To appear in 2000 in book *Cognitive and Neural Models for Word Recognition and Document Processing* (World Scientific Press) edited by Nabeel Murshed.

Experiments with Adaptation Methods in On-line Recognition of Isolated Latin Characters

Jorma Laaksonen, Vuokko Vuori, Matti Aksela, and Erkki Oja

Helsinki University of Technology
Laboratory of Computer and Information Science
P.O.Box 5400
Fin-02015 HUT, Finland

Jari Kangas

Nokia Research Center
P.O.Box 100
Fin-33701 Tampere, Finland

Contents

Experiments with Adaptation Methods in On-line Recognition of Isolated Latin Characters

Jorma Laaksonen, Vuokko Vuori, Matti Aksela, and Erkki Oja

Helsinki University of Technology
Laboratory of Computer and Information Science
P.O.Box 5400
Fin-02015 HUT, Finland

Jari Kangas

Nokia Research Center
P.O.Box 100
Fin-33701 Tampere, Finland

Abstract

The purpose of this paper is to summarize our work on adaptive on-line recognition methods for handwritten characters. Reports on the work have been published in various conference proceedings and book chapters. As each publication covers only some specific part of our work, it is hard to see the whole picture and get a good overview of the whole work. Instead of trying to explain in detail all the techniques and experiments, we compare them with each other and give more general results.

By adaptation we mean that the system is able to learn new writing styles and thus improve its performance. We have had two different approaches to the adaptation: experiments have been carried out with both individually adaptive classifiers and adaptive committees of static classifiers.

The main techniques applied in our work include the k-Nearest Neighbor and the Local Subspace Classification rules, Dynamic Time Warping and Levenshtein distances, Learning Vector Quantization, and Dynamically Expanding Context.

1 Introduction

Since 1997 we have studied methods for adaptive on-line recognition of isolated characters [10]. A hypothetical application in our minds has been a portable digital assistant (PDA) into which all input would be entered with a stylus. Such a system should be capable of decent recognition of any user's writing right from the beginning and still be able to increase its accuracy during use. In our view, the adaptation of the recognizer should take place unnoticed by the user, i.e. simultaneously with the system's normal use.

In this paper, we summarize all our experiments and experiences this far. In Section 2, we first give a short overview of the recognition system. Then, in Sections 3 and 4 we describe the data used and the preprocessing and normalization stages applied in all experiments. Section 5 addresses the feature extraction methods we have utilized and Section 6 the different classification techniques we have applied. The various dissimilarity measures used in the Dynamic Time Warping (DTW) classifier are then analyzed in more depth in Section 7. As our recognition system is user-independent and adaptive, the formation of the initial prototype set and its modification during the on-line adaptation stage are essential to the operation. These questions are addressed in the next two sections. Finally, we summarize our results until now in Section 10, consider some implementation issues in Section 11, and have a look at our future plans in Section 12.

2 General Overview of the System

The recognition system used in our experiments is based on various forms of template matching. It consists of one or more separate classification units which compare input characters with their own prototype sets. If more than one classification unit are used simultaneously, they form a committee classifier. The recognition system is adapted to the new user's writing style either by adding, inactivating, or modifying the prototypes in the individual recognizers, or by adding new, more detailed decision rules in the committee classifier. These two forms of adaptation can be carried out simultaneously or sequentially.

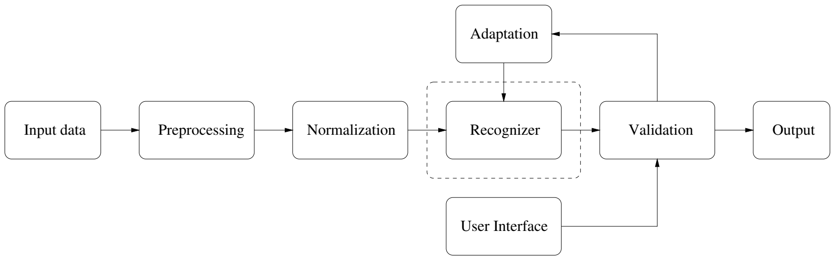


Figure 1: *Architecture of the handwriting recognition system.*

The overall architecture of the recognition system is illustrated in Figure 1. The information flow begins at the data collection device and goes through the preprocessing and normalization units before it reaches the recognition unit. The recognition unit is adapted according to validated recognition result which is the output of the system. Figure 2 describes the recognition unit in more detail.

Various classification techniques have been experimented with. These include Dynamic Time Warping, matching of symbol strings, and non-parametric statistical classification based on the k-Nearest Neighbor rule and the Local Subspace Classification rule.

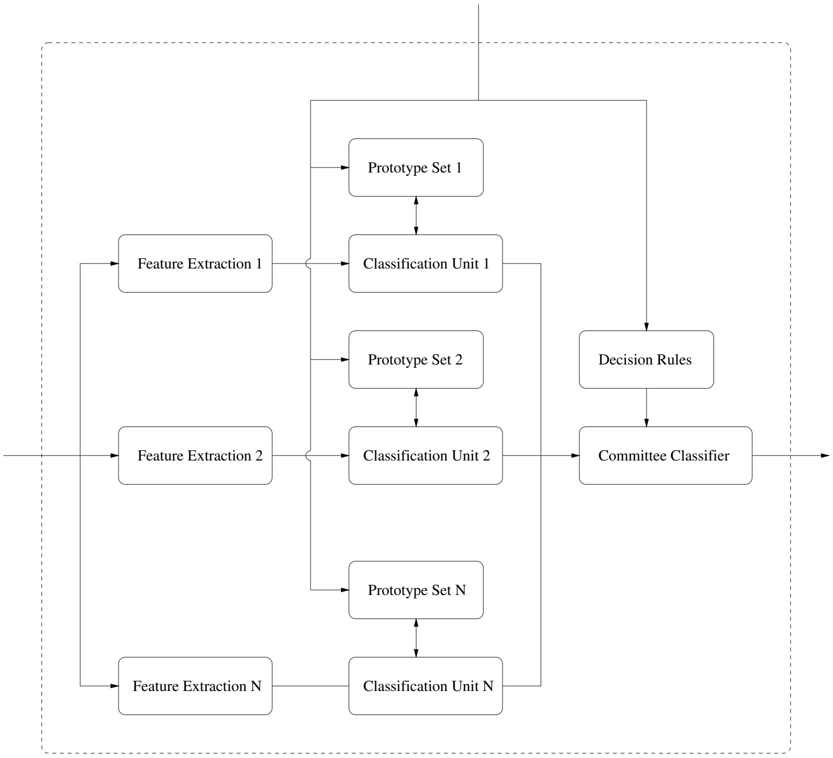


Figure 2: *Architecture of the recognizer unit.*

The recognition system is adapted on-line and in a self-supervised fashion. The correct classes of the input characters are deduced from both the recognition results and the user's actions. The initial prototype set of each classifier is formed by clustering character samples written by several subjects. Therefore, it covers multiple writing styles already at the beginning of the use.

Completely new writing styles can be learned quickly when characters input by the user are added into the prototype set as such. In addition, the existing prototypes can be gradually reshaped so that they better represent the user's style of writing. This adaptation is carried out with an algorithm based on Learning Vector Quantization (LVQ) [5]. Prototypes which are confusing and are therefore more harmful than useful can be inactivated.

The decision rule of a committee classifier is initially as simple as majority-voting. New decision rules are produced according to Dynamically Expanding Context (DEC) principle [4, 16]. This technique adds new decision rules whenever the existing rules fail to correctly classify an input character. The new rules utilize more information on the outputs of the committee members and are thus more specific than the original ones.

Database	*Subjects*	*Left-handed*	*Females*	*Characters*
DB1	22	1	1	$\sim 10\,400$
DB2	24	2	10	$\sim 29\,300$

Table 1: *Summary of the databases used in the experiments.*

3 Data

All character data were collected with a pressure sensitive Wacom ArtPad II tablet attached to a Silicon Graphics workstation. The resolution of the tablet is 100 lines per millimeter and the sampling rate is at maximum 205 data points per second. The loci of the pen point movements consist of the x- and y-coordinates, pen's pressure against the writing surface, and time stamp. The writing area was a rectangle of size 50 mm $\times$ 20 mm placed at the center of the tablet. The characters were written one at a time. Writers were advised to use their natural handwriting style. The data was saved in UNIPEN format [2]. Important details of the databases are summarized in Table 1.

Database 1 consists of characters which were written without any visual feedback. The pressure level thresholding the pen movements into pen up and pen down movements was set individually for each writer. The distribution of the classes (a-z, A-Z, å, ä, ö, Å, Ä, Ö, 0-9, (,), /, +, -, %, $, @, !, ?, :, ., and ,) was somewhat similar to that of the Finnish language. Database 2 was collected with a program which showed the pen trace on the screen and recognized the characters on-line. The minimum writing pressure for showing the trace of the pen on the screen and detecting pen down movements was the same for all writers. The distribution of the character classes (a-z, A-Z, å, ä, ö, Å, Ä, Ö, and 0-9) was nearly even. None of the writers of Database 1 appeared in Database 2. Database 1 was used for forming the initial prototype set and Database 2 was used as a test set.

4 Preprocessing and Normalization Methods

Prior to the classification and adaptation phases, the input characters need to be preprocessed and normalized. Naturally, the characters included as prototypes in the classifiers need to be similarly processed so that they are comparable with the input characters. The preprocessing operations applied in our work are very simple as they were mainly used for finding a suitable sampling method and frequency for the Dynamic Time Warping type classifier.

As the first task, characters are always preprocessed with an operation called *NoDuplicatePoints* (*NDP*) so that the sequential data points having same coordinate values are merged into a single data point. The sampling frequency can be altered with two operations: 1) *Decimate*(n) (*Dec*) keeps every $(n + 1)$th data point and abandons the intermediate ones, 2) *Interpolate*(n) (*Int*) interpolates n equally spaced points between every original data point pair. The former operation reduces both sampling rate and the amount of information in the data. The latter operation only increases the sampling rate as the data points added do not contain any additional dynamic information and are slightly misplaced from the actual smooth path of the pen. The data points can alternatively be made spatially equidistant with *EvenlySpacedPoints*(d) (*ESP*) operation, where d is the desired distance between the adjacent points.

The unknown character and the prototypes are moved into the same location so that they can be properly matched. This is carried out by moving their center points to the origin of the coordinate system. The normalization method *MassCenter* (*MC*) moves the mass center of the character to the origin and the normalization method *BoundingBoxCenter* (*BBC*) does the same to the center of the bounding box. The size variations in the characters are normalized with an operator called *MinMaxScaling* (*MMS*) which scales the size of the character so that the length of the longer side of the bounding box is the same for all characters. The aspect ratios of the characters remain unchanged.

Section 7.1 presents a quantitative comparison between different combinations of the described preprocessing and normalization methods when used together with various dissimilarity methods in the DTW classifier.

5 Feature Extraction Methods

With DTW classifier, no feature extraction methods were used as the matching was done to the coordinate sequences resulting from the preprocessing and normalization operations. The symbol string-based and LSC classification methods, to be described in more detail in Sections 6.2 and 6.3, required specific feature extraction steps. The used features were symbol strings of pen direction for the modified Levenshtein distance-based classifier and principal components of thickened stroke images of the characters for LSC classification.

5.1 Symbol Strings

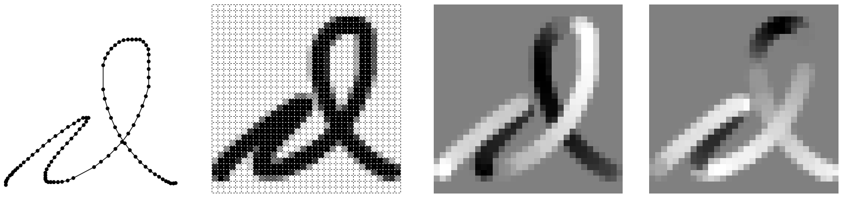
Pen traces were transformed to directional symbol strings for classification. Prior to the actual formation of the symbol string, all strokes in the character were joined. At this point, information on where the pen was off the tablet was stored. The characters were normalized using the *BoundingBoxCenter* and *MinMaxScaling* operators. This resulted in a centered and scaled one-stroke character in a 1000×1000-sized box with the pen-up points marked.

The discretization of the character was performed by setting a minimum length l for each directional symbol and following the pen-trace until this distance was reached, the trace ended, or a pen-up point was encountered. Then, the direction of the resulting vector was calculated and quantized to one of a predetermined number of values. The number d of directions was varied from 4 to 32. The parts of the character with the pen up were marked with single symbols having values different from those with the pen down.

Two approaches to using length information were experimented with. The first was to set for each direction symbol a default length which was the assigned discretization distance. The second approach was to store the actual length of the corresponding line segment together with the direction symbol. Also, a corner detection method, which was most sensitive to changes near the center of the character, was applied. This kind of corner detection approach is most useful due to the fact that the beginning and end parts of handwritten characters are often written less carefully and thus contain less information than the central part.

5.2 Thickened Strokes

In order to form feature vectors of fixed dimensionality and thus suitable for straightforward statistical classification, we devised two feature extraction methods [7]. In these approaches, information about the time sequence of the strokes is not used at all. In the first version, the straight lines connecting the measured xy-points were thickened to the width of $2r$ units in a coordinate system where the image was centered in a 1024×1024-sized frame. The thickening process was carried out by drawing filled circles of radius r along the path of the stylus. The original frame was then downsampled to the size of 32×32 by averaging. Figure 3(a) displays a handwritten character 'd' first in its original form as a sequence of coordinate points connected by straight lines. Figure 3(b) illustrates the same character after downscaling the thickened stroke. The particular value of $r = 50$ has been used. The effect of the averaging in downsampling can be observed as grey shades around the character boundary.



(a) (b) (c) (d)

Figure 3: *Handwritten 'd' in various forms: (a) original points connected with lines, (b) thickened image in 32×32 frame, (c) vertical direction image, and (d) horizontal direction image.*

In the second variation, two 32×32-sized images were created instead of one. The directions of the lines connecting the sampled pen positions were used as additional information when creating the images. In the first one, illustrated in Figure 3(c), the vertical component of the direction of pen movement was used in thickening the path. The filling value was obtained as $f_v = \sin\theta$ where θ is the line direction in polar coordinates. Likewise, the horizontal part $f_h = \cos\theta$ was used in the second image as depicted in Figure 3(d). In both illustrations, white represents positive and black negative values, respectively.

The feature extraction process was in both cases continued by concatenating the pixel values of the grey-scale images. This gave rise to 1024-dimensional pattern vectors in the former and to 2048-dimensional vectors in the latter case. The covariance matrix of the training data set was calculated after feature extraction. The first 64 eigenvectors of the covariance matrix were used in projecting the pattern vectors to a 64-dimensional feature space using the Karhunen-Loève transform (KLT).

6 Classification Techniques

For the classification of handwritten characters, we have used various techniques. These include Dynamic Time Warping (DTW), symbol string matching, and statistical classification of fixed-length feature vectors extracted from the character images. In addition, we have used committee classifiers which have been formed from the above types of individual classifiers.

All the classification techniques have been adaptive. In the case of individual classifiers, this has been realized by adding or modifying the prototypes used by the classifier. In the committee classifier, the adaptation has been implemented by adding new decision rules whenever the existing ones have been unable to produce the correct recognition result. The classification techniques are addressed in more detail in the following sections. Later in Section 9, the implementation of adaptive behavior in each of the classifier types is elaborated.

6.1 Dynamic Time Warping

Elastic matching performed with the DTW algorithm is a nonlinear matching method originally used in speech recognition. It was developed at the beginning of 1970s and was introduced as a method for recognition of handwriting in the late 1970s. Dynamic Time Warping can be used in comparison of all kinds of continuous functions of a continuous parameter, typically time. It is based on the idea that the speed of the process underlying the function can vary. The effects of these variations can be omitted in the comparison of two functions if the parameter axes of the functions are warped. In this case, warping means compressing and stretching of the parameter axes locally. In practice, the functions which are originally continuous are converted into discrete sequences by sampling [15].

In our work, we have used the DTW algorithm for nonlinear curve matching. Classification is carried out by first evaluating the dissimilarity measures between the unknown character and all the prototypes and then applying the k-Nearest Neighbor rule [1]. In the current implementation, the prototypes are pruned according to their number of strokes prior to the matching phase. As a result, only characters with the same number of strokes are matched. In addition to pruning, the prototypes are ordered based on the locations of the starting and ending point of the first stroke. These two techniques improve the efficiency of finding the nearest prototypes for the input character.

The dissimilarity measures based on the DTW algorithm and used in our work are described briefly in Section 7 and in detail in [17]. Section 9.2 describes various ways how the adaptation of the DTW classifier has been implemented in our studies.

6.2 Modified Levenshtein Distance

The distance measure for the symbol strings was based on the Levenshtein distance [12]. This distance measure allows three kinds of operations: replacements, removals and additions. Each of these can be assigned a specific cost function. In our experiments, this distance measure was somewhat modified so that information regarding the neighboring symbols was also used in determining the costs. Extra cost for alteration of symbols referring to input with the pen off the tablet was also added. This modification helps to preserve the information available in the original stroke-based structure. The cost was also dependent on the lengths of the symbols when length information was used. Based on these costs, the actual distance between characters was calculated with a dynamic programming algorithm [15].

6.3 Local Subspace Classifier

The Local Subspace Classifier (LSC) method [6] models the distribution of the pattern classes in a nonparametric fashion by using existing prototypes to span lower-dimensional local subspaces in the feature space. Instead of measuring distances to the discrete prototypes, as with the k-Nearest Neighbor (k-NN) classification rule, the distance is now defined between the input sample and the linear manifold nearest to it.

When calculating the distance between input vector $\mathbf{x}$ and pattern class j, the $D+1$ prototypes belonging to class j and nearest to $\mathbf{x}$ are first searched for. A D-dimensional linear manifold $\mathcal{L}_j$ of the d-dimensional real space can then be spanned by these prototypes. When $\mathbf{x}$ is projected orthogonally onto this manifold, a residual vector $\widetilde{\mathbf{x}}_j$ results. The classification of $\mathbf{x}$ is then performed according to the shortest $\widetilde{\mathbf{x}}_j$ among classes $j = 1, \dots, C$ where C is the number of classes. In any case, the residual length from the input vector $\mathbf{x}$ to the linear manifold is equal to or smaller than the distance to the nearest prototype, i.e. $||\widetilde{\mathbf{x}}_j|| \leq ||\mathbf{x} - \mathbf{m}_{0j}||$. It can be seen that the LSC method degenerates to the 1-NN rule when $D = 0$.

In a modification of the basic LSC method, named the Convex Local Subspace Classifier (LSC+), it is required that $\mathbf{x}$ is projected onto a convex subspace spanned by the prototypes. If the orthogonal projection does not fulfill this condition, prototype vectors are iteratively removed from the basis and the projection recalculated until an orthogonal projection to a convex subset of the nearest prototypes is found. An interested reader can find the details in [6].

6.4 Committees

Instead of single classifiers, a committee classifier can be used [8]. The outputs of a set of classifiers are combined in a committee machine which makes the final classification decision according to its internal rules. The most simple rule is to perform majority voting among the member classifiers and output the most voted class. This majority-voting action may be seen as only a default rule for combining the inputs. When this simple rule fails, adaptation takes place. In our experiments, the adaptation has been implemented by the use of the Dynamically Expanding Context (DEC) principle of Kohonen [4, 16] described in detail in Section 9.3.

In our system, the context upon which the rules operate is formed from the set of outputs from the committee members. First, the members are ranked in the order of individual recognition accuracy, and second, more than one classification output from each member can be examined. When using more than one output from each member classifier, the context can expand in two different ways.

7 DTW Dissimilarity Measures

We have performed experiments with six dissimilarity measures based on the DTW algorithm. The main difference between the measures is the associated cost of matching a data point. The DTW algorithm finds the optimal matching of the data points which corresponds to the minimum sum of the costs and satisfies the boundary and continuity conditions. The continuity condition common to all the dissimilarity measures requires that all the data points are matched and in the same order as they have been produced. The dissimilarity measures are defined on stroke basis. Connected parts of the drawn curve in which the pressure between the pen and writing surface exceeds a given value are considered as strokes. In case of all but one dissimilarity measure, the boundary conditions ensure that the first and last points of two strokes are matched against each other.

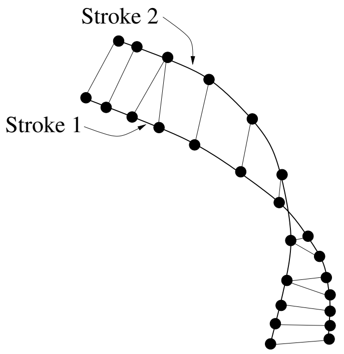


Figure 4: *Optimal matching of data points found by the DTW algorithm.*

The dissimilarity measures are called *Point-to-point* (*PP*), *Normalized point-to-point* (*NPP*), *Point-to-line* (*PL*), *Normalized point-to-line* (*NPL*), *Kind-of-area* (*KA*) and *Simple-area* (*SA*) distances. *PP*-distance uses the squared Euclidean distance between the data points as a matching cost. The optimal matching of two strokes is illustrated in Figure 4 for *PP*-distance. In the case of *PL*-distance, the data points are matched to lines interpolated between the data points. Therefore, the boundary conditions applied with the other dissimilarity measures cannot be used. Instead of matching the first, or the last, data points against each other, only one of them is matched against the line connecting the other one to its neighboring data point. *NPP*- and *NPL*-distances are otherwise similar to *PP*- and *PL*-distances, respectively, but the sums of the matching costs are divided stroke-wise by the number of matchings. Due to the normalization, a pairs of long and short strokes have equal contributions to the total distance. In addition, the increase in the dissimilarity measure caused by different data point densities of the strokes is reduced. *KA*-distance also matches data points against data points but the Euclidean distances from the matched data points to their neighboring data points are considered too. *SA*-distance uses the area between the strokes approximated with triangles or quadrilaterals as the matching cost. These two distances measure the area left between the matched strokes and are therefore more sensitive to the shapes of the strokes than their data point densities. All the DTW-based dissimilarity measures are described in full detail in [17].

Dissimilarity measure	*Dec*	*NDP*	*MMS*	*MC*	E_{ave} %
Point-to-point	2	•	•	•	15.9
Point-to-line		•	•	•	17.1
Normalized point-to-point	1	•	•	•	16.1
Normalized point-to-line	2	•	•	•	17.1
Simple-area		•	•	•	29.8
Kind-of-area(1,0)	2	•	•	•	18.3

Table 2: *The summary of best recognition results for all the DTW-based dissimilarity measures. Notation E_{ave} stands for the average recognition error rate.*

7.1 Comparison

The most suitable combination of preprocessing and normalization methods yielding the lowest recognition error was selected separately for all the DTW-based dissimilarity measures. These selections were carried out in the following way: 1) the prototype set was formed from the characters of Database 1 with a semiautomatic clustering algorithm and each class was represented by seven prototypes, 2) the test set included the lower case letters and digits of the eight first writers of Database 2, 3) the recognizer consisted of a single, nonadaptive 1-NN classifier, 4) the combination producing the lowest recognition rate for all the writers on the average was selected.

Operations *NoDuplicatePoints* and *MinMaxScaling* were found to be beneficial for all the dissimilarity measures. The former operation removes data points which do not contain any additional information on the shape of the character but are caused, for example, by the hesitation of the writer. The latter operation improved recognition rates because the writers were allowed not only to use their own writing style but also to write the characters in any size they preferred to. The selection of the centering method was also straightforward as *MassCenter*-operation yielded better results than *BoundingBoxCenter*-operation for all but one writer.

The experiments showed that neither *EvenlySpacedPoints*- nor *Interpolate*-operation were able to improve the average recognition accuracy. The former operation loses all the dynamical information on the writing process such as the velocity and acceleration of the pen point which are implicitly contained in unevenly distributed data points. The latter operation was quite useless as the original data points were sampled with a frequency high enough. Therefore, the best average recognition result was obtained with *Decimate*-operation or with no additional preprocessing at all.

The results of the experiments are summarized in Tables 2 and 3. The best combination of preprocessing and normalization methods and the corresponding average recognition error rate are shown for each DTW-based dissimilarity measure in Table 2. Note that these are just base-line error percentages, meant solely to compare the preprocessing and normalization. With adaptation, the results improve significantly as will be seen in Section 10. From Table 2 it can be seen that dissimilarity measure *Point-to-point* and the normalized version of it yielded clearly lower average recognition error rates than the other measures. According to Table 3, *Point-to-point* was also the best choice for a dissimilarity measure in case of all but one writer. However, it should be noticed that the variation in the recognition accuracy between the writers was signifi-

Writer	*PP*	*NPL*	*Dec*	*Int*	*NDP*	*MMS*	*MC*	*BBC*	*E %*
DB2:7	•		3		•	•	•		5.7
DB2:6	•		4		•	•	•		5.8
DB2:4	•		2		•	•	•		11.2
DB2:3	•		1		•	•	•		12.2
DB2:8	•		1		•	•	•		16.6
DB2:1	•			1	•	•	•		17.8
DB2:2		•	1		•	•	•		19.0
DB2:5	•		6		•	•		•	24.8

Table 3: *The best combination of the DTW-based dissimilarity measure, preprocessing, and normalization method for each writer. Notation E stands for the recognition error rate.*

cant. In addition, selection of a good parameter value for *Decimate*, which was the best preprocessing method for the clear majority of the writers, is a writer dependent task.

According to the result of these experiments, the best average recognition result for several writers can be obtained if *NoDuplicatePoints*-operation followed by *Decimate*(2)-operation is used as a preprocessing method, characters are normalized with *MinMaxScaling*- and *MassCenter*-operations, and the dissimilarity measure is *Point-to-point*. These settings have been used in the adaptation experiments whose results are presented in Section 10.

8 Creation of Prototype Sets

Depending on the computational requirements of a classification algorithm, it may be necessary that only a subset of the available training data is used in on-line recognition. If such a case, a special prototype set needs to be extracted from the totality of existing character data, e.g. with clustering [3]. In our work, the prototype sets have been formed by first clustering character samples of Database 1 written by several subjects and then selecting the middlemost items of the clusters to present the corresponding styles of writing. We have experimented with three different algorithms for the clustering task.

The first clustering algorithm is semiautomatic. It starts with one cluster containing all the samples. The clusters are split until a predefined count is reached. The number of different writing styles per character class and stroke number variation was manually examined. This clustering algorithm was used to create the sets for the DTW classifiers. In these runs, the number of prototypes was the same for every class, namely seven. The number of clusters per stroke number variation was selected so that it roughly corresponds to the respective share of all the writing styles of that character. The algorithm is explained in detail in [11].

The second clustering algorithm utilizes the opposite approach. At the beginning of the algorithm, there are as many clusters as there are character samples. Next, those two cluster whose middle items are the most similar are merged and the middle item of the new cluster is found. Then again, two clusters are merged into one in a similar way. The algorithm continues until there is only one cluster left. The benefit of this algorithm is that it does not require any prior information on the number of writing styles. In all stages of the algorithm, the user can check if all the center items represent writing styles different enough and decide whether the merging should be stopped.

The first two clustering algorithms were compared by creating equally-sized prototype sets with each. Database 2 was then used to evaluate the recognition accuracy of the DTW classifier with both prototype sets. This experiment showed that there were no real differences between the two clustering algorithms. The recognition results seemed to be more dependent on the data than the algorithm used in the clustering process.

The third clustering technique used was the traditional K-means algorithm [13, 14]. It was used to form the initial user-independent 1-NN classifier needed in the LSC experiments. A set of typical representatives for each character class were selected with it. The value for K was varied in the experiments between 1 and 10, thus resulting to prototype set sizes between 39 and 390. In the symbol string-based recognition, the computational requirements were lowest. Therefore it was feasible to use the entire Database 1 as the prototype set and a special prototype extraction phase was unnecessary.

9 On-line Adaptation Techniques

The key feature of our recognition system is its ability to adapt to a new writing style. This can be achieved in two fundamentally different ways. Adaptation is carried out on-line either by modifying the prototype sets of the separate classification units or the decision rule of the committee classifier. The adaptation process is supervised without any direct interaction with the user. Instead, the correct classes of the input characters are inferred from the recognition results and the user's reactions to them. In the following sections, such a supervision scheme together with the user interface, and adaptation strategies both for prototype sets and decision rules are described.

9.1 Supervision of Adaptation

We assume that the device in which an on-line character recognition system has been implemented has an input subprogram with the following properties: 1) Input characters are written into the desired positions on the display. Alternatively, the user first selects the input position by pointing it with the pen and then writes the characters into a special writing area. 2) Handwritten characters are recognized and replaced by the machine-printed recognition results right after they have been input. 3) Recognition errors and writing mistakes are corrected by inputting a new character on top of the machine-printed character. The recognition result of the latest input character is assumed to be the correct class for all the characters drawn into the same position. The validation unit of the recognition system (see Figure 1) takes care of the labeling of the input characters according to this assumption. 4) All input characters are stored. 5) The adaptation is carried out character by character after a whole text sequence, for example a line, has been accepted by the user.

As recognition errors and writing mistakes are corrected in the same way, erroneous learning situations of three types can take place. First, the user may not notice all the recognition errors or does not care to correct them. Second, if the user makes writing mistakes and corrects them, some of the learning samples will be incorrectly labeled. Third, carelessly written and thus atypical or malformed characters might be used as learning samples. However, there were no malformed learning samples in our experiments as the data had been manually examined and cleaned.

9.2 Adaptation of Prototype Sets

With the DTW-based classifiers four different prototype set adaptation strategies have been applied. They are named as *Add*, *Inactivate*, *Lvq*, and *Hybrid* [17, 18]. Adaptation strategy *Add*(k) examines the classes of the k prototypes nearest to the input character. The classification is carried out according to the k-NN rule. The input character is added to the prototype set if any one of these prototypes belongs to a wrong class, even if the classification was correct. Adaptation strategy *Inactivate*(N) is used for inactivating those prototypes which are more harmful than useful. After each recognition, *Inactivate*-strategy checks if the prototype nearest to the input character has been the nearest one at least N times and whether its class has been incorrect more often than correct. In that case, the prototype is removed from the set of active prototypes.

When a character written by the user is basically similar to a prototype of the correct class, for example it has the same number and order of strokes, but of slightly different shape, the existing prototype can be reshaped instead of adding the input character to the prototype set. This can be performed with an adaptation strategy called *Lvq*(α) based on a modified version of Learning Vector Quantization (LVQ) [5, 9]. Parameter α controls the degree of reshaping. If the value of α is near to zero, the nearest prototype is reshaped only slightly. With larger values of α the modifications to the prototypes have more effect. Adaptation strategy *Hybrid*(α,k) combines the *Add*(k) and *Lvq*(α) strategies. The k nearest prototypes are examined. If any one of them belongs to the same class as the input character, the nearest prototype is modified with *Lvq*(α). Otherwise, the input character is added to the prototype set.

In the Local Subspace Classifier experiments we started with a user-independent 1-NN classifier created from Database 1 with the K-means algorithm. For each writer, the user-dependent LSC prototype set was initially empty. The adaptation of the LSC classifier was then performed according to two distinct rules controlling the inclusion of the input character into the classifier. The ‘E’ rule stated that the prototype was added only if the LSC classifier had misclassified the input. The ‘A’ rule forced the addition of every input character. Every input character was classified with both the user-independent 1-NN classifier and the adaptive user-dependent LSC classifier. The joint classification decision of the two was given by the one with shorter distance to either to the nearest prototype or the nearest local subspace, respectively. This was possible as the both types of classifiers are based on the Euclidean distance metrics and measure the residuals in same units. If the class provided by the 1-NN classifier was incorrect, the corresponding prototype in the K-means-initialized prototype set was removed. The input character was added to the LSC prototype set according to either of the ‘A’ and ‘E’ rules. As a result, the size of the 1-NN classifier decreased while the size of the LSC classifier increased during the adaptation. As a consequence, the classification decisions were increasingly determined by the latter.

The symbol string-based classifier used similar ‘A’ and ‘E’ rules as the LSC classifier in adaptation. A notable difference between the two methods was that the prototype set of the string-based classifier was initialized by using all samples in Database 1 instead of a K-means-clustered subset. Also, the initial prototypes were never removed even when they caused false recognitions.

9.3 Adaptation of Committee Decision Rules

We have used the principle of Dynamically Expanding Context (DEC) [4, 16] to implement the adaptation of a committee classifier. This technique adds new decision rules whenever the already existing rules fail to correctly classify an input character. The new rules utilize more information on the outputs of the committee members and are thus more specific than the original ones. The general principle of DEC can be formulated as a set of production rules of the form $x(A)y \rightarrow (B)$, where A and B are the input and output symbols, respectively, and x and y are the left and right contexts, respectively, of the input symbol. The combined length of the x and y contexts determine the level of the rule. Each time a rule is found to be in conflict with the actual transformation needed, a new higher-level rule is added. This new rule is, due to the increased amount of context involved, not conflictive.

The DEC principle has been somewhat modified for our current purposes in on-line recognition of handwritten characters. In our setting, there are a set of individual DTW-based classifiers for recognition of handwritten characters. The classifiers have been first initialized and then ranked in the order of decreasing recognition performance. These classifiers are then used to form a committee classifier and the modified DEC principle is used to create the production rules for the committee. The outputs of the member classifiers as well as the second-ranking recognition results from each of them are used as one-sided context when forming the DEC rules.

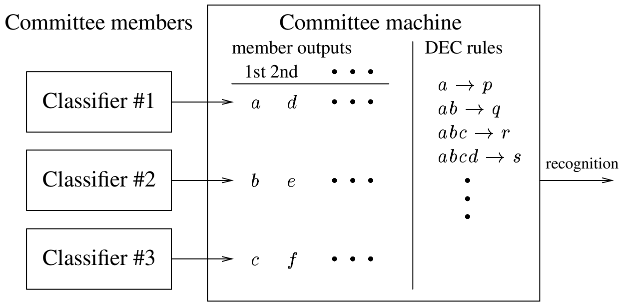


Figure 5: *The basic setting of the DEC-based adaptive committee classifier.*

Figure 5 displays a schematic diagram of the DEC-based adaptive committee classifier. In the illustration, there are three member classifiers. The first rank outputs from the classifiers are denoted by a, b, and c. Likewise, the second rank outputs are denoted d, e, and f, respectively.

Each time a new character has been input to the system, the outputs of the member classifiers are compared to the existing DEC rules. If no match is found, a default decision is applied. This default action can be, e.g. to use the first output of the best individual classifier. If one or more rules match the situation, the highest-level one, i.e. the one with the largest context, is applied and the output symbol specified by the rule is used. If the recognition result is then found to be incorrect, a new rule with more context is included in the rule base. Eventually, as new DEC rules are being added, all the available context information will be used by the rules. All error situations thereafter call for additional rules but the context cannot be expanded anymore. Therefore, it is allowed that there exist more than one highest-level rule for a single context. In this case, the number of correct applications is maintained for each rule. The rule with the highest correctness value is then selected.

We experimented with some variations in the setting of the committee classifier. These variations included: 1) The default rule for cases when there were no applicable rules in the rule set was (a) to obey the opinion of the best individual classifier, or (b) to perform majority voting among the members. 2) It could be demanded that in every DEC rule $A \rightarrow x$, $x \in A$. This means that at least one of the symbols in the context needs to be correct in order to produce a new transformation rule. In the opposite case, this constraint was not enforced. 3) The size of the context could be made fixed. This means that when the number of committee members was four and the context size was fixed to four, the intermediate level two and three rules were not generated at all. Instead, the level four rules were used right from the first error. Context sizes smaller than four were also allowed. These selections reduced the actual number of committee members. 4a) Either only the first-ranking outputs from the committee classifiers were used, or (b) also the second-ranking results were utilized.

10 Results of Adaptation Experiments

This section summarizes the results of all the experiments made with adaptive recognizers. The results are grouped in Table 4 so that in every group the first line shows the results of a comparable nonadaptive classifier. Then, results with different adaptation strategies or parameter values are shown for each recognition method. The recognition error rates shown have been obtained as the average of the results of the last eight subjects in Database 2. Two error rates are shown. First the *total* error percentage measured for each writer during the whole test run, typically about 500 characters. This figure reflects the initial error rate and, when compared with the nonadaptive results, the speed of adaptation. The *final* error rate was evaluated for the last 200 characters of each writer. It thus gives better impression of the obtainable recognition accuracy after adaptation. It was noted during the experiments that the writing style of some subjects got rather poor during the last characters due to fatigue and lowered motivation. This can also be observed in the results of some classifiers where the *final* error rate is higher than the *total* rate.

In the DTW experiments, suitable values for the parameters of *Add*(k), *Inactivate*(N), *Lvq*(α), and *Hybrid*(α,k) strategies were selected by using data from the first sixteen writers of Database 2 as a test set. The best results were obtained with *Add*(k) strategy when $k = 4$, and with *Hybrid*(α,k) strategy when $k = 3$ and $\alpha = 0.3$. The same value

recognizer	*error rate %* *total*	*final*	*# of units* *start*	*end*
DTW	14.1	14.1	273	273
DTW-*Add*(4)	3.1	1.8	273	453
DTW-*Lvq*(0.3)	9.9	8.6	273	273
DTW-*Add*(4)+*Inactivate*(3,0)	3.0	1.6	273	450
DTW-*Hybrid*(3,0.3)	4.2	2.5	273	278
DTW-*Hybrid*(3,0.3)+*Inactivate*(16,0)	4.3	2.8	273	278
SS(d=32,l=15)	26.1	27.1	8461	8461
SS-E(d=32,l=15)	15.2	13.4	8461	8549
SS-A(d=32,l=15)	10.5	7.6	8461	9041
1-NN(K=10)	39.0	42.1	390	390
1-NN-E(K=7)	22.0	19.0	273	346
1-NN-A(K=7)	16.1	11.2	273	796
LSC-E(K=10,D=4)	18.6	13.9	390	483
LSC-A(K=9,D=4)	13.5	8.1	351	895
majority voting	14.6	15.9	1	1
adaptive committee reference	14.5	15.0	1	1
DEC(b, inc, 2nd v)	11.6	11.3	1	41
DEC(b, inc, 2nd h)	11.7	11.8	1	16
DEC(v, 2nd v)	12.0	12.0	1	70
DEC(b, 2nd v)	11.1	11.1	1	64
DEC(v, inc, 2nd v)	12.5	12.0	1	46
DEC(v, inc)	12.9	13.5	1	18
DEC(b, inc)	12.7	13.4	1	17

Table 4: *The results of all adaptive recognition experiments together with their non-adaptive reference results. The last two columns show the numbers of prototypes or committee rules before and after adaptation. All figures are averages of the results of the last eight subjects in Database 2.*

of α worked also best with *Lvq*(α) strategy. *Inactivate*(N) strategy did improve the recognition accuracy only when it was applied with *Add*(k) strategy. In that case, the best value for N was 3. The best recognition result in the whole series of experiments was obtained with DTW when adaptation strategy *Add*(4) was used together with *Inactivate*(3,0). It can also be seen that the results of the DTW classifier are clearly superior already in the nonadaptive case when compared with the other single classifier methods. The superiority of the DTW-based methods is at clearest in the case of the *final* error rate.

We found out that the symbol string classifier produced its best results when using $d = 32$ directions and the discretization distance of $l = 15$. Using both the actual length of the symbol and the corner detection technique proved to be beneficial for the classification accuracy. The 'A' rule of adaptation produced only about one half of the *final* errors the 'E' rule made. Further, the nonadaptive version's error rate was approximately twice that of the 'E' rule. As we could due to the computational lightness of the symbol string classifier use the whole Database 1 as the user-independent initial prototype set, it might be possible to obtain better nonadaptive and initial recognition rates if we would have more character samples to start with.

In the LSC experiments we found out two facts: First, the two proposed stroke thickening methods for feature extraction performed equally well. Therefore it was reasonable to use the first one as it was easier to implement and use. Second, LSC+ algorithm did not perform better than the simpler LSC algorithm. For efficiency reasons, it was then justified to use the latter. Therefore, Table 4 only displays results for the first thickening method and LSC. The dimensionality d of feature vectors was selected experimentally by decreasing it gradually from 64, which was the dimensionality of data after the Karhunen-Loève transform. The best result with nonadaptive 1-NN classifier was obtained when d was 45. This value was then exclusively used. The optimal value for K, the number of initial user-independent character prototypes per class, was selected individually for each method between 1 and 10.

The table first shows the result for the nonadaptive 1-NN classifier and then an adaptive 1-NN classifier when both the 'E' and 'A' adaptation strategies have been used. This classifier was formed similarly to the adaptive LSC classifier, i.e. user-independent prototypes were removed and user-dependent ones added. It can be seen that the LSC-based adaptive classifier outperforms the adaptive 1-NN classifier in both the 'E' and 'A' cases. For both classifiers, the 'A' strategy seems to be better than the 'E' strategy. Also, in all cases the adaptive classifiers are clearly better than the nonadaptive one. The numbers of final prototypes in the last column of the table are the sums of the numbers of the remaining user-independent and the added user-dependent ones.

In the committee classifier experiments, we formed four individual classifiers which were user-independent and nonadaptive. They all were DTW-based and formed all combinations of the *MassCenter* vs. *BoundingBoxCenter* normalization methods and *Normalized point-to-point* vs. *Point-to-line* distance measures. The *total* error rates of the member classifiers varied between 15.0 and 19.7 percent. For the committee experiment there are two reference results. First, a simple static majority-voting scheme for the first-ranking member outputs. The second reference method was adaptive but it used a very simple adaptation rule: A count of the number of correct recognitions was maintained for each member classifier. Every input character was then classified according to the opinion of that classifier which at that particular moment had the highest success count. This lead to a sort of adaptive selection of the best single classifier for each individual test subject.

A selection of best results with different variations of DEC settings are displayed in the table. Notations 'b' and 'v' stand for best single classifier and majority voting, respectively, as the default decision rule. The existence of 'inc' option indicates that the output symbol was required to belong to the context in the rules. '2nd v' means that the first-ranking outputs of all the member classifiers were used as the context before using the second-ranking output from the best member. On the contrary, '2nd h' indicates that the second-ranking output of the best member was used before the first-ranking output of the second best single classifier, and so on. In all cases, it was better to use expanding contexts than to fix the context size to some value between 1 and 8. It can be seen that the average number of resulting rules varies considerably between the DEC variations. For example, with only 16 rules it was possible to obtain the third best results. Generally, however, variations which produced more rules also produced better results. All the DEC-based committees outperformed the nonadaptive and adaptive reference committees. Also, all committees were superior to all the single nonadaptive member classifiers.

11 Implementation Issues

Until recently, the recognition system described above has been running on large-scale UNIX systems with generous computational and memory resources. The existing system has been used for both batch runs and on-line recognition testing successfully. In order to collect experience on the use of adaptive on-line recognition, we are now implementing the system in a real PDA.

Our testing equipment for this purpose is a hand-held device running Windows CE on a MIPS R3000 processor with restricted memory and storage capabilities. Due to the limited computation power, some modifications have been made to the recognizer system. Most significant alterations necessary for the portable implementation include a drastic cut-down in floating-point calculations, which due to the lack of a specific floating-point unit are extremely cumbersome in the small-scale platform. The DTW-based recognition routine was altered to function fully by integer calculations, resulting in a more than ten-fold speedup for the actual recognition routine at negligible cost to recognition accuracy.

Also the need to investigate specific speed-optimization approaches to the DTW-based *Point-to-point* distance measure primarily used in the recognition system arose from these difficulties. The studied speed-optimization approaches included a method to predict the final cost of the match at the start of the calculations, using a pre-classification algorithm comparing the lengths of the strokes to be matched, and using stricter continuity rules in the DTW algorithm. All of these approaches seem to be able to produce notable gains in speed while impairing the recognition accuracy of the system very little. Still, the actual implementation in the smaller platform is yet to be tested. Currently, the implementation seems promising, as the recognition accuracy is at the same level as in the large-scale platform and the recognition speed is also approaching an acceptable level.

12 Conclusions and Future Directions

In this paper, we have demonstrated that there exist various ways to create adaptive recognition systems for on-line handwriting. In all experiments, adaptation was able to increase the accuracy of recognition. As a whole, the results obtained with methods based on the Dynamic Time Warping classifier were superior to others. It can be stated that DTW was the only technique that yielded average final error rates low enough to be acceptable in a real-world PDA application. The average final error rate of 1.6 percent can also be considered as negligible compared to the probability of typing errors and spelling mistakes. Still, as the DTW approach is computationally more demanding than its competitors, there may exist situations where some other method has to be selected due to implementation constraints.

Adaptive committees may prove to be a useful classification technique, if 1) we can find a set of member classifiers which individually have sufficiently low error rates, and 2) the errors of the member classifiers are mutually uncorrelated enough. In our current experiments, the members of the adaptive committee were all based on the DTW classifier. Therefore, their errors were supposedly too dependent. Also, the committee members could be adaptive classifiers by themselves. Such a doubly-adaptive system may, however, turn out to be somewhat instable.

Currently, we are implementing the system in a palm-size PC. It will be used in collecting new data and performing usability evaluations. Meanwhile, we are running analyses on the sensitivity of different adaptation strategies to erroneous training samples. More advanced methods for detecting erroneous data and recovering the system from the effects of bad learning samples are being devised. Also, we are experimenting with new techniques for implementing the adaptation. These should allow the adaptation process to automatically become inactive and active again depending on the changes in the system's performance.

References

[1] T. M. Cover and P. E. Hart. Nearest neighbor pattern classification. *IEEE Transactions on Information Theory*, 13(1):21–27, January 1967.

[2] I. Guyon, L. Schomaker, R. Plamondon, M. Liberman, and S. Janet. Unipen project of on-line data exchange and recognizer benchmark. In *Proceedings of International Conference on Pattern Recognition*, pages 29–33, 1994.

[3] J.A. Hartigan. *Clustering Algorithms*. John Wiley & Sons Inc., New York, 1975.

[4] Teuvo Kohonen. Dynamically Expanding Context, with application to the correction of symbol strings in the recognition of continuous speech. In *Proceedings 8th International Conference on Pattern Recognition (8th ICPR)*, pages 1148–1151, Paris, France, Oct. 27–31 1986.

[5] Teuvo Kohonen. *Self-Organizing Maps*, volume 30 of *Springer Series in Information Sciences*. Springer-Verlag, 1997. Second Extended Edition.

[6] Jorma Laaksonen. Local subspace classifier. In *Proceedings of the International Conference on Artificial Neural Networks*, pages 637–642, Lausanne, Switzerland, October 1997.

[7] Jorma Laaksonen, Matti Aksela, Erkki Oja, and Jari Kangas. Adaptive local subspace classifier in on-line recognition of handwritten character. In *Proceedings of IJCNN'99*, Washington, D.C., July 1999.

[8] Jorma Laaksonen, Matti Aksela, Erkki Oja, and Jari Kangas. Dynamically expanding context as committee adaptation method in on-line recognition of handwritten latin characters. In *Proceedings of ICDAR'99*, pages 796–799, Bangalore, India, September 1999.

[9] Jorma Laaksonen, Jarmo Hurri, Erkki Oja, and Jari Kangas. Comparison of adaptive strategies for on-line character recognition. In *Proceedings of International Conference on Artificial Neural Networks*, 1998.

[10] Jorma Laaksonen, Jarmo Hurri, Erkki Oja, and Jari Kangas. Experiments with a self-supervised adaptive classification strategy in on-line recognition of isolated handwritten latin characters. In *Proceedings of the 6th Workshop on Frontiers of Handwriting Recognition*, pages 475–484, Taejon, Korea, August 1998.

[11] Jorma Laaksonen, Vuokko Vuori, Erkki Oja, and Jari Kangas. Adaptation of prototype sets in on-line recognition of isolated handwritten latin characters. In Seong-Whan Lee, editor, *Advances in Handwriting Recognition*, pages 489–497. World Scientific Publishing, 1999.

[12] V.I. Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Sov. Phys.-Dokl.*, 10(8):107–710, February 1966.

[13] Y. Linde, A. Buzo, and R.M. Gray. An algorithm for vector quantizer design. *IEEE Transactions on Communications*, COM-28(1):84–95, January 1980.

[14] J. MacQueen. Some methods for classification and analysis of multivariate observations. In *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability*, pages 281–297, 1967.

[15] D. Sankoff and J. B. Kruskal. *Time warps, string edits, and macromolecules: the theory and practice of sequence comparison*. Addison-Wesley, 1983.

[16] Kari Torkkola. *Short-Time Feature Vector Based Phonemic Speech Recognition with the Aid of Local Context*. PhD thesis, Helsinki University of Technology, Espoo, Finland, 1991.

[17] Vuokko Vuori. Adaptation in on-line recognition of handwriting. Master's thesis, Helsinki University of Technology, 1999.

[18] Vuokko Vuori, Jorma Laaksonen, Erkki Oja, and Jari Kangas. On-line adaptation in recognition of handwritten alphanumeric characters. In *Proceedings of International Conference on Document Analysis and Recognition (ICDAR'99)*, 1999.