Verifying the UNIPEN devset

Louis Vuurpijl[1], Ralph Niels[1], Merijn van Erp[1],
Lambert Schomaker[2], and Eugene Ratzlaff[3]
[1]Nijmegen Institute for Cognition and Information
[2]University of Groningen
[3]IBM Research

Abstract

This paper describes a semi-automated procedure for the verification of a large human-labeled data set containing online handwriting. A number of classifiers trained on the UNIPEN "trainset" is employed for detecting anomalies in the labels of the UNIPEN "devset". Multiple classifiers with different feature sets are used to increase the robustness of the automated procedure and to ensure that the number of false accepts is kept to a minimum. The rejected samples are manually categorized into four classes: (i) recoverable segmentation errors, (ii) incorrect (recoverable) labels, (iii) well-segmented but ambiguous cases and (iv) unrecoverable segments that should be removed. As a result of the verification procedure, a well-labeled data set is currently being generated, which will be made available to the handwriting recognition community at the IWFHR9 conference.

1 Introduction

In a large collaborative effort, a wide number of research institutes and industry have generated the UNIPEN standard and database [2]. Originally hosted by NIST, the data was divided into two distributions, dubbed the *trainset* and *devset*. Since 1999, the International UNIPEN Foundation (iUF) hosts the data, with the goal to safeguard the distribution of the trainset and to promote the use of online handwriting in research and applications. In the last years, dozens of researchers have used the trainset and described experimental performance results. Many researchers have reported well established research with proper recognition rates, but all applied some particular configuration of the data. In most cases the data were divided, using some specific procedure, into three subsets for training, testing and validation. Therefore, although the same source of data was used, recognition results can not really be compared as different decomposition techniques were employed. Furthermore, in most reported cases, a particular set of badly segmented or wrongly labeled data was removed or changed, which makes the comparison of results even more difficult.

For some time now, it has been the goal of the iUF to organize a benchmark on the remaining data set, the devset. Although the devset is available to some of the original contributors to UNIPEN, it has not officially been released to a broad audience yet. It is the goal of our current paper to describe the procedure for verifying the devset, i.e. validating and correcting the data. This procedure should ensure the quality of a proper new benchmark data set, to be made available to the global handwriting recognition community.

The original UNIPEN devset is organized in 9 sets, as listed in Table 1. It is known that labeling and segmentation errors are present in both UNIPEN sets. An estimate of the number of errors in the trainset is given in [3]. It was reported that approximately 4% of the samples are errors. Other reported errors in both sets are described

in, e.g., [1, 7], reporting about segmentation errors, and in [4] and [5], reporting about segments that were obviously too wide or too narrow. In a recent effort made by Ratzlaff [5], scripts were generated that divide the data into configurable train and evaluation sets. The scripts used for decomposing the data were recently made available through `http://www.alphaworks.ibm.com/tech/comparehwr`. Assuming that users have the data, these scripts now provide the possibility to generate uniform subsets and thus create the opportunity to report on UNIPEN benchmarks that are comparable between researchers. However, a number of segmentation errors still remains in the data and moreover, the scripts do not check on labeling errors.

set	**nfiles**	**#**	**description**
1a	508	8598	isolated digits
1b	1087	16414	isolated upper case
1c	1672	37506	isolated lower case
1d	973	9898	isolated symbols (punctuations etc.)
2	2144	72416	isolated characters,mixed case
3	1267	44416	isolated characters in the context of words or texts
6	2072	46353	isolated cursive or mixed-style words (without digits and symbols)
7	2143	52700	isolated words, any style, full character set
8	3592	11059	text: (minimally two words of) free text, full character set
total	15458	299360	

Table 1: UNIPEN devset organization. Sets 4 and 5 (isolated printed words) are empty in both the trainset and devset.

The focus of this paper is to report on the quality of the UNIPEN data by examining the observed and detected errors in detail. To this end, a semi-automated procedure is described that distinguishes between a number of sample categories. The first step of this process is automated. A number of classifiers are combined to increase the confidence in cases where samples may be accepted. In the second step, the rejected samples are manually verified. As a result of this procedure, the following classes of samples are produced, where all but the first category require human supervision:

1. *Correct segments*, containing samples that are accepted with sufficient confidence by the procedure. This category is not further inspected in the procedure. In the next section it is explained how it is ensured that the amount of errors that slip through the automated selection process can be minimized.

2. *Segmentation errors*, containing samples that are badly segmented. In UNIPEN, segmentations are specified through a so-called delineation, which marks the beginning and end of a sample (also called *segment*). Segmentation errors are caused by wrong delineations. In some cases these errors can be recovered, which is explained in Section 3.

3. *Labeling errors*, containing samples with wrong labels. Such errors may be caused by the writer producing the samples or by the human labeler, who may have interpreted

the handwriting incorrectly. There is a fuzzy line between obvious labeling errors and cases where the label cannot be determined because of sloppy handwriting, or because the shape of a sample is ambiguous.

4. *Ambiguous samples*, containing shapes that can be interpreted in at least two ways. Most often, such shapes cannot be interpreted without context.

5. *Unfamiliar samples*, containing allographs that are unfamiliar to a classifier or human expert. Such samples typically are encountered in multi-lingual databases or databases with writers from different origin, as is the case in UNIPEN.

Figure 1 displays some examples from the latter four categories. The first row depicts samples with a proper label, but that have a poor quality, because of sloppy handwriting. In UNIPEN, such samples would be labeled as having a BAD quality. Rows 2,3,4 in Figure 1 depict problems of actual mislabeling, erroneous segmentation and interpretation ambiguity, respectively.

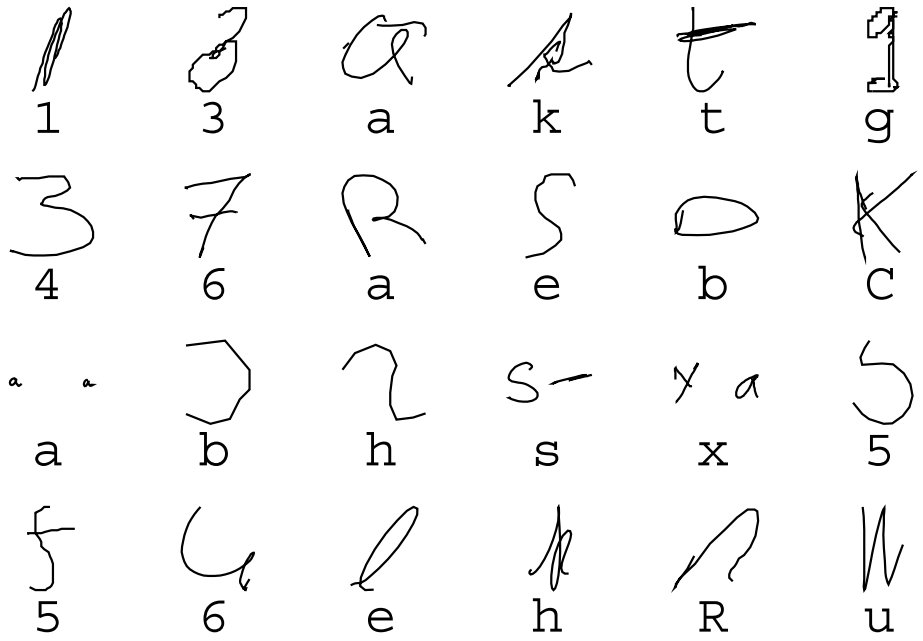

Figure 1: Problematic cases in UNIPEN data.

There is a particular note to be made on the first and last categories, containing samples with inferior quality or which are ambiguous. There are examples of other, well-segmented and labeled data sets that are used for training and testing handwriting recognizers, yielding high recognition rates. Although it is valid to report such results in literature, it also leads to systems that fail in real-life conditions. Rather than removing such bad samples, we opt for leaving them in the database and label the quality as BAD, or as a more suitable category like INFERIOR or AMBIGUOUS. The latter two qualifications are not contained in the current UNIPEN definition, however.

The verification procedure that is described in this paper has been completed for the first three character sets (1a,1b,1c) and is currently being applied on the remaining sets. Results

for these character sets are presented here and the preliminary set up of the verification procedure for word and text segments is discussed as well. In the next section, Section 2, the automated verification procedure for accepting or rejecting samples is discussed. Any samples that are rejected must be visually inspected. This process is described in Section 3. The procedure for verifying word and text segments is described in Section 4.

2 The automated verification procedure

The requirements for the verification procedure are straightforward: The first requirement is that the number of accepted samples (the *yield*) should be maximized. This reduces the laborious amount of manual verification and correction. The second requirement is that the amount of errors should be kept to a minimum. It is our goal to reduce the amount of errors in the data significantly below 1%. In the verification procedure, two parameters rule the yield and error quantities: (i) the quality and amount of the classifiers and (ii) the way in which the output hypotheses from multiple classifiers are combined for accepting or rejecting a sample.

2.1 Quality of the accepted samples

Given a division of samples in two categories: the accepted and rejected samples, a test can be performed to assess (with a certain significance α) whether the number of errors that may be expected in the accepted samples is below a certain fraction ϵ. The test says that if no errors are observed in a randomly drawn subset of N accepted cases, it is valid to assume that the error fraction is below ϵ with confidence $1-\alpha$. Let the probability of drawing an erroneous sample i from this pool be ϵ_i, which equals ϵ for all samples if samples are drawn with replacement. In this case, the total probability of detecting no errors in the subset is defined as:

$$\alpha = \prod_i^N (1-\epsilon_i) = (1-\epsilon)^N$$

So, in order to be certain with a probability α that only a fraction of ϵ errors occur in the data, it has to be verified that N randomly drawn samples contain no errors, with:

$$N = {}^{(1-\epsilon)}log(\alpha) \quad (1)$$

An event is usually considered as statistically significant, if the probability of the event occurring randomly is smaller than 5% (or a stricter 1%). Here, the event is "no errors in the subset of N samples". It is our goal to ensure with significance $\alpha = 0.01$ that maximally 1% of the accepted samples are errors and therefore, we must visually verify that no errors occur in $N = 459$ samples. Note that neither α nor ϵ can reach 0% with this test. Also note that this test does not use any information on how the accepted set was constructed. This would require an intricate knowledge of the behavior of the classifiers and their interdependency, which is beyond the scope of this research.

2.2 Quality and amount of classifiers employed

In this subsection, the quality and yield of the employed individual classifiers for character verification are discussed. Four different classifiers are used, trained on data from the

UNIPEN trainset. The trainset and devset comprise distinct samples, but do not distinguish between writers. This makes classifiers trained on the first set very suitable for recognizing the latter set, as it may be expected that samples from the training set do not differ to a large extent. Different feature schemes are being employed, describing spatial-temporal (trajectory, running angle, angular velocity) characteristics and spatial (bitmap) characteristics. All four classifiers were trained on the 1a, 1b and 1c sets of the UNIPEN trainset, from which 36130 digits, 65483 isolated upper case, and 157264 isolated lower case segments where extracted. A multi-layered perceptron (MLP) using these features, a knn classifier (k=5) with the same features, the DTW (dynamic time warping) algorithm described in [8], and the allograph matcher as described in [9] were used for classifying the three sets 1a, 1b and 1c from the devset. The knn classifier matches each of the devset samples to the training samples from the corresponding set. Although it uses the same feature vector as the MLP, the completely different knn, DTW, and MLP algorithms ensure an distinct view on the data.

	set 1a: digits		set 1b: upper case		set 1c: lower case	
Classifier	dev	errors	dev	errors	dev	errors
MLP(0)	94.3	3	87.5	2	87.1	1
MLP(.7)	90.6	2	86.4	1	74.7	1
MLP(.8)	84.4	1	76.4	1	66.4	1
MLP(.9)	60.3	none	55.1	none	51.0	none
DTW(1)	85.8	3	92.9	none	81.9	none
DTW(2)	91.6	1	83.5	1	78.5	1
DTW(3)	88.3	1	77.2	2	71.5	1
DTW(4)	83.1	none	66.6	1	61.3	none
DTW(5)	72.9	none	49.4	none	45.1	none

Table 2: Yield (percentage) and correctness of the MLP and DTW algorithms. Similar yields and amount of errors are produced by the other two classifiers for the thresholds 1...5.

All four classifiers can use an individual threshold for deciding to accept or reject samples. Each classifier only accepts samples if two conditions hold: (i) the threshold is passed and (ii) the output of each classifier corresponds to the label of the original devset. All other cases are rejected. Table 2 depicts the typical yield for two classifiers, given a certain threshold, for the devset. In case of the multi-layered perceptron, $MLP(t)$ corresponds to the percentage of accepted samples where the activation of the best output unit passes t. In case of the latter three classifiers, respectively $KNN(k)$, $DTW(k)$ and $HCLUS(k)$ represent the percentage of accepted samples for which the k closest neighbors are correct. For each individual classifier, a randomly drawn set of N reference samples was selected to be visually inspected. The column "errors" indicates the number of errors detected in the accepted samples from a particular classifier, for a given threshold.

2.3 Increasing the yield while passing the test

As can be observed in Table 2, in very strict settings each classifier is able to pass the test. However, this is at the cost of rejecting a large amount of samples. Therefore, a number of

different combination schemes were evaluated that increase the yield, while still passing the test. The assumption is that when a particular number of classifiers accept a sample (i.e. mutually agree on the outcome, which equals the label), this can be considered as a success. All classifiers are treated equal in this procedure. Below, different yields for respectively one, two, three, or four classifiers that agree on each sample in the devset are listed. Numbers that are marked with a '(y)' did pass the test.

nc	yield 1a	yield 1b	yield 1c
4	90.5(y)	59.3(y)	57.5(y)
3	91.4(y)	83.4(y)	81.1(y)
2	95.6	91.1(y)	89.8(y)
1	98.2	96.1	95.2

Table 3: Yield and results on passing the tests for cases where *nc* classifiers must agree.

When comparing these results to Table 2, it can be observed that the yield is much higher in the case of combining classifiers than when using individual (strict) threshold values. At the same time, even with combinations of only two out of four classifiers, all tests (except in the case of digits) are passed. This is an excellent example of using multiple classifiers for increasing the robustness of pattern recognition. Rather than increasing the decision threshold, the different views on the data ensure that only samples are accepted when two or more distinct observations agree.

The 1a, 1b and 1c sets from the devset were automatically divided in two categories (accepted and rejected) by using the marked combinations from Table 3. After this first step of the procedure, respectively 7858 (1a), 14952 (1b) and 33671 (1c) samples are accepted, where it is assumed that these samples contain less than 1% errors, i.e. samples that should have been rejected.

2.4 Verification of the procedure

In order to verify the correctness of the procedure, a major effort was performed by manually verifying all processed segments from the 1a, 1b and 1c sets. For each data set, the original data were split into multiple files, each file containing the data for only a single label type. So, for example, the digits set (1a) was split into ten files, one for each of the digits 0-9. The data were then displayed 100 at a time in a 10x10 grid. This allows for rapid review of the data for verification purposes. It also provides a context for reviewing a single writers work as a group, and for viewing and comparing several writing styles at the same time. This sorted and multiple view context is especially helpful to discern between labeling errors, sloppy instances of a particular allograph, and for discovering unusual allographs or writing styles that might otherwise be evaluated as bad or mislabeled. The data are then evaluated and appropriately assigned.

This manual verification process was performed independently of the manual labeling process described in the next section. Based on the completely verified data set, it is possible to assess the correctness of the assumption made in Equation 1. This assessment was performed by comparing the set of samples that were judged as erroneous by the manual

verification process, to the set of samples that were automatically accepted by the procedure described above. As a result of this comparison, no samples that were accepted for 1a appeared to have errors. Only 0.061% falsely accepted samples from the 1b set appeared to have slipped through and for the 1c set, this number was less than 0.064%. These numbers indicate that although statistically, the number of automatically accepted samples contain less than 1% errors, the real (validated) estimates are much better.

3 The manual labeling process

All samples that were rejected in the previous process are candidates for errors. Although the majority of these samples are probably correct (as only 4% errors were expected [3] and about 10% of the samples are rejected), they must be verified through human supervision. Here, three main categories of interactive operations have to be performed: (i) marking false rejects, i.e. samples that were rejected by the ensemble but that were judged as correctly labeled after visual inspection, (ii) correcting wrong labels, i.e. samples that were correctly rejected and should definitely be labeled differently, and (iii) correcting wrong segmentations, i.e. samples that could not be accepted because they were badly segmented. Please note that as indicated in the introduction, labels and segmentations in any of these categories may be distinguished in various levels of quality (sloppiness) and confidence (depending on ambiguity or familiarity of the allographs).

For each collection (1a, 1b, and 1c) of the UNIPEN devset, the appropriate ensemble of classifiers was used to filter out samples that could not be recognized with sufficient confidence. These samples were alphabetically sorted and displayed via the UNIPEN displayer `upview`. Upview is a program for fast visualization of large amounts of UNIPEN data. Similar to the viewer described in Section 2.4, upview depicts segments in a matrix organization. Specific routines for processing particular samples can be engaged by clicking on the corresponding segment. If one of the three kinds of interactive operations mentioned above should be applied to a segment, the human verifier can click on the segment by using either the left, middle or right button of his mouse. Correcting falsely rejected samples (the majority of cases) can be performed very efficiently in this manner. As samples were depicted in alphabetical order, anomalies can be detected fast.

Correcting false rejects Upon manually overriding the rejected samples, two options were made available to the supervisor. The first option marks the segment as correctly labeled, but with a proper quality. In the second option, the segment is still marked as correctly labeled, but the quality is labeled bad. The latter option is typically in place for the samples depicted in the first row of Figure 1.

Correcting wrong labels Similar to the cases where rejected samples had to be accepted, labeling errors can be distinguished in two categories: wrong labels with bad quality and wrong labels with good quality.

Marking segmentation errors As an example of a segmentation error that can be visually detected, consider the sample below, which represents a wrongly segmented character. Such segmentation errors are marked as recoverable and are stored for a later correction process.

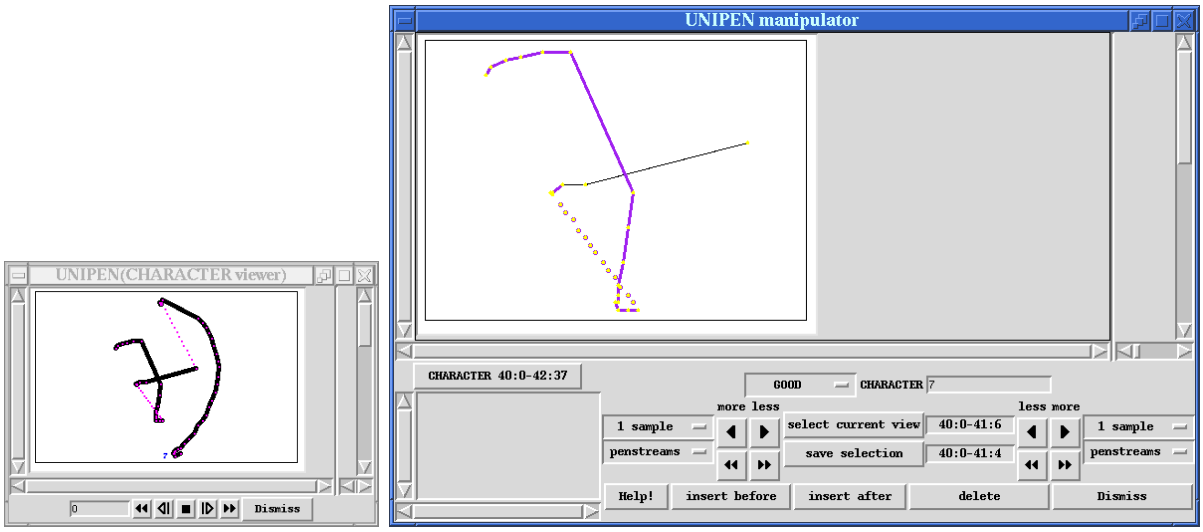

Figure 2: Wrongly segmented character and UNIPEN manipulator for recovering the correct segmentation.

Handling undecidable errors The former three cases can be identified through careful examination of the depicted segments using Upview. However, as depicted in Figure 1, some cases cannot be determined as they either contain segmentation errors or ambiguous shapes. These cases are marked as undecidable and are stored for further processing, for which the tools depicted in Fig 2 are employed.

Two human handwriting experts performed the manual labeling process described above. As the results depicted in Table 4 show, it appears that many samples provide causes for uncertainty. The main reason for this uncertainty is that judging the quality of a handwriting sample is a subjective process. Judgments on quality (i.e., when is a sample too sloppy or not, allograph familiarity (is the label correct, is the shape ambiguous?), and even segmentation errors are examples of subjective decisions.

	set 1a			set 1b			set 1c		
category	both	A	B	both	A	B	both	A	B
1 Label OK, qual. OK	348	27	160	507	163	67	1962	664	266
2 Label OK, qual. bad	46	204	18	42	60	180	291	240	680
3 Label wrong, qual. OK	6	4	7	20	36	15	78	296	23
4 Label wrong, qual. bad	0	5	5	7	14	11	11	12	160
5 Segmentation error	95	3	43	554	48	49	258	8	90
6 Undecidable	0	2	12	8	3	2	4	11	12

Table 4: Manual categorization of rejected samples by two handwriting experts 'A' and 'B'. The columns labeled "A" and "B" indicate judgments made by a single expert. Columns marked "both" list the number of cases in which both experts agree.

Estimating a lower bound on the number of correctly accepted samples in the original 1a, 1b and 1c sets can be performed by adding the number of overruled samples on which both experts agree (categories 1 and 2 in Table 4) to the number of samples accepted in the automated verification step described in Section 2. As the latter is guaranteed to have

maximally 1% errors, it can be deduced that the maximum percentage of errors in the original sets is respectively 3.0 (1a), 4.5 (1b) and 3.2 (1c).

The sixth category (undecidable) as well as all cases where both experts do not agree are stored for subsequent processing, either using more context (e.g. considering the surrounding coordinate trajectories) or discussing these cases with further experts. However, as may be concluded at this point, there is a considerable amount of samples for which judging between labels or quality is ambiguous. It will be examined whether a more elaborate distinction in, e.g. INFERIOR (shape, segmentation) or AMBIGUOUS (shape, label) is required.

4 Verifying words

The procedure described above was tested on three character sets. The same procedure is now being used for verifying the other character sets 1d, 2 and 3. In order to semi-automatically verify the word and text categories 6, 7 and 8, a more elaborate procedure is required. Although we have not completed the word and text verification procedure, the approach that is currently being implemented is described briefly below.

In word recognition of unknown trajectories containing (X,Y,Z) coordinates, an approach that is often followed is to find proper character segmentation points and to generate a character hypothesis lattice of possible word outcomes. The hypothesis space can become very large, but is restricted by the size of the lexicon, which is used to prune irrelevant character paths from the lattice. In the case of word *verification*, the lexicon contains only one word. In the word verification procedure, the word recognizer will first try to find the character paths that match the word label. If this does not succeed, the word will be rejected. If it does succeed, one or more character paths will be contained in the lattice. In the second stage, the trajectories of subsequent characters in a path will be verified as described in the character verification procedure. If all characters in a path are accepted, the word can be accepted. If one of the characters in a path cannot be accepted with sufficient confidence, the word must be rejected.

We have performed experiments with this approach, by using the velocity-based segmentation algorithm described in [6]. The method works quite well, but is restricted to fluent handwriting. Unfortunately, not all words in the UNIPEN database confirm to the required temporal characteristics that are needed to perform segmentation on points of minimal velocity. In particular, some data are acquired through mice or tablets with a low temporal resolution. Therefore, our current efforts are targeted on implementing other segmentation schemes, like those based on points of maximum curvature or Y-extrema. The current word verifier has a yield of 91% with very low error rates. However, these numbers have to be sustained in further experiments.

5 Conclusions

This paper presents a procedure for detecting and solving errors present in the UNIPEN devset. The procedure is currently being applied to all character sets from the database. The goal of this work is (i) to remove unrecoverable labeling and segmentation errors, (ii) to correct labels and segmentations for cases where this is possible, and (iii) to review the quality of UNIPEN segments.

It is shown that by using classifier combination schemes, a large portion of the data samples can be automatically checked, while keeping the remaining error margin well within a respectable 1% range. The samples that are rejected by the classifier combination have been checked manually, resulting in a number of ambiguous cases that need to be further investigated.

It has been debated that in particular the ambiguous cases or cases with bad quality present problems for handwriting classifiers and that rather than removing these samples from the database, a more elaborate qualification scheme is required.

Our current efforts are targeted on finalizing the verification process for the remaining categories and further processing samples that have not been decided upon.

Acknowledgments

The authors would like to thank Jules Ellis for his support on verifying the validity of the statistical tests. Hans Dolfing and Jianying Hu are kindly acknowledged for their contributions to recovering the original devset labels.

References

[1] C. Bahlmann, B. Haasdonk, and H. Burkhardt. On-line handwriting recognition with support vector machines—a kernel approach. In *Proc. of the 8th IWFHR*, pages 49–54, 2002.

[2] I. Guyon, L. Schomaker, R. Plamondon, R. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmarks. In *Proceedings of the 12th International Conference on Pattern Recognition, ICPR'94*, pages 29–33, Jerusalem, Israel, October 1994. IAPR-IEEE.

[3] J. Hu, S. Lim, and M. Brown. HMM based writer independent on-line handwritten character and word recognition. In *Proc. of the IWFHR-6, Taejon*, pages 143–155, 1998.

[4] M. Parizeau, A. Lemieux, and C. Gagné. Character recognition experiments using unipen data. In *Proc. of 6th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 481–485, 2001.

[5] E. Ratzlaff. Methods, report and survey for the comparison of diverse isolated character recognition results on the unipen database. In *Proc. of 7th Int. Conf. on Document Analysis and Recognition (ICDAR)*, pages 623–628, 2003.

[6] H.-L. Teulings and L.R.B. Schomaker. Un-supervised learning of prototype allographs in cursive script recognition using invariant handwriting features. In J.-C. Simon and S. Impedovo, editors, *From Pixels to Features III*, pages 61–73. North-Holland, Amsterdam, 1992.

[7] V. Vuori. Clustering writing styles with a self-organizing map. In *Proc. of the 8th IWFHR*, pages 345–350, 2002.

[8] V. Vuori, M. Aksela, J. Laaksonen, E. Oja, and J. Kangas. Adaptive character recognizer for a hand-held device: Implementation and evaluation setup. In *Proc. of the 7th IWFHR*, pages 13–22, 2000.

[9] L. Vuurpijl and L. Schomaker. Finding structure in diversity: A hierarchical clustering method for the categorization of allographs in handwriting. In *ICDAR'97*, pages 387–393. IEEE, 1997.