appears when the correct segmentation does not fit with the pre-defined rules of the algorithm. Table 2 reports the average number of

sub-images generated by the three over-segmentation algorithms as well as the average number of times the recognizer must be invoked.

If performance is the only criterion adopted to choose a segmentation algorithm, then based on Table 1 the algorithm #2 should be picked. As discussed previously, the algorithm #2 is more complex than the first one since it uses characteristics form the skeleton of the image in addition to the contour and profile. Consequently, it can segment more efficiently slanted images. On the other hand, the number of sub-images yielded by this algorithm and the number of classifier calls is quite bigger. The algorithm #3 is much better for segmenting multiple-touching images, but it is very expensive.

**Table 2. Average of the number of sub-images generated by the over-segmentation algorithms.**

| Algorithm | Number of Sub-images | | Recognizer |
|---|---|---|---|
| | Average | Std. Dev. | Calls |
| 1 | 3.2 | 1 | 10 |
| 2 | 5.0 | 1.5 | 18 |
| 3 | 7.3 | 2.2 | 66 |

In spite of the relatively inexpensive CPU power, the cost of segmentation algorithms still is a big issue in the field of handwritten numeral string recognition. Real systems, for example, have to process a numerical amount in about one second. In light of this, a cheaper algorithm such as the #1, would be preferable than the others discussed here.

## 5 Conclusion and Future Works

In this work we have described a database to assess segmentation algorithms but that could be used for other purposes as well. So far we have considered just touching pairs, but we plan to improve this database generating string with 3, 4, 5, and 6 digits. As soon as we finish generating all these data, we plan to make it available to the handwriting recognition community.

We also performed several experiments using different segmentation algorithms. In addition to the performance of the algorithms we have address their cost, which is very important when dealing with segmentation-based recognition applied to real systems.

For future works we plan to extract some knowledge of the synthetic database in order to reduce the number of segmentation cuts produced by the over-segmentation algorithms.

## Acknowledgements

This research has been supported by The National Council for Scientific and Technological Development (CNPq) grant 150542/2003-8. The authors would like to thank Felipe Ribas and Leonardo Prates for their important contribution in this work.

## 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] H. Bunke. Recognition of cursive roman handwriting: Past, present, and future. In $7^{th}$ *ICDAR*, pages 448–459, 2003.

[3] Y. K. Chen and J. F. Wang. Segmentation of single- or multiple-touching handwritten numeral string using background and foregound 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 $11^{th}$ *ICPR*, pages 619–622, 1992.

[5] S.-M. Choi and I.-S. Oh. A segmentation-free recognition of handwritten touching numeral pairs using modular neural networks. *IJPRAI*, 15(6):949–966, 2001.

[6] R. Fenrich. Segmentation of automatically located handwritten words. In $2^{nd}$ *IWFHR*, pages 33–44, 1991.

[7] 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.

[8] 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.

[9] 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.

[10] 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 PAMI*, 24(11):1438–1454, 2002.

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

[12] X. Wang, V. Govindaraju, and S. Srihari. Holistic recognition of touching digits. In $6^{th}$ *IWFHR*, pages 295–303, Taejon, Korea, 1998.

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