High-level Verification of Handwritten Numeral Strings

L. S. OLIVEIRA[1-3], R. SABOURIN[1-3], F. BORTOLOZZI[1], C. Y. SUEN[3]

[1]PUCPR Pontifícia Universidade Católica do Paraná
PPGIA Programa de Pós-Graduação em Informática Aplicada
LARDOC Laboratório de Análise e Reconhecimento de Documentos
Rua Imaculada Conceição 1155, 80215-901 - Curitiba, PR - BRAZIL
{soares,fborto}@ppgia.pucpr.br

[2]ETS Ecole de Technologie Supérieure
LIVIA Laboratoire d'Imagerie, de Vision et d'Intelligence Artificielle
1100, rue Notre Dame Ouest, Montreal, (Quebec) H3C 1K3, CANADA
sabourin@gpa.etsmtl.ca

[3]CENPARMI Centre for Pattern Recognition and Machine Intelligence
1455 de Maisonneuve Blvd. West, Suite GM 606 - Montreal, H3G 1M8, CANADA
suen@cenparmi.concordia.ca

Abstract. In this paper we discuss the use of high-level verification on handwritten numeral strings. First of all, we introduce the concept of levels of verification and present the baseline system used to carry out the experiments. Two different strategies were developed: absolute and one-to-one verifiers. A thorough error analysis is also presented in order to identify under which conditions high-level verification is more appropiate. Experimental results are presented on NIST SD19 database.

1 Introduction

Automatic processing of handwritten numeral strings has been attempted in several application areas such as the reading zip codes on envelopes, courtesy amounts on bank cheques, data filled in tax and census forms. Such applications have been motivated by the availability of relatively inexpensive CPU power, and the possibility to reduce considerably the manual effort involved in these tasks.

Strategies for numeral string recognition can be divided into segmentation-then-recognition [17] and segmentation-based recognition [15]. In the first approach, 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 as well as the pre-defined rules of the segmenter.

The second strategy is based on a probabilistic assumption where the final decision must express the best segmentation-recognition score of the input image. Usually, the system yields a list of hypotheses from the segmentation module and each hypothesis is then evaluated by the recognition. Finally, the list is post-processed taking into account the contextual information. Although this approach gives better reliability than the previous one, the main drawback lies in the computational effort needed to compare all the hypotheses generated. In this strategy, segmentation can be explicit when based on cut rules [12, 25] or implicit when each pixel column is a potential cut location[1, 20].

There have been significant improvements in the recognition rate of the handwritten numeral string recognition systems in the last decade. One of the most significant reasons is that researchers have combined different feature sets and classifiers in order to achieve better recognition rates [8, 14]. However, when considering real business constraints, there may be some difficulties such as run time inefficiency, system complexity, and coordination of different organizations. Another strategy that can increase the recognition rate in a relatively easy way with a small additional cost is through the use of high-level verification. Such a scheme consists on refining the top few candidates in order to enhance the recognition rate economically. The focus of this work is to show the implementation of this strategy in an already developed handwritten numeral string recognition system.

For this purpose, we will consider the system presented in [13]. It takes a segmentation-based recognition approach where an explicit segmentation algorithm determines the cut regions and provides a multiple spatial representation. In opposite to the systems that process just isolated numerals [7, 22], our system has to solve a crucial problem: distinguishing, at the recognition stage, a sequence corresponding to an inter-character segmentation from another

1530-1834/01 $10.00 © 2001 IEEE

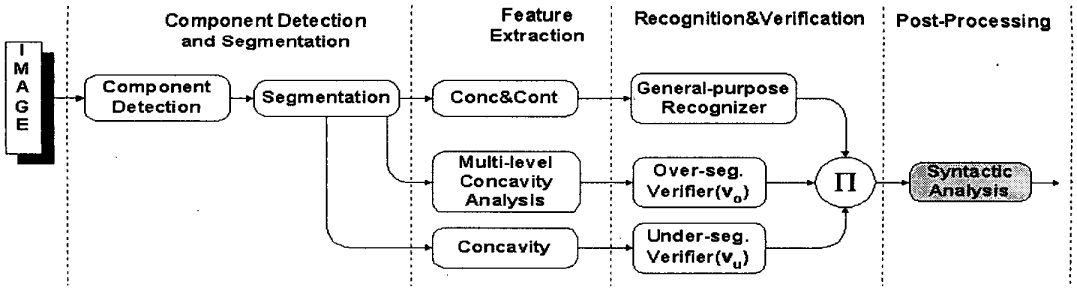


Figure 1: Block diagram of a numeral string recognition system.

relative to an intra-character segmentation. In order to overcome this problem we have used a strategy based on Recognition and Verification where the recognition function takes into account only a general-purpose recognizer while the verifiers evaluate the result supplied by the recognizer. The integration of all modules is done through a probabilistic model inspired by information theory [5].

Initially, we introduce the concept of levels of verification and present a brief overview of the system. Afterwards, we describe two different strategies of high-level verification as well as the results achieved by them. All experiments reported in this work were carried out on NIST SD19 database. In order to get a better comprehension of the system, we show a strong error analysis. Finally, we outline the perspectives for future works and present our conclusions.

2 Levels of Verification

The Recognition and Verification scheme looks straightforward, with a verification module embedded in the traditional classification system, which has a general-purpose recognizer only. The goal of the general-purpose recognizer is to attribute a given input to one of the n existing classes of the system, while the pattern verifier assumes the role of an expert to evaluate precisely the result of the recognizer in order to compensate for its weakness due to particular training, and consequently to make the whole system more reliable. Usually, a pattern verifier is applied after a general-purpose recognizer and it is designed to "plug and play", i.e., it is used without knowing the implementation details of the recognition modules.

Takahashi and Griffin in [21] define three kinds of verification: absolute verification for each class (Is it a "0" ?), one-to-one verification between two categories (Is it a "4" or a "9" ?) and verification in clustered, visually similar, categories (Is it a "0", "6" or "8" ?).

In addition to these definitions, we introduce the concept of levels of verification, where two levels are considered: high-level and low-level. We define as high-level verifiers those that deal with a sub-set of the classes considered by the general-purpose recognizer. The goal of the verifiers at this level is to confirm or deny the hypotheses produced by the general-purpose recognizer by recognizing them [21, 26]. We define as low-level verifiers those that deal with meta-classes of the system such as characters and part of them. The purpose of a low-level verifier is not to recognize a character, but rather to determine whether a hypothesis generated by the general-purpose recognizer is valid or not [19].

3 System Overview

As we can see in Figure 1, basically the system is composed of four parts: component detection and segmentation, feature extraction, recognition&verification and post-processing.

The component detection module operates in two steps: connected component analysis and grouping. The former segments the string image into connected components and eliminates very small ones by filtering while the latter tries to minimize the effects of fragmentation by detecting potential parts and grouping each of them to its nearest neighbor.

The segmentation module that we have used in this system is based on the relationship of two complementary sets of structural features, namely, contour/profile and skeletal points. Such an algorithm takes into account an over-segmentation context and its final objective is to provide the best list of hypotheses of segmentation paths without any a priori knowledge of context, such as the number of characters to be segmented. More details about this method can be found in [12, 11].

The feature extraction is composed of three different feature sets. The first one, which feeds the general-purpose recognizer, uses a mixture of concavity [9] and contour based features. The second one, which feeds the over-segmentation verifier, is based on a multi-level concavity analysis [13]. The last feature set, which feeds the under-segmen-

tation verifier, takes into account the same concavity analysis used by the general-purpose recognizer.

Although many types of neural networks can be used for classification purposes [18], we opted for a multi-layer perceptron (MLP) which is the most widely studied and used neural network classifier. Therefore, all classifiers presented in this work are MLPs trained with the backpropagation algorithm [3]. The training and validation sets were composed of 195,000 and 28,000 samples from hsf_{0,1,2,3} series respectively while the test set was composed of 60,089 samples from hsf_7 series. The recognition rates (zero-rejection level) achieved by the general-purpose recognizer were 99.66%, 99.45% and 99.13% on the training, validation and test sets respectively.

As we can observe from Figure 1, the recognition system considers two verifiers to cope with the over-segmentation and under-segmentation problems. The objective of these verifiers is to validate the general-purpose recognizer hypotheses by using the following meta-classes: characters, part of characters and under-segmented characters. The first verifier (v_0) was trained with 28,000 samples and it reached a recognition rate of 99.40% on the test set (14,000 samples) while the second verifier (v_u) was trained with 9,000 samples and it reached a recognition rate of 99.17% on the test set (4,000 samples). In [13] we show that such verifiers provided an improvement of about 7% in the recognition rate for numeral string. However, it is important to emphasize that these verifiers are low-level verifiers and hence we will not discuss how they work in this paper.

Regarding the post-processor, the sole information that it can use (considering NIST numeral string database) is the string length. Thus, the syntactic analysis just considers the string length to post process the hypotheses yielded by the system. Such a model is represented by a grey box in Figure 1 due to the fact that it can be either part of the system or not. In the following sections we will reference the system depicted in Figure 1 without the post-processor module (no a priori knowledge of the string length) as the baseline system.

4 High-Level Verification

As mentioned previously, the goal of the high-level verification is to confirm or deny the hypotheses produced by the general-purpose recognizer by recognizing them. In this section we discuss two different strategies of high-level verification: absolute and one-to-one. We will describe how such verifiers were implemented as well as why they contribute to improve the recognition rate of the system in some cases and why they fail in others.

4.1 Absolute High-Level Verifier

In this experiment ten absolute verifiers (one for each numerical class) were considered. Each verifier was trained with two classes: digit and noise. For example, for the verifier of the digit class "0", we have used all zeros of the training set (19,500 samples) for the digit class and the same number of other digits for the noise class. We have tried different feature sets such as concavity analysis in 8-Freeman directions and Moments [6]. The feature set that produced better results was the same one used by the general-purpose recognizer. Table 1 shows the recognition rates reached by each absolute high-level verifier.

Table 1: Recognition rates achieved by the absolute high-level verifiers.

Class	RR (%)
0	99.66
1	99.08
2	99.58
3	99.20
4	99.80
5	99.66
6	99.50
7	99.84
8	99.28
9	99.10

According to our probabilistic model, the output of the recognition module will be the product of four probabilities: the probability supplied by the general-purpose recognizer (P_1), the probability supplied by its respective high-level verifier (P_2) and the probabilities yielded by the two low-level verifiers (P_3 and P_4) (see Figure 2).

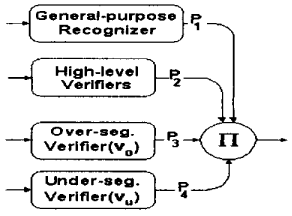


Figure 2: Recognition and Verification module with high-level verifiers.

We have observed that this strategy of verification supplies an improvement to naturally isolated digits, however, when the system faces problems such as touching and fragmentation, it does not seem very appropriate. Table 2 presen-

ts the results on different string lengths[1] on NIST SD19 (hsf_7 series).

As we can see, the overall performance achieved for numeral strings by the system that considers the absolute verifiers is worse than the baseline system. This can be explained by the fact that strings with more than one digit present several cases of fragmentation and touching.

Table 2: Recognition rates (% - zero-rejection level) for numeral strings: Comparison among different strategies.

String Lenght	Number of Strings	Absolute Verifier	One-to-one Verifier	Baseline System
1	60089	98.72	98.02	98.06
2	2370	96.65	96.10	96.88
3	2385	95.03	94.98	95.38
4	2345	92.97	92.91	93.38
5	2316	92.01	91.03	92.40
6	2169	92.60	91.77	93.12
10	1215	89.51	89.00	90.24

This strategy seems well suitable for systems that have a weak general-purpose recognizer or systems that do not face problems such as touching and fragmentation very often. In such cases, the verifier can re-rank the list of hypotheses in order to get the correct answer.

5 One-to-One High-Level Verifier

The second strategy of high-level verification that we have implemented was the one-to-one verifier. Such a strategy is straightforward and makes it easy to concentrate on the local difference between two classes. In order to determine the main confusions of the baseline system, we carried out an error analysis on the validation set of isolated digits and we observed 39 different confusions (theoretically, the number of possible confusing digit pairs is $10 \times 9/2 = 45$). We can solve 75.05% and 62.76% of all errors focusing on top-39 and top-20 confusions respectively. Therefore, it seems more cost effective focusing on top-20 confusions, since we have to deal with about 50% of the confusions produced by the system. Table 3 presents top-20 confusions with their respective frequencies.

We trained each verifier with 40,000 samples (20,000 for each class of digit involved) using the same feature set that we have used for our general-purpose recognizer. For these verifiers we have tried different feature sets such as Edge Maps [2] and histograms, but the one that brought better results was the same one used by the general-purpose recognizer. Table 2 presents the results achieved by this strategy for different string lengths.

[1] 1-digit string means that isolated digits are submitted to the system modules without any a priori knowledge of the image. In addition to recognition errors, segmentation and fragmentation errors are also considered and consequently the system reaches smaller recognition rates than when the isolated digit is submitted directly to feature extraction and recognition.

Table 3: Top 20 digit confusion with frequencies

Confusion	Frequence	Confusion	Frequence
8-0	48	7-1	17
3-2	40	9-5	17
2-1	29	7-2	16
4-0	28	9-8	15
7-3	28	8-2	15
9-7	28	6-4	14
9-4	27	6-5	12
5-3	22	8-5	11
6-0	22	9-0	11
8-3	22	8-4	10

As we can notice, this strategy gave worse results than the previous one. We expected better results at least for 1-digit string problem, once the confusions were detected on the isolated digit database. But even for this case the results were unsatisfactory. In order to enhance the results supplied by this strategy, it is necessary to improve the verifier training set by including misrecognized samples. The difficulties of implementing such a solution lie in two points:

1. Lack of samples to improve the database. If we consider our most frequent confusion (8-0), we have just 48 cases.

2. If we include a few misrecognized samples in the training set of the verifier, probably we will introduce noise to our models. We can visualize this problem from Figure 3.

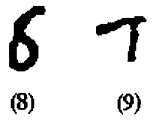


Figure 3: Misrecognized samples: 8 confused with 6 and 9 confused with 7.

In order to identify the different sources of error of the system and find why the high-level verification schemes achieved unsatisfactory results, we decided to carry out an error analysis on numeral strings instead of isolated digits.

6 Error Analysis on Numeral Strings

In order to gain a better insight of the system, we divided the errors into four classes: confusions generated by the general-purpose recognizer, confusions generated by the low-level verifiers, errors caused by segmentation and errors caused by fragmentation. This analysis was carried out considering the baseline system. Table 4 describes all sources of errors as well as its frequency per string size.

We can read this table in the following way. For 2-digit strings, we have detected 74 errors, which corresponds to a global error rate of 3.12%. This error is divided into the following: 43 due to the general-purpose recognizer, 15 to low-level verifiers, 9 to segmentation and 7 to fragmentation. Focusing on the top-10 confusions of the general-purpose recognizer, we can correct 31 of the 43 errors found. In the following subsections, we will give more details about each source of error.

6.1 General-purpose recognizer

In order to generate the total number of confusions for numeral strings, we carried out the analysis for each string length independently and afterwards we summarized the results to compare with the confusions obtained for isolated digits. The first interesting fact we observed was that we have different confusions for different string lengths. This means that a verification strategy based on one-to-one verifier could supply improvements for some string lengths but will be difficult to optimize globally the system with this strategy. Table 5 summarizes the top-10 confusions for each string length.

If we compare the top-10 confusions found on numeral strings (Table 5) with the top-10 confusions found on isolated digits (Table 3), we can observe that the confusions do not respect the same order, and some of the numeral string confusions even do not exist for isolated digits.We can cite for example, 7-2, 4-1 and 7-1 pair confusions. We observed that effects generated by the segmentation algorithm such as ligatures produce several confusions. Figure 4 shows the problem of the confusion between 9-7 (top-1 confusion).

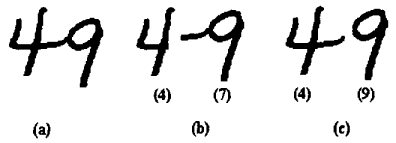


Figure 4: Confusion between 9 and 7: (a) Original image, (b) Best hypothesis of Segmentation-Recognition and (c) Correct hypothesis.

6.2 Low-Level Verifiers

We divided the error of the low-level verifiers into two classes: confusion generated by the over-segmentation verifier (v_o) and confusion generated by the under-segmentation verifier (v_u). We have observed that the latter is responsible for 87.7% of the errors generated at this level, while the former generates just 12.3% of the errors. About the second low-level verifier, the confusions are generated when isolated digits are classified as under-segmented digits. The classes where this kind of confusion occurs are the digit classes "6" (44%), "0" (22%), "8" (17%), "4" (12%) and "2" (5%). Figure 5 shows some examples of these classes of digits.

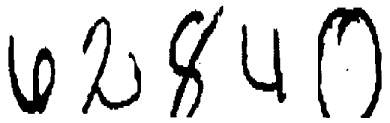


Figure 5: Digits confused with under-segmented class by the second low-level verifier. This kind of 8 and 0 are sometimes misverified by the first verifier (v_o).

About the first low-level verifier (v_o), the confusion is generated when the verifier does not succeed in detecting the over-segmented parts. Such a fact usually happens with digit classes "0" (61%) and "8" (39%). The over-segmentation in these two classes is generated when the digits are opened (Figure 5). This kind of effect usually is produced by pre-processing.

6.3 Segmentation

The segmentation errors can be caused either by under-segmentation, which is due to a lack of basic points in the neighbourhood of the connection stroke, or wrong segmentation. More details about segmentation errors can be found in [12].

6.4 Fragmentation

The confusions produced by fragmentation are found basically when the algorithm groups the fragmented part with the wrong neighbor. Usually, it fails for images that have neighbors (left and right) with similar distances to the fragmented part (Figure 6b) and for images with poor quality (Figure 6a)

7 Discussion

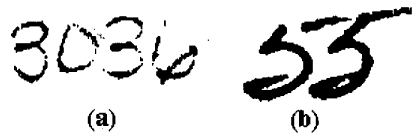
So far, we have described two different strategies of high-level verification in order to improve the recognition rate of

Table 4: Distribuition of the system errors

String	G-P. Recognizer				Verifier		Segmentation		Fragmentation		Total	
Length	Errors	%	Top-10	%	Errors	%	Errors	%	Errors	%	Errors	%
2	43	1.81	31	1.31	15	0.64	9	0.38	7	0.29	74	3.12
3	78	3.26	40	1.67	15	0.64	9	0.38	8	0.33	110	4.61
4	104	4.46	53	2.27	20	0.86	17	0.73	13	0.56	154	6.61
5	126	5.43	72	3.10	11	0.48	14	0.60	25	1.08	176	7.59
6	98	4.49	65	2.98	31	1.42	13	0.59	8	0.37	150	6.87
10	89	6.73	55	4.16	8	0.60	9	0.68	23	1.74	129	9.75

Table 5: Top-10 digit confusion with frequencies per string length.

2-digit	3-digit	4-digit	5-digit	6-digit	10-digit	Total
7-2: 6	4-1: 5	5-3: 7	7-3: 11	9-7: 11	9-7: 9	9-7: 41
9-4: 5	7-1: 5	9-7: 7	7-9: 9	7-3: 9	6-2: 8	8-0: 39
8-0: 4	4-1: 5	4-1: 6	8-0: 9	8-0: 9	8-0: 7	3-2: 31
3-2: 4	3-2: 5	3-2: 5	7-2: 8	3-2: 8	2-1: 7	9-4: 30
6-5: 2	2-1: 5	2-1: 5	9-4: 7	7-2: 7	7-1: 7	7-2: 27
4-1: 2	6-0: 3	6-0: 5	9-8: 6	9-4: 6	9-4: 5	7-3: 24
8-2: 2	9-4: 3	9-4: 5	3-2: 6	8-3: 6	3-2: 3	5-3: 20
8-5: 2	8-0: 3	8-0: 5	5-3: 6	7-1: 6	4-2: 3	4-1: 20
9-0: 2	7-3: 3	7-3: 4	9-5: 5	5-3: 5	6-5: 3	7-1: 20
9-7: 2	2-0: 3	2-0: 4	6-4: 5	8-2: 5	9-1: 3	2-1: 18



(a) (b)

Figure 6: Fragmentation problems.

the system. We also presented a strong error analysis carried out on numeral strings. As we can notice, both strategies (absolute and one-to-one) do not achieve satisfactory results on numeral strings. Such strategies become interesting either when there is a diversity of samples (confusions) to train the verifiers or when the system has a weak general-purpose recognizer, e.g., the system presented by Britto Jr. et al in [1].

The strategy based on absolute verifiers has brought an improvement for 1-digit string. Such a fact emphasizes that high-level verifiers should be built in order to cope with more complex problems, e.g., all sources of errors presented in the last section. However, we have seen that it is not a trivial problem. As described in Table 5, the confusions for numeral strings are not concentrated in a few classes and for this reason a different strategy of optimization should be adopted in our case. One strategy could be to find different feature sets to feed the high-level verifiers. But in this case, the system should overcome the same kind of problems faced by multi-classifier systems, e.g., run time inefficiency and system complexity.

Since our two main sources of errors are related to classification problems (general-purpose recognizer and low-level verifiers), we think that a plausible strategy to optimize the overall performance of the system lies in feature subset selection [10, 24]. A very popular approach to carry out this task in large-scale problems is genetic algorithms [16], since the encoding of a feature subset into a chromosome is straightforward and the function that is optimized does not need to be smooth and can, therefore, be directly the classification accuracy (direct error minimization).

In spite of the fact the current system can be optimized in some respects, we already have recognition rates comparable or better than those reported in the literature. Table 6 summarizes recognition rates claimed by different authors on NIST SD3/SD19 (hsf_7). Ha et al [23] used about 5,000 strings of the NIST SD3. Lee and Kim [20] used 5,000 strings but they did not specify the used data. It is important to remark that the results achieved for Fujisawa's

Table 6: Recognition rates on NIST SD3/SD19 - hst_7 (zero-rejection level) ([2]No post-processor, [4]With post-processor)

String	Ha [23]		Lee [20]		Fujisawa [4]		Our System[2]		Our System[4]	
Length	Strings	RR %	Strings	RR %	Strings	RR %	Strings	RR %	Strings	RR %
2	981	96.2	1000	95.23	1000	89.79	2370	96.88	2370	97.21
3	986	92.7	1000	88.01	1000	84.64	2385	95.38	2385	95.80
4	988	93.2	1000	80.69	1000	80.63	2345	93.38	2345	94.11
5	988	91.1	1000	78.61	1000	76.05	2316	92.40	2316	93.22
6	982	90.3	1000	70.49	1000	74.54	2169	93.12	2169	95.90
10							1215	90.24	1215	91.07

system were published in [20]. As we can see, even considering a larger number of strings we reach better results than the other systems. Figure 7 shows the examples of misclassified fields while Figure 8 shows the examples of fields containing touching or broken characters that were correctly recognized by the baseline system.

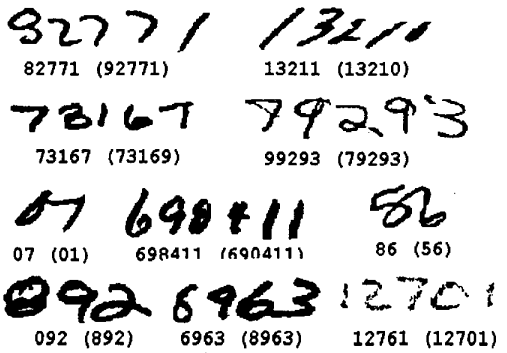


Figure 7: Examples of misclassified fields.

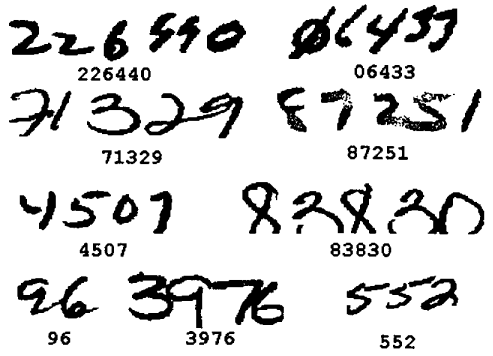


Figure 8: Examples of correctly recognized fields.

8 Conclusion

We have presented in this paper some experiments considering strategies of high-level verification. Two different schemes were developed, namely, absolute verifiers and one-to-one verifiers. We have introduced the concept of levels of verification and described the baseline system, which takes into account a segmentation-based recognition approach with an explicit segmentation algorithm. We have seen that the absolute verifier strategy brought an improvement in the recognition rate for 1-digit string but it reached worse results on strings with more than one digit due to the different kinds of errors between isolated digits and string of digits.

Based on the experiments described in this work, we can conclude that one of the best ways to optimize a system with a good overall performance lies in the optimization (deletion, addition, and modification) of the feature sets. Therefore, our next efforts will be focused on the optimization of the feature sets employed in the system through a genetic algorithm approach. Finally, some results claimed by different authors on NIST SD19 database have been compared.

Acknowledgements

The authors wish to thank Pontifícia Universidade Católica do Paraná and Paraná Tecnologia which have supported this work.

References

[1] A.Britto-Jr., R.Sabourin, F. Bortolozzi, and C.Y.Suen. A two-stage HMM-based system for recognizing handwritten numeral strings. To appear in 6^{th}ICDAR, Seattle-USA, September, 2001.

[2] Y. C. Chim, A.A. Kassim, and Y.Ibrahim. Dual classifier system for handprinted alphanumeric character recognition. *Pattern Analysis and Application*, 1:155–162, 1998.

[3] D.E.Rumelhart, R.Durbin, R.Golden, and Y.Chauvin. Backpropagation: The basic theory. In Y.Chauvin and D.E.Rumelhart, editors, *Backpropagation:Theory, Architectures and Applications*, pages 1–34. Lawrence Erlbaum, Hillsdale, NJ, 1995.

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

[5] G.E.Kopec and P.A.Chou. Document image decoding using markov source models. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(6):602–617, 1994.

[6] M. K. Hu. Visual pattern recognition by moment invariant. *IEEE Transaction on Information Theory*, 8:179–187, 1962.

[7] J.Cai and Z.Q.Liu. Integration of structural and statistical information for unconstrained handwritten numeral recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(3):263–270, 1999.

[8] J.Kittler, M.Hatef, R.Duin, and J.Matas. On combining classifiers. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 20(3):226–239, 1998.

[9] L.Heutte, J.Moreau, B.Plessis, J. Plagmaud, and Y.Lecourtier. Handwritten numeral recognition based on multiple feature extractors. In 2^{nd} *ICDAR*, pages 167–170, 1993.

[10] L.Kuncheva and L.C.Jain. Designing classifier fusion systems by genetic algorithms. *IEEE Trans. on Evolutionary Computation*, 4(4):327–336, 2000.

[11] L.S.Oliveira, E. Lethelier, F. Bortolozzi, and R.Sabourin. A new approach to segment handwritten digits. In 7^{th} *IWFHR*, pages 577–582, Amsterdam-Netherlands, 2000.

[12] L.S.Oliveira, E. Lethelier, F. Bortolozzi, and R.Sabourin. A new segmentation approach for handwritten digits. In 15^{th} *ICPR*, volume 2, pages 323–326, Barcelona-Spain, 2000.

[13] L.S.Oliveira, R.Sabourin, F.Bortolozzi, and C.Y.Suen. A modular system to recognize numerical amounts on brazilian bank cheques. To appear in 6^{th}ICDAR, Seattle-USA, September, 2001.

[14] L.Xu, A.Krzyzak, and C.Y.Suen. Methods of combining multiple classifiers and their applications to handwritten recognition. *IEEE Trans. on Systems, Man, and Cybernetics*, 22(3):418–435, 1992.

[15] O. Matan and C.J.C.Burges. Recognizing overlapping hand-printed characters by centered-objects integrated segmentation and recognition. In *IJCNN*, pages 504–511, 1991.

[16] M.Mitchell. *An introduction to genetic algorithms*. MIT Press, Cambridge - MA, 1996.

[17] M.Shridhar and A.Badreldin. Recognition of isolated and simply connected handwritten numerals. *Pattern Recognition*, 19(1):1–12, 1986.

[18] R.P.Lippmann. Pattern classification using neural networks. *IEEE Communications Magazine*, 27(11):47–64, 1989.

[19] S.J.Cho, J.Kim, and J.H.Kim. Verification of graphemes using neural networks in an HMM-based on-line korean handwritting recognition system. In 7^{th} *IWFHR*, pages 219–228, Amsterdam-Netherlands, 2000.

[20] S.W.Lee and S.Y.Kim. Integrated segmentation and recognition of handwritten numerals with cascade neural networks. *IEEE Trans. on Systems, Man, and Cybernetics, Part C:Applications and Reviews*, 29(2):285–290, 1999.

[21] H. Takahashi and T.D.Griffin. Recogniton enhancement by linear tournament verification. In 2^{nd} *ICDAR*, pages 585–588, Tsukuba - Japan, 1993.

[22] T.M.Ha and H.Bunke. Off-line, handwritten numeral recognition by perturbation method. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(5):535–539, 1997.

[23] T.M.Ha, M.Zimmermann, and H.Bunke. Off-line handwritten numeral string recognition by combining segmentation-based ans segmentation-free methods. *Pattern Recognition*, 31(3):257–272, 1998.

[24] W.Siedlecki and J.Sklansky. A note on genetic algorithms for large scale on feature selection. *Pattern Recognition Letters*, 10:335–347, 1989.

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

[26] J. Zhou, Q. Gan, A. Krzyzak, and C.Y.Suen. Recognition and verification of touching handwritten numerals. In 7^{th} *IWFHR*, pages 179–188, Amsterdam-Netherlands, 2000.