Analysis by Synthesis in Handwriting Recognition

by

Boris L. Elbert

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Science in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 1997

Author .....
Department of Electrical Engineering and Computer Science
September 25, 1996

Certified by .....
Robert C. Berwick
Professor
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Departmental Committee on Graduate Students

Analysis by Synthesis in Handwriting Recognition

by

Boris L. Elbert

Submitted to the Department of Electrical Engineering and Computer Science
on September 25, 1996, in partial fulfillment of the
requirements for the degree of
Master of Science in Electrical Engineering and Computer Science

Abstract

An approach to on–line cursive handwriting recognition, based on discrete motor control commands and analysis by synthesis to solve an inverse–dynamic problem, is proposed and evaluated. In this model a continues input handwriting data is transformed into a discrete sequence of control commands. The patterns are generated internally in the analyzer according to predefined sequences of commands until the best match with the input data is obtained. To find this best match the time-alignment algorithm, based on dynamic programming, has been designed.

Because of the nature of the proposed model, where all possible shapes and stiles of cursive handwriting are described as a sequence of discrete control commands, practically any need for an extensive training and parameter estimation has been eliminated.

The actual cursive handwriting recognition system (**WordCracker**) is presented. Constraint dictionaries containing either single letters or three–letter words were constructed and used to evaluate **WordCracker**. In both cases low error rates have been achieved.

Our experiments demonstrated the potential for the proposed model to be developed into a writer independent, full lexicon cursive handwriting recognition system.

Thesis Supervisor: Robert C. Berwick
Title: Professor

Acknowledgments

I would like to thank Robert Berwick, my adviser, for his help and support. His experienced understanding of what is possible and worthwhile has guided me through this project.

I would also like to thank Partha Niyogi whose profound knowledge of the problem and willingness to help saved me a lot of time and prevented from chosing a wrong direction in this research.

Thanks to my brother Igor Elbert for his comments and corrections in C programming.

A big thank-you to my wife Elena and my son Dennis for their patience and forgiveness.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

For the past three decades, there has been an increasing interest among researchers in problems related to machine simulation of the human reading process. Intensive research has been carried out in this area with a large number of technical papers and reports devoted to character and word recognition [33, 20, 19]. This subject has attracted an immense research interest not only because of the very challenging nature of the problem, but also because it provides means for automatic processing of large volumes of data such as postal codes and addresses [5], office automation [30], signature recognition and verification [28, 25, 24], and other business and scientific applications.

The recent emergence of pen computers with high resolution tablets has made available dynamic (temporal) information as well as created the need for robust on-line handwriting recognition algorithms. Considerable effort has been spent in the past years on on-line cursive handwriting recognition [35, 27, 26, 36], but there are no robust, low error rate recognition schemes available yet.

1.1 Problem Description

One of the major difficulties of the cursive word recognition descends from the great variability observed in different samples of script issued from the same writer over time or from different scriptors. So it is difficult to find a reliable description of a

word able to represent all the admitted occurrences of the input shape.

When focusing on the techniques of machine simulation of the human reading process many diverse fields are addressed. Techniques from cognition, psychophysics, statistics, and computer science can all be applied to help constrain problem domain.

Cursive handwriting is a complex graphic realization of natural human communication. Its production and recognition involve a large number of highly cognitive functions including vision, motor control, and natural language understanding.

Research of the motor aspects of handwriting has suggested that the pen movements produced during cursive handwriting are the result of "motor programs" controlling the writing apparatus. This view was used for natural synthesis of cursive handwriting [4]. Some of these works are based on a similar to ours approach [35, 29]. None of the previous works, however, have solved the inverse–dynamic problem of finding the "motor code" used for the production of cursive handwriting.

In this paper we propose a new on–line handwriting recognition model. This model is based on the *motor–control theory* [12, 3, 40, 14, 15] and uses *analysis by synthesis approach* [13] to solve the inverse–dynamic problem. This combination of the motor–control theory and analysis by synthesis produce a natural way of robust cursive handwriting recognition.

In this model a continues input handwriting data is transformed into a discrete sequence of *control commands*. The patterns are generated internally in the *articulator*. The articulator takes as an input the predefined sequence of control commands describing a particular entry in the *dictionary*. The length of the entry does not matter. It might be a single letter, word, or even a sentence. The probability of how well this particular entry matches the input data is calculated. This process is repeated until the best match is found.

In this paper we also present an implementation of a new on–line handwriting recognition system (**WordCracker**). This system has been created using the described above model. We show that all cursive letters of English alphabet (which we consider to be basic units) can be represented as a sequence of limited number (20 in our implementation) control commands. The larger units of handwriting (words) are

the concatenation of the basic ones.

The underlying idea that all possible shapes and stiles of cursive handwriting can be reduced to the same sequence of discrete control commands eliminates practically any need for extensive training and parameter estimation (the problem of all current recognition schemes). We show a very good **WordCracker** performance (on the constrained dictionaries) with only a few parameters of the script needed to be estimated. The possibility of complete eliminating parameter estimation in the future is discussed.

All of these proves that the proposed model has good chances to become a new and very promising approach in on–line cursive handwriting recognition, and that **WordCracker** is an easy to use, friendly, and robust cursive handwriting recognition system.

1.2 Goals

The need for a new robust on–line cursive handwriting recognition system motivated the design of the model and implementation of the system proposed in this paper.

While the scope of this thesis is not to create a person independent, full lexicon on–line system for recognizing cursive handwriting, a desired attribute of the proposed system is extensibility towards this goal.

One of the major goals that we tried to achieve (and I think we have succeeded) was to design and implement a really "natural" recognition scheme. The scheme that uses the same underlying mechanisms that we humans use for handwriting production and recognition. Not only because of the really interesting and challenging nature of this problem, but also because, we think, that this is the only right way of addressing such kind of problems. Not to reinvent something, but understand how it works in "real world" and try to simulate it in "computer world", using all the power we have nowadays. This is the only way to the future.

Another goal is to create a robust system which does not require an extensive training, which is easy to use and provides a friendly graphical user interface (GUI).

This allows easier experimentation and demonstrates the possibility of creating a commercial product in the future.

1.3 Outline

Chapter 2 discusses previous work in related areas. Chapter 3 describes the model architecture and technical issues of **WordCracker** implementation. The experiments performed to evaluate the system and results are discussed in Chapter 4. Conclusions and discussion of future work are included in Chapter 5.

Chapter 2

Previous Work

2.1 General Overview

The systems proposed up to now to solve the problem of handwriting recognition can be generally divided into off–line and on–line.

In the field of off–line recognition of handwritten words several works have been devoted to word description techniques using the structural approach. Simon and Baret [34] and Hull et al. [16] divided the regular part from the singular part of the trace before performing coding; this idea is spreadly used by many groups in the world (see also [32]). Moreover, Simon and Baret' work used a dictionary containing a set of codes of the words used in the field of bank cheques. In Simon's approach the matching procedure was carried out starting from the analysis of anchors in the chain that defined robust features and then using dynamic matching for the other parts of the code. On the other hand, Hull et al. proposed a new word recognition technique without explicit segmentation of the word. Specifically this approach, developed for the whole word recognition in postal addresses, extracted a chain code from the contour of the whole word and then used this code to derive singular features. The approach appears to be stable with respect to variability in writing and it is also supported by the biological behavior of human beings. Camillerapp et al. [1] proposed a system for off–line handwriting recognition based on a structural approach. Each word was represented by its graph model deduced directly from the grey–level image

by detecting specific primitives along the baseline og the word. A new method based on a syntactic description of the words for automatic recognition of off–line Arabic cursive handwritten words was also proposed by Zahour et al. [41].

In the field of on–line recognition of run–on handwriting the results have mainly been obtained using a unified tablet–display such as a paper–like computer interface. Fujisaki et al. [11] developed a system that classified strokes, generates character hypotheses, by means of a hypothesis generator, and verified them by means of a hypothesis tester to estimate the most suitable character sequence for each word. They also used two different types of linguistic constraints: the first constraint was based on the character type transition probability, the second one evaluated sequences by character tri–grams. Schomaker and Teulings [31] considered the stroke–based systems for cursive script recognition versus the character–based systems. E. Doojies [4] used the idea of "motor programs" controlling the writing apparatus for natural synthesis of cursive handwriting. Y. Singer and N. Tishby [35] and D. Rumelhart [29] made an attempt to construct a dynamical model of handwriting for recognition and solve the inverse-dynamic problem of revealing the motor code used for the production of cursive handwriting.

2.2 Motor-Control Theory

The idea of "motor programs" that control the handwriting is quite old. S. Grillner [12] introduced the notion of oscillators and tied handwriting to locomotion. A few works have been devoted to devising a measurement apparatus [3, 40, 14, 2, 22]. An initial approach to modeling a handwriting trajectory has been done by Mermelstain and Eden [23], who segmented writing for fitting with quarter sine waves. J. Danier van der Gon and J. Thuring [3] assumed assumed a rectangular form to the accelerations. J. McDonald [22] fit trapezoids to the accelerations. M. Yasuhara [40] assumed an exponential rise and decay time to an acceleration plateau. The end result of this process is a list of acceleration burst durations and amplitudes which when applied to the corresponding model yields synthetic writing close to the measured human

handwriting.

J. Hollerbach [15] introduced an oscillatory model of handwriting. In this theory there is a preexisting and underlying repeated pattern of letter shapes. This pattern propagates indefinitely unless it is modulated. Rather than an active process of forming letter shapes, there already exist letter shapes typical of the oscillation pattern and the modulations serve to remold the preexisting letter shapes into the desired letters. A modulation will change the underlying oscillation pattern to a new one, which will propagate indefinitely unless it is also modulated. In this model the motor programs controlling the process of handwriting are considered as the sequence of modulations. The oscillatory process acts as an interpretive program that "interprets" the motor program, which are the sequence of modulations, in the context of the current oscillation.

Y. Singer and N. Tishby [35] extended the Hollerbach's oscillatory motion theory and developed a parameter estimation and regularization scheme which was used for the analysis, synthesis, and coding of cursive handwriting.

2.3 Formalization of Cursive Handwriting

The first attempt to formalize the description of cursive handwriting was made by M. Eden and M. Halle [8, 7, 6]. In their model they defined a set of four primitive symbols ("bar", "hook", "arch", and "loop"), where each primitive symbol was a point pair, partially ordered. That is, two points were ordered one above the other or to the right or both, and a sense vector was specified for the tangent to the continuous line to be drawn between the two points. Two conventions were introduced. These specified that the rotation of each tangent vector from one end of the stroke to the other was 180^0 and was monotone. They generated a set of 11 symbols by rotating and reflecting these primitives. Finally they obtained a set of 33 strokes by allowing the symbols to be located in on of three partially overlapping horizontal fields.

Each letter in the language was defined as a unique, finite sequence of strokes. Additional rather complicated rules were used for collating strokes (different rules

applied to the strokes within a single letter and between letters).

A harmonic oscillator theory was used to describe the actual production of handwriting. The cursive script generated using these techniques was close to real human cursive handwriting. No attempts to use this scheme for handwriting recognition were made.

2.4 Analysis by Synthesis Approach

The notion of "analysis by synthesis" has been introduced by M. Halle and K. Stevens [13] in 1962. In this paper they proposed an outline of a speech recognition model in which mapping from signal to message space was accomplished through an active or feedback process. Patterns were generated internally in the analyzer according to a adaptable sequence of instructions until a best match with the input signal was obtained. Since the analysis was achieved through active internal synthesis of comparison signals, the procedure was called "analysis by synthesis".

Unfortunately, for the last 30 years, since this idea has been introduced, only a few works have been devoted to the actual attempts to use analysis by synthesis for handwriting (or speech) recognition.

There were a few attempts to construct dynamical models for speech recognition based on the predictive neural networks [18, 38, 21, 37]. Ken-ichi Iso [17] used an approach similar to ours. He proposed a speech recognition method based on the dynamical model of speech production. It was the first work where linguistic and articulatory information were actually separated. His model consisted of an articulator and its control command sequences. The latter had the linguistic information of speech and the former had the articulatory information which determined transformation from linguistic intentions to speech signals.

In the field of handwriting recognition Rumelhart [29] proposed a dynamical model, but he did not actually solved the inverse–dynamic problem of "revealing" the "motor code" used for production of cursive handwriting.

Y. Singer and N. Tishby [35] extended Hollerbach's osillatory model of handwriting

[15] and using analysis by synthesis approach to solve the inverse problem developed a new parameter estimation and regularization scheme.

None of the previous works, however, have actually tried to combine in one model the ideas of *motor control theory*, *formalization of cursive handwriting*, and *analysis by synthesis approach.*

The idea of creating a new cursive handwriting recognition scheme, based on all of these principles, which will be close to the way humans do writing production and recognition motivated the research described in this paper.

Chapter 3

Model Description and WordCracker Implementation

3.1 Model Architecture

The proposed model consists of a handwriting articulator and control commands for the articulator. Each single letter (which is considered as a basic unit) has a control command sequence. The control command sequence for larger segments of handwriting (word or sentence) is obtained as a concatenation of the ones for the basic units.

Figure 3-1 on page 17 shows the model architecture. Input handwriting data is represented by a feature vector sequence (length T),

$$a_1, ..., a_t, ..., a_T. \tag{3.1}$$

Each feature vector has P components (P-dimensional vector). We used X and Y-coordinates of the pixel to describe the feature vector (that is, in our case $P = 2$). Control command sequence for each basic unit of handwriting (letter) is represented by a control command vector sequence (length N),

$$c_1, ..., c_n, ..., c_N. \tag{3.2}$$

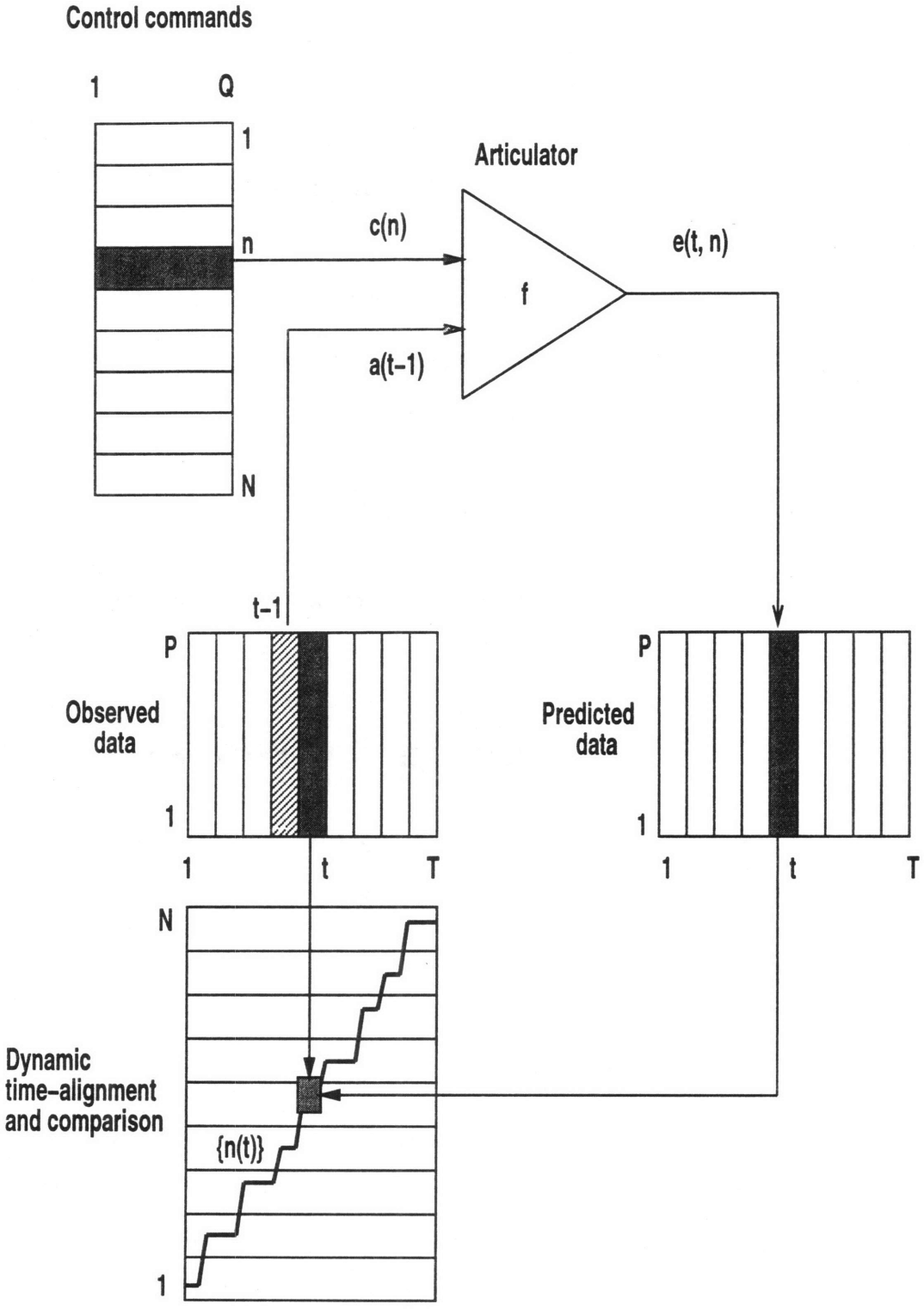


Figure 3-1: Model architecture

Each control command is a Q-dimensional vector. The control command sequence for a word is a concatenation of the ones for single letters (sometimes, with addition of a few extra commands to account for the transition from one letter to another).

The articulator, which is a nonlinear predictor with control input c_n, provides a mapping between the input feature vector a_{t-1} and predicted feature vector e_t, given the control command vector c_n,. The articulator is represented by a nonlinear vector function f,

$$e_{t,n} = f(a_{t-1}, c_n). \tag{3.3}$$

The distance between input feature vector a_t and predicted feature vector $e_{t,n}$ is defined by prediction error $d_{t,n}$,

$$d_{t,n} = \|a_t - e_{t,n}\|^2. \tag{3.4}$$

This can be generalized to Gaussian probability of producing the feature vector a_t, given the control command c_n, $P(a_t|c_n)$,

$$-lnP(a_t|c_n) = \frac{1}{2}\sum_{p=1}^{P}\{\frac{(a_{t,p} - e_{t,n,p})^2}{\sigma_{n,p}^2} + ln(2\pi)\sigma_{n,p}^2\}, \tag{3.5}$$

where $a_{t,p}$ is a p-th component of a_t and $e_{t,n,p}$ is a p-th component of $e_{t,n}$.

The probability of the handwriting data observed, given the control command sequence, can be computed as

$$P(a_1, ..., a_T|c_1, ..., c_N) = \max_{\{n(t)\}} \prod_{t=1}^{T} P(a_t|c_{n(t)}), \tag{3.6}$$

where function $n(t)$ determines time-alignment between input handwriting feature vector sequence and control command sequence. The optimal time-alignment $\{n^*(t)\}$ which gives maximum probability is determined by dynamic programming. Using this probability as a score, we can perform handwriting recognition.

3.2 Implementation Issues

The model described above gives a general outline of the system (WordCracker) presented in this paper. Although, WordCracker has been implemented strictly according to the model, a few implementation specific issues need additional explanation.

3.2.1 Articulator and Control Commands

The articulator is a nonlinear predictor, which takes as an input the previous (in the time domain) handwriting feature vector (in our implementation it is two-dimensional vector) and one of the allowed control commands. Given these, the articulator computes the *current predicted* feature vector. After that the distance between *predicted* and *observed* feature vectors is calculated. The result is passed to the time-alignment module, where, based on the total observation sequence and the whole set of allowed commands, the best probable path is computed. This algorithm is repeated for every entry in the dictionary and the one which results in the best probability is the *predicted* meaning of the observed data.

The idea behind the control commands was to describe the pen movement over the time in terms of simple curves (such as lines and ellipses) so that, at any instance of time the next point on the curve can be computed given the previous one and, maybe, some extra parameters (we called it *state of the command* and will talk about it later). So all the information that should be passed to articulator is the previously observed feature vector (coordinates of the previous point), the curve identifier (command itself) and the state of the command, if this particular command needs it.

Control commands

In our system we use a set of 20 commands to describe all possible shapes of cursive handwriting. In general all the commands can be divided into three large groups.

1. The commands which are ellipse-based. They require additional information about the current state of the command.

2. The commands which are based on the straight lines. These commands do not require any additional data.

3. Other commands, which are based on the straight lines, but required additional information about state of the command.

Ellipse-based commands These commands were designed with the assumption that the trace of the pen can be modeled by the arc of the ellipse. To compute, at any instance of time, the next position of the pen (that is, its X- and Y-coordinates), given the previous position and the type of the ellipse (which is specified by its large and small axises and by the axis tilt.), we need to

1. Compute the position of the ellipse center.

2. Increment the phase which was used to calculate the previous pixel position by specified Δ,

$$New\_Phase = Old\_Phase + \Delta. \tag{3.7}$$

3. Compute the next pixel position, given the center and new phase.

All the ellipse-based commands follow the described above steps. The differences between the particular commands from this group are the differences in the ellipse parameters, which can vary greatly from one command to another, but they are all fixed for the each particular command. These differences are:

- Length of the axises. Both axises can be modified independently in different commands.

- Tilt. It can vary from $-\frac{\pi}{2}$ to $\frac{\pi}{2}$.

- Direction of the movement (clockwise or counterclockwise).

- Starting phase. It can vary from $-\pi$ to π.

The additional parameter "state of the command" is required for this group of commands. It is specific for every command and it is passed to the articulator every time

the particular command from this group is executed. It currently contains the only one field describing the phase, that was used in the previous step of executing the same command. This field is set only if the time-alignment module has decided that the system should repeat the same command. Otherwise this field contains NULL, which indicates that this is the first time the command is used and the current phase should be set to the starting phase, defined for this command.

The ellipse-based commands currently used in WordCracker are as follow:

{ counterclockwise, normal tilt, normal axis sizes, starts from $\frac{\pi}{2}$ }
{ counterclockwise, normal tilt, normal axis sizes, starts from 0 }
{ counterclockwise, normal tilt, normal axis sizes, starts from π }
{ counterclockwise, twice the normal tilt, X-axis twice smaller than normal, starts from $\frac{\pi}{2}$ }
{ counterclockwise, normal tilt, both X- and Y-axis are five times reduced, starts from $\frac{\pi}{2}$ }
{ counterclockwise, normal tilt, both X- and Y-axis are three times reduced, starts from $\frac{\pi}{2}$ }
{ clockwise, normal tilt, normal axis sizes, starts from $-\frac{\pi}{2}$ }
{ clockwise, normal tilt, normal axis sizes, starts from $\frac{\pi}{2}$ }
{ clockwise, normal tilt, normal axis sizes, starts from $-\pi$ }
{ clockwise, normal tilt, both X- and Y-axis are three times reduced, starts from $-\frac{\pi}{2}$ }

Commands based on the straight lines These commands were designed with the assumption that the trace of the pen can be modeled by the straight tilted line. To compute, at any instance of time, the next position of the pen, given the previous position, we need to now only the tilt and the direction of movement(up, down, left, or right) which are fixed for every command.

The "state of the command" parameter of articulator is not used for this group of command and always set to NULL.

The commands based on the straight lines currently used in WordCracker are as follow:

{ down, right to left, normal tilt }
{ down, right to left, two times normal tilt }
{ down, left to right, two times normal tilt }
{ up, left to right, normal tilt }
{ up, left to right, two times normal tilt }
{ up, left to right, three times normal tilt }
{ left, no tilt }
{ right, no tilt }

Other commands There are two commands in this group: *cross for t* and *cross for f*. Both commands are required additional information about the state of the command, that is if the command has been executed before or not. If not, than they calculate the single "jump" of the starting pixel to the left and up (the exact algorithm to do that is different for each command). If the command has been used before, than it repeats the { right, no tilt } command described above.

The parameters called "normal axis size", "normal tilt", and "Δ" are obtained during the parameter estimation procedure and described in the corresponding section.

Articulator

The function **articulator()** is defined as follows:

```
int articulator ( data PreviousObserv,
                  int Command,
                  data *NextPredictedObserv,
                  int NotUsed,
                  StateOfCommand *State ),
```

PreviousObserv. This is the previous observation. The type of this parameter is **data**, which is defined as

```
typedef struct
        {
          long x;
          long y;
        } data;
```

This structure contains the information about the pixel position on the screen.

Command. This is an integer identifier of the particular command. In other words this parameter specifies what type of a curve should be used to compute the next pixel position.

NextPredictedObserv. This is a pointer to the **data** datatype. The **articulator** will fill it with the data corresponding to the next predicted pixel position.

NotUsed. This parameter is currently not used.

State. This is the current state of the command. The type of this parameter is *StateOfCommand*, which is defined as

```
typedef struct
        {
          double Phase;
          double NotUsed;
        } StateOfCommand;
```

This structure contains additional information, which is used by the commands which require some additional data. If the command does not required additional data, this parameter is ignored.

3.2.2 Time-alignment Algorithm

The time-alignment procedure is implemented as a modified Viterbi algorithm for HMM [39, 9].

In general we need to solve a problem of finding the single best command sequence,

$$\mathbf{C} = \{c_1, ..., c_t, ..., c_T\}, \tag{3.8}$$

for the given observation sequence,

$$\mathbf{A} = \{a_1, ..., a_t, ..., a_T\}. \tag{3.9}$$

To do this we need to define the quantity

$$n_i^*(t) = \max_{c_1, c_2, ..., c_{t-1}} P[c_1, c_2, ..., c_{t-1}, c_t = i, a_1, a_2, ..., a_t \mid n(t)], \tag{3.10}$$

that is, $n_i^*(t)$ is the best score (highest probability) along a single path $n(t)$, at time t, which accounts for the first t observation and ends in the command i. By induction we have

$$n_j^*(t+1) = \max_i \{n_i^*(t) \cdot p_j[a_{t+1} \mid a_t]\}, \tag{3.11}$$

where $p_j[a_{t+1} \mid a_t]$ is a probability of observation a_{t+1} after executing command j, given observation a_t.

To actually retrieve the command sequence, we need to keep track of the argument that maximized, Eq. (3.11), for each t and j. We do do this via the array $\psi_j(t)$. The complete procedure for finding the best state sequence can now be stated as follows:

1. Initialization

$$n_i^*(1) = 1, \qquad 1 \le i \le N \tag{3.12}$$

$$\psi_i(1) = 0, \qquad 1 \le i \le N. \tag{3.13}$$

2. Recursion

$$n_j^*(t) = \max_{1 \le i \le N} \{n_i^*(t-1) \cdot p_j[a_t \mid a_{t-1}]\} \qquad 2 \le t \le T, \quad 1 \le j \le N \tag{3.14}$$

$$\psi_j^*(t) = \arg \max_{1 \le i \le N} \{n_i^*(t-1) \cdot p_j[a_t \mid a_{t-1}]\} \quad 2 \le t \le T, \quad 1 \le j \le N. \tag{3.15}$$

3. Termination

$$P^* = \max_{1 \le i \le N} [n_i^*(T)] \tag{3.16}$$

$$n_T^* = \arg \max_{1 \le i \le N} [n_i^*(T)]. \tag{3.17}$$

4. Path (command sequence) backtracking

$$n_t^* = \psi_{t+1}(n_{t+1}^*). \tag{3.18}$$

If we apply this algorithm to relatively large sequences of observation two problem arise:

1. Underflow, because the actual value of resulted probability is very low.

2. Too many multiplications, which are computationally very expensive.

We can solve both problems by using logarithms of the probability instead of the real values. In this case we can implement this procedure without any multiplications, replacing them with summations. The resulted calculation required for this implementation is on the order of N^2T additions.

3.2.3 Parameter Estimation

The major difference between the proposed model and all the other technics used for handwriting recognition is that our model does not require any training at all. It does require to estimate four parameters, such as "normal X-axis size", "normal Y-axis size", "normal tilt", and '"normal Δ". WordCracker does it by asking the user to write two times letter "o" and two times letter "n". WordCracker gets all the information about these four parameters from this data. It takes 30 seconds to do this as apposed to 20 - 60 minute training, which is usual for standard HMM implementation.

"Normal axis sizes" and "normal tilt" can be obtained from the data collected during writing letter "n". These parameters are used to estimate the average height, width, and tilt of the letter.

"Δ" parameter can be obtained by dividing 2π by the average number of pixiles in the letter "o". This parameter acounts for the differnces in the speed of writing among different users.

3.2.4 Observations and Remarks

A few additional words should be said about specific details of time-alignment algorithm implementation and related problems that occured during the construction of the control command sequences for particular letters.

These letters are **"a"**, **"d"**, and **"q"**. The problem was that all of them could be described by the same control command sequence. This means that a special effort should have been taken to distinguesh among them after the general algorithm desided that one of these letters was the most probable match to the handwritten data.

This problem was solved at the level of time-alignment algorithm. During the computation of the probability for the best sequence of commands over time, given the current observation feature vector, the algorithm keeps track of all the commands on this path. After the calculation is finished the path can be backtraced. It makes it possible to find out how long the system stayed in the particular command and as a result it allows to distinguish among **"a"**, **"d"**, and **"q"**. In fact, the only difference between **"d"** and other two letters is that the system stays longer in the "go up" command. Similarly for **"q"** the system stays considerably longer in "go down" command.

Chapter 4

Experimental Evaluation and Results

The recognition software system, **WordCracker**, based on the model discussed in the previous section, was written in C and evaluated under the SunOS and IRIX operating systems.

To evaluate **WordCracker** two dictionaries have been constructed. One contained 26 single letters of English alphabet and the other contained 100 three–letter words. The words in the second dictionary were carefully chosen to represent most of the possible letter combinations occuring in common English words. To make the recognition more difficult, special attention has has been paid to include the words which differ only in one letter (words like "fan", "can", and "man"). The testing has been done using both a Wacom tablet and a pen, and a mouse as a drawing device. The use of a mouth instead of a pen introduced additional Gaussian noise and made the whole recognition process more difficult and the results more reliable.

Dictionary	Percentage of correctly recognized words
Single letters	97.6%
Three letter words	92.4%

Table 4.1: Percentage of words correctly recognized

The results of **WordCracker** performance in handwriting recognition using these two dictionaries are given in Table 4.1. The results show that the system performs significantly better on the single letter dictionary than on the dictionary consisting of the three–letter words. This difference in recognition error can be explained by two factors:

- The size of the dictionary. The single word dictionary included only 26 entries, compared to 100 entries in the three letter word dictionary.
- The average length of the entry. Because a command sequence for a word was constructed as a concatenation of these for the single letters and the whole word was treated as a basic entry, the recognition error was increasing with the size of the entry.

In the single letter dictionary only a few letters were responsible for the overall recognition error. These letters are:

- **"a","d"**, and**"q"**. These three letters use the same control command sequence and a special procedure was used to distinguish among them (see section **Observation and Remarks** for details). Because of the nature of this additional procedure the recognition of these letters is very sensitive to the speed of the handwriting and rapidly decreases as speed goes up.
- **"o"** and **"c"**. These two letters are very close to each other (**"c"** can be considered as an unfinished **"o"**). As a result the system sometimes misrecognized **"o"** as **"c"**, but never the other way around.
- **"e"** and **"l"**. These letters also use very similar control command sequences. The recognition was considerably robust while both letters were written of the expected sizes (as estimated during the *Parameter estimation procedure*). The errors started to occur during the attempts to scale **"l"** down or **"e"** up.

In the cases of all the other letters the error rate was very low and the recognition was close to 100%.

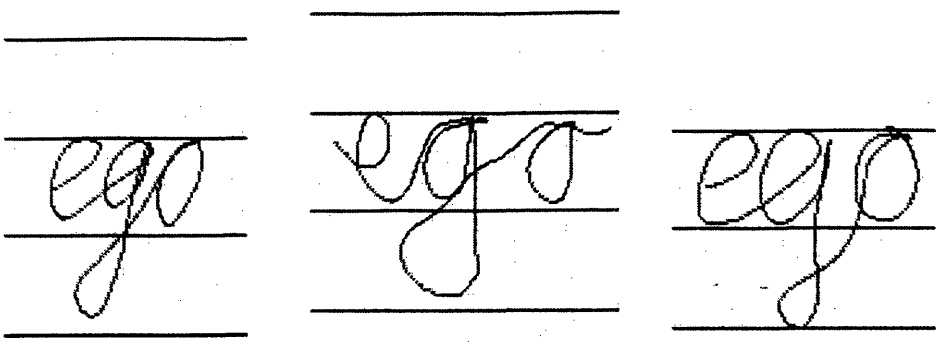


Figure 4-1: Examples of different writing styles used for the word "ego". In all cases the word has been recognized correctly.

In the three–letter dictionary the error was distributed evenly among all the words and was increasing with the increase of handwriting speed, which resulted from a worse sampling at higher speed.

The system seems to be considerably robust to the change of writing style and scaling. Figure 4-1 shows different styles of handwriting used by the different subjects to write the word **"ego"**.

This insensitivity to the style change can be explained by the fact that all the subjects regardless of the personal habits in writing used the same underlying motor commands when they were attempting to write a particular letter (and as a result, a particular word). It also proves that the chosen basic commands were in general correct.

The scaling problem was solved on the level of time-alignment module. On this level the difference between two identical letters, one of which is scaled up or down, is just the difference in how long the system is staying in each of the allowed commands. This approach also allows to change the size of different parts of the letter independently. This effect results in the ability to recognize different letter shapes and styles.

These two results, namely ability to accommodate different writing shapes and

stiles and good scaling performance, make it possible to eliminate parameter estimation module from the future implementation of a new version of **WordCracker**.

All the presented results and consideration make the proposed model a very promising approach to on-line cursive handwriting recognition, and **WordCracker** a very robust handwriting recognition system, which requires a minimal training and is able to accommodate various sizes, shapes, and styles of handwriting.

Chapter 5

Conclusions and Future Work

Although the idea that the pen movements in the production of cursive handwriting are the results of a simple "motor program" is quite old, the task of revealing this "motor code" remains a difficult inverse–dynamic problem.

In this paper we have presented a robust scheme which transforms the continuous pen movements into discrete *motor control commands*. These commands can be interpreted as a possible high level coding of the motor system.

A cursive handwriting recognition system, **WordCracker**, has been created based on the mentioned above model. Using analysis by synthesis to solve the inverse-dynamic problem, low error rates on constrained dictionaries were achieved (2.4% on the single letter dictionary and 7.6% on the three letter dictionary).

The system does not require practically any training whatsoever. Currently it needs to estimate a few parameters of the script (which is very fast and easy for the writer), but even these estimations can be eliminated in the future. The idea of representing any possible style of handwriting as a sequence of limited number of motor commands and system good performance in scaling make this goal a relatively easy task.

Another future goal is to test **WordCracker** on much bigger dictionaries (3000–5000 words). The realization of this goal require faster hardware and probably some optimization of recognition algorithm.

The recognition system presented in this paper does not purpet to be a solution

to machine cursive handwriting recognition. Issues such as recognition of groups of several words or sentences were ignored. These topics are the objectives for the future research.

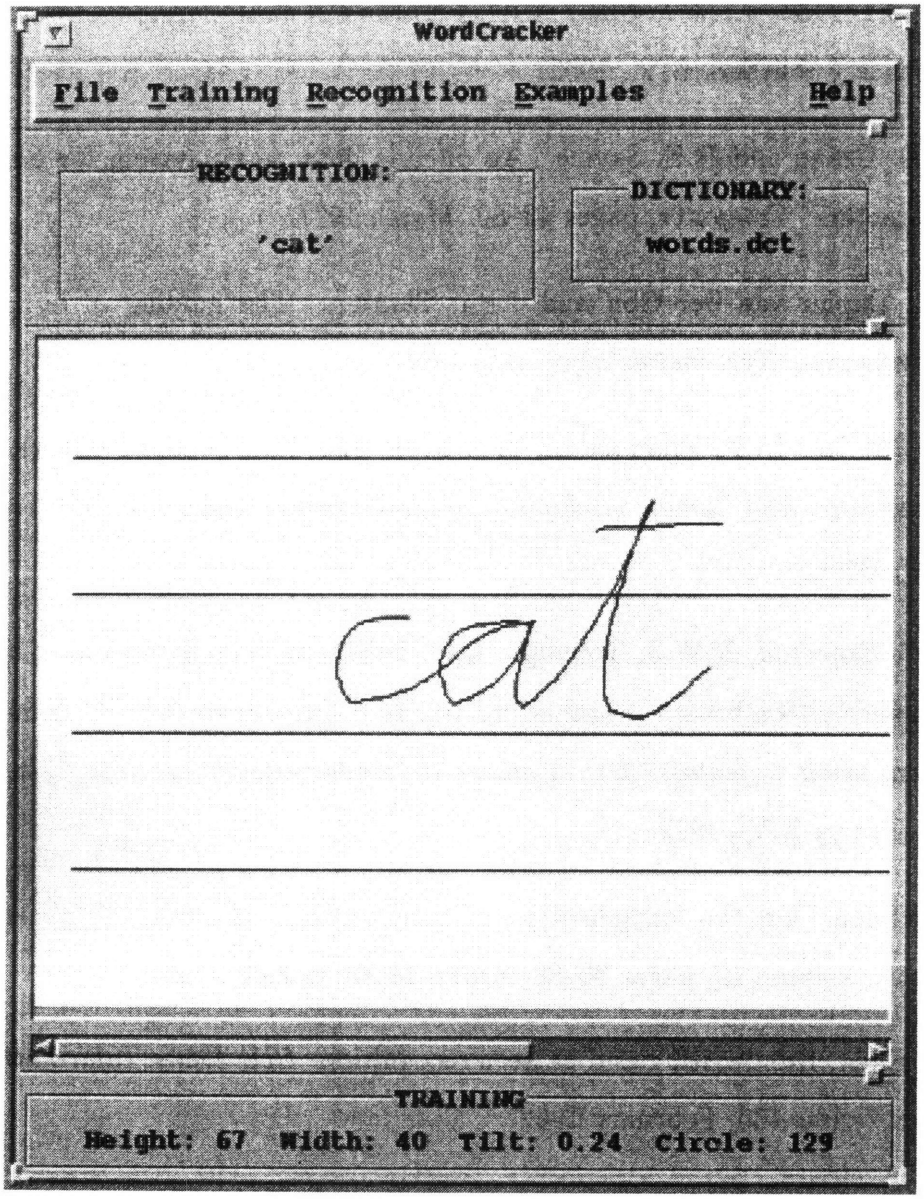
The *discrete motor control* representation largely reduces the variability in different writing styles and writer specific effects. Since different writing styles are transformed to the same representation, the transformation itself can be used for text independent writer identification and verification tasks.

Although the relationship between this representation and the actual cognitive representation of handwriting remains open, though there is some psychophysical experimental evidence linking the recognition time to the writing time for handwriting [10].

Another conclusion that can be drawn from the analysis of the presented model is, that the proposed approach is not limited to the handwriting recognition and can be used as a general purpose recognition scheme for any kind of dynamical (temporal) data. Automated speech recognition, that does not require days of training, or real-time lips reading are the potential applications of this approach.

Appendix A

WordCracker



Bibliography

[1] J. Camillerapp, G. Lorette, H. Oulhadj, and J.C. Pettier. Off-line and on-line methods for cursive handwriting recognition. In S. Impedovo and J.C. Simon, editors, *From pixels to features III: Frontiers in handwriting recognition.* Elsever Science Publishers B.V., 1992.

[2] H.D. Crane and R.E. Savoie. An on-line data entry system for hand-printed characters. *Computer*, pages 43–50, March 1977.

[3] J.J. Danier van der Gon and J.Ph. Thuring. The guiding of human writing movements. *Kybernetik*, 2:145–148, 1965.

[4] E. Doojies. In R. Plamondan, S.Y. Suen, and M.L. Simner, editors, *Computer recognition and human production of handwriting*, pages 119–130. World Scientific, 1989.

[5] A.C. Downton, R.W.S. Tregidgo, C.G. Leedham, and Hendrawan. Recognition of handwritten british postal addresses. In S. Impedovo and J.C. Simon, editors, *From pixels to features III: Frontiers in handwriting recognition.* Elsever Science Publishers B.V., 1992.

[6] M. Eden. On the formalization of handwriting. In *Proc. of Siposia in Appl. Math.*, volume 12, pages 83–88. Amer. Math. Society, 1961.

[7] M. Eden. Handwriting and pattern recognition. *IRE Trans. Information Theory*, IT-8(2):160–166, February 1962.

[8] M. Eden and M. Halle. The characterization of cursive writing. In C. Cherry, editor, *4th London Symposium on Information Theory*. Butterworth, 1961.

[9] G.D. Forney. The vitervi algorithm. *Proc. IEEE*, 61:268–278, March 1973.

[10] J.R. Frederiksen and J.F. Kroll. Spelling and sound: approaches to the internal lexicon. *J. Exp. Psyc. Hum. Percept. Perform.*, 2(3):361–379, 1976.

[11] T. Fujisaki, H.S.M. Beigi, C.C. Tappert, M. Ukelson, and C.G. Wolf. Online recognition of unconstrained handprinting: a stroke–based system and its evaluation. In S. Impedovo and J.C. Simon, editors, *From pixels to features III: Frontiers in handwriting recognition*. Elsever Science Publishers B.V., 1992.

[12] S. Grillner. Locomotion in vertebrates: Central mechanisms and reflex interaction. *Physiol. Rev.*, 55:247–304, 1975.

[13] M. Halle and K. Stevens. Speech recognition: a model and a program for research. *IRE Trans. Information Theory*, IT-8(2):155–159, February 1962.

[14] N.M. Herbst and C.N. Liu. Automatic signature verification based on accelerometry. *IBM J. Res. Develop.*, pages 245–253, May 1977.

[15] J.M. Hollerbach. *An oscillation theory of handwriting*. PhD thesis, Massachusetts Institute of Technology, March 1980.

[16] J.J. Hull, T.K. Ho, J. Favata, V. Govindaraju, and S.N. Srihari. Combination of segmentation-based and wholistic handwritten word recognition algorithms. In S. Impedovo and J.C. Simon, editors, *From pixels to features III: Frontiers in handwriting recognition*. Elsever Science Publishers B.V., 1992.

[17] Ken ichi Iso. Speech recognition using dynamic model of speech production. *CMU-CS-92-187*, pages 1–9, September 1992.

[18] Ken ichi Iso and Takao Watanabe. Speaker–independent word recognition using a neural prediction model. In *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 441–444. IEEE, April 1990.

[19] S. Impedovo, editor. *Fundamentals in handwriting recognition.* NATO ASI Series. Computer and System Sciences Vol.124. Springer–Verlag, 1992.

[20] S. Impedovo and J.C. Simon, editors. *From pixels to features III: Frontiers in handwriting recognition.* Elsever Science Publishers B.V., 1992.

[21] E. Levin. Word recognition using hidden control neural architecture. In *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 433–436. IEEE, April 1990.

[22] J.S. McDonald. Experemetal studies of handwriting signals. *RLE Technical Report 443*, 1966.

[23] P. Mermelstein and M. Eden. Experiments on computer recognition of connected handwritten words. *Information and Control*, 7:255–270, 1964.

[24] F. Nouboud. Handwritten signature verification: a global approach. In S. Impedovo, editor, *Fundamentals in handwriting recognition*, NATO ASI Series. Computer and System Sciences Vol.124. Springer–Verlag, 1992.

[25] G. Pirlo. Algorithms for signature verification. In S. Impedovo, editor, *Fundamentals in handwriting recognition*, NATO ASI Series. Computer and System Sciences Vol.124. Springer–Verlag, 1992.

[26] R. Plamondan and C.G. Leedham. *Computer processing of handwriting.* World Scientific, 1990.

[27] R. Plamondan, S.Y. Suen, and M.L. Simner. *Computer recognition and human production of handwriting.* World Scientific, 1989.

[28] R. Plamondon. A mofel based dynamic signature verification system. In S. Impedovo, editor, *Fundamentals in handwriting recognition*, NATO ASI Series. Computer and System Sciences Vol.124. Springer–Verlag, 1992.

[29] D.E. Rumelhart. Theory to practice: a case study — recognizing cursive handwriting. *Proc. of 1992 NEC Conf. on Computation and Cognition*, 1992.

[30] K. Sakai, Y. Kurosawa, and T. Mishima. Total approach for practical character recognition system development. In S. Impedovo, editor, *Fundamentals in handwriting recognition*, NATO ASI Series. Computer and System Sciences Vol.124. Springer–Verlag, 1992.

[31] L.R.B. Schomaker and H.-L. Teulings. Stroke–versus character–based recognition of on–line, connected cursive script. In S. Impedovo and J.C. Simon, editors, *From pixels to features III: Frontiers in handwriting recognition.* Elsever Science Publishers B.V., 1992.

[32] J.C. Simon. A complemental approach to feature detection. In J.C. Simon, editor, *From pixels to features: proceedings of a workshop held at Bonas, France, 22–27 August, 1988.* Elsever Science Publishers B.V., 1989.

[33] J.C. Simon, editor. *From pixels to features: proceedings of a workshop held at Bonas, France, 22–27 August, 1988.* Elsever Science Publishers B.V., 1989.

[34] J.C. Simon and O. Baret. Cursive words recognition. In S. Impedovo and J.C. Simon, editors, *From pixels to features III: Frontiers in handwriting recognition.* Elsever Science Publishers B.V., 1992.

[35] Y. Singer and N. Tishby. Dynamical encoding of cursive handwriting. *Biological Cybernetics*, February 1994.

[36] C.C. Tarpet, C.Y. Suen, and T. Wakahara. The state of the art in on-line handwriting recognition. *IEEE Trans. Pattern Anal. Mac Intell*, 12(8):787–808, 1990.

[37] J. Tebelskis and A. Waibel. Large vocabulary recognition using linked predictive neural networks. In *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 437–440. IEEE, April 1990.

[38] N. Tishby. A dynamical systems approach to speech processing. In *IEEE Proceedings of the International Conference on Acoustics, Speech and Signal Processing*, pages 365–368. IEEE, April 1990.

[39] A.J. Viterbi. Error bounds for convolutional codes and an asymptotically optimal decoding algorithm. *IEEE Trans. Information Theory*, IT–13:260–269, April 1967.

[40] M. Yasuhara. Experimental studies of handwriting process. *Rep. Univ. Electro–Comm.*, 25(2):233–254, 1975.

[41] A. Zahour, B. Taconet, and A. Faure. Machine recognition of arabic cursive writing. In S. Impedovo and J.C. Simon, editors, *From pixels to features III: Frontiers in handwriting recognition.* Elsever Science Publishers B.V., 1992.

5778-61