Rijksuniversiteit Groningen Faculteit der Wiskunde en Natuurwetenschappen

Vakgroep Informatica

Automated Signature Identification and Verification

Herman Dullink

Begeleiders:
Prof.dr.ir. L. Spaanenburg
Dr. J.A.G. Nijhuis

RuG

SUMMARY

Security in commercial traffic has become a major socio–economic concern. One method could be signature authentication. In this report we scan the literature on authentication systems and methods. Then we perform some verification experiments to achieve a first impression of whether and how it is feasible to identify a person through his signature. Finally we discuss some options for future developments.

SAMENVATTING

In de moderne samenleving is een sterke behoefte aan electronische beveiliging ontstaan. Bij bancair verkeer is daarbij het automatisch (h)erkennen van de handtekening in discussie. Dit rapport geeft een korte inleiding tot deze materie, waarna enkele neurale uitvoeringsvormen verder onder de loupe worden genomen.We sluiten af met enige suggesties voor verder onderzoek.

PREFACE

Automated identification and verification based on the personal hand–written signature seems an attractive alternative for the (usually) four digit PIN system as used on many bank transaction and security systems. Banks are constantly on the look–out for an alternative to PIN–codes to reduce the susceptibility of their transaction systems to potential fraud.

Available interactive pen–based input devices allow automatic identification and verification systems to examine the hand–written signature generation process. Such a system should then be able to verify the dynamic features as the duration, tempo and motions of the generation process as well as the static features as size, shape and aspects of the signature. Signature verification systems are based on the assumption that a number of the dynamic features of signatures are reproducible and unique to a signer.

However, several (psychological and physic) analyses and experiments show that over 30 years the hand–writing process is not a (very) stable one, and that the signature generation process is even worse. Also one's signature deteriorates from many (subtle) changes over the years. Even the stability of the hand–writing process varies from time to time and seems to be dependent on the individual's condition and mood.

When extracted features from several signatures sampled from one individual are compared against each other, large relative deviations are found. By introducing an allowance range at the decision, the chance that a signature from a legal user will be rejected by a verification system can be reduced. This, however, results to forged signatures and even random signatures to become acceptable as genuine ones.

To overcome these problems, researchers are constantly searching for features and comparison techniques that are stable and discriminant enough for a (close to) zero fail verification. The better results are achieved using personalised feature sets, personalised allowance ranges, and many sample signatures as reference. Still, these systems are far from perfect. For zero acceptance failing, one has to sacrifice 100% forgery detection and vice versa.

Judging from the results found in several articles and by my own experiments, I find the current technology not suitable for automatic identification and verification. As it fails to reveal the psychomotor system unless a level of fraud is expected, it can never detect uniqueness. This can be overcome by using hand–written (code) words or sentences instead of personal signatures. Normal handwriting is considered a more stable process than generating signatures. As with PIN, code words or sentences can be kept secret while the personal signature can be found on many letters, (official) documents and, for example, the bank or credit card needed for the transactions.

This alternative os a totally new technology, that starts by considering the signature as the result of the psychomotor system as a dynamic system with chaotic behaviour. We have not traced significant literature in this direction and fear that a lot of research remains to be done.

H. Dullink
Groningen, 30.8.96

TABLE OF CONTENTS

1. INTRODUCTION

Already quite some research is done in the field of automated processing of handwriting. The main targets seem to be banking and security systems on one side, and document processing on the other. As many document storing and processing systems are getting automated and digital, an automatic system transferring hand–written documents to electronic documents is a great help to archivists. This document however is mainly about hand–written signatures. Hand–written signature verification is considered superior to many other biometric authentication techniques as finger prints or retinal patterns.

Banks are constantly examining and searching for verification systems that could replace the widely used Personal–Identification–Number (PIN) system. Especially senior citizens are having hard times remembering the (usually four digit) PIN of their bank card, having worse times when using several cards with each an unique PIN. Some attractive applications are based on the personal hand–written signature. In fact, the personal hand–written signature is the traditional form of identification on, for example, cheques.

Where security systems should be very hard to break, a banking system should be very easy to use. A system refusing a transaction because a valid identification input isn't recognized as such is not likely to be accepted by a Bank, even if the probability of such a 'failing' is too small to write down as a percentage. Having millions or billions transactions a day this 'too small' probability will result in thousands 'failings' per day.

1.1. Studies and terms

The studies of handwriting and automated systems are categorized in Fig. 1. (Optical) character (re)cognition systems refer to man–computer interfaces processing the 'semantic' information of text to recover its contents irrespective of the author. Practical examples are digitizing or scanning of documents and pen–computers. As hand–written signatures are the subject in this document, these studies and systems will not be discussed here.

For documents with unknown origin, retrieving the author of the document is often desired. The same can be said about signatures. Especially when a signature is identified as a fraud. Signature authentication has a close relation to writer identification. Here one would only desire to know whether or not the signature is authentic.

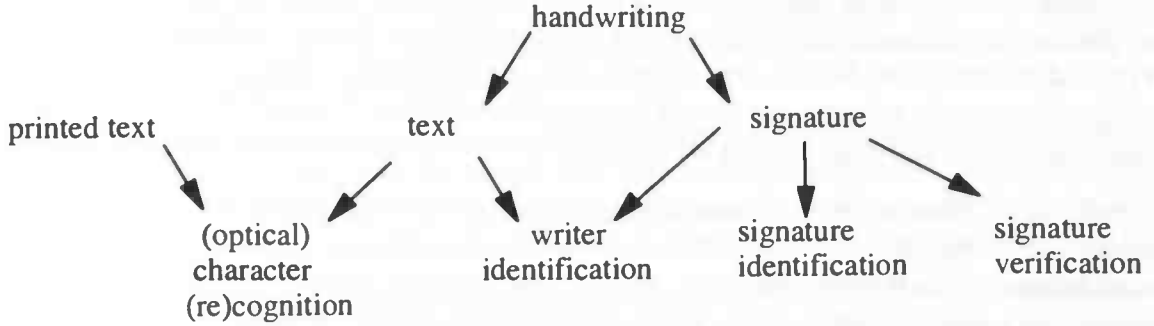


Fig. 1: Overview of current handwriting studies

The study of signature authentication is omitted in Figure 1 as most articles I've read use the term signature verification as in 'verifying authenticity'. The subtile differences between the two terms are that small I'll use the term 'signature verification' as a general term.

> *In my opinion, signature verification and signature identification deal with the signatures only, in contrast to the writer identification and signature authentication where the generation process is dealt with.*

Signature identification is also put into this figure just to make it complete. I found only one reference to this subject. I suspect this is because of the little practical use of an automated signature identification system.

Two terms often used in this field are on–line and off–line. An **on–line system** implies the data acquisition happens at the moment text or the signature is produced, usually using a dynamic (interactive) input device such as an electronic pen, drawing tablet or digitizer. Although dynamic input devices are used, on–line does not imply the system actually uses the dynamic features of the acquired data. I found many studies using static features only.

On–line writer identification techniques are not well developed. Maarse and colleagues [14] reported an experiment where twenty subjects were asked to write the same eight lines of text. From these 160 lines, half were used for training, the rest for discriminant analysis based on several static and dynamic features. No studies on using hand–written signatures for writer identification are found.

An **off–line system** implies the data acquisition happens after the moment text or the signature is produced. The signature is then scanned into the system as an binary or photographic image using a scanner or camera.

Two other terms used are static and dynamic. **Static features** usually tell about the shape of the signature; **dynamic features** tell about the process of signature generation. Which features are static or dynamic can be quite obvious (size, aspects, velocity, acceleration, jerk) to matter of opinion (duration of generation and pauses).

1.2. The handwriting process

One name that pops out in the field of signature verification is Réjean Plamondon from the Ecole Polytechnique, Université de Montréal, Canada. Together with Guy Lorette and Franck Leclerc, he has produced several articles, including two [11][17] which give a very complete reflection of studies in the field of automated signature verification and writer identification.

The design of a signature verification or a writer identification system is based on the fact that people do not write according to a standard penmanship, and that the deviation from the norm is individual dependent. Which parts of the process are writer dependent and writer independent, which way specific information should be extracted and how it is reflected in local or global characteristics of an image or a signal are questions which remain unanswered.

According to Plamondon, fast handwriting is considered a biophysical process, a ballistic movement controlled without instantaneous position feedback. Some central nervous mechanisms within the brain activates, with a predetermined intensity and duration, the proper muscles in a predetermined order. Where the boundary stands between fast handwriting and slower handwriting processes, where position and visual feedback would probably apply, is unclear.

Several conceptual, physical and empirical models have been developed to analyze the complexity of the handwriting generation processes. From these models, handwriting can be viewed as the output: $O(t) = [\Sigma(t), \Theta(t), \Gamma(t)]$ of a space–time variant system, described by the curvilinear displacement ($\Sigma(t)$, the angular displacement $\Theta(t)$ and the torsion $\Gamma(t)$ of the trajectory according to the theory of intrinsic parametrisation of curves.

Another important name is Fransiscus Johannus Maarse [14]. Within the Human Performance section of the Department of Experimental Psychology of the Catholic University of Nijmegen (The Netherlands), Maarse and colleagues have carried out several studies on the aspects of handwriting as a form of psychomotor behavior. Maarse's studies [14] include the study of the motoric aspects of handwriting involving the modelling of peripheral processors which relate to the muscle–joint systems by which the actual handwriting movements are produced and to the motor commands to these muscle–joint systems. The peripheral motor commands are under the control of a higher, more central motor program. The results show that natural handwriting can be simulated more adequately by models in the velocity domain.

From his psychomotor analyses, he found the ballistic movement not being entirely stable. A standard deviation of more than 10% is found in the patterns which are the result of writing down one and the same letter several times by one and the same author. He also carried out studies on signal processing of handwriting movements. From his Fourier analysis of handwriting signals it appeared that handwriting does not contain (many) harmonics. The most important frequency components are found around 5 Hz, while nearly all components are confined to frequencies below 10Hz. This suggests that parts of the handwriting signal may be approximated by a phase–modulated signal.

1.3. An attempt to standardize

Sofar we have seen a lot of confusion about what is actually meant by identification and verification. Though we will not try to change the world, discussing this subject with other people indicates that a strict and consistent usage of such terms would be beneficial. From the dictionary we receive the following descriptions:

- **identification** is the process to determine a complete agreements between two (sets of) objects.
- **verification** is the process to determine the unique agreements between two (sets of) objects.

This still leaves room for interpretation. We see it thus that verification includes identification (being a further prove on an identified object) but that the opposite is not true: an identified object is not necessarily verified. Suppose we have a box with objects. We can then classify on roundness to identify the marbles but we need additional information to verify that the identified marble has been put in by a specific person.

Off–line. Let's take first the image of a signature. The drawing has been created in the past and we can work on the resulting picture. In this way we can only identify this picture as looking like one or more of the previously stored pictures. Without additional measures such as making sure by legal means that a signature is uniquely connected to a person, we can not verify it.

On–line. If the signature is handled while being written we can fetch dynamic information next to the picture. Assuming that this dynamic information is unique for a person, we can both identify and verify the signature. This is what is understood by signature verification, but the assumption has not been verified in this respect that sofar the unique features have not been identified.

On–going. Though not pursued in this report, we like to introduce here a third way in which we can rely mostly on the dynamic information. We see the sampled signature as a time–series. By this means we can in principle identify the psychomotor system itself and use the signature only for further verification.

1.4. Project Description

In order to acquaint ourselves with this field of research and to verify the intricacies of the various proposals, we set out to construct a low–cost system from a small digitizer and a PC. Once a suitable driver is written we will conduct a number of experiments, largely aiming for the use of neural classification techniques. The actual learning will be performed by InterAct on an HP–workstation.

The reason why we confine outselves to an experimental set–up instead of claiming a potential solution lies in the relatively unchartered territory we are entering. Despite the fact that there are partial solutions and lengthy reviews, no all–encompassing solution on a strictly scientific basis is at hand. The reason is most probably the high–dimensionality of the problem.

Suppose there are 60 milliard persons to be authenticated, then the solution space could be seen as a binary (yes/no) encoding with 25 variables. The problem space is a multiple of this as (a) every person has an everchanging signature, and (b) a signature is constructed of an unknown number of partially known elements. Moreover in the problem space the variables are not mutually independent. Sofar the full extent of this classification problem has not been tested, as only a limited amount of regionally collected signature sets have been used.

In the following, that part of the project as performed in partial fulfillment of the M.Sc. degree is documented. First the structure of a typical signature verification system is treated. Then we go into the details of the various attempts to verify a signature by neural means. Ensuing we describe some of our experiments and finally draw some conclusions.

2. SYSTEM ARCHITECTURE

Plamondon [18] and Pirlo [17] have developed the architecture for a general signature verification system as shown in Fig. 2. As seen in this figure, four types of problems have to be solved: (a) data acquisition, (b) pre–processing, (c) feature extraction, (d) the comparison and decision process. Any part of the system is equally important as every next part relies on the previous one.

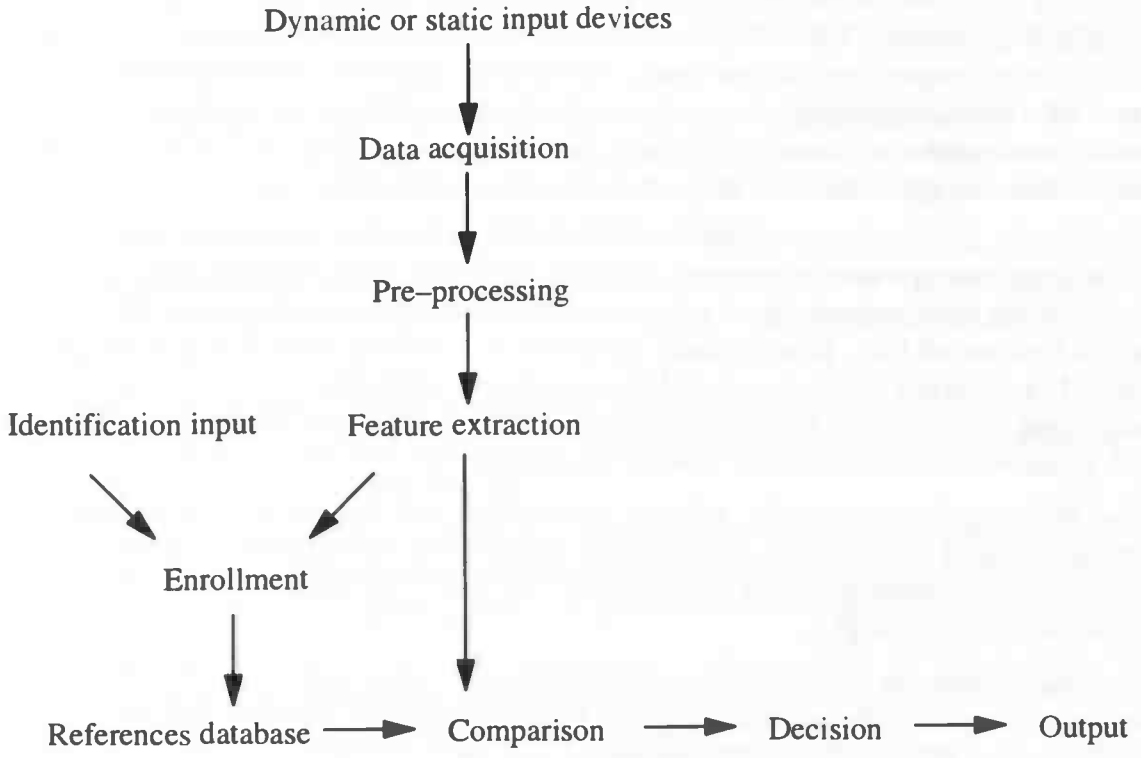


Fig. 2: Dataflow diagram of a general signature processing system

The raw data acquired are pre–processed to remove spurious information, to filter the significant signals or images, and to validate acquisition. A further aspect is the normalization of the signature in order to provide input to the ensuing feature extraction in a standardized manner.

In the feature extraction process, specific functions or parameters are computed from the filtered input data and are used to represent a signature. This is the architectural phase, where probably the most future research will be required, because it is still debatable what the features of a signature are. At the moment, we see here a lot of speculation and wishful thinking.

To perform comparisons, a reference must be generated for each user of the system. These "golden" features are retrieved from exemplary input during a system set–up phase and subsequently enrolled in the reference database together with a unique identification number.

At the comparison stage, ID information is used to extract the proper reference set from a database. In an identification system all the reference sets from a database are used. The input signature and the reference signature are matched against each other and a similarity (or dissimilarity) is determined.

Finally, a decision process evaluates the comparison output (similarity or dissimilarity) with respect to a threshold and the signature is accepted or rejected (or identified in case of an identification system).

2.1. Data acquisition & pre–processing

Static and dynamic handwriting processing systems have their own distinct input devices. A static system uses a (photo–) camera or scanner. As opposed to a dynamic or interactive input device, it is unclear which part of the data acquired (image) is the actual signature or the background. The pre–processing step has to localize the signature, slice it out, perform a threshold and some filtering, break it into segments and do some data reduction as image data is relative large. As the background where the signature was written on is often 'noisy' (for example bank cheque), extracting the signature image from its background isn't trivial and restoration techniques are also necessary. Due to these problems, static signature verification has always been considered by researchers to be the more difficult approach and to give worse results than dynamic signature verification. For this reason, and for the fact that static systems are off–line, I stopped the research on static systems at this point.

The digitizer, also known as a graphics or XY tablet, is the most widely used input device for dynamic signature processing systems, although many other special designed devices have been used. The digitizer measures the x and y co–ordinates and the applied pressure along the pen's axis as function of time. In most cases, the pressure is nothing more than pen–up/pen–down status. F. J. Maarse [14] used a special developed pen containing a miniature load cell with a range from zero to 10.23 N and a resolution of 0.01 N. This pen is supplied with a regular ball–point refill, so it can be used to write on plain paper.

From the co–ordinate signal(s), velocity, acceleration and jerk signals can be derived. The special designed devices can also measure acceleration (as opposed to position) using a so–called accelerometric pen as developed by J.S. Lew [13], calligraphic characteristics [2] and/or the angle of the pen.

From his studies on discretisation, quantization and noise errors of XY tablets, Maarse concluded the minimum requirements of an XY tablet include a sampling frequency of 100Hz to 200Hz (or at least 64Hz to cope with noise error) and a resolution of approximately 0.1mm.

No details about pre–processing are found in the articles, except for segmentation. The applied pre–processing techniques include reducing spurious noise, detecting gaps in the pen–down signal, amplifying, conditioning, re–sampling, truncating and normalizing of the signal(s), encoding direction and detecting the pen–up/pen–down status. Noise reduction is usually carried out by Fourier analysis. The quality of the signal(s) generated by the digitizing hardware determines which techniques are or have to be used.

From his Fourier analysis, Maarse [14] proposes a low–pass filter having a flat passband up to about 10Hz. In order to reduce inevitable Gibbs oscillations at abrupt movement changes, a sinusoidal transition band from 10 to 37 Hz is proposed as well.

To solve time–axis alignment and partial mismatch problems, segmentation is used to break up the signals in smaller parts that can be compared locally. Various segmentation techniques are proposed. The most simple technique is detecting the connected components; the pen–down information is used to locate parts situated between a pen–down and the next pen–up. Another simple technique is detecting the individual strokes by looking for rest points (velocity near 0).

An interesting approach has been proposed by Brault and Plamondon [1]. In this approach the segmentation points are found with the help of a 2–steps procedure. The first step weights the perceptual importance of every signature co–ordinate. The second step identifies the segmentation points as the points which are locally more significant from a perceptual point of view (corners and turning points).

Another technique is based on a dynamic splitting procedure [3]. The basic idea is to perform the splitting by using the information about both the reference signatures and the input signature. By an elastic matching procedure, Candidate Splitting Points of the reference and the input signatures are used to identify the set of coupled strokes.

2.2. Feature extraction & classification

There are two approaches for signature representation; function and parameter. Using the function approach, the (pre–processed) signals are used as whole, or represented by parametric or complex mathematical functions.

Geometric parameters, timing parameters and more specific dynamic parameters related to a signal are involved in writer identification and signature verification. Some have derived more than one hundred parameters from the three signals generated by a force–sensitive pen. Generally speaking two types of parameters are distinguished.

1. **Local parameters**: maximal or minimal values of signals features extracted from specific segments such as number of maxima and minima, local curvature, initial direction, etc.
2. **Global parameters**: total time, number of segments, connected components or pen lifts, duration of connected components, means and standard deviations, number of zero crossings, area, proportions, etc.

From an initial set parameters, a subset is considered for verification. Theoretically, the parameter selection must be carried out according to a maximum discriminant power criterion between genuine and skilled forged signatures. In practise random forgeries are often used since they are easier to obtain.

For most applications, the parameters used are general, i.e. the same for all writers. Some have proposed the use of personalized parameters for each user.

Generally speaking the comparison technique evaluates the fit between the reference signature and the test specimen to calculate the similarity (or dissimilarity) between the two signatures. At the decision stage, this similarity is compared to an *a priori* fixed threshold (T_0) to decide if the test signature should be accepted or rejected.

The straightforward method for function comparison is to use a linear correlation, but, due to certain difficulties, this is not valid for signatures. The first problem is related to overall signal duration. In spite of a certain amount of consistency, total duration varies from one signature to another, even if they are produced by the same writer. Since signals are sampled at a constant rate, this provides a final number of sampled points which is different in each specimen.

The second problem has to do with non–linear time–axis distortions between two different signature specimens. Random variations exist, which can create portions of signals, deletions, additions and gaps due to pauses or hesitations from the writer.

Different approaches have been proposed to solve these two problems while at the same time determining the best fit between two corresponding functions. For compressing or expanding the time–axis, non–linear correlation through dynamic programming matching correlation previously used for speech recognition has been extended for use in signature verifications. In another experiment, a peak matching technique via a peek match table using a global mean–square error criterion has been considered. Other algorithms include regional correlation, dynamic time warping and tree matching. A study performed by Plamondon shows that differences between these algorithms are signal dependent.

Parameter comparison is the most straightforward. In this representation scheme, signatures are often described by vectors of parameters in a feature space, and their closeness is evaluated with the use of specific metric. In most experiments, the means and standard deviations of each of the components from several vectors are stored as the reference.

The dissimilarity vector (D) is then calculated from the reference (R) and test (T) vector as D = | R – T | and normalized by dividing each of the components of D by the corresponding standard deviation. The length of D (Euclidean or city–block distance) is compared against a threshold T_0; If |D| < T_0, then the signature is accepted, else the signature is rejected.

In most experiments, T_0 is unique and general, in the sense that it has a single value for all users. Loosening or tightening T_0 for each signer has also been proposed to take into account the specificity of interpersonal variations.

Plamondon [12] proposed a technique where the personal thresholds are obtained by comparing the reference signatures among themselves, taken two–by–two and storing the worst result of verification.

2.3. Performance evaluation

The performance of a verification system is generally evaluated according to the error representation of a two–class pattern recognition problem, that is, with the type I (FRR : false rejection) and type II (FAR : false acceptation) error rates. These rates vary with the acceptance/rejection threshold T_0. The ideal and practical situations are summarized in Fig. 3. The ideal situation would occur if the "best features" were selected to separate completely true signatures from forgeries. In this case no imitation or degeneracy would occur. In practical situations, as the feature selection problem has not been satisfactorily solved, the feature choice is not optimal and one has to cope with accepted "imitations" and rejected degenerated signatures.

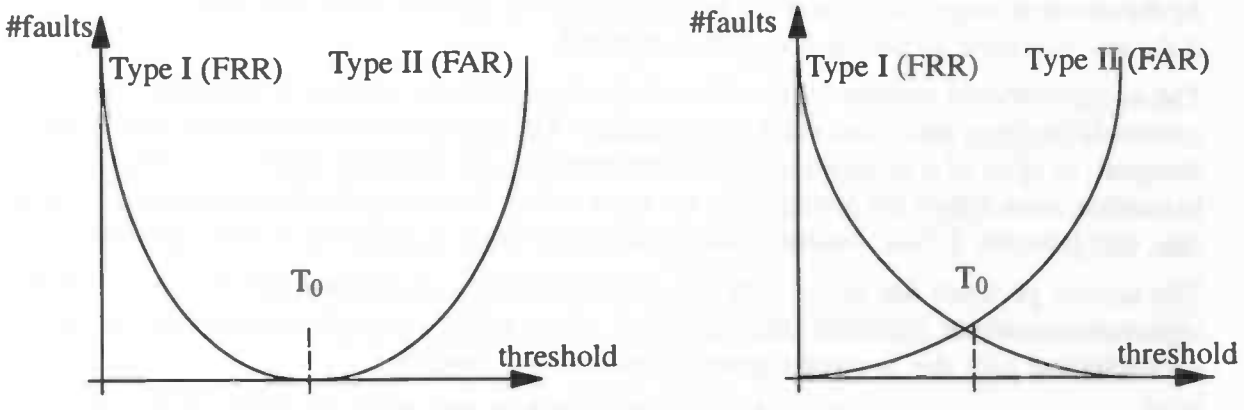


Fig. 3: FRR/FAR graph ideal (left) and real (right)

When creating my own FRR/FAR graphs, I got quite confused about which line is (supposed to be) the FRR, and which one is the FAR. I found the FRR/FAR graph will be mirrored when a dissimilarity value is compared against the threshold, as opposed to a similarity value! The lines in Figure 3 are based on dissimilarity values. That is, if dissimilarity < T_0 then the signature will be accepted.

Obviously the choice of T_0 depends on the application considered, according to the false acceptance or false rejection risk. Thresholds leading to equal type I and type II error rates (FRR = FAR) are often proposed. Plamondon's state of the art articles [11][17] include some tables giving an overview of the performance of various on–line signature verification systems with error rates varying from 0% to 50%.

Plamondon's overview and references can be found in appendix A. More recent results are included in this overview as well.

Plamondon stated that for several reasons these final results cannot be directly compared. There is no data common to the different experiments, due in particular to the lack of public signature data bases. The experimental protocols vary considerably from one experiment to another. The tests are not carried out in similar conditions (laboratory or field, industrial or commercial). Also, very large differences in the size of the train and test sets are seen; from a few signers to several thousand. Sets are collected over different time–spans, from within minutes to months.

A general result has been pointed out in several studies performance degradation of dynamic signature verification system is due mainly to the very bad signature consistency of a few users. To cope with these users, some proposals have been made either not to enrol them in the system, or largely relax the threshold on them.

This makes me wonder how many of the 'very good performance' are the result of this kind of 'cheating'.

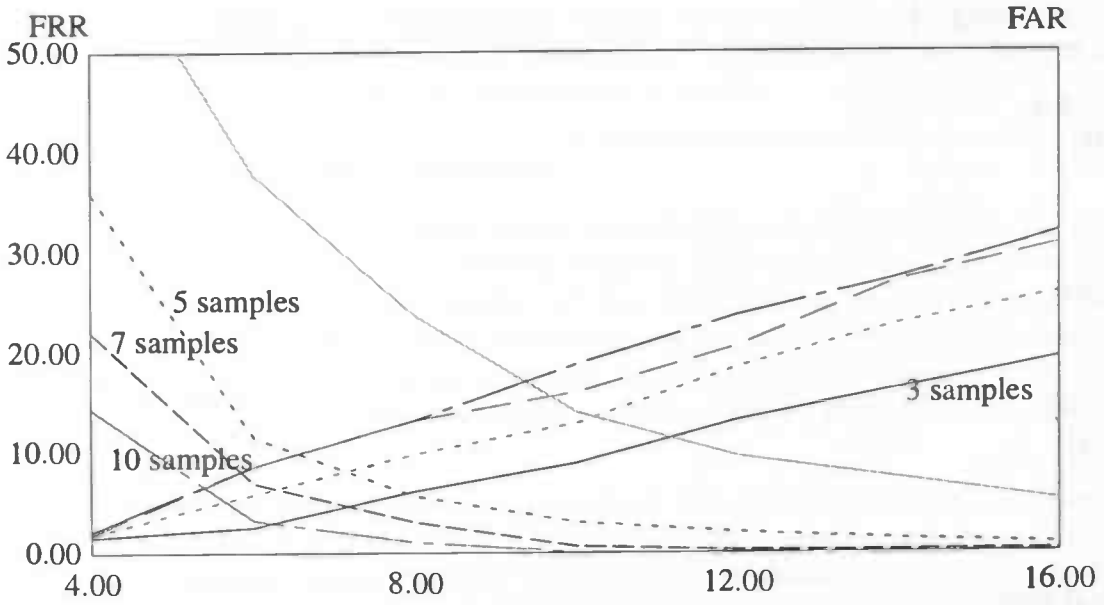


Fig. 4: FRR/FAR graph: Gopal K. Gupta's initial results

One important reason the results cannot be compared is that the threshold values are usually not given. Only in Gopal K. Gupta's article [4], a few tables of FRR/FAR values against different thresholds were included. Using these tables, (parts of) FRR/FAR graphs could be created for a better examination of its performance. In his article Gupta proposes a signature verification technique that he believes is considerably simpler than most techniques that so far have been proposed. His simple technique is based on using features which can be derived very easily and directly from the XY tablet's data.

Fig. 4 shows his initial experiment where he evaluated different number of sample signatures (3, 5, 7 and 10). Seven global parameter features were used: Overall time, number of velocity zero crossings in the X direction, number of velocity zero crossings in the Y direction, number of acceleration zero crossings in the X direction, number of acceleration zero crossings in the Y direction, total pen–up time and the overall path length. The difference vector between the mean reference vector and a test vector was normalized by dividing each element of the difference vector by the corresponding element in the standard deviation vector. The L_2 norm (Euclidean distance) of this normalized vector is then compared against the threshold.

Adding number of acceleration zero values in the X direction and number of acceleration zero values in the Y direction, better results were obtained, as shown in Table 1 and NO TAG. When 10 sample signatures from each individual are used, a 0.0% FRR with a FAR of 12.0% can be obtained.

Table 1 – Gopal K. Gupta's end results

	5 sample signatures		10 sample signatures	
Threshold	FRR	FAR	FRR	FAR
4	54.3%	0.6%	27.1%	0.9%
6	20.2%	1.2%	6.0%	2.5%
8	9.8%	3.7%	1.9%	7.4%
10	5.4%	8.0%	0.0%	12.0%
12	2.6%	10.5%	0.0%	15.1%
14	1.8%	14.5%	0.0%	20.0%
16	1.0%	18.1%	0.0%	25.5%

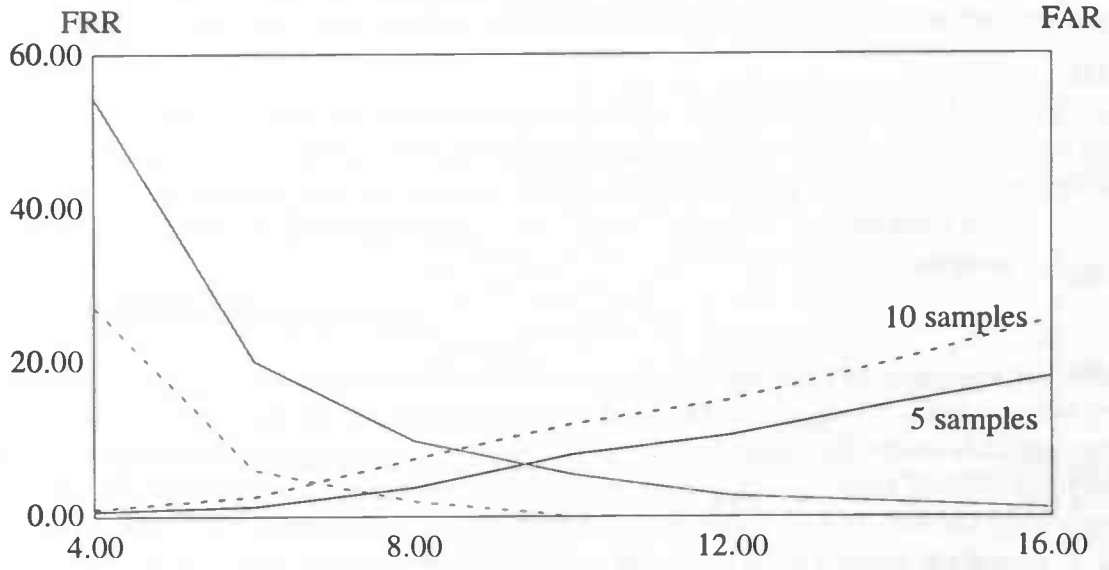


Fig. 5: FRR/FAR graph: Gopal K. Gupta's end results

Gupta also presented a table with the reference signatures obtained for his user #1 (see Table 2) to provide a better understanding of what reference signatures were obtained with different number of sample signatures. With these results, Gupta made the following comments:

> *Relatively, the pen–up time has the largest standard deviation but the pen–up time is very small, an average of only 65 milliseconds since this signature has only*

three segments and therefore the pen is lifted only two times. Therefore a relative large standard deviation does not necessarily imply that a forger would be able to forge easily this value.

Table 2 – Gupta's reference signatures for user #1

Number of sample signatures used	3		5		7		10		
Feature	Mean	SD	Mean	SD	Mean	SD	Mean	SD	SD/Mean
Overall Time	879.7	27.9	938.8	41.2	901.3	31.4	897.4	46.9	5.2%
X–Velocity Crossings	32.0	2.9	35.6	3.0	35.3	3.5	34.1	4.0	11.7%
Y–Velocity Crossings	48.7	2.1	48.8	2.0	47.0	1.9	48.1	2.2	4.6%
X–Acceleration Crossings	67.0	13.5	77.8	5.9	71.3	6.2	68.1	7.6	11.2%
Y–Acceleration Crossings	57.3	1.2	63.0	4.2	57.9	3.6	58.3	4.8	8.2%
Pen–up Time	12.7	5.4	17.8	4.4	14.1	4.3	13.5	5.1	37.8%
Overall Path Length	2176	49	2291	86	2251	117	2261	102	4.5%

An identification system is evaluated according its recognition rate. Not much articles are written about on–line signature identification systems. Lorette [12] used seven parameter features, four static (number of connected components, number of loops, quantified cumulated phase for signatures on their whole and initial direction of track pen), and three dynamic features (total duration, duration of connected components, mean and maximum velocities in connected components). To deal with realistic situations only the first five signatures for each writer were kept to form the reference set, the next ten signatures formed the test set. Using statistical clustering methods, he achieved a recognition rate of 93.6%.

2.4. Summary of research problems

Actually all parts of the authentication system provide still numerous problems. The reason is a lack in fuzzy technology. We evidently have to find ways to adaptively change the pre–processing on basis of the quality of the identified features. This seems like a chicken–or–egg problem. If a signature could be represented in a way that is unique to the writer, it could show the influence of the unique human motoric system. On the other hand, we have no clue on how the motoric system could make a signature unique. Hail again to the chicken and the egg.

For the time being, and in an effort to be practical, one could refrain to way to handle the undefined fuzziness. Can this be limited? It should be because not only can a signature be forged but also can a signature that becomes non–characteristic a forgery to somebody else. So forgery is not necessarily criminal but can be excessive fuzziness. Evidently we need more influentic and therefor we go into more detail in the next chapter.

3. MORE ARCHITECTURAL DETAILS

An obvious source of inspiration for the verification problem is in speech recognition. Here, it is known from classical research that speech is constructed from phonemes. These are small bursts of a well–defined frequency spectrum. Hence, utterings can be segmented into potential phonemes, after which these phonemes can be classified and the sound can be reconstructed. Such a process of segmentation, classification and assembly is also feasible for printed text. Nothing would be easier than to assume that the same technique applies to written text. There is just a tiny problem on our path: we have no substitute for the phoneme.

Word recognition and signature verification have just a superficial resemblance. Words are constructed from characters; hence words can be recognized from its classified components out of a known, stable and limited set. We find here the fundamental antagonism between the segmentation of a word into characters and the classification of the resulting segments as identifiable characters, that will constantly re–appear in this chapter. The segmentation problem is relieved when we assume hand–printed words, i.e. words already consisting of loose characters. On this level we find the most striking similarity between speech and word recognition. Alas, signatures do not necessarily have to be a word. A signature may easily contain non–characters: another reason why it is so hard to categorize a signature.

One of the greatest advances in signature verification is the increasingly frequent use of neural networks. Neural networks have found their way into identity verification systems and are now used in signature segmentation, static signature verification and dynamic signature verification.

The advantages of neural networks are that they can be trained to recognise signatures and their characteristics are such that they could be used to classify signatures as genuine or forged as a function of time through a retraining process based on recent signatures. Their primary disadvantage is often the large number of specimens required to ensure that the network does in fact lean.

In this chapter we take one step back and try to identify: what makes it so hard to identify stable characteristics within a signature that are unique for a writer. We will attempt this by first looking at the signature as it is, then we discuss the process of writing and lastly we review neural architectures suited for such situations.

3.1. A signature as a picture.

When the signature must be analysed as a whole, the full image must be analysed on a per–pixel basis. This is often used for the classification of hand–written digits [5]. Here the raw images can be preprocessed to obtain 16 by 16 normalized pixel maps, on which skeletonization is performed followed by the identification of rudimentary pixel–oriented features such as lines and end–points. The resulting feature map may use 99 binary values to indicate the presence or absence of a feature at a given location.

The resulting description can be viewed as a multi–dimensional structure in which classification can be performed by minimum–distance measurements. The alternative is to see each digit as a collection of points in a multi–dimensional space and perform clustering. Both approaches aim to find a degree of similarity (dissimilarity). It maybe clear that the results are strongly affected by the effectiveness of the representation. However, as the operation is performed on multi–dimensional elements it will be hard to ensure that the information is learnable on a neural network.

As the individual person has a distinct variety in his signatures, the first question is to find commonalities that can be explored by looking at image parts, such as lobs and arcs. On this basis there are two things to be reckoned with:

1. lobs and arcs can come and go, i.e. a signature is per person not always the same set of lobs and arcs. Correlation is required to find correspondence between the variety of signatures of a single person.
2. lobs and arcs are not always on the same location, i.e. a signature is per person non–linearly transformed. In order to normalize the representation, stretching and compression w.r.t. the distinguished features is required; the most popular technique here is called "Elastic Matching".

The provisional conclusion is that additional information is needed to provide a cognitive weight to those elements. In general, such a system will consist of the following steps:

1. a number of different signatures from one person must be gathered to establish the cognitive characteristic elements.
2. these signatures must be segmented into lobs and arcs.
3. the representation must be stretched and/or compressed.
4. the cognitive relevant elements must be determined.
5. steps 2, 3 and 4 must be repeated for slightly differing segmentations to optimize the classification.

This clearly shows the fuzzy nature of segmentation: an algorithmic definition is not available and the boundaries between segment types may vary over person and over time. The experience in the area of license–plate recognition (in principle a variation on hand–printed words as the printed characters are blurred by the lack in resolution and the speed / obervation angle of the driving object) has shown that such a fuzzy segmentation problem can already be very cumbersome for relatively easy problems [16].

In the case of signature verification, the reference signature may be assumed. In order to verify the authenticity of a writing, it must first of all be known which writer is supposed to be authentic. In that case, it may be possible to have a single classifier per writer (or just a small number of writers of which a single one is supposed to be correct and the others have a confusing similarity). This eases the task to a 2–stage approach, whereby first the potential writers are identified and subsequently in a second pass the single authentic writer must be identified [23]. This technique has proven itself already in the above named license–plate recognition application.

Though for the above off–line applications we may come a long way in achieving signature identification, we are fundamentally missing information about the human motor system. The input is simply a picture and there numerous ways in which a picture can be perfectly forged. In other words, this can only serve as a stepping stone towards a much more sophisticated system, where the writer is also interrogated for his unique motoric system.

3.2. A signature as a signal

During on–line application, we have information about the human motoric system through its effect on the dynamics of the signature. However, we run again into the problem that there is no clear knowledge on the characteristic components of a signature. One usually makes the

proposition, that a signature is a sequence of strokes, whereby the strokes can be classified and allow categorization upon assembly. The confusion becomes apparent, when looking at the various definitions of a stroke:

1. **cognitive feature**: a component displaying a distinct perceptual importance [18]. This is probably the most vague definition. On the other hand, when perceptual importance can be further detailed to its frequent occurrence in a variety of signatures and/or the fact that evidently the writer has attention to that part of the signature then it might be possible to come to a more explicit definition.
2. **time segment**: the part between two consecutive velocity sign changes. The assumption is here that there must be a reason why the velocity changes and this may have something to do with perceptual relevance [19], [15].
3. **connected component**: the part between pen–up and pen–down moments [17]. Again the explanation may be that this indicates moments of perceptual relevance.

All this seems like a wild goose chase. Most of the proposals in literature try to mask this deficiency in clear algorithmic definition may introducing probabilistic models. A first choice could be a feedforward network with error back–propagation learning. In order to handle the dynamics of the signature it should be of the time–delayed variation. For practical realisation such time–delays are clock–synchronized, which in principle should be justified to the sampling clock of the input device.

The popular way of representing a stroke is by a 5–tuple: the begin–point, three equally–spaced intermediate points and the end–point. As a result this vector connotates the time segment of the stroke divided into four intervals. As the respective strokes are written at different speeds / time intervals, over–sampling is required and therefore the number of required time–delays in the neural network takes astronomous proportions.

The alternative method to grasp the dynamics of the signature is by recurrency. In its simplest form, this could be a recurrent network, whereby the internal delay elements may be clock–asynchronous. Many variations on this network type have been proposed, but the most effective one seems to be the Kohonen feature map: a locally connected network using the winner–takes–all principle. This appears to be able to identify the strokes with perceptual relevance, which can be used in a subsequent feedforward network for the actual classification.

3.3. Some characteristic examples

Morasso has attempted to identify cursive hand–writing by two types of network: (a) a self–organizing network in analogy with Kohonen's phonetic typewriter [9], and (b) a feed–forward network in analogy with Rosemberg & Sejnowski's NeTalk [22]. He starts from the signal dynamics by looking at the velocity profile. A stroke is defined as the segment between two valleys in the velocity profile. These 20–40 samples are condensed to a 5–point polygonal curve including the boundary points. A hand–writing specimen (such as a signature) is then a sequence of strokes, each identified by 10 numbers.

In the first experiment, Morasso trains a self–organizing network using a modified Hebb learning rule by stroke code extracted from 200 6–10 character long words over 5–10 epochs. The network is arranged over a hexagonal grid taking 5x5 to 30x30 neurons. The resulting graphotopic map provides a very compressed representation of the deep structures that underly the handwriting production process. This is verified over a dedicated reconstruction phase by following the centers of the activated bubbles over the application of the stroke codes. It was found that a network of 15x15 was minimal required for a decent reconstruction.

In order to fuzzify the segmentation effects, Morasso continues in his second experiment by researching the correspondence between strokes and character doublets over a 3–layer feed–forward network. For Italian words, the network takes 88 input, 70 hidden and 42 output neurons. Training is performed on 144 patterns with a 0.25 learn rate to minimize the average square error. The learn curve sharply decreases during the first 300 steps and levels off after 600. The final state is a 97% correct classification by looking at the most active neuron for each of the two characters.

Then on generalization, the results were very negative and the paper does not give numerical data. The author remarks:

> *It is probable that in order to take into account the idiosyncratic features of one's writing style, it is necessary to incorporate some representation of regularity in the coding scheme of the input units. Failing to do so might cause an insufficient generalization ability similar to the one found here. The logic way to go is to combine the two types of networks, exploiting the strong points of both of them: (a) the ability of self–organizing networks to discover idiosyncratic regularities and to represent them in a distributed way, and (b) the simplicity/efficiency of multi–layer perceptrons in learning associations.*

There is no further evidence that this observation is actually usable except for the recent application of such a strategy in the diagnosis of motor engines [6].

Another major drawback of using neural networks in this field is the slowness of learning. Higashino from the Hitachi's Central Research Laboratory in Tokyo, Japan has designed a neuro–computer with a tremendously increased learning speed [8]. One neuro–computer system is capable of 2.3 billion connection updates per second.

For input, an XY tablet and a special designed pen with 64 digitised pressure levels are used. The sample rate is 50 points/sec. The data (vertical speed, horizontal speed, pressure) is normalised into 256 points. The training patterns are stored in a special memory so that these are transferred into the neuron circuit at the pipeline speed (100ns). The circuit consists of eight wafers providing a total of 1,152 neurons.

> *The neural network is partial connected due to memory limitation. I Wonder if this would this affect performance? According to Higashino, the partial connection is also based on the idea that the correlation of the widely separated signal becomes smaller generally, and makes possible to decrease the entire number of connections and to shorten the learning time.*

The learning system of neuro–computer only supports the back–propagation algorithm [7]. To classify signatures as authentic, this algorithm needs both authentic and non–authentic signatures for learning. Non–authentic signatures are generated by deforming authentic signatures, as in practice forged signatures do not exist when the signature is registered. Deformation is done by multiplying the signal values by a constant factor. A factor near 1.0 (0.7 – 1.3) will result in a small deformation. These deformed signatures will be added to the authentic patterns. A much larger (3.0 – 4.0) or much smaller (0.1 – 0.3) factor will result in a large deformation. These deformed signatures will are used as the non–authentic patterns.

> *A very important point is made here: in practice, forged signatures do not exist when the original(s) is/are produced!*

With 527 signatures from 70 individuals, the average FRR is 7.97% (worst–case 40.00%), and the average FAR is 0.61% (worst–case 4.05%). Cases with a high FRR are concentrated on specific persons who wrote many signatures.

Higashino's article is the only one with a table containing individual results. Most individuals have an almost perfect result of 0% FRR with a FAR < 0.5%. The few 'bad' cases (with a FRR up to 40% and a FAR up to 4.05%) result in the high overall error rates.

3.4. A plan for an experiment

For building the pool of experience as required to better understand the authentication problem, we will in this report focus on the use of the 3–layer feed–forward network. It should not be able to give perfect results for reasons discussed before but it

1. connects directly to the current practice in our group.
2. allows to see how bad this works for signature verification.

So we can foresee three stages in experimentation:

1. provide an effective interface with the digitizer taking into account the various small but mostly realization–oriented problems as for instance caused by the non–synchronized clocks in the respective system parts;
2. evaluate how effective some straightforward features named in literature are, and
3. come with a first idea on which course the research should take to have some chance on a future solution.

4. EXPERIMENTAL RESULTS

This part of this document describes the experiments I have performed to examine, verify and solve some of the problems, methods, techniques, theories and thoughts.

The experiments were performed on two different systems. A low–end PC with Turbo C and Turbo Pascal compilers and tools were used for most experiments. An Hewlett–Packard series 9000 workstation with Neuro–Tech's Interact(tm) and the GNU C compiler were used for the neural experiments.

As stated in the previous chapter, we need a pen–based input device for data acquisition. For my experiments I didn't need a very luxurious and expensive input device as used on portable pen–computers. Instead I've chosen for a relative cheap five by five inch graphics tablet, also known as a digitiser, normally used for Computer Graphic Art and Computer Aided Design applications. Such a graphics tablet can be connected directly to a Personal Computer, workstation or terminal through the serial communications interface (RS–232C). For the user, the drawbacks of such a cheap graphics tablet are the electronic pen which is connected to the graphics tablet through a wire, and the lack of visual feedback apart from a computer display.

> *The first drawback is only minor inconvenience at the experimental stage. Tablets with cordless pens are already available. The second drawback is something to live with. As with entering the PIN into a system, it is probably not a good idea to display the input on, for example, a transaction terminal.*

The software that comes with the graphics tablet includes a Windows driver and a mouse simulator for DOS applications. After a first attempt to get data from the graphics tablet using this software, I came to the conclusion that to get a proper signal, raw data from the graphics tablet has to be used. The main problem was time. The mouse simulator didn't send samples at a constant rate. I tried to give every sample a time–stamp, but the time delay within the mouse simulator wasn't constant either. By examining the documentation given from the graphics tablet's manufacturer and some minor experimenting, I found the graphics tablet having the following properties:

- The graphics tablet has a default resolution of 500 Lines Per Inch (LPI) with an accuracy of 0.01" (0.254mm) and a constant sampling rate of about Points Per Second (PPS).
- An electronic pen is used as the positioning device. The pen's tip has a switch to detect whether or not pressure is applied onto the pen.
- Samples are sent through the serial communications port at a constant rate. A sample consists of x and y positions, the status of the pen–tip and button(s) and whether or not the positioning device is in detecting range of the graphics tablet (proximity).
- Command codes can be given to the graphics tablet. Normally the graphics tablet only sends samples when the pen's tip or a button is pressed down. A command to make the tablet send samples continuously is available. Another command to retrieve the tablet's resolution (2500 by 2500 sample points) can be used to detect the graphics tablet.

  More details, incl. the format of the graphics tablet's data can be found in appendix B

The next problem: when to stop sampling. Using the button on the electronic pen to stop the data acquisition appeared to be impractical as it was often pressed accidentally when people tried to

use the pen like any other normal stylus. To solve this problem, I've programmed my 'sampler' to stop the data acquisition when it doesn't receive 'pen–down' samples for more than a second.

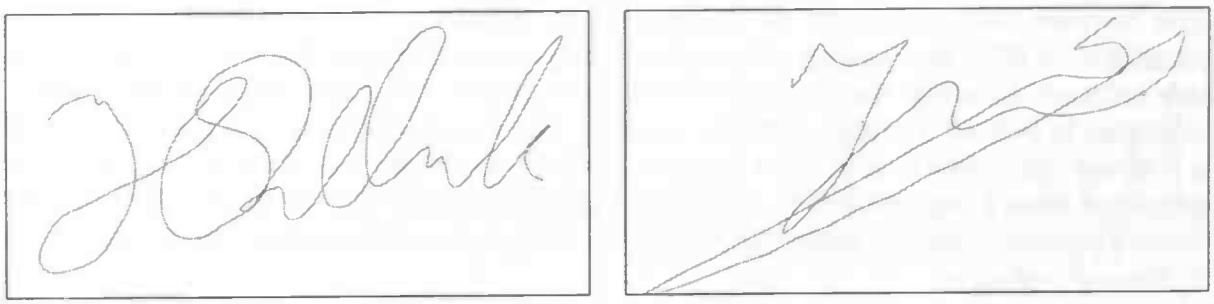
> *With a sample rate of 128 samples per second, a connected component of 128 'pen–up' samples could be used to determine when to stop. The digitiser however does not generate samples when the electronic pen is out of proximity which is often the case when a user has stopped writing!*

Using the sampler, sixteen to twenty–five signatures from thirteen individuals (see Appendix E) each were digitised and put into a database. The 'raw' data varied from 75 to 1500 sample points per digitised signature.

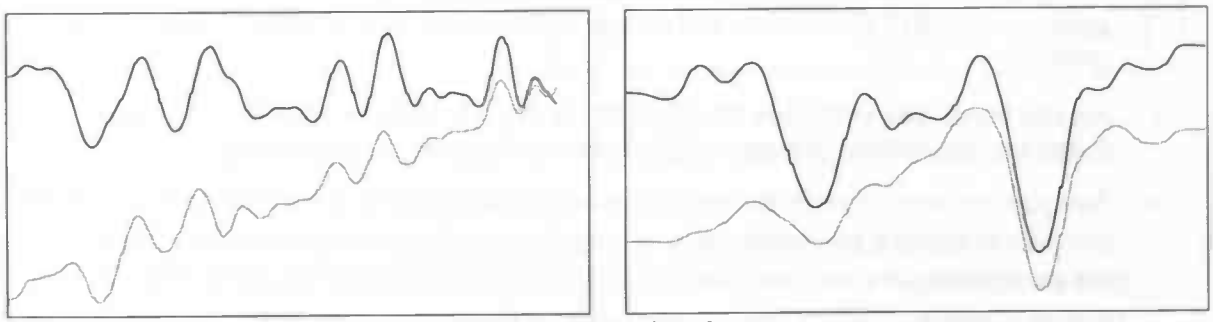
> *Full source of my sampler can be found in appendix C. Appendix D contains the data format of the digitised signatures.*

4.1. Pre–processing

To determine the type(s) of pre–processing, a close examination of the raw data has to be made. Using a simple program and a graph utility, we can visualise the signature, its position signals, velocity signals and acceleration signals. Fig. 6 shows my own and another individual's signature and corresponding position signals. The darker line represents the *Y* component of the signal, the other represents the *X* component



raw signatures



position signals

Fig. 6: My own and another individuals raw signature with position signals

Fig. 7 shows my own velocity and acceleration signals. Again, the darker line represents the *YY* component of the signal. Both the velocity and acceleration signals aren't as smooth as the position signals. The velocity signals have drops in them, the acceleration signals have peaks in them, and both seem to have a periodic appearance. When looking closely to the actual sample values, one can see these drops and peaks are the result of double samples received from the digitiser. Apparently, the position detecting device and the serial communication device within the graphics tablet don't share the same clock circuit. As the clock circuit of the communication

device is slightly faster than the position detecting device, a copy of the last sample is send every ten to twenty samples. Therefore, the first pre–processing step is to detect the double samples and filter them out.

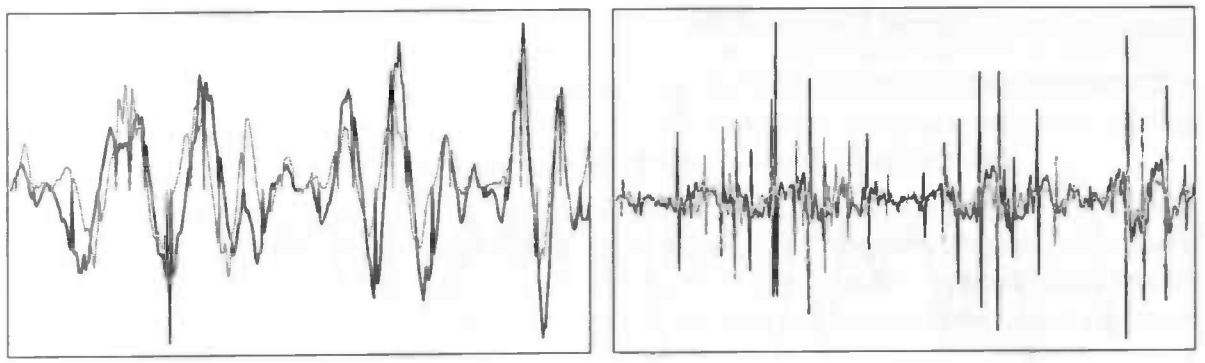


Fig. 7: My raw signature velocity (left) and acceleration (right) signals

Detecting these double samples isn't trivial as double samples are also generated at rest points. But as double samples don't give problems at rest points, we can concentrate on the other parts. The code as shown in Pseudo–code 1 does the trick, and can be used to do the filtering 'in–place'. Fig. 8 shows the filtered velocity and acceleration signals.

```
i <- 1;
j <- 1;
while i < N-1 do
  if input[i].Xvel = 0 and input[i].Yvel = 0 and
    (input[i-1].Xvel * input[i+1].Xvel > 1) or
    (input[i-1].Yvel * input[i+1].Yvel > 1)
  then i <- i+1;
  output[j] <- input[i]
N <- j;
```

Pseudo–code 1 – Double samples filter

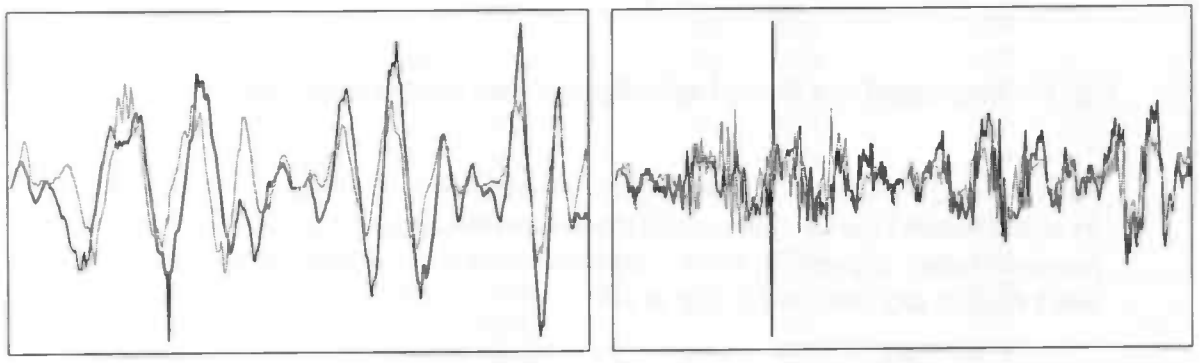


Fig. 8: Filtered velocity (left) and acceleration (right) signals

As seen especially in the acceleration signal, noise is definitely present. A simple low–pass filter as shown in Pseudo–code 2 can filter out the majority of the noise. Fig. 9 shows the resulted velocity and acceleration signals.

```
output[0] <- input[0];
for i = 1 to N - 1 do
 output[i].Xvel <- (input[i-1].Xvel + input[i].Yvel) div 2;
 output[i].Yvel <- (input[i-1].Yvel + input[i].Yvel) div 2;
```

Pseudo–code 2 – Simple low–pass filter

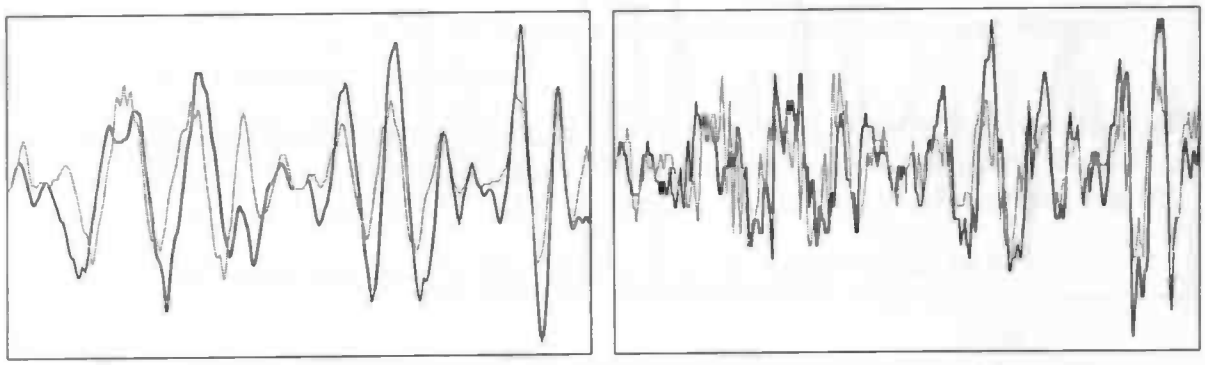


Fig. 9: Low–Pass Filtered velocity (left) and acceleration (right) signals

The next pre–processing step is segmentation. Fig. 10 shows the easiest form of segmentation: the connected components. The connected components can easily be found using the pen–down signal.

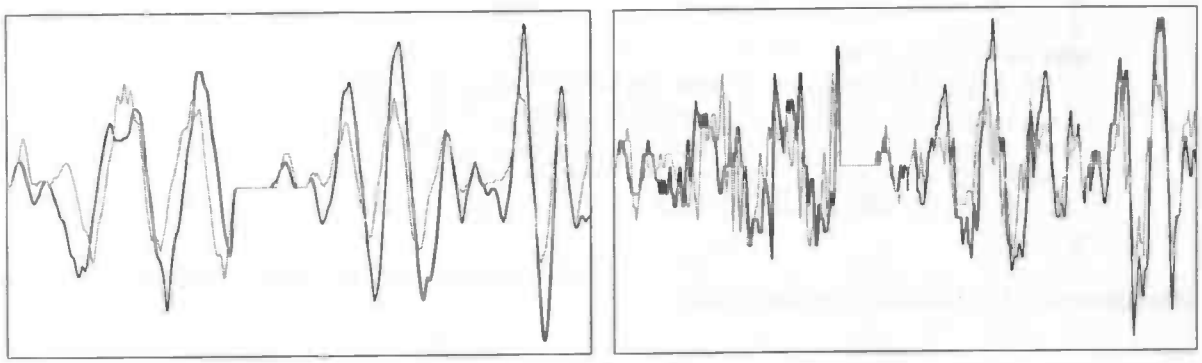


Fig. 10: Segmented and filtered velocity (left) and acceleration (right) signals

The (minor) problems with data acquisition and pre–processing should be clear by now. I haven't carried out (much) more experiments as the required form(s) of pre–processing depend(s) on the application. Some (usually global) features don't require any pre–processing at all.

4.2. Initial neural experiment

Using the InterAct tool an initial experiment has been done to see how well a basic feed–forward neural network [7] would handle the signature signals as whole. As Higashino [8] did with his Neural computer, I re–sampled the x and y velocity and the pen–down signals to 256 points. The feed–forward network used has 768 input neurons, as many output neurons (13) as individuals or classes, and four times the number of output neurons for the number of hidden neurons (52). The back–propagation algorithm was applied to learn 5 signatures from each individual (giving a total of 65). The other signatures were used to test the network's performance.

The network's output can be used for both signature identification and signature verification. When looking at all the output neurons' activity after having fed the network with a test signature, we can determine how the network has classified (identified) it. When looking at only one output neuron, we can determine for how much the network has accepted a test signature as being from the corresponding class.

Three methods were used to determine which class was identified. The first method is simply chosing the most active output neuron (max.). The second is method is very much as the first, but a signature is not identified if the most active output neuron isn't active at least for 50% (> 50%). The third method is identical to the second, except the output neuron has to be active for at least 75% (> 75%).

The network has little trouble classifying the given signatures. Within a few thousands of learn cycles the network hardly changed its behaviour. This, however, cannot be said from the test signatures. The 'results' of one of the many test runs are shown in Table 3. Varying the neural experiment's parameters as the number of learn cycles, number of hidden neurons and the number of learn signatures didn't improve the results. I got worse results when I tried to learn with less than 5 signatures or left out the pen–down information.

Table 3 – Results initial neural experiment

	max.		**> 50%**			**> 75%**		
Individual	+	–	+	–	?	+	–	?
1	33%	67%	20%	33%	47%	13%	13%	74%
2	100%		91%		9%	64%		36%
3	64%	36%	27%		73%	9%		91%
4	100%		91%		9%	64%		36%
5	90%	10%	75%		25%	70%		30%
6	65%	35%	65%	10%	25%	55%		45%
7	100%		95%		5%	70%		30%
8	81%	19%	57%		43%	33%		67%
9	100%		94%		6%	69%		31%
10	85%	15%	85%	15%		60%	10%	30%
11	59%	41%	45%	14%	41%	27%		73%
12	64%	36%	55%	27%	18%	18%		82%
13	100%		100%			76%		24%
All	**80%**	**20%**	**70%**	**7%**	**23%**	**50%**	**2%**	**48%**
Worst	**33%**	**67%**	**20%**	**33%**	**73%**	**9%**	**13%**	**91%**

As seen in Table 3, the difference between the overall errors and the worst–case errors are quite large. Judging from these results, I'm under the impression that high recognition rates tell more about the number of 'bad cases' the system had to cope with than about the performance of the system.

The verification results have been transformed to a FRR/FAR graph (see Fig. 11). The FRR and FAR lines intersect at threshold T_0 = 19% with equal error rates of 10%. With this threshold, the worst–case (individual) has an FRR of 27% and a FAR of 20%. Using an threshold of 50% (as in Higashino's experiment), we get an average FRR of 27% with an FAR of 0.8%, and a worst–case of FRR = 60% and FAR = 2.2%.

> *These are not very good results. I suspect this is due to the fact the network was especially trained for identification.*

If we allow a worst–case of FRR = 10% per individual and we give each individual his/her own threshold value, we get slightly better results. The worst–case has a FAR of 48%. The overall rates are FRR = 6% and FAR = 22% where we otherwise would get an overall FAR of 58%.

> *Using individual thresholds is thus the way to go to reduce overall FAR.*

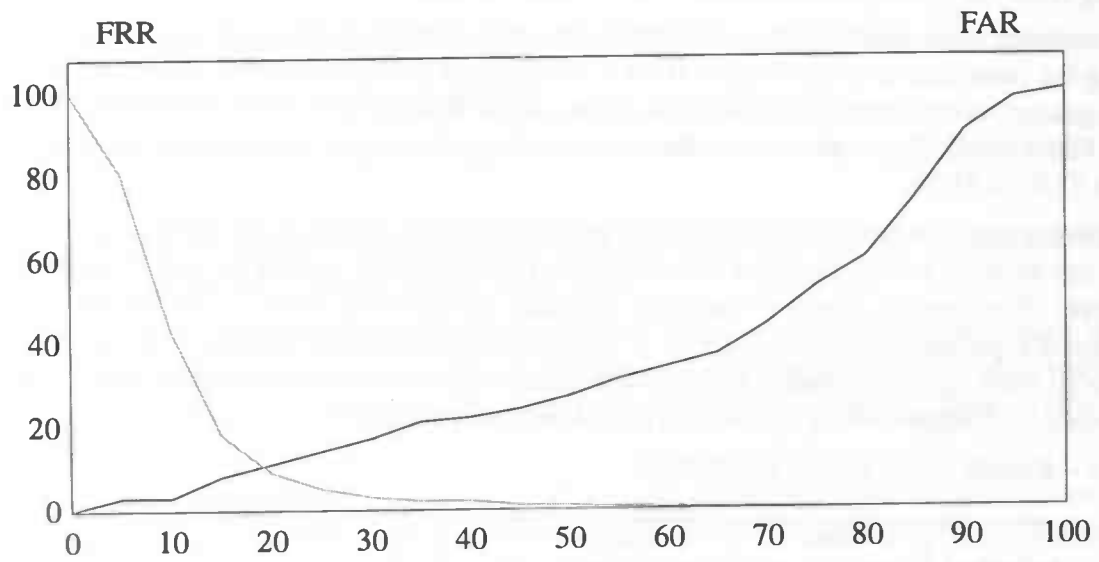


Fig. 11: FRR/FAR graph: initial experiment

4.3. Some more experiments

In the initial experiment, the whole (re–sampled) signal is fed into the neural network hoping it could figure out by itself which features to use. Apparently this is not the case, so I've stepped to the parameter features approach.

For the next experiment the following mainly dynamic parameter features are used: total time, number of connected components, duration per connected component. The same feed–forward neural network from the initial experiment, but with (naturally) less input neurons is used. Again, 5 signatures from each individual are used as the training set.

The performance of this system is (surprisingly) very similar to the initial experiment. Again, the FRR and FAR lines intersect with error rates of 10%. Now the threshold T_0 is 23% as seen in Fig. 12. Further these results seem at first sight not really different from the state–of–the–art.

As with the initial experiment, the same 'inconsistent' cases (38% of all cases to be exact) are causing the relative high error rates. To me, these results suggest that the feature extraction process is the most critical. If this process can't find features that are discriminant enough, the overall system will fail.

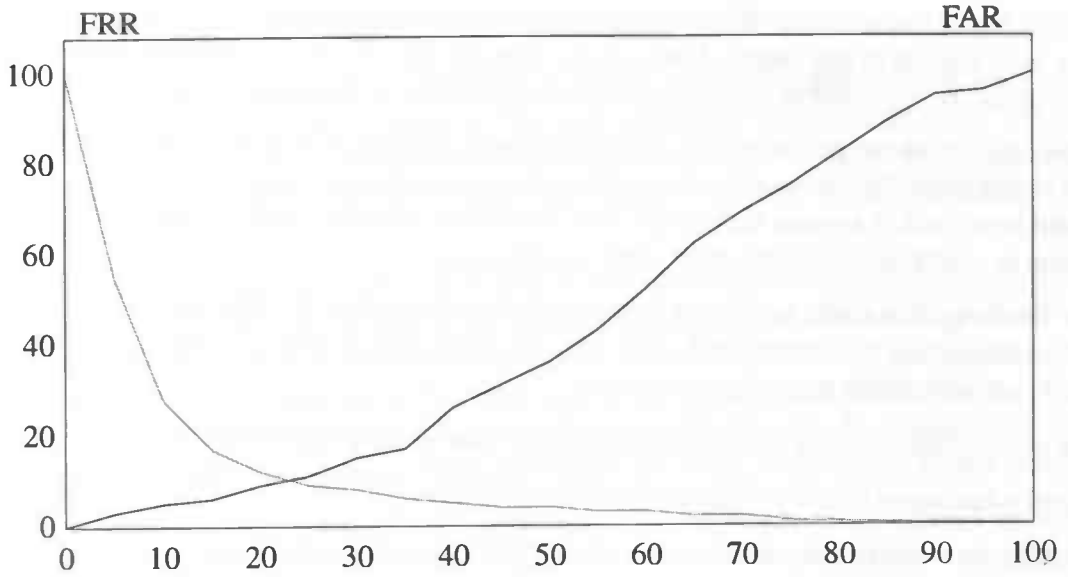


Fig. 12: FRR/FAR graph: parameter experiment

4.4. Provisional results

At the moment, judging from the literature I've read and from my own experiments, I have serious doubts whether hand–written signatures are suited for automated banking and security systems. The comparison process is unable to make a 100% correct decision due to the interpersonal variation in the signatures. This variation leaves little information about the signature generation process.

Gopal K. Gupta found relative standard deviations varying from 5.2% to 37.8%. With a quick look into my own signature database, I found the often used 'overall time' feature having a relative standard deviation varying from 5% to 20% per individual. If the signatures are represented as clusters in a feature space, and if these clusters overlap each other due to the variation, then no clustering method or neural network is able to give satisfying results. Somehow, one has to increase the feature space to reduce overlapping clusters.

In my opinion, limiting the feature space by using personalised features is not a good idea. This could lead to situations where features which could detect certain classes of forgeries are left out in the comparison process.

Another problem is that signatures change during the years (evolution). Some signatures have become small pieces of line–art from which, with some fantasy, the owner's initials can be identified. To solve this, the verification system should be capable to 'evolve' with the user. The system should use recent (and accepted) signatures to update the user's reference signature. Especially with neural networks, this doesn't have to be a large problem. This 'evolving' could solve another problem: how to get enough sample signatures. In the literature we see systems performing better when using a lot of sample signatures.

From a totally different point of view, I found another reason why hand–written signatures are not very suited as an alternative for PIN. The key feature of PIN is that it is only known to the user and the system, while personal signatures can be found on letters, (official) documents and, for example, the bank card. Therefore I suggest to use hand–written 'code' words or sentences which, as PIN, can be kept secret from potential forgers.

Using hand–written code words has some interesting advantages. Simple static feature comparison and/or text recognition techniques can be used as well as dynamic feature

comparison. Static features add little security to a signature verification system if the signature's pattern is well known to the forger. Many static features are too obvious in such cases. Some good examples are the number of up & down movements and the number of lobs.

In my opinion, I think the public will be convinced quick that a system using hand–written code words is better than PIN as words are much easier to remember. Also, as most users won't understand how such a system will work, they may have doubts whether dynamic signature verification is much more reliable than static verification.

Whether hand–written code words are indeed a better alternative for PIN than hand–written signatures has yet to be determined. The idea of using hand–written code words sounds promising, but also raises some questions:

- Is normal handwriting a more stable process than writing signatures?
- And what about hand–writing without visual feedback?
- Are code–words easier to remember than PIN? (one would think so)

Some general questions are raised as well:

- How does the user's medical condition and/or the user's mood affect the hand–writing process?
- And if, due to such a user's condition, the first attempt fails, can we guarantee the transaction will succeed within three trials?

Further study is required to get these answers. As only the hand–written signature has been replaced by a hand–written word or sentence in this idea, all other processes, studies and theories of the general identification and verification system as shown in Figure 2 (page 8) can still be used.

Using either hand–written signatures or (code) words, an on–line dynamic verification system has to be able to extract user specific dynamic features from the hand–writen input signal(s). Apparently, conventional function and parameter feature extraction techniques and a feed–forward neural network are unable to do this without having several users with relative high false rejection error rates (>10%). I suspect this is due to not being able to find features that have the needed discriminant power.

The perfect feature extraction process should be able to figure out the discriminant features by itself. For the system manager or user, which specific features are used is not important as long the features used can separate forged signature from genuine ones.

5. CONCLUSIONS AND FUTURE RESEARCH

Although some good and interesting results are found, signature identification and verification systems are still far from being perfect. The most important cause seems to be 'inconsistent' users. As a result, features that can be used to discriminate these users' genuine signatures from forgeries are very hard (and sometimes impossible) to get with the currently available technology.

That signature verification seems not suited as a replacement for the PIN system does in my opinion not imply that other forms of verification based on handwriting aren't suitable. Hand–written 'code' words or sentences might be more suitable for several reasons. The most important reason is that similar to PIN these words can be kept secret. This allows verification systems to use a more 'relaxed' form of verification. Whether hand–written code–words are indeed more suitable than hand–written signatures has to be found out in future research.

As stated by Réjean Plamondon, final results from the many experiments carried out in this field cannot be directly compared as the conditions vary considerably from one experiment to another. Error ratios have little meaning when their conditions are not known. Most important, these ratios are mean values. In practice, one would like to know for how many cases a system has a relative high probability to fail.

BIBLIOGRAPHY

[1] Jean–Jules Brault and Réjean Plamondon, **Segmenting Handwritten Signatures at Their Perceptually Important Points**, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 15, No 9, pp. 953–957, Sept. 1993.

[2] P. De Bruyne, and R. Korolnik, **Measurement of static and dynamic calligraphic features**, *International Carnahan Conference on Security Technology: Crime Countermeasures*, pp. 100–115, 1991.

[3] G. Dimauro, S. Impedovo and G. Pirlo, **On–line Signature Verification by a Dynamical Segmentation Technique**, *Proceedings of IWFHR–3*, Buffalo, NY, pp. 262–271, May 1993.

[4] Gopal K. Gupta and Rick C. Joyce, **A simple approach to dynamic hand–written signature verification**, *preliminary paper*, 1985.

[5] I. Guyon, I. Poujaud, L. Personnaz et al., **Comparing different neural network architectures for classifying handwritten digits**, *Proceedings IJCNN II*, Washington, pp. 127–132, July 1989.

[6] S. Hafner, **personal communication**, 1996.

[7] Simon Haykin, **Neural Networks, A Comprehensive Foundation**, McMaster University, Hamilton, Ontario, Canada, ISBN 0–02–352761–7

[8] Junichi Higashino, **Signature verification system on neuro–computer**, *Proceedings 11th IAPR Internation Conference on Pattern Recognition*, pp. 517–521, Aug.–Sept. 1992.

[9] T. Kohonen, **The neural phonetic typewriter**, *IEEE Computer*, pp. 11–22, March 1988.

[10] Franois Lamarche and Réjean Plamondon, **Segmentation and feature extraction of handwritten signature patterns**, *Proceedings of 7th International Conference on Pattern Recognition*, Vol. 2, pp. 756–759, Montral, 1984.

[11] Franck Leclerc and Réjean Plamondon, **Automatic signature verification: the state of the art – 1989–1993**, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 8, No. 3, pp. 643–660, 1994.

[12] Guy Lorette, **On–line handwritten signature recognition based on data analysis and clustering**, *Proceedings of 7th International Conference on Pattern Recognition*, Vol. 2, pp. 1284–1287, Montral, 1984.

[13] J.S. Lew, **(Optimal) Accelerometer layouts for signature verification**, *Research Reports RC–7154 & RC–7374*, IBM Watson Research Center, Yorktown Heights, NY, 1978.

[14] Fransiscus J. Maarse, **The study of handwriting movement**, *Proefschrift*, Nijmegen, Februari 1987, ISBN 90–265–0811–5

[15] P. Morasso, **Neural models of cursive script handwriting**, *Proceedings IJCNN II*, Washington, pp. 539–542, July 1989.

[16] J.A.G. Nijhuis, J.A.G., M.H. terBrugge, K.A. Helmholt et al, **Car License Plate Recognition with Neural Networks and Fuzzy Logic**, *Proceedings ICNN'95*, Vol. V, Perth, Western Australia, pp. 2232–2236, November 1995.

[17] Giuseppe Pirlo, **Algorithms for Signature Verification**, *Fundamentals in Handwriting Recognition*, Château de Bonas, France, June–July 1993, pp. 139–152.

[18] Réjean Plamondon and Guy Lorette, **Automatic signature verification and writer identification – the state of the art**, *Pattern Recognition*, Vol. 22, No. 2, pp. 107–131, 1989.

[19] Réjean Plamondon and Marc Parizeau, **Signature verification from position, velocity and acceleration signals: A comparative study**, *Proceedings of 9th International Conference on Pattern Recognition*, Vol. 1, pp. 260–265, 1988.

[20] Réjean Plamondon , **The design of an on–line signature verification system: from theory to practice**, *International Journal of Pattern Recognition and Artificial Intelligence*, Vol. 8, No 3, pp. 795–811, 1994.

[21] Réjean Plamondon, P. Yergeau and J.J. Brault, **A multi–level signature verification system**, *From Pixels to Features III – Frontiers in Handwriting Recognition*, S. Impedovo and J.C. Simon eds., Elsevier Publ., pp. 363–370, 1992.

[22] C.R. Rosemberg and T. Sejnowski, **Parallel networks that learn to pronounce the English text**, *Complex Systems*, vol. 1, pp. 145–168, 1987.

[23] R. Sabourin and J.–P. Drouhard, **Off–line signature verification using directional PDF and neural networks**, pp. 321–325.

LIST OF FIGURES

LIST OF TABLES

APPENDICES

Appendix A – Performance overview and references

Performance of on–line signature verification systems

Authors	Input signals and feature description	Training and/or test database specimens(S) writers(W)	Comparison Method	Error rates	Comments
Achemlal, Mourier, Lorette and Bonnefoy [1] (1986)	x and y positions 10 personalised parameters selected among 40	600 genuines training set: (10 S 60 W) 600 forgeries 1 imitator	Weighted city–block distance	FRR : 11% FAR : 8%	Use of skilled forgeries 3 trials
Beatson [2] (1986)	x and y positions movements of the pen in the air included, spatial and dynamic features	1000 genuines 100 forgeries		FRR : 1% FAR : 2%	No information on training set commercialised by Signify Corp. As SIGNIFY system
Bechet [3] (1984)	x and y velocities normalisation	300 genuines training set: (4 or 5 S 48 W) 1500 forgeries 12 imitators	Euclidean distance and score	FRR : 5% FAR : 5%	Over 3 months
Bonnefoy and Lorette [4] (1981)	x and y positions parameters selected from a training set of 18	342 genuines (15 S 14 W): training (12 S 11 W): test no forgeries	Sequential recognition decision–tree	FRR : 0–6%	Only one trial over one year
Brault and Plamondon [5] (1984)	average acceleration, sum of accelerations, number of samples	243 genuines (5 S 50 W)	Histogram classifier global and local likeliness coefficients	FRR : 1.2% FAR : 1%	Among 243 signatures random forgeries
Chorley, Olding, Parks, Pobgee and Watson [6] (1975)	x and y position 10 parameters selected among 100	200 genuines (5 S 70 W): training, 549 forgeries 40 imitators	Windows in 10 dimensions space	FRR : 4% FAR : 0–0.05%	After 4 weeks reference: 5 consistent signatures 3 trials random and skilled forgeries commercialised by Transaction Security as VERISIGN and SECURISIGN systems
Crane and Ostrem [7] (1983)	x and y force and pressure 25 parameters selected among 44 reference vector	5220 genuines (2 10 S) 58 W: training set 648 forgeries 12 imitators	Weighted Euclidean distance (standard deviation distance)	FRR : 1.5% FAR : 1.5% FRR : 2.25% FAR : to 3%	Over 4 months – 3 trials random forgeries Skilled forgeries
DeBruyne [8] (1983)	x and y positions 4 parameters selected among 18	71 genuines from 11 W 52 forgeries	Weighted parameters grading maxilikelihood ratio test	FRR : 3% FAR : 2%	Reference parameters evaluated from 10 specimens
Dyche [9] (1969)	x and y positions 40 parameters based on time length and on positions. Speed and acceleration self and cross moments	Training: 333 genuines from 1 signer 167 forgeries from 3 amateur forgers Tests: 125 signatures from each classes	Likelihood ratio nearest neightbour linear boundary	FRR : 0% FAR : 0%	Similar results using a subset of 15 parameters

continued

continued

Farag and Chien [10] (1972)	*x* and *y* positions recording duration 2 S (256 coordinates) chain coded vector histogram	200 genuines (20 S 10 W) 250 forgeries: 10 users imitated 5 imitators	City block distance Mahalanobis distance	FRR : 14–23% FAR : 14–24%	
Groupement, Carte Bleue [11] (1984)	*x* and *y* positions	5656 genuines (5 S 460 W): training 3509 S : test	Correlation 0–99	FRR : 3.5%	Reference 3 consistent signatures 3 trials over one year credit card in a public store
Gupta and Joyce [4] (1995)	*x* and *y* positions 9 global features	5220 genuines from 58 subjects 648 forgeries from 12 forgers	Weighted distance	FRR : 0.5% FAR : 10%	10 sample signatures used for reference signature
Haberman and Fefjar [12] (1976)	pressure 9 local or global features	209 W (170 M. 39 F) 2645 FRR tests 106,505 FAR tests		FRR : 6.81% FAR : 3.19%	Higher error rates for females using the VERIPEN system
Impedovo, Castellano, Pirlo and Dimauro [13] (1990)	*x* and *y* positions	232 genuines 434 forgeries	Spectral analysis	FRR : 3.5% FAR : 4.2%	
Lam and Kamins [14] (1989)		8 genuines from one signer 152 forgeries from 19 forgers	Spectral and discriminant analysis	FRR : 0% FAR : 2.5%	
Hale and Paganini [15] (1980)	force 15 Haar coefficients 18 waveform physical features	Typical experiments: 500 genuines 97 S 951 verification attempts 181 forgery attempts (non–observing signers) 59 forgery attempts (observing signers)	Weighted distance	FRR : 1.5% FAR : 1.2% 2.5%	Non–observing signer Observing signer general result for 9976 signatures
Herbst and Liu [16] (1979)	*x* and *y* acceleration and pressure	6000 genuines (6 S 201 W): training forgeries 40 users imitated 10 S/user 40 imitators	Regional correlation	FRR : 1.7% FAR : 0.4% FAR : 0.022%	3 trials — over 6 months skilled forgeries With random forgeries
Herbst and Liu [17] (1979)	*x* and *y* acceleration	1042 genuines (5 S 40 W): training 750 forgeries		FRR : 2.4% FAR : 3.2%	3 trials
Ibrahim and Levrat [18] (1979)	*x* and *y* position, velocity and acceleration global parameters peak matching zero crossing	47 genuines (1 S 10 W): training 36 forgeries 2 imitators	Euclidean distance	FRR : 19% FAR : 5.5%	Signing with one finger isntead of a pen
Lorette [9] (1983)	*x* and *y* position global & local & dynamic 7 parameters	210 genuines (5 S 14 W): training (10 S 14 W): test no forgeries	Clustering analysis	FRR : 6%	Only one trial

continued

continued

Mauceri [19] (1965)	acceleration and pressure (pen paper contact) 10 indices related to frequency spectraq 19 local and global features	2350 genuines (45 S 40 W) no forgeries 20 S/W learning 25 S/W test	Squared distance Weighted squared distance	60% recognition rate 90% recognition rate	Partial results from 10 subjects only
Mohankrish –nan, Paulik and Khalil [20] (1993)		928 genuines (58 S 16 W)	Nonstationary autoregressive model	FRR : 8% FAR : 8%	Random forgeries
Sato and Kogure [21] (1982)	*x* and *y* position and pressure complex normalised functions shape, motion and pressure vector	100 genuines (10 S 11 W) 330 forgeries 1 user 10 S/user 3 imitators	Dynamic Programming Matching Mahalanobis pseudo–distance	FRR : 1.8% FAR 0%	No information on training set
Sternberg [22] (1975)	pressure reduced to a few "measures" via mathematical transformation	1000 genuines (10 S 100 W) 50 imitators each one forging 8 subjects sign 3 times random forgeries		FRR : 0.7% FAR : 1.8% FAR : 1.8%	Over 2 months with delibrate forger commercialized by veripen as SIGNAC system
H. Taguchi, K. Kiriyama, E. Tanaka and K. Fujii [23] (1989)		105 genuines 105 forgeries		FRR : 6.7% FAR : 0%	
Worthingto n, Chainer, Williford and Gundersen [24] (1985)	*x* and *y* acceleration and pressure	5000 genuines (6 S 108 W): training 4700 S : test	5 similarity measures	FRR : 1.77% FAR : 0.28% 2.33%	Tests over 9 months
Yang, Widjaja and Prased [25] (1993)		496 genuines no forgeries	hidden Markov models	FRR : 4.44% FAR : 1.79%	Techniques inspired from speech recognition
Yasuhara and Oka [26] (1977)	*x* and *y* position and pressure calculated force function	100 genuines (3 S 10 W): training (7 S 10 W): no training	D.P. matching Euclidean distance	FRR : 3% FAR : < 1%	Key words used instead of signature
Zimmerma n and Varady [27] (1985)	pressure (one bit) Walsh functions power spectrum 40 low sequency harmonics	90 genuines (3 S 9 W): training	Fisher discriminant functions linear classifier	FRR : 30–50% FAR : 4–12%	Over several weeks
Zimmerma nn and Werner [28] (1978)	*x* and *y* positions acceleration derived from *FIR* filter	Not specified	Straightforward digital correlation	Less than 10% misrecognition	Preliminary tests

{1} M. Achemlal, M. Mourier, G. Lorette and J.P. Bonnefoy, **Dynamic signature verification**, *Proc. 4th Int. Conf. Exhib. On Comput. Security*, Monte–Carlo (1986).

{2} R. Beatson, **Signature dynamics in personal identification**, *Proc. 4th World Congr. Comput. Commun. Security and Protection*, pp. 179–196, Paris (1985).

{3} L. Bechet, **Reconnaissance dynamique de la signature**, *6th Conv. Eur. Financial Marketing Ass.* Pp. 93–97, Montreux, 1984.

{4} J.P. Bonnefoy, P. Jounet, G. Lorette and M. Gaudaire, **Reconnaissance automatique en temps réel de signatures manuscrites: définition et mise en oeuvre d'un critère général**, *3rd Congrès Reconnaissance des Formes et Intelligence Artificielle*, pp. 267–275, Nancy, 1981.

{5} J.J. Brault and R. Plamondon, **Histogram classifier for characterization of handwritten signature dynamics**, *Proc. 7th Int. Conf. Pattern Recognition*, Vol. 1, pp. 619–622, Montréal, 1984.

{6} National Physical Laboraty, **Verisign: a system for automatic verification of signatures**, Technical report, 1975.

{7} H.D. Crane and J.S. Ostrem, **Automatic signature verification using a three–axis–force–sensitive pen**, *IEEE Trans. Syst. Man. Cybernetics*, **13**, 329–337, 1983.

{8} P. DeBruyne, **Signature verification using holistic measures**, *Comput. Security* **4**, 309–315, 1985.

{9} J.W. Dyche, **Positive personal identification by handwriting**, *Proc. Carnahan Conf. On Electron. Crime Countermeasures*, University of Kentucky ORES, Lexington, 1969.

{10} R.F. Farag and Y.T. Chien, **On–Line signature verification**, *Proc. Int. Conf. on On–line Interactive Comput.* p.403, Brunel University, London, 1972.

{11} Groupement Carte Bleue, **L'expérience de reconnaissance dynamique de signatures des établissements Carte Bleue**, Rapport final, 1984.

{12} W. Haberman and A. Fefjar, **Automatic identification of personnel through speaker and signature verification: system description and testing**, *Proc. 1976 Carnahan Conf. on Crime Countermeasures*, J.S. Jackson and R.W. de Vore, eds, pp.23–30, Univ. Of Kentucky, 1976.

{13} S. Impedovo, M. Castellano, G. Pirlo and G. Dimauro, **On–line signature verification system through stroke analysis**, *Proc. AFCET Int. Conf. on New Concepts in Computer Science*, 1990, pp. 47–53.

{14} C.F. Lam and D. Kamins, **Signature recognition through spectral analysis**, *Pattern Recogn.* **22**, 1, pp. 39–44, 1989.

{15} W.J. Hale and B.J. Paganini, **An automatic personal verification system based on signature writing habits**, *Proc. 1980 Carnahan Conf. on Crime Countermeasures*, pp.1210125, Univ. Of Kentucky, Lexington, KY, 1980.

{16} N.M. Herbst and C.N. Liu, **Automatic signature verification**, *Computer Analysis and Perception*, C.Y. Suen and R. De Mori, eds, pp. 83–105. CRC Press, Boca Raton, FL, 1982.

{17} N.M. Herbst and C.N. Liu, **Automatic verification of signatures by means of acceleration patterns**, *Proc. IEEE Comp. Soc. Conf. on Pattern Recognition and Image Processing*, pp. 332–337, 1977.

{18} B. Ibrahim and B. Levrat, Improved security through on–line signature recognition, *Proc. Engl. Conf. on Applic: Inf. Technol. Of the IFIP*, 491–498, London, 1979.

{19} A.J. Mauceri, **Feasibility study of personal identification by signature verification,** Report No. SID 65 24 RADC TR 65 33, Spacfe and Information System Division, North American Aviation Anaheim, CA, 1965.

{20} N. Mohankrishnan, M.J. Paulik and M. Khalil, **On–line signature verification using a nonstationary autoregressive model representation,** *IEEE Int. Symp. On Circuits Systems*, Vol. 4, Chicago, USA, 1993, pp. 2303–2306.

{21} Y. Sato and K. Kogure, **On–line signature verification based on shape, motion and handwriting pressure,** *Proc. 6th Int. Conf. on Pattern Recognition*, Vol. 2, pp. 823–826, Munich, 1982.

{22} J. Sternberg, **Automated signature verification using handwriting pressure,** 1975 Wescon Technical Papers, Nos 31/4, Los Angeles, 1975.

{23} H. Tagushi, K. Kiriyama, E. Tanaka and K. Fujii, **On–line recognition of handwritten signatures by feature extraction of pen movements,** *Syst. Computers Jpn.*, **20,** 10 (1989) 1–14.

{24} T.K. Worthingtion, T.J. Chainer, J.D. Williford and S.C. Gunderson, **IBM dynamic signature verification,** *Computer Security, IFIP 1985*, pp. 129–154, Elsevier, North Holland, 1985.

{25} L. Yang, B.K. Widjaja and R. Prasad, **On–line signature verification applying hidden Markov models,** *Proc. Of 8th Scandinavian Conf. on Image Analysis*, Tromso, Norway, 1993, pp. 1311–1316.

{26} M. Yasuhara and M. Oka, **Signature verification experiment based on nonlinear time alignment: a feasibility study,** *IEEE Trans. Syst. Man. Cybernetics* **17,** 212–216, 1977.

{27} K.P. Zimmermann and M.J. Varady, **Handwriter identification from one–bit quantized, pressure patterns,** *Pattern Recognition* **18,** 63–72, 1985.

{28} K.P. Zimmermann and C.L. Werner, **SIRSYS — a reasearch facility for handwritten signature analysis,** *Proc. 1978 Carnahan Conf. on Crime Countermeasures*, J.S. Jackson and R.W. De Vore, eds. Pp. 153–155, University of Kentucky, Lexington, 1978.

Appendix B – Specifications ACECAT II graphics tablet

Physical Information:

Active Area	5.0" x 5.0" (127mm x 127mm)
Outside Dimension	8.0" x 9.5" (204mm x 240mm)
Weight	1.05Lbs (0.48kgs)
Tilt Angle	3.7

Functional Information:

Resolution	Up to 2000 Lines Per Inch (LPI)
Accuracy	+/–0.01" (+/–0.254mm)
Sampling Rate	Up to 128PPS (max.)
Proximity	0.27" (7.0mm) 2–buton stylus–pen
Interface	RS–232C
Data Format	ACECAT, MM1201

Electrical:

Power Source	Via keyboard adapter cable
Input Voltage	5 VDC
Current Draw	120 mA

Environmental:

Operation Temp.	41(to 122(F (5(to 50(C)
Storage Temp.	14(to 122(F (–10(to 50(C)
Humidity	20% to 95% (Non–Condensing)

De ACECAT II hardware has two operation modes; the native and the MM1201 (an industry standard twelve by twelve inch digitiser) emulation mode. In native operation mode, the RS–232C port is set to 9600 baud, 8 bit, odd parity and 1 stop bit. When pressure is applied onto the stylus–pen, up to 128 times per second the ACECAT generates a sample with the format given in Figure 13. A '1' as value of the most significant bit of byte 1 only can be used for synchronisation. The 14–bit co–ordinates are send as two 7–bit strings, the least significant bits first.

The ACECAT is capable of receiving commands from the host. Below are some very usable commands which can be send through the RS–232C port:

* **40h** – This command tells the ACECAT II to send samples continuously (unless out of proximity), so we don't have to deal with gaps in the time axis. Command **00h** can be used to select the default behaviour.

* **61h** – This command tells the ACECAT II to send back the graphics tablet's resolution. The power–up default is 2500 by 2500 points (500 LPI). These values can be used to detect the ACECAT.

```
| byte 1 | byte 2 | byte 3 | byte 4 | byte 5 |
|1p??bbbb|0xxxxxxx|0xxxxxxx|0yyyyyyy|0yyyyyyy|
p = proximity        b = button status
x = x co-ordinate    y = y co-ordinate
```

Fig. 13: ACECAT data format

Appendix C – data acquisition source code

For programming PC hardware (the serial port in this case) I find myself using the assembler more often than a HLL as C or PASCAL. Slowly I'm getting used using a HLL (and some in–line assembly...), but my sampler still has some assembly source for the serial communication part including the interrupt handler:

```
; DIG2, the easy part
code    segment
        assume  cs:code
.286
; UART routines
        public init, restore, transmit, receive, check
index   dw ?
port    dw ?
oldIsr  dw ?,?
bsize   equ 64
f       dw 0
l       dw 0
n       db 0
buf     db bsize dup (?)
; Initialize UART routines
init    proc
  push  bp
  mov   bp,sp
  xor   ax,ax                   ; Get Port address from BIOS
  mov   bx,ss:bp[4]
  mov   cx,1
  shl   bx,1
  mov   es,ax
  mov   dx,es:400h[bx]
  mov   cs:port,dx
  and   cl,dh
  mov   bx,cx
  shl   bx,2
  mov   cs:Index,bx
  mov   ax,es:2Ch[bx]           ; Get 'old' Isr vector
  mov   cs:oldIsr[0],ax
  mov   ax,es:2Eh[bx]
  mov   cs:oldIsr[2],ax
  mov   es:2Ch[bx],offset newIsr ; Set 'new' Isr vector
  mov   es:2Eh[bx],cs
  in    al,dx                   ; Clear buffer
  mov   al,80h                  ; Select baud-rate
  add   dl,3
  out   dx,al
  mov   ax,000Ch                ; baud-rate = 9600
  sub   dl,3
  out   dx,ax
  mov   al,0Bh                  ; 8 bit, odd parity, 1 stop
  add   dl,3
  out   dx,al
  inc   dx                    ; Enable transfers and interrupts
  out   dx,al
  mov   ah,0F7h                 ; Enable interrupts (PIC)
  rol   ah,cl
  in    al,021h
  and   al,ah
  out   021h,al
  mov   al,1                    ; Enable interrupt 'received data'
  sub   dl,3
  out   dx,al
  pop   bp
  ret   2
init    endp
restore proc
  xor   ax,ax                ; Disable interrupt 'received data'
  mov   dx,cs:port
  inc   dx
  out   dx,al
```

```
  add   dx,3                    ; Disable transfers and interrupts
  out   dx,al
  mov   es,ax
  mov   ah,008h                 ; Disable interrupts (PIC)
  mov   cl,1
  and   cl,dh
  shl   ah,cl
  in    al,021h
  or    al,ah
  out   021h,al
  mov   ax,cs:oldIsr[0]         ; Restore 'old' Isr
  mov   bx,cs:Index
  mov   es:2Ch[bx],ax
  mov   ax,cs:oldIsr[2]
  mov   es:2Eh[bx],ax
  ret
restore endp
transmit proc
  push  bp
  mov   bp,sp
  mov   dx,cs:port              ; Wait for e.g 'clear to send'
  add   dl,5
t1: hlt
  in    al,dx
  test  al,20h
  jz    t1
  inc   dl
  in    al,dx
  dec   dl
  and   al,30h
  cmp   al,30h
  jne   t1
  mov   al,ss:bp[4]             ; Transmit data
  sub   dl,5
  out   dx,al
  pop   bp
  ret   2
transmit endp
receive proc
  cmp   cs:n,0                  ; Wait for received data
  jne   r1
  hlt
  jmp   receive
r1: mov   bx,cs:f               ; Return data & increment pointer
  mov   al,cs:buf[bx]
  inc   bx
  and   bx,(bsize-1)
  mov   cs:f,bx
  dec   cs:n
  ret
receive endp
check   proc
  mov   al,cs:n
  ret
check   endp
newIsr  proc
  push  dx bx ax
  mov   bx,cs:l                 ; Get data
  mov   dx,cs:port
  in    al,dx
  mov   cs:buf[bx],al
  inc   bx                      ; Store data
  and   bx,(bsize-1)
  mov   cs:l,bx
  inc   cs:n
  mov   al,20h                  ; End_Of_Interrupt
  out   20h,al
  pop   ax bx dx
  iret
newIsr  endp
code    ends
        end
```

The main body is written in Turbo PASCAL. Still many links to the assembly past can be found...

```
program digitiser;

(* Defines *)
 type sample = array[0..4] of byte;
 var  timer : byte absolute 0:$46C;
      video : array[0..38399] of byte absolute $A000:0;

(* UART functions *)
 procedure init(comm : word); external;
 procedure restore; external;
 procedure transmit(x : byte); external;
 function  receive : byte; external;
 function  check : byte; external;
 {$L dig2.obj}
 procedure getXY(var s : sample); var i : byte;
  begin for i := 1 to 4 do s[i] := receive end;

(* Keyboard functions *)
 function checkkey : word; assembler;
  asm mov ah,11h; int 16h; jnz @key; xor ax,ax; @key: end;
 function getkey : word; assembler; asm mov ah,10h; int 16h end;

(* VIDEO functions *)
 procedure box(x,y,w,h : word; v : byte); begin
   inc(x,y*80); repeat fillchar(video[x],w,v); inc(x,80); dec(h) until
h = 0
 end;
 procedure blt(x,y1,y2,w,h : word);
  begin
   y1 := y1*80+x; y2 := y2*80+x;
   repeat
    move(video[y1],video[y2],w); inc(y1,80); inc(y2,80); dec(h)
   until h = 0
  end;
 procedure black(x,y : word); begin
   y := 35145 - y * 80 + x shr 3;video[y] := video[y] and lo($FF7F shr
(x and 7))
 end;
 procedure invert(x,y: word); begin
   y := 35145 - y * 80 + x shr 3; video[y] := video[y] xor $80 shr (x
and 7)
 end;
 procedure white(x,y : word); begin
   y := 35145 - y * 80 + x shr 3; video[y] := video[y] or $80 shr (x
and 7)
 end;
 procedure text(x,y : word; s : string);
  type font = array[-64..255,0..15] of byte;
  var fp : ^font absolute 0:$7C; i,r : byte; p : word;
  begin
   inc(x,y*80);
   for i := 1 to ord(s[0]) do begin
    p := x + i;
    for r := 0 to 15 do begin
     video[p] := fp^[ord(s[i]),r]; inc(p,80);
     video[p] := fp^[ord(s[i]),r]; inc(p,80)
    end
   end
  end;
 procedure scroll; var x,y : word;
  begin
   x := 2564;
    for y := 0 to 375 do begin move(video[x + 2560], video[x], 16);
inc(x,80) end;
```

```
   for y := 0 to 31 do begin fillchar(video[x],16,0); inc(x,80) end
  end;
 procedure writes(s : string); begin scroll; text(4,408,s) end;

(* Main program *)
const max = 4096;

var b, oldmode, t : byte;
    buff : array[0..max-1] of sample;
    key, n, m : word;
    f : file;
    fname : string[12];

begin
 init(0);
{ Detect AceCat II }
 transmit($61); b := 0; t := timer + 2;
 repeat until receive >= 128; getXY(buff[0]);
  if  (buff[0,1]=68)and(buff[0,2]=19)and(buff[0,3]=68)and(buff[0,4]=19)
then begin
  transmit($40);

 { Init Video }
  asm mov ah,0Fh; int 10h; mov oldmode,al; mov ax,11h; int 10h end;

  for b := 0 to 239 do begin
   fillchar(video[b * 160], 80, $55); fillchar(video[b * 160 + 80], 80
,$AA)
  end;
  box(4,32,16,416,0); box(24,32,52,416,255);

 { Init variables }
  key := 0; n := 0; fname := 'DIG00/.DAT';
  writes('DIG v2.1');
 { Main Loop }
  repeat
   writes('Ready');
   repeat
    if check > 0 then b := receive else asm hlt end
   until (b and 193 = 129) or (checkkey > 0);
   if checkkey > 0 then begin
    key := getkey;
    if hi(key) = 31 then
    { Save recording }
     if n > 0 then begin
      repeat
       inc(fname[6]); if fname[6] = ':' then begin
        fname[6] := '0'; inc(fname[5]);
          if fname[5] = ':' then begin fname[5] := '0'; inc(fname[4])
end
       end;
       assign(f,fname); reset(f); close(f)
      until IOResult > 0;
      rewrite(f,sizeof(sample));
      if IOResult = 0 then begin
       BlockWrite(f, buff, n, m);
       if n = m then begin writes('Data saved as'); writes(fname) end
       else begin writes('Error writing'); writes('data file') end;
       close(f)
      end else begin writes('Error creating'); writes('data file') end
     end else writes('No data!!')
   end else begin
   { Recording }
    n := 0; writes('Recording');
    box(24,32,52,416,255);
    repeat
     buff[n,0] := b; getXY(buff[n]);
     if odd(b) then begin
```

```
      black((word(buff[n,1]) + buff[n,2] shl 7) shl 2 div 25,
            (word(buff[n,3]) + buff[n,4] shl 7) shl 2 div 25);
      t := timer + 20
     end;
     inc(n); b := 0;
     repeat
      if check > 0 then b := receive else asm hlt end
     until (t = timer) or (b >= 128)
    until (n = max) or (t = timer);
    while buff[pred(n),0] and 193 <> 129 do dec(n);
   end
  until key = $11B;

  asm mov ah,0; mov al,oldmode; int 10h end;

 end else writeln('Can''t detect AceCat II');
 restore
end.
```

Appendix D – format of .DAT files

The format of the .DAT files generated by the sampler from appendix B is an array of the ACECAT data format given in appendix A. In C notation:

```
typedef struct {
 unsigned char status;
 unsigned char xpos_lo, xpos_hi;
 unsigned char ypos_lo, ypos_hi;
} ACECAT;
```

The C code below can be used to convert ACECAT data to conventional C data types.

```
void convert (short int no_samples, ACECAT samples[],
              short int xpos[], short int ypos[], char pendown[]);
{
 int i;
 for (i = 0; i < no_samples; i++) {
  xpos[i] = samples[i].xpos_lo + (samples[i].xpos_hi << 7);
  ypos[i] = samples[i].ypos_lo + (samples[i].ypos_hi << 7);
  pendown[i] = samples[i].status & 0x01;
 }
}
```

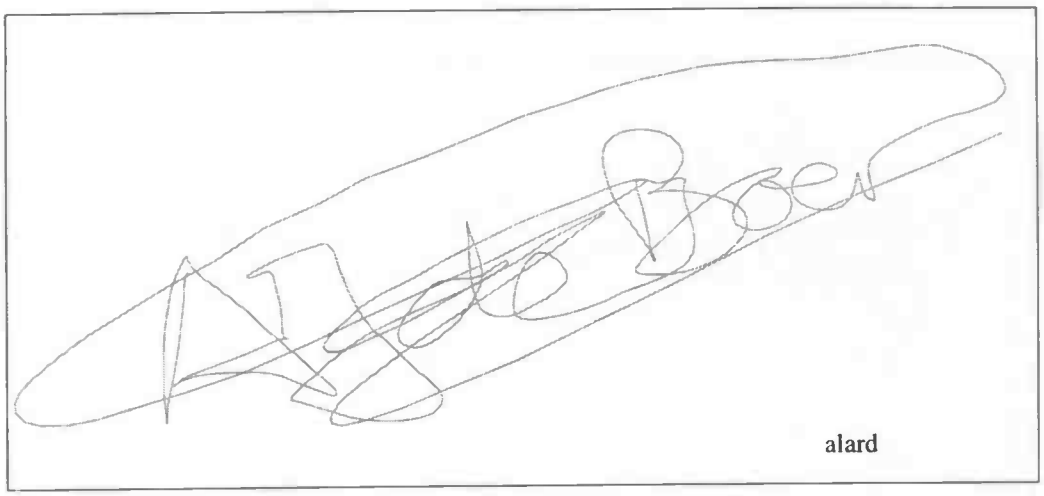
Appendix E – Assembled signatures

Overview assembled signatures:

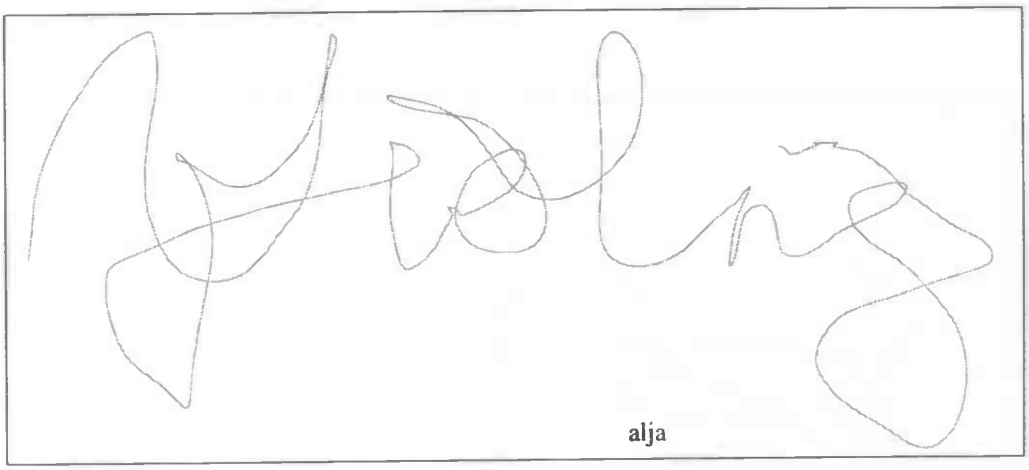
Writer	#sig	total time mean	total time SD/mean	time 1st CC mean	time 1st CC SD/mean	#X-crossings mean	#X-crossings SD/mean	#Y-crossings mean	#Y-crossings SD/mean
Herman	25	372.6	6.1%	140.2	6.9%	19.0	12.5%	24.1	8.7%
Alard	16	557.8	7.1%	48.8	12.5%	38.9	10.4%	41.3	10.7%
Alja	16	580.2	5.1%	63.6	28.4%	33.8	15.1%	33.2	13.7%
Eddy	26	499.9	7.5%	211.8	62.6%	34.2	10.3%	35.2	9.9%
Emilia	21	599.0	0.9%	179.7	22.4%	34.0	18.5%	34.9	16.5%
Hans	26	84.8	5.4%	82.0	18.8%	4.8	21.6%	6.1	21.4%
Hessel	20	344.6	9.2%	64.1	12.3%	24.0	14.3%	25.1	12.1%
Immelien	25	470.0	3.8%	101.9	18.7%	37.1	12.0%	34.1	7.6%
Jan-Jaap	25	281.8	7.2%	281.8	7.2%	14.0	12.4%	13.6	11.6%
Jos	27	162.0	17.4%	117.2	18.5%	5.6	40.7%	11.0	22.0%
Josien	16	440.9	10.9%	58.7	14.3%	28.1	13.4%	29.2	11.1%
Mark	16	591.1	6.6%	275.9	74.3%	38.9	13.5%	41.2	10.9%
Walter	25	335.1	25.2%	308.8	35.5%	18.8	16.7%	17.8	20.4%

CC stands for 'connected component'

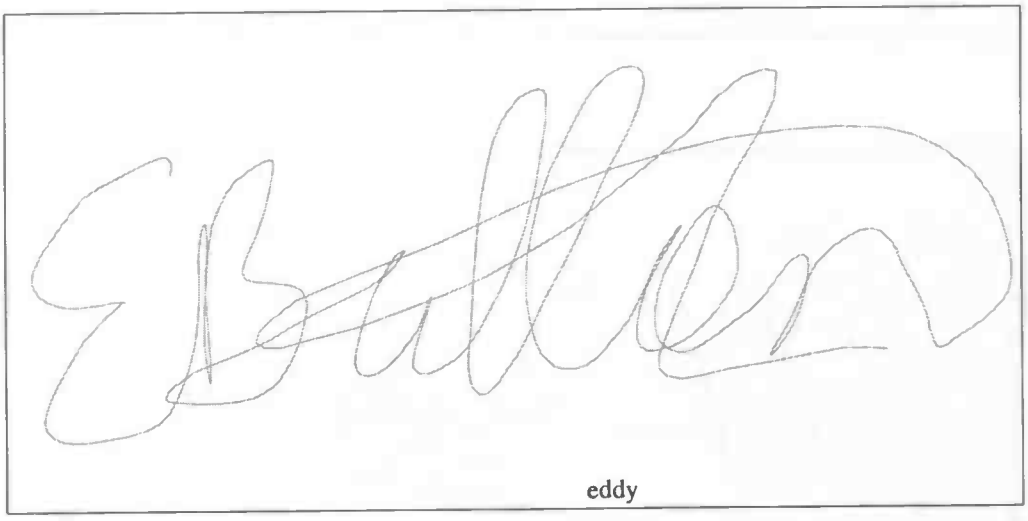
herman



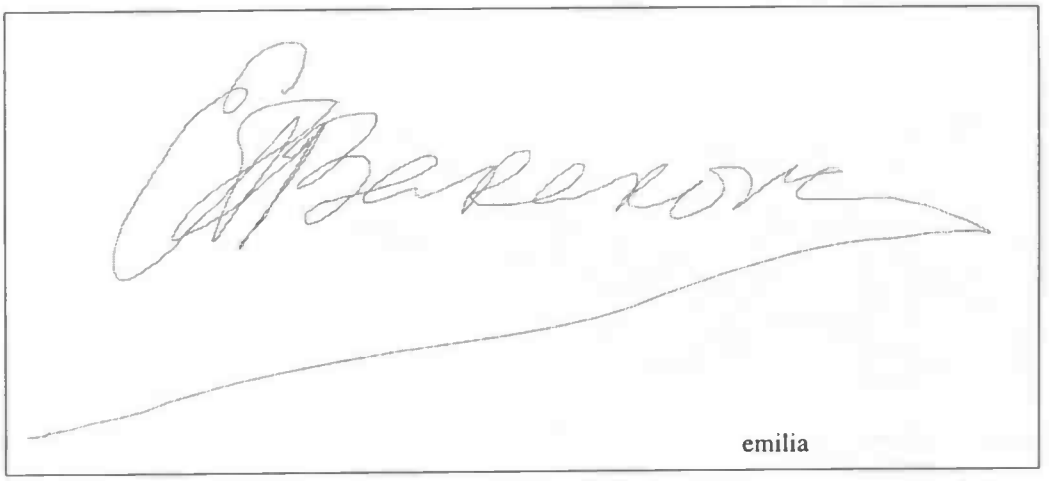
alard



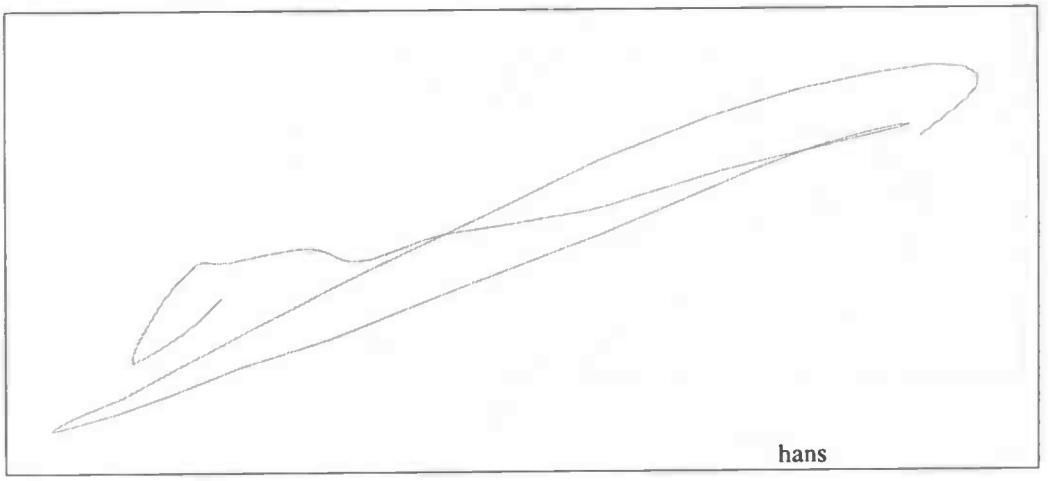
alja



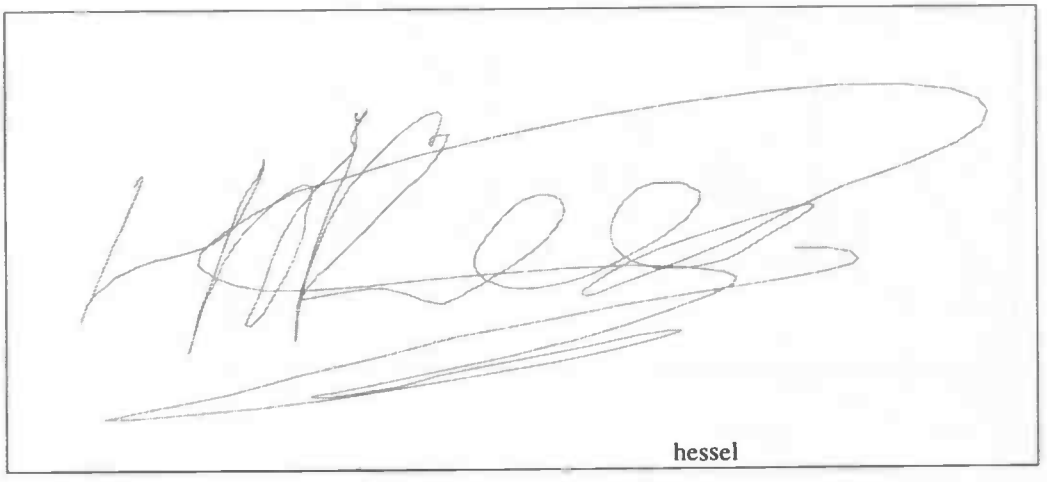
eddy



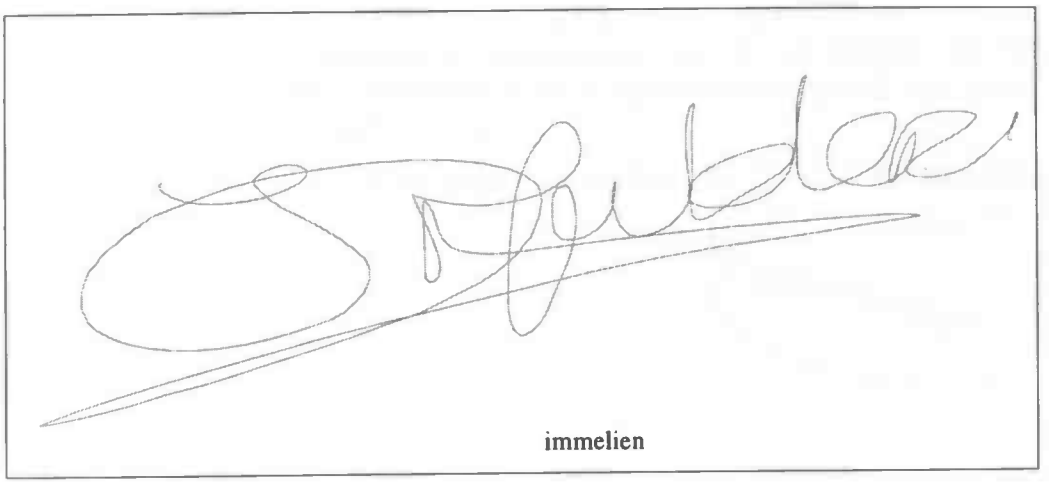
emilia



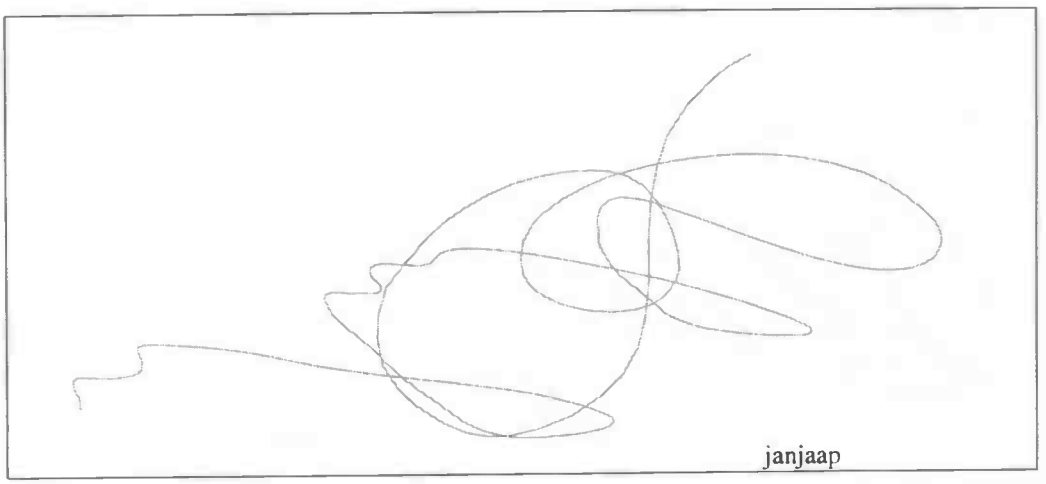
hans



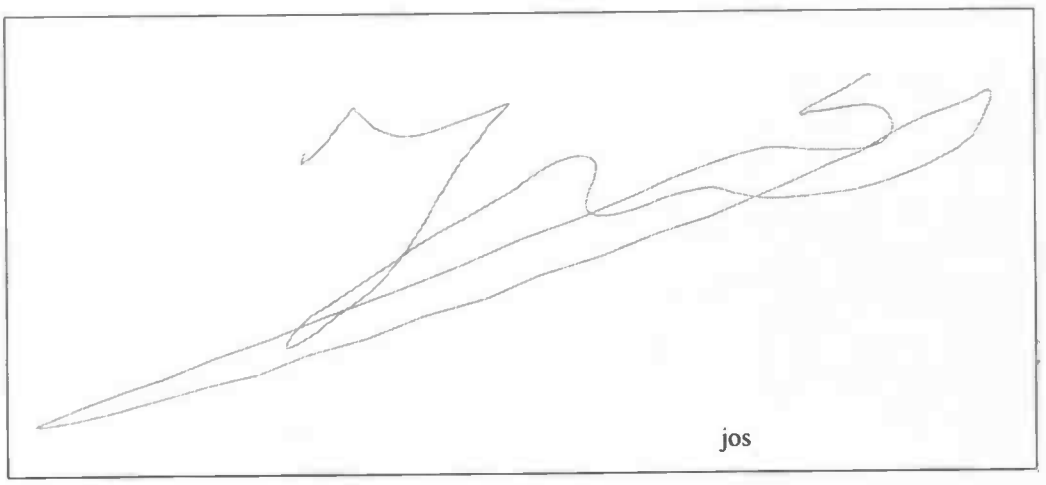
hessel



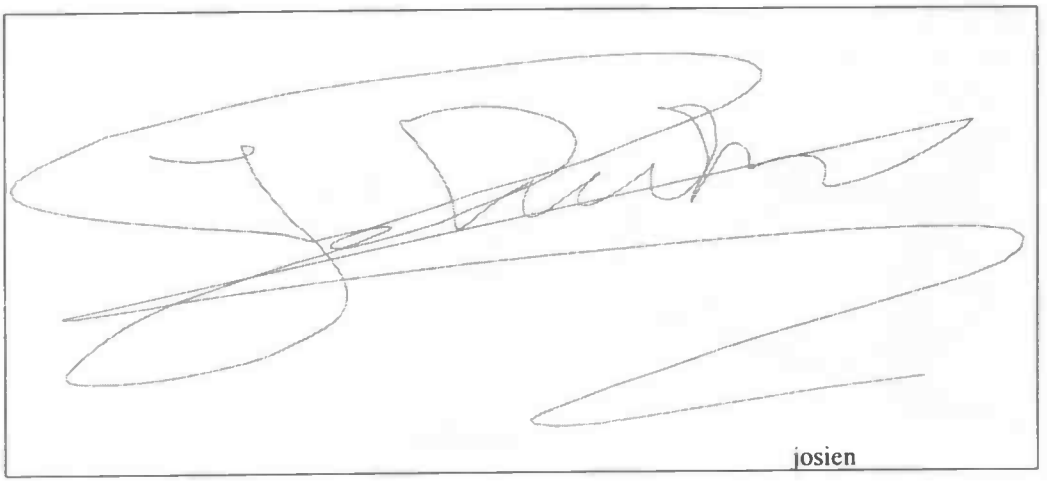
immelien



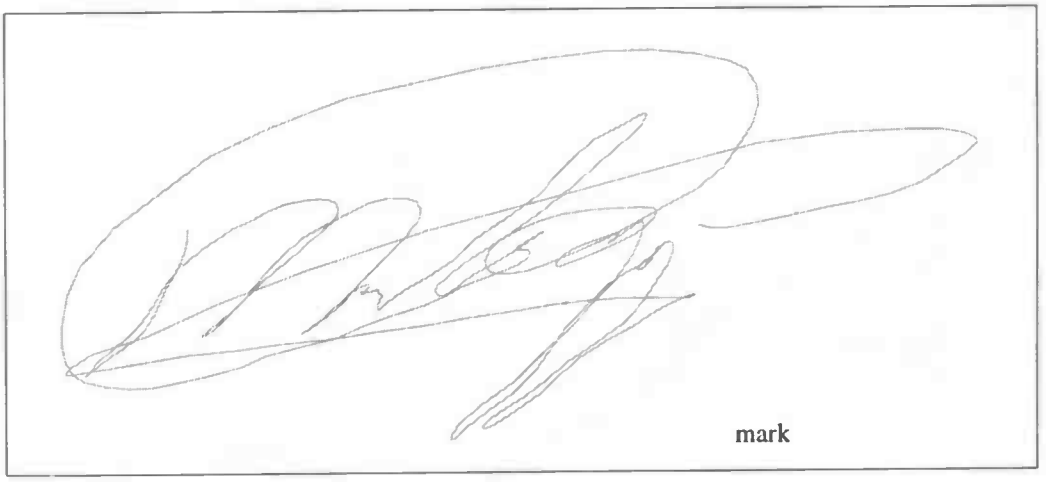
janjaap



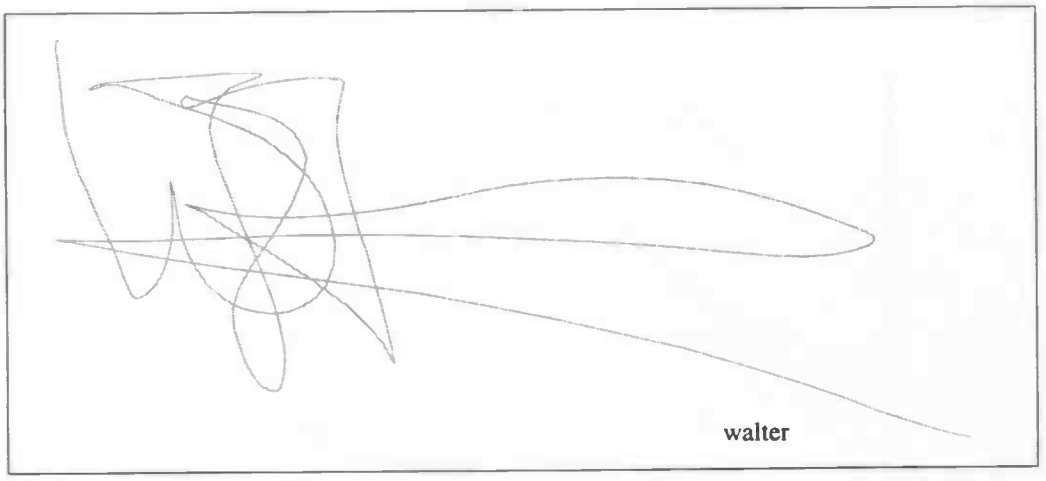
jos



josien



mark



walter

ACKNOWLEDGEMENTS

Gratitudes are expressed to the 'brain' group of the department of technical computing sciences, including M. Diepenhorst and M. ter Brugge and the volunteers who also have contributed several signatures each: Alard, Alja, Eddy, Hessel, Immelien, Jan–Jaap and Josien.

Special gratitudes are expressed to my mentors Prof. Dr. Ir. L. Spaanenburg and Jos Nijhuis.