

Disclaimer: This document was part of the DSP Solution Challenge 1995 European Team Papers. It may have been written by someone whose native language is not English. TI assumes no liability for the quality of writing and/or the accuracy of the information contained herein.

***Implementing a DSP Kernel for Online
Dynamic Handwritten Signature
Verification Using the TMS320 DSP
Family***

**Authors: H. Dullink, B. van Daalen, J. Nijhuis,
L. Spaanenburg, H. Zuidhof**

EFRIE, France
December 1995
SPRA304



IMPORTANT NOTICE

Texas Instruments (TI) reserves the right to make changes to its products or to discontinue any semiconductor product or service without notice, and advises its customers to obtain the latest version of relevant information to verify, before placing orders, that the information being relied on is current.

TI warrants performance of its semiconductor products and related software to the specifications applicable at the time of sale in accordance with TI's standard warranty. Testing and other quality control techniques are utilized to the extent TI deems necessary to support this warranty. Specific testing of all parameters of each device is not necessarily performed, except those mandated by government requirements.

Certain application using semiconductor products may involve potential risks of death, personal injury, or severe property or environmental damage ("Critical Applications").

TI SEMICONDUCTOR PRODUCTS ARE NOT DESIGNED, INTENDED, AUTHORIZED, OR WARRANTED TO BE SUITABLE FOR USE IN LIFE-SUPPORT APPLICATIONS, DEVICES OR SYSTEMS OR OTHER CRITICAL APPLICATIONS.

Inclusion of TI products in such applications is understood to be fully at the risk of the customer. Use of TI products in such applications requires the written approval of an appropriate TI officer. Questions concerning potential risk applications should be directed to TI through a local SC sales office.

In order to minimize risks associated with the customer's applications, adequate design and operating safeguards should be provided by the customer to minimize inherent or procedural hazards.

TI assumes no liability for applications assistance, customer product design, software performance, or infringement of patents or services described herein. Nor does TI warrant or represent that any license, either express or implied, is granted under any patent right, copyright, mask work right, or other intellectual property right of TI covering or relating to any combination, machine, or process in which such semiconductor products or services might be or are used.

TRADEMARKS

TI is a trademark of Texas Instruments Incorporated.

Other brands and names are the property of their respective owners.

CONTACT INFORMATION

US TMS320 HOTLINE	(281) 274-2320
US TMS320 FAX	(281) 274-2324
US TMS320 BBS	(281) 274-2323
US TMS320 email	dsph@ti.com

Contents

Abstract	7
Product Support on the World Wide Web	8
Introduction.....	9
Signature Verification	10
Simple Dynamic Handwritten Signature Verification Technique	13
Data Acquisition.....	13
Preprocessing.....	13
Feature Extraction	13
Enrollment.....	14
Comparison	14
Decision-Making	14
Introducing Neural Clustering	16
Software Architecture	20
Discussion.....	25
Summary.....	26
Index of Terms	26
References	26

Figures

Figure 1.	Data Flow Diagram of a Signature Verification System	10
Figure 2.	Data Flow Diagram of a Neural Signature Verification System	16
Figure 3.	InterAct Observations on a Small Experiment.....	18
Figure 4.	A leveled network.....	22
Figure 5.	Neural Software Architecture and Table Composition	23

Tables

Table 1.	Evaluating Different Number of Sample Signatures from Gupta and Joyce.....	14
----------	--	----

Implementing a DSP Kernel for Online Dynamic Handwritten Signature Verification Using the TMS320 DSP Family

Abstract

Biometric authentication techniques are in high demand for entrance monitoring and security systems. The techniques must be cheap, reliable and, foremost, unintrusive to the authorized person.

The only technique to meet all three requirements is handwritten signature verification. A recent development shows how focussing on the dynamic properties enables a high quality result at a drastically reduced computational effort. Here, we discuss an elegant realization, based on the TMS320 line of digital signal processors, that serves as a kernel in small to large safety and security related systems.

This document was an entry in the 199x DSP Solutions Challenge, an annual contest organized by TI to encourage students from around the world to find innovative ways to use DSPs. For more information on the TI DSP Solutions Challenge, see TI's World Wide Web site at www.ti.com.



Product Support on the World Wide Web

Our World Wide Web site at www.ti.com contains the most up to date product information, revisions, and additions. Users registering with TI&ME can build custom information pages and receive new product updates automatically via email.



Introduction

Authentication has become an essential part of highly computerized services and/or security-sensitive installations in modern society. A number of techniques researched over the past decade checks on the person's face, voice, iris or fingerprint.

For optimal security, practical systems usually apply at least two different authentication techniques. However, the techniques tend to require additional measurements to be usable or inoffensive to the person involved.

A popular means of authentication historically has been the handwritten signature. Though such signatures are never the same for the same person at different times, there appears to be no practical problem for human beings to discriminate visually the real signature from the forged one. It will be extremely useful when an electronic device can display at least the same virtuosity.

Nevertheless, signatures do get forged. Most of the authentication techniques used by machines to detect such forgeries are very complex. Even the most trained professionals find it difficult to identify a painting by Van Gogh, Brueghel, or Rembrandt by looking at the signature. They also look at other features, such as (chemical) composition of the paint and the type of brush used. Signatures on modern artwork are made impossible to forge by mixing the ink with certain small parts of the artist's DNA. For large-scale uses, such as banking, we cannot permit ourselves this luxury. Moreover, with the introduction of electronic pen-based human interfaces, the use of ink has disappeared and one has to rely on other means for automatic authentication.

Automatic verification is far from perfect.¹ Most techniques require complex functions and a lot of computing power; however, as shown in this paper, some techniques are simple enough for a low-cost DSP.

In this application report, we first introduce the basic signature verification terminology. Then we review the work of Oupta and Joyce as it provides a reasonable starting point for realization in TMS320 DSP-technology.^{2 3} We add neural clustering techniques to enhance the discriminating power and arrive at a very simple and low-cost solution that can be embedded in existing pen-based systems, such as handheld computers and transaction units.

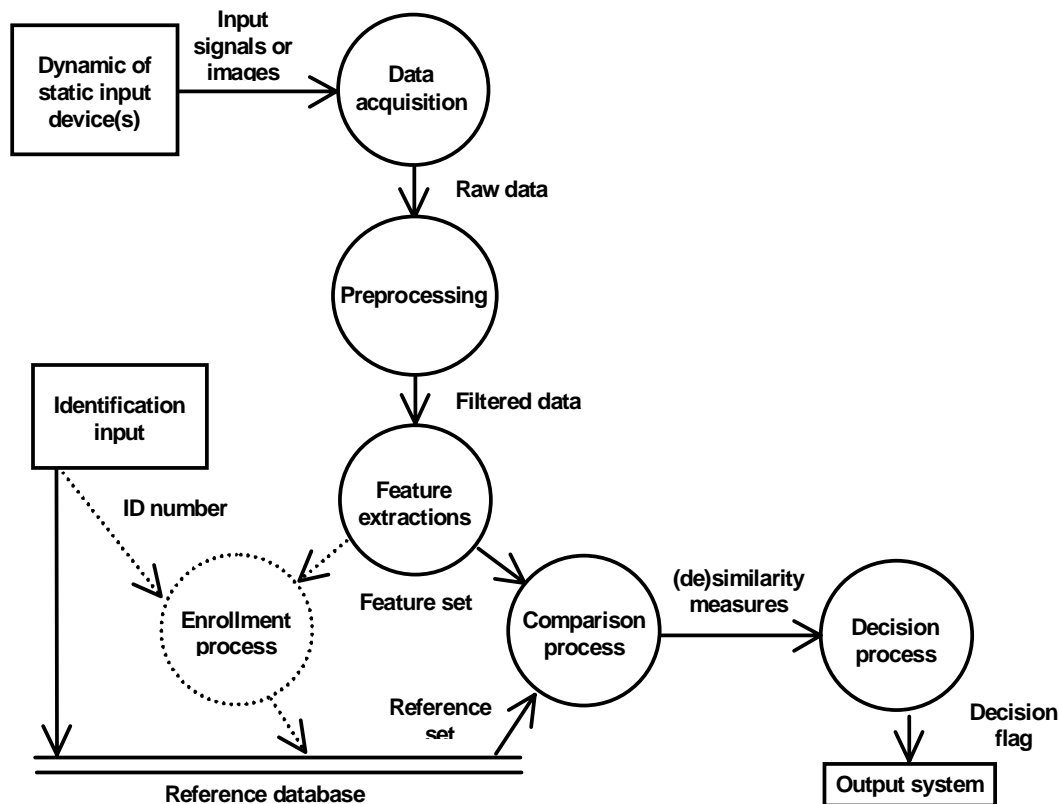
Signature Verification

The design of a signature verification system requires solutions to five types of problems (see Figure 1):

- ❑ Data acquisition
- ❑ Preprocessing
- ❑ Feature extraction
- ❑ Comparison process
- ❑ Performance evaluation

A static signature verification system receives a 2-D image as input from a camera or scanner. Such a system requires a lot of memory and computing power to process the images. The major algorithmic challenge is the required invariance to the current disposition of the writer: no two signatures are fully identical, even after transformation.

Figure 1. Data Flow Diagram of a Signature Verification System





A dynamic signature verification system gets its input from a digitizer or other, usually pen-based, dynamic input device. The signature is then represented as one or several time-varying signals. In other words, the verification system focuses on how the signature is being written rather than how the signature was written. This provides a better means to grasp the individuality of the writer but fails to recognize the writing itself.

Intuitively, this must be correct, being fully in-line with science fiction literature: "The irregularity of the hammer blows used by each artisan followed characteristic patterns to an extent that the maker can be identified without question by sampling that pattern. Collectors developed the method to verify authenticity. It's as definite as an eye print, more positive than any skin-print anomaly," as Herbert stated on pg.165.⁴

The performance of a signature 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 rate) and type II (FAR: false acceptance rate) errors. As the ideal case (i.e., 0 percent on both errors) is questionable to exist, a choice has to be made depending on the application between one of the two error rates equal to zero or the minimization of the total error, $FRR + FAR$. For entry systems, the false rejection is the most important; for security systems, the false acceptance is most important.

Dynamic signature verification methods can be classified in two main groups. The first group contains methods dealing with functions as features. In this case, the complete signals (i.e., position, pressure, velocity, acceleration vs. time, etc.) are considered as, or represented by, mathematical time functions whose values directly constitute the feature set. In the second group, the methods refer to parameters as features (total time, means, number of zero crossings, etc.) which are computed from the measured signals.

The algorithms used in the preprocessing part are not detailed here. Depending on the type of digitizer and features used, preprocessing reduces spurious noise, detects gaps, amplifies, filters, conditions, digitizes, truncates, normalizes and/or encodes.

The algorithms used in the comparison and decision parts differ by group. Comparing functions comes with problems such as consistency, nonlinear time axis distortions and random variations. Solutions to these problems include regional correlation, dynamic time warping and tree matching.



Parameter comparison is very straightforward. Vectors of parameters in a feature space often describe the signatures and their closeness is evaluated with the use of specific metric, such as Euclidean distance. The decision part needs a threshold on this metrical distance value to control the FAR and FRR, and is thus a very important factor in the performance of the system.



Simple Dynamic Handwritten Signature Verification Technique

Gopal W. Gupta and Rick C. Joyce describe a simple dynamic signature verification technique based on parameter features that are easy to determine and/or compute: ²

- Total time
- Number of sign changes in the x and y velocities and x and y accelerations
- Number of zero values in the x and y accelerations
- Pen-up time
- Total path length

Data Acquisition

The digitizer used in their experiments is a graphics tablet to capture a signature as samples of (x, y) coordinate pairs 100 to 200 times a second. With such equipment, it is straightforward to compute velocities and accelerations from the data.

Preprocessing

Although no preprocessing has been used in their experiments, they recognize that smoothing the data, i.e. averaging out the measurement errors, would be helpful to obtain better approximations to the velocities and accelerations. In our own experiments, using a commercial 5- by 5-inch graphics tablet with a sampling rate of 120 Hz, the ACECAT-II, we found that removing dropouts and peaks should be sufficient.

Feature Extraction

As mentioned above, seven features are computed from the (pre-processed) data. Computing x and y velocities, accelerations, and number of zeros in the accelerations are trivial. As the digitizers use samples at a frequent rate, we can use the digitizer time units of 1/100 to 1/200 seconds, i.e., the number samples, for the overall time and pen-up time. The values of the features are placed in a vector T.

Enrollment

For the comparison process we need a reference. Gupta and Joyce simply used the mean and standard deviations of the values of the features of 5 to 10 sample signatures to obtain two vectors R and S.

Comparison

To compare the signature, we simply compute the distance vector $D = R - T$, and normalize D by dividing each value by the corresponding standard deviation in the vector S to obtain a vector Z whose norm is then computed. In practice, it is possible to have standard deviations of zero. In such case, a value or 10 percent of the mean value is used for the standard deviation.

Decision-Making

The computed norm is now compared to a pre-defined threshold. The signature is authenticated only if the norm is smaller than the threshold. The value of the threshold depends on the application. Table 1 presents the results of their initial experiment. Their aim was to approach an FRR of 0 percent or very close to it. No skilled forgeries are used, i.e., FAR represents the acceptance of random signatures, also called *zero-effort* FAR.

Also, only seven features are used: total time, number of sign changes in the x and y velocities and x and y accelerations, per-up time, and the total path length.

Table 1. Evaluating Different Number of Sample Signatures from Gupta and Joyce

	Number of Sample Signatures							
	3		5		7		10	
Threshold	FRR	FAR	FRR	FAR	FRR	FAR	FRR	FAR
4	64.2%	1.5%	35.9%	1.8%	22.0%	1.8%	14.3%	2.1%
6	37.8%	2.5%	11.5%	5.8%	6.9%	8.6%	3.2%	8.6%
8	23.7%	6.1%	5.6%	9.8%	3.0%	13.2%	1.0%	13.2%
10	14.0%	8.9%	3.0%	12.9%	0.6%	16.0%	0.0%	18.8%
12	9.6%	13.2%	2.0%	18.5%	0.2%	20.6%	0.0%	23.7%
14	7.4%	16.3%	1.3%	22.8%	0.2%	27.1%	0.0%	27.4%
16	5.2%	19.4%	0.8%	25.9%	0.2%	30.8%	0.0%	32.0%

Having more features does not necessarily improve performance. In fact, when trying to minimize the total error, the best results were obtained using a set of four features:



- Total signature time
- Number of acceleration sign changes in the x-direction
- Total pen-up time
- Number of zero values in the x-acceleration

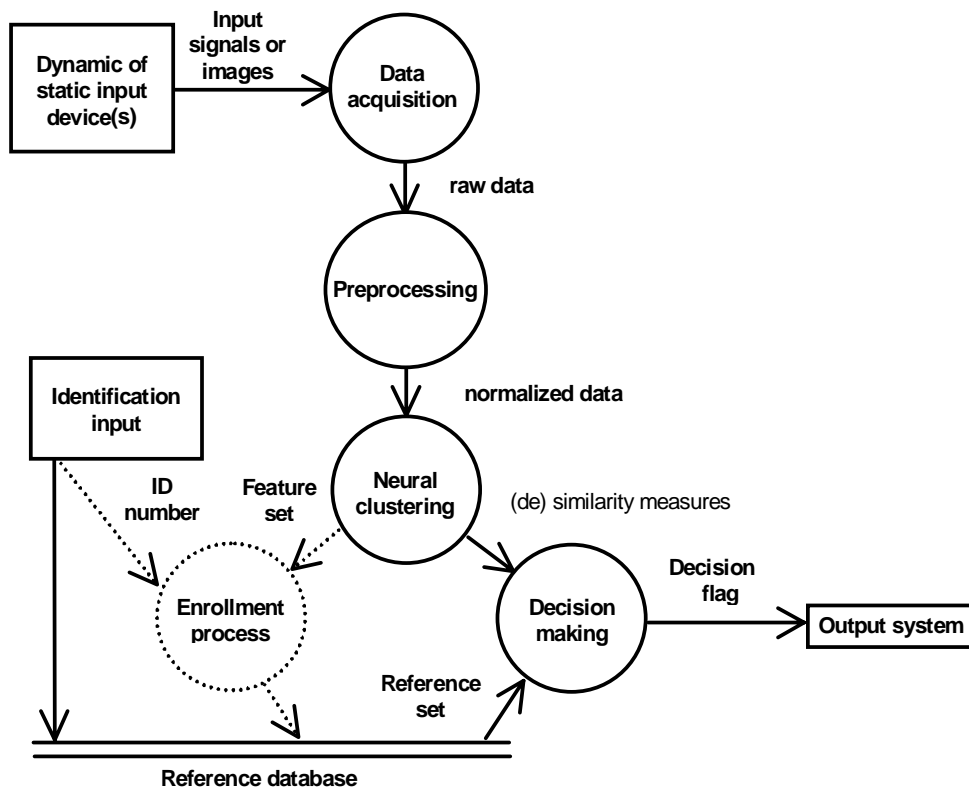
These four features gave a result of 2.5 percent FRR with a FAR of 8.6 percent (when the threshold value was 5). Minimizing FAR or FRR to 0 percent resulted in a 78.9 percent FRR and a 17.8 percent FAR respectively, using only three features. G. Gupta and R. Joyce are confident that they can reduce the zero-effort FAR considerably by including a small number of shape-related features in the set of features. Such features are currently under study.

Introducing Neural Clustering

It is not unusual in recognition and identification problems to see that a reasonable result can be obtained by almost any technique already available. A typical example is the real-time recognition of vehicle license plates, where literature shows that at least 92 percent recognition can be achieved.

The real problems start trying to improve beyond this level. It is not unthinkable that the claim of Gupta et al. on further improvement to taking static parameters also into account is mere wishful thinking. From the experience gained over a wide range of recognition and identification problems, from on-line diagnosis of turbogenerators to the real-time recognition of license-plates, we have therefore assumed that nonlinear clustering based on neural techniques will again introduce further improvements.^{5 6}

Figure 2. Data Flow Diagram of a Neural Signature Verification System





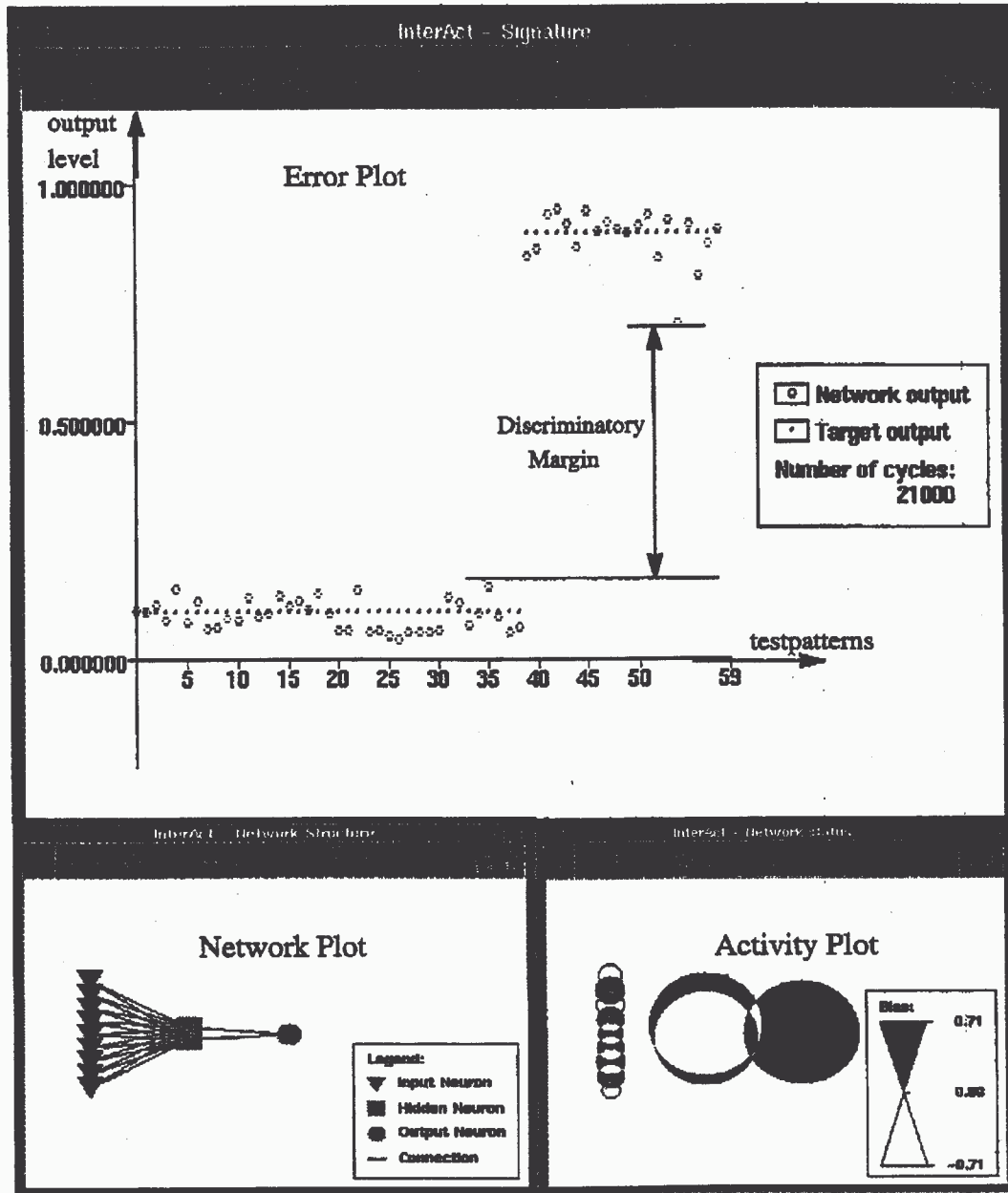
As shown in Figure 2, this leaves the architectural scheme of the signature verification system largely unaltered. In developing the neural network, we are first confronted with the choice of topology and learn rule. Though the use of temporal effects would be preferable for the current experiments, we have refrained from such computationally intensive schemes and preferred in the first instance a straight feed-forward network. Because of the inherent non-linearity of neural clustering, this already provides a step beyond the previous work of Gupta and Joyce.

After data acquisition and preprocessing of the time-varying input signal, the neural network measures the likeness to the various trained patterns. A final decision-making step is still required to qualify the most likely result from all others. Depending on the application, these results are thresholded to minimize the FRR and/or FAR. In other words, the nature of the application differs only in this final stage.

For illustration purposes, we discuss here first a small experiment with three people using the InterAct environment. Figure 3 shows some typical observations.

- ❑ The network plot showing the topology used for the identification,
- ❑ The activity plot showing by node size the degree by which the neurons participate in the identification,
- ❑ The error plot showing the acceptance error over the applied test patterns.

Figure 3. InterAct Observations on a Small Experiment



From a random selection of 25 students, we assembled a set of signatures at different moments in time to experiment with the network dimensions. On a conventional feed forward network, the neural recognition is on a per person basis, where the number n of input neurons ranges between 4 and 9 (as required for adequate enrollment), while usually 2 - 3 hidden and 1 output neuron suffices.



The networks are trained off-line in the InterAct environment and finally produced in a small, table-oriented software model. During this generation process, the non-linear transfer characteristics of the individual neuron are sharpened to minimize the computational effort during execution on the target hardware while maintaining the trained functionality.

Software Architecture

The signature will be written on the ACECAT-II graphics tablet and continuously provide the pen status and (x,y) location with a sampling rate of 120 Hz. This information is scanned for outliers and dropouts and stored in a linear list. From inspection of this list, the nine data items for further processing follow:

- 1) *Total time (T)*, i.e., the number of samples in the list (from first pen-down to last pen-up) with the sampling rate as common time-base.
- 2) *Number of zero crossings in x-velocity (XV)*, i.e., the number of sign changes in the differences in the pair over the x coordinates.
- 3) *Number of zero crossings in y-velocity (YV)*, i.e., the number of sign changes in the differences of the pair over the y coordinates.
- 4) *Number of zero crossings in x-acceleration (XA)*, i.e., the number of sign changes in the differences of the pair over the x velocities.
- 5) *Number of zero crossings in y-acceleration (YA)*, i.e., the number of sign changes in the differences of the pair over the y velocities.
- 6) *Number of zero values in x-acceleration (XAZ)*, i.e., the number of samples with a zero x-acceleration value.
- 7) *Number of zero values in y-acceleration (YAZ)*, i.e., the number of samples with a zero y acceleration value.
- 8) *The overall pen-up time (PU)* i.e., the number of samples with the pen up.
- 9) *The overall path length (PL)*, i.e., the sum of the Euclidean distances between the samples

As accommodated by the next procedure.

```
T = N; XV = YV = XA = YA = XAZ = YAZ = PU = PL = 0
last_xv = last_xa = last_yv = last_ya = 0;
for (i = 1; i < N; i++)
{ xv, = sample [i].x - sample [i-1].x;
  if(xv ^ last_xv < 0) XV += 1;
  xa = xv - last_xv;
  if (xa ^ last_xa < 0) XA += 1;
```



```

if (xa==0) XAZ += 1;
last_xv = xv;
last_xa = xa;
yv, = sample [i].y - sample [i-1].y;
if(yv ^ last_yv < 0) YV + = 1;
ya = yv - last_yv;
if (ya ^ last_ya<0) YA +=;:
if (ya==0) YAZ += 1;
last_yv = yv;
last~ya = ya;
If (sample[i].pen_up) PU += 1;
PL += sqrt(xv * xv + yv * yv)}

```

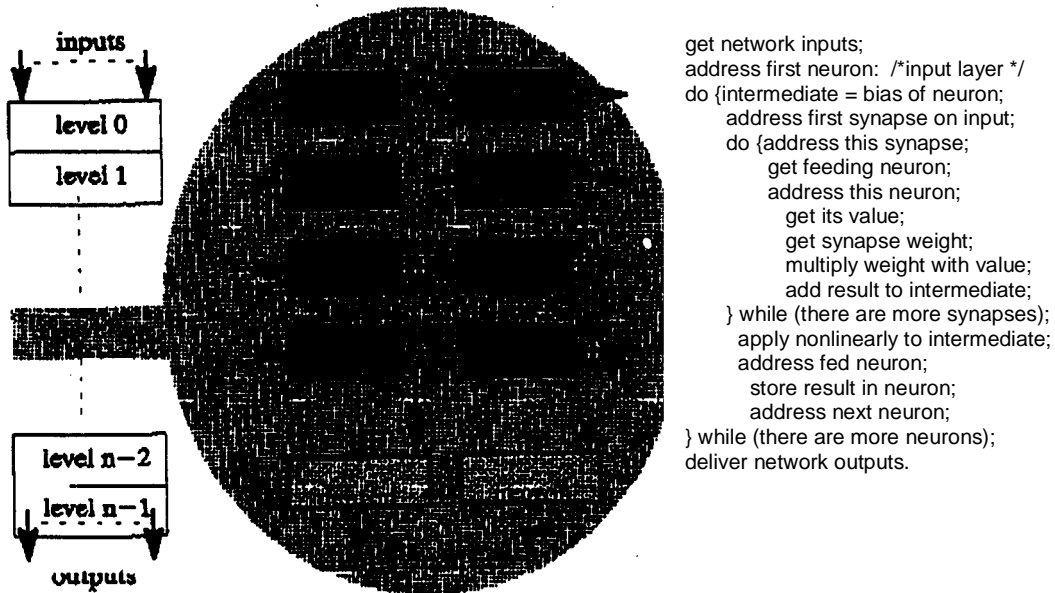
This procedure needs between 300 and 350 cycles per sample, which on a TMS320C10-14 provides a 100 μ s response time. With the sampling rate of 200 Hz, there is sufficient room for the further processing to be on-line. Further, the data storage requirements are 16 words.

The feed-forward artificial neural network is an assembly of neurons that are connected through synapses in such a way that only a single direction of the data flow is supported, i.e., all signals flow from the inputs to the outputs. A second restriction is based on the structuring of the network in so-called levels: all signals pass all levels in consecutive order, or, in other words, all synapses that emanate from level i connect to level $i+1$. As a consequence, the feed-forward neural network is extremely easy to emulate.

By moving from the input level through the intermediate (hidden) levels to the output levels, all calculations are guaranteed to be based on fresh synapse values. Note, that so far there is no need to limit ourselves to synapse connections between neighboring levels, but we have introduced this restriction solely as a precaution against eventual problems when introducing on-chip learning at a later stage of development.

Figure 4 illustrates feed-forward networks. The network is first shown to be a succession of levels; secondly, a single level is detailed to be a computational matrix.

Figure 4. A leveled network

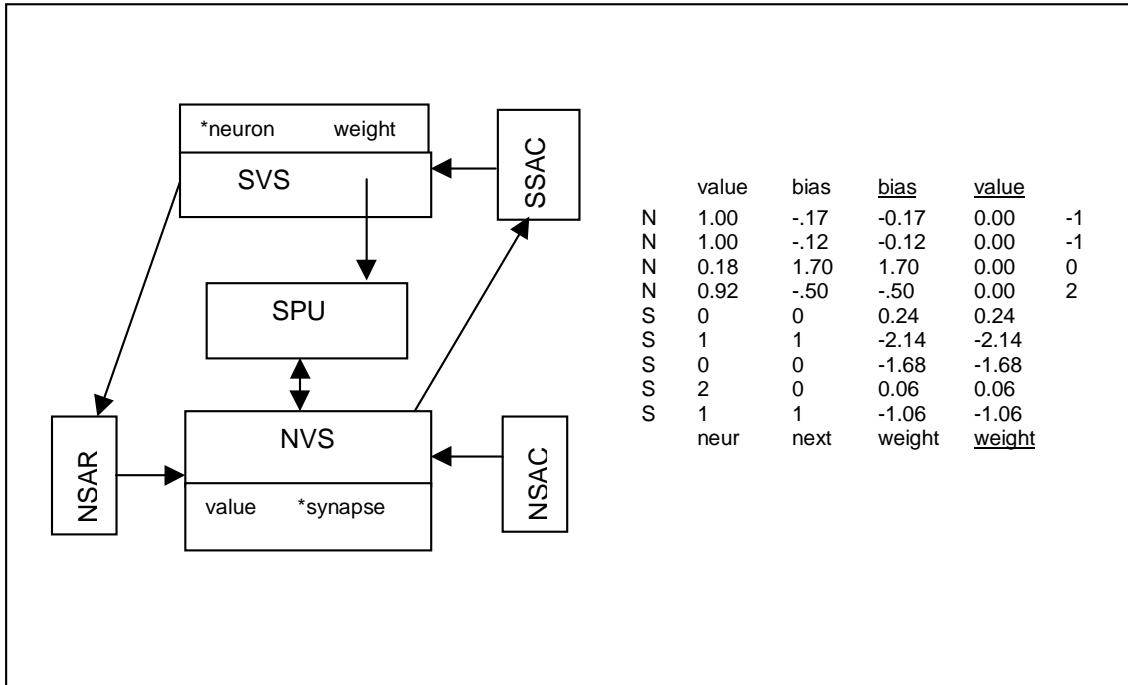


Now look at the computational meaning of synapse and neuron. Each synapse takes a value from the output of a designated neuron in the previous level and multiplies this value with the synapse weight. The result is passed onto the input of the neuron in the present level. There it will be summed with the multiplication results from all other feeding synapses. Hence, the emulation for a single neuron requires addressing all feeding synapses in succession, whereby each synapse in turn takes values from other, specifically addressed neurons. It is therefore of importance for the emulation to have the synapses in a calculation order.

We can now turn this informal description into a hardware architecture. As discussed above, we need to handle all levels in consecutive order, all neurons in a level in any order and, finally all synapses connected to the input of a single neuron according to the above procedure. This leads naturally to the following arrangement.



Figure 5. Neural Software Architecture and Table Composition



The operation on the tables is supported by two generally applicable emulation routines: one for the synapse multiplication, and one for the neural summing and discrimination. The latter procedure reads simply as follows:

```

NVS[0].value = net_inputs [0]; NVS[1].value =
net_inputs[1];

/* scan all entries in the NVS table ..... */
for (counter = 0; counter < NPS->nr_neurons;
counter+ +)
{ result = NVS[counter].bias;
/* loop through the SVS table ..... */
for (ssac = NVS[counter].offset_s; ssac > = 0 ;
ssac+ +)
{result + = SVS[ssac].weight *
NVS[SVS[ssac].offset_n].value;
if (SVS[ssac].lasts == 1) break; }
/* collect for new neuron status */
if (NVS[counter].offset_s >=0)
NVS[counter].value=PC_nonlin2(result);
result=0; }

```



This procedure needs 25 cycles per synapse, which leads for our network to 750 to 800 cycles for a complete recall. On a TMS320C10-14 this leads to a 225 μ s execution time. Further, the storage requirements on the data memory are 130 words. The preprocessed data can not be used directly as input to the neural network. The obligatory normalization of the preprocessed data is elementary in the current application, as the features derived from the signature lie usually in very narrow bands.

From a realistic signature, 300 and 600 samples are taken. Thus, even without a direct handling of the sampled data, the time elapse for a complete authentication will be worst case 0.1 seconds. Further, data memory requirements of the software parts are sufficiently less than the available 256 words to accommodate for the needs of other service routines.

So far, the experience with the above-described dynamic signature verification system is largely limited to experiments on the constituting parts. However, from the wealth of experience gained by the work of Plamondon and Lorette and the impressive results gained by the similar approach of Nujhuis et. al., we have confidence that the current recognition rate of better than 99 percent and the FAR of less than 0.01 percent can still be improved.¹⁶



Discussion

The technique described requires very basic operations as subtracting, comparing, incrementing counters, multiplying, and dividing. With a sample rate of 200 Hz, even the slowest member of the first generation of TMS320 DSP devices is capable of doing the needed operations while waiting for the next sample (e.g., with a 200 Hz sampling rate, the TMS320C10-14 has about 17850 instructions to process one sample). However, the facility of fixed-point arithmetic had a major impact in the selection of this hardware platform.

So far, we have limited the discussion to the technical ability of identifying a person by his/her signature. It stands to reason that such a technique will have problems when the person has to be identified out of a prospective set of millions of signatures. Such a search space would need a number of features that are simply not of statistical significance. The verification problem is however a different nature. Here, we have to ensure that the signature is sufficiently alike to the one presented in the past. This past is represented by the learned status of the neural network and, as shown above, a table gives this neural network status by a string of numbers. This allows for a condensed (possibly encrypted) and downloadable representation.

Thus, we have constructed an authentication kernel comprising of some personalized tables and some general-purpose procedures, that can be part of a variety of entrance monitoring and security systems. We will briefly discuss two typical applications. In computer systems, the encrypted password file monitors the login procedure. For pen-based computers, we suggest the alternative of entering the user's signature. The clear advantage is that the need to remember the password is removed, while on the other hand the shared use of passwords becomes no longer possible.

A similar remark can be made with regards to the use of credit and smart-cards in commercial transactions. Here, a signature can be used instead of a secret number to relieve the individual of the need to remember this number for every card in his possession. THIS is of special interest for those telephone payments where the use of a signature on the bill is already enforced.

Summary

Neural networks are especially suited to understand the dynamics of movements. Writing down a signature is such a movement. It is discussed that dynamic signature verification is a technical reality and that such provides a real personal code in safety and security-related operations. Typical examples are login control for pen-based computer systems and smart-card commercial transactions. Details of the authentication kernel are given for the TMS320C10-14.

Index of Terms

Authentication	9
Clustering	15
Digitizer	10
Features	11
FAR	11
FRR	11
Identification	15
InterAct	16
Neural Network	20
Preprocessing	13
TMS320C10-14	20
Signature	9
Smart-Card	24
Verification	7
Dynamic	11
Static	15

References

- ¹ G. Plamondon and G. Lorette, "Automatic signature verification and writer identification: The state of the art", *Pattern Recognition*, Vol. 22, No.2, pp. 107-131, 1989.
- ² G. K. Gupta and R. C. Joyce, *A Simple Approach to Dynamic Hand-Written Signature Verification*, preliminary paper, 1995.
- ³ *TMS320C1x User's Guide*, Texas Instruments, 1991.
- ⁴ R. Herbert, *Whipping Star*, New English Library, January 1972.
- ⁵ E.I. Barakova, L. Spaanenburg, and J. Zaprjanov "Neural Fault Diagnosis of a Turbogenerator by Vibroacoustic Data", *Int. Conf. on Signal Processing, Applications & Technology ICSPAT'95* (Boston, MA, USA, 24-26 October 1995) pp.1454-1458.
- ⁶ A.G. Nijhuis, M.H. terBrugge, K.A. Helmholt, J.P.W. Pluim, L. Spaanenburg & R.S. Venema, and M.A. Westenburg. *Car License Plate Recognition with Neural Networks and Fuzzy Logic*, presented at ICNN'95 (Perth, Western Australia) November 1995.