ONLINE HANDWRITING RECOGNITION

Online recognition of handwriting has been a research challenge for almost 40 years (1). Contrary to offline techniques (2–9), which deal with the automatic processing of the image $I(x, y)$ of handwritten words as collected using a camera or a scanner, online approaches aim at recognizing words as they are written, using a digitizer or an instrumented pen to capture pen-tip position (x_t, y_t) as a function of time.

Table 1 shows the basic differences between offline and online handwriting recognition. Figures 1(a) and 1(b) illustrate a typical image $I(x, y)$ of a word and the (x_t, y_t) position of the

Table 1. Distinction Between Online and Offline Handwriting Signals

Method	Representation	Transducer
Offline	$I(x, y)$	Optical scanner
Online	(x_t, y_t)	Digitizer, tablet

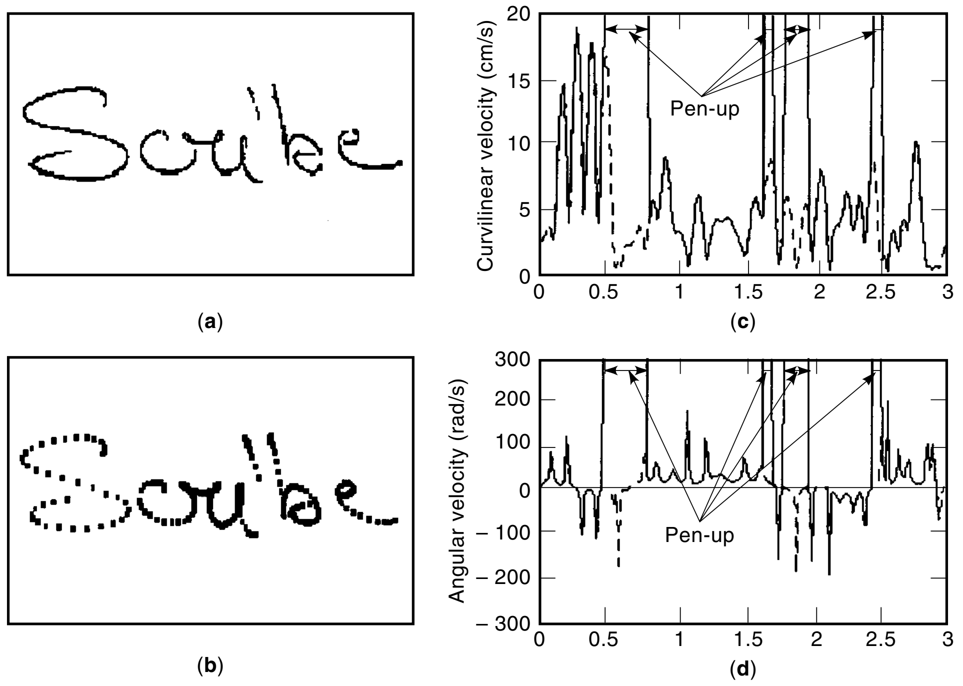


Figure 1. (a) Binarized image of a handwritten word; (b) digitized *XY* trajectory of the pen tip; (c) tangential velocity of the pen tip; (d) angular velocity of the pen tip.

pen that was used to write it, as a function of time *t*. As can be seen, the major difference between online and offline data is that the time sequence of movements for online data is directly available from the acquired signals. This allows also the computation of other signals, like velocity signals, from the original data [for examples, see Figs. 1(c) and 1(d)].

Figure 2 depicts a simplified data flow diagram of an online handwriting recognition system, highlighting the main functions performed by such a system. The (x_t, y_t) coordinates collected by the digitizer are first filtered to remove spurious noise. Then some specific characteristics of the signals are extracted and compared with those of letters or words kept in a reference lexicon. The candidate letters or words produced by the recognition algorithms are then analyzed in the context of lexical, syntactical, semantical, and application-based pragmatical rules to make a decision and to propose the most realistic solution. Numerous variations of this basic description have been proposed to take into account the various feedback loops and parallel paths that are essential to design an efficient system. This article surveys the principal trends that have been put forward and published since the previous general reviews in the field (10,11).

Because of space limitation, this chapter focusses on the recognition of the English language. Readers interested in other languages can consult others surveys (12) or several books, journals, and conference proceedings where such projects are reported (13–15). We also neglect to cover applications dealing with automatic signature verification (16,17), shorthand (18), and gesture recognition as well as interactive tools to help children or disabled persons to learn handwriting. Moreover, we pay particular attention to methods and approaches that can be used in an electronic pen–pad environment. Numerous midterm projections (19) have predicted that these computers without keyboard will be part of the next computer generation, allowing them to mimic the pen–paper interaction in numerous tasks that involve both point-

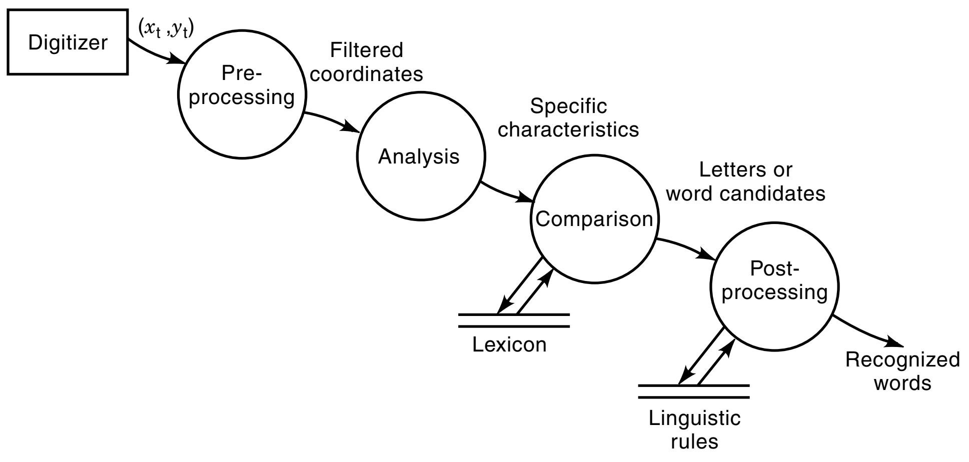


Figure 2. Data flow diagram of an online recognition system.

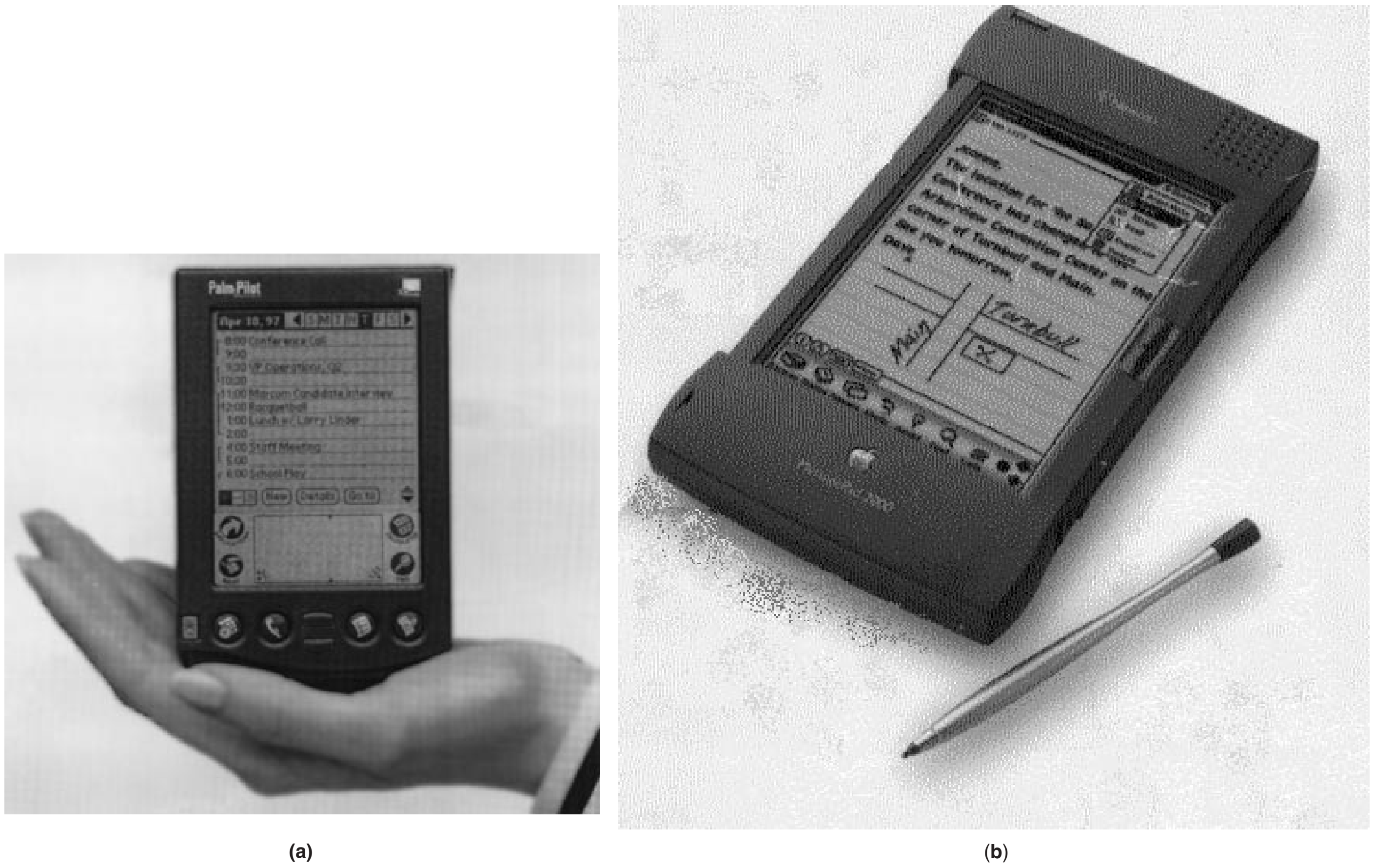


Figure 3. Examples of electronic pen-pads. A transparent digitizer is superimposed on a liquid crystal display. The overall system encompasses hardware and software to process the electronic ink. (a) Pilot (3Com and the 3Com logo are registered trademarks, and PalmPilot and the PalmPilot logo are trademarks of Palm Computing, Inc., 3Com Corporation, or its subsidiaries) and (b) Newton model.

ing and data entry (20,21) (see typical products Fig. 3). The problem of determining writing style (printed, cursive, or mixed) when it is not known a priori is not considered in detail [see, for example, (22)]. Similarly, we do not address nontextual input recognition. For maximum flexibility, a pen-based system should allow for more than just textual input (consider the openness of a traditional pad of paper to accept anything the user might want to write). Such a system must be prepared to accept nontextual data such as equations and diagrams (23–25) and graphical inputs (26–29). Another issue not considered here is the representation and compression of handwritten data (30–33).

The rest of the article is divided into four parts. First, we briefly survey the basic knowledge related to the handwriting generation processes to better understand the ground of the methodologies that are described in the next three sections dedicated, respectively, to data acquisition and preprocessing, script recognition, and linguistic postprocessing. We conclude by pointing out some of the remaining problems and suggesting some promising research directions.

HANDWRITING

Handwriting stands among the most complex tasks performed by literate human beings. It requires sensorimotor control mechanisms involving a large spectrum of cerebral activities, dealing with the emotions, rational thought, and communications. As such, the study of handwriting constitutes a very broad field that allows researchers with various backgrounds and interests to collaborate and interact at multiple levels with different but complementary objectives (34–45).

As suggested by many of those groups, handwriting involves several functions (see Fig. 4). Starting from a communication intention, a message is planned at the semantical, syntactical, and lexical levels and converted somehow into a set of allographs, graphemes, and strokes to generate a pen-tip trajectory that is recorded on a planar surface physically or electronically. The resulting document will be read later by the person to whom the message is dedicated.

Consistent with this view, some design methodologies incorporate this theoretical framework in the development of online handwriting processing systems. So far, numerous models have been proposed to study and analyze handwriting. Depending on the emphasis placed on the symbolic information or on the neuromuscular architecture, two complementary approaches have been followed: top-down and bottom-up. The top-down approach has been developed mainly by those researchers interested in the study and application of the various aspects of the high-level information processing from semantical, syntactical and lexical aspects down to basic motor

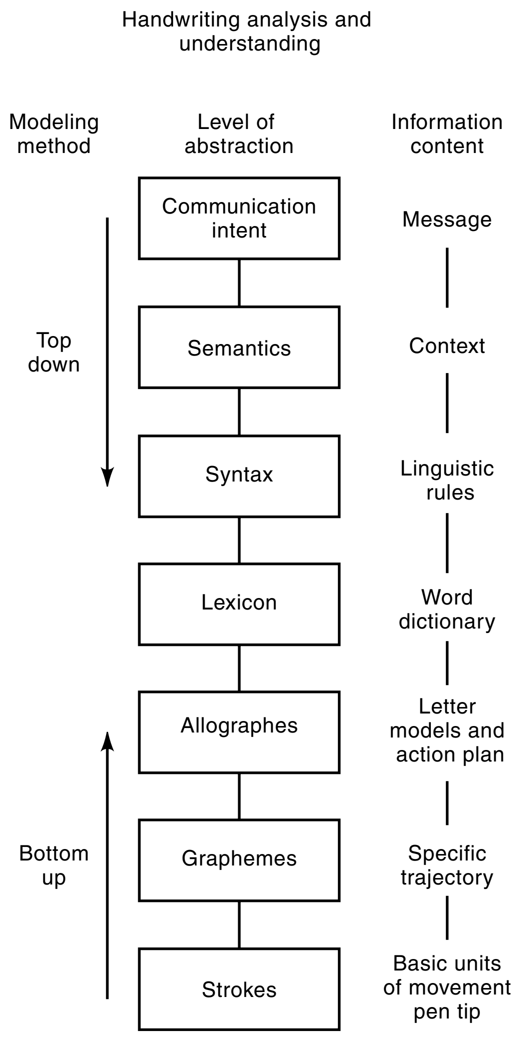


Figure 4. General overview of the different neuromuscular processes involved in handwriting.

control problems like the coding of fundamental units of movement, the sequencing and retrieval of motor commands, the control and learning of movement, and the evaluation of task complexity. The bottom-up approach has been used by those interested in the analysis and synthesis of the low-level neuromuscular processes starting at the stroke level to recover graphemes, allographs, the ultimate goal being to recover the message. In the domain of automatic recognition, top-down models are generally used as postprocessing techniques to validate the output of a specific recognition system, whereas bottom-up model is mostly used for recovering strokes graphemes or allographs.

Most of the top-down models are not specific to handwriting and have been developed for other language processing purposes like automatic speech recognition and automatic translation. In the domain of handwriting, these models focus mostly on lexical and syntactical knowledge to correct incorrect words in a processed sentence. As will be seen in more details in the section dealing with postprocessing, two approaches are used to integrate lexical information in a system: a statistical and a combinational approaches. The first approach is based on the statistics of letter occurrence (46–49), whereas the second mostly deals with the calculation of distances between character strings (50,51). The integration of syntactical knowledge can also be done using two different models: structural and probabilistic models. In the first case, sentences are analyzed using sentence structure grammars to decide whether a sentence is grammatically meaningful (52). In the second case (53), probabilities of co-occurrence of three words (54) in the lexicon or of tri-part of speech (55–57) constitute the basis of model development. It must also be noted that very often a combination of models and methods lead to better results.

As for many well-mastered tasks, human subjects generally work at the highest and most efficient level of abstraction possible when reading a handwritten document (semantical, syntactical, lexical). When difficulties are encountered in decyphering a part of the message using one level of interpretation, they often switch to a lower level of representation (allographs, graphemes, strokes) to resolve ambiguities. In this perspective, the lower levels of abstraction, although generally used in the background, constitute a cornerstone on which a large part of the higher and more abstract process levels relies. For example, according to motor theories of perception (58), it is assumed that motor processes enter into the genesis of percepts and that handwriting generation and perception tasks interact and share sensorimotor information. Handwriting recognition tasks also require, directly or indirectly, an understanding of the handwriting generation processes from a bottom-up approach.

Bottom-up models [see Plamondon and Maarse (59) for a survey of the models published prior to 1989 and Plamondon et al. (60) for a comparative study of numerous models] can be divided according to the type of basic movements that are used to analyze and reconstruct a complex pen-tip trajectory (61,62). Discrete models (61,63,64) reconstruct complex movements using temporal superimposition of a set of simple discrete movement generally referred to as strokes. Continuous movements emerge from the time overlap of discontinuous strokes. Oscillatory models (65–68) consider oscillations as the basic movement and complex movement generation is produced by controlling the amplitude, the phase, and the frequency of the fundamental wave function. A single discontinuous stroke is seen here as a specific case of an abruptly interrupted oscillation.

Among these models, various approaches are used to trackle the problem of handwriting generation. For example, Schomaker et al. (69) use a computational approach to describe handwriting based on motor control principles and on the knowledge of idiosyncratic feature of the handwriting of a given writer. Bullock et al. (62) and Morasso et al. (70) use nonlinear dynamic approaches, whereas Flash and Hogan (71), Edelman and Flash (72), and Uno et al. (73) use models that exploit minimization principles—the first group minimizes jerk, the second minimizes the snap, and the third minimizes the torque. Other groups (66–68) use basic analytic functions to describe an oscillatory movement, without any further justification.

Figure 5 shows the results of a typical analysis by synthesis that can be done using a discrete bottom-up model. According to this recent model (61), a handwritten word is a sequence of components that is part of the trace between pen-down and pen-up (59,74). Each component is made up of a superimposition of circular strokes, each one being analyti-

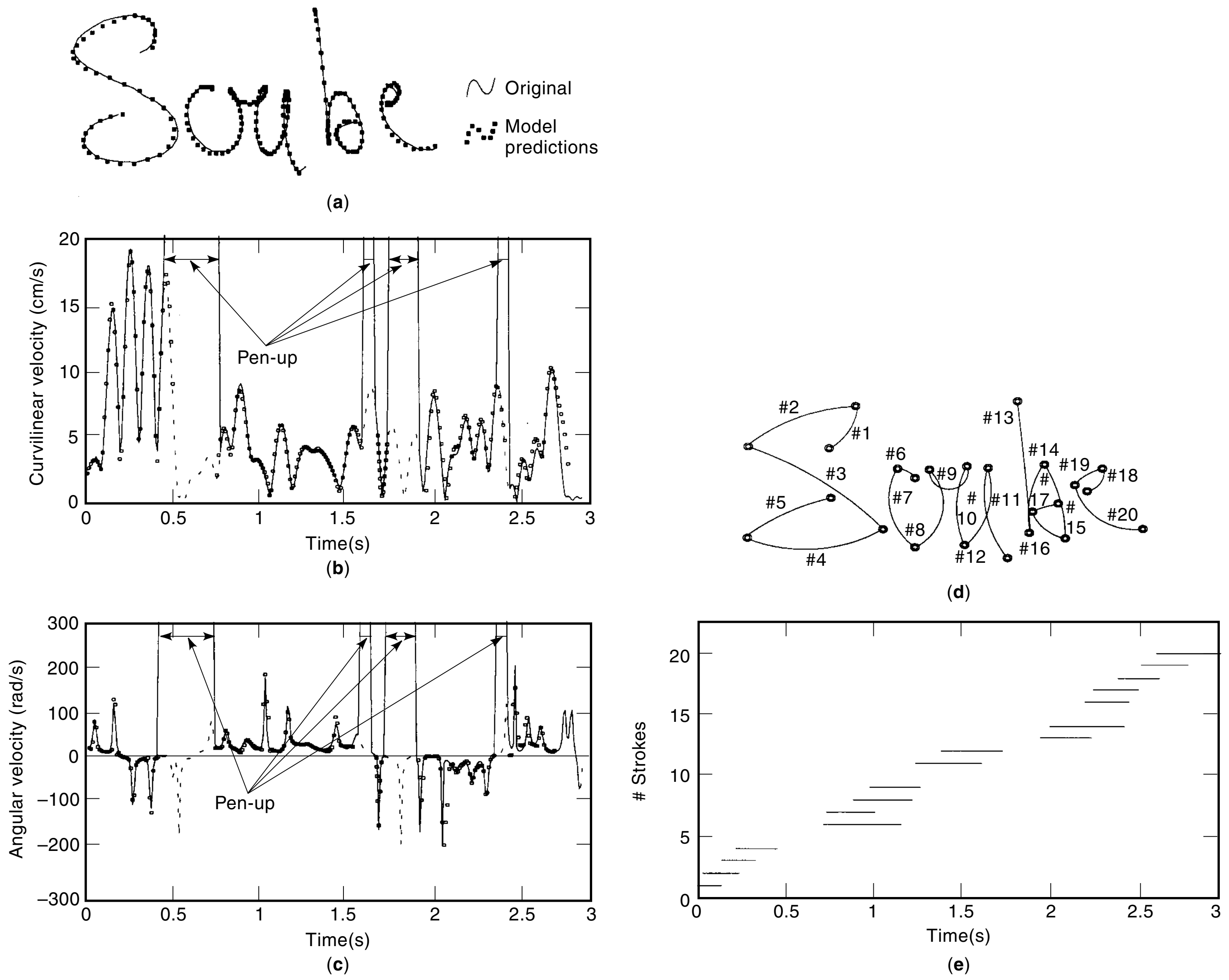


Figure 5. Typical result of an analysis-by-synthesis experiment using a bottom-up handwriting generation model.

cally described by a velocity vector $\boldsymbol{v}(t)$ whose magnitude obeys a delta-lognormal law (75,76):

$$|\boldsymbol{v}(t)| = |\boldsymbol{D}_{1(P_0,\theta_0,C_0)}|\Lambda(t;t_0,\mu_1,\sigma_1^2) - |\boldsymbol{D}_{2(P_0,\theta_0,C_0)}|\Lambda(t;t_0,\mu_2,\sigma_2^2) \quad (1)$$

where

$$\Lambda(t;t_0,\mu_i,\sigma_i^2) = \frac{1}{\sigma_i\sqrt{2\pi}(t-t_0)}\exp\left(\frac{-[\ln(t-t_0)-\mu_i]^2}{2\sigma_i^2}\right) \quad (2)$$

with $t > t_0$

where

D_i = amplitude of the agonist ($i = 1$) and antagonist ($i = 2$) commands,
t_0 = time occurrence of the synchronous input commands,
μ_i = log time delay of the agonist ($i = 1$) and antagonist ($i = 2$) systems,
σ_i = log response time of the agonist ($i = 1$) and antagonist ($i = 2$) systems,

and with an orientation that evolves along the circle arc according to

$$\angle\boldsymbol{v}(t) = \theta(t) = \theta_0 + C_0\int_{t_0}^{\infty}|\boldsymbol{v}(t)|\,dt \quad (3)$$

A whole component can thus be described in the velocity domain, as the vectorial summation of velocity vectors:

$$\boldsymbol{v}(t) = \sum_{i=1}^{n}\boldsymbol{v}_i(t-t_{0i}) \quad (4)$$

with each $\boldsymbol{v}_i(t - t_{0i})$ described by Eqs. (1) and (3), for $t > t_{0i}$.

Using nonlinear regression, a set individual strokes can be recovered from the velocity data and each of them can be characterized by a set of nine parameters as described earlier.

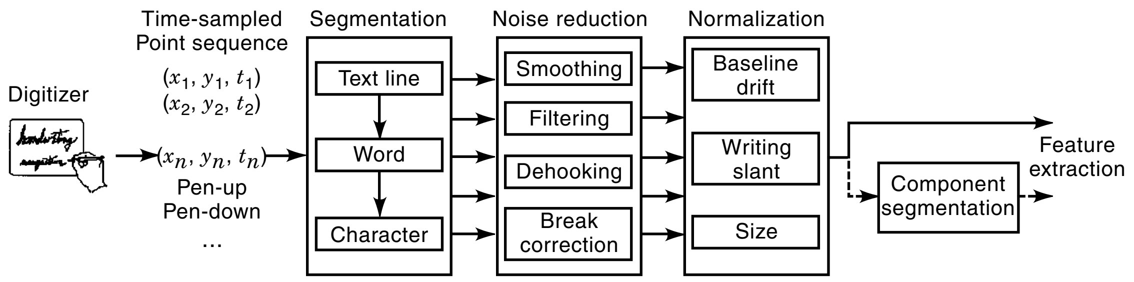


Figure 6. Typical end-to-end preprocessing.

As can be seen in this example [Fig. 5(d)] the word *scribe* is made up of four major components, the first component encompasses five strokes whereas the second, third, and fourth have seven, five, and three strokes, respectively. Figure 5(e) shows how these strokes superimpose in time. Figures 5(d) and 5(e) can be interpreted as a spatial and a temporal representation of the action plan that has been used to generate the specific word. Figures 5(a)–5(c) show that an excellent reproduction of the word can be achieve both in the spatial and in the velocity domain using this model. Such a global approach can be use to provide basic knowledge for cursive script segmentation and character and word recognition, as will be seen later.

DATA ACQUISITION AND PREPROCESSING

The purpose of preprocessing is to correct for problems in data acquisition, to compensate for some of the natural variation that arises between writers, and to segment handwriting into more basic units for later recognition.

The input to preprocessing is assumed to be a sequence of points sampled over time, (x_t, y_t). These record the trajectory of the pen tip over the surface of the digitizer as previously mentioned. In addition, components or pen-down and pen-up events are noted using specific contact switch, pressure or strain-gauge and the like. Because of the online nature of the problem, the data has both spatial and temporal aspects that can be employed as appropriate. Depending on the classification stages that follow, the output from preprocessing can be features extracted either from word units or from more primitive character or subcharacter units.

The basic stages of preprocessing include segmentation, noise reduction, and normalization. The order in which these are performed can vary from system to system, and specific steps may be repeated and/or skipped altogether. Figure 6 illustrates the end-to-end process for a typical case. Earlier overviews of preprocessing can be found in Tappert et al. (77), Nouboud and Plamondon (11,78), Tappert (79), and Guerfali and Plamondon (80).

Segmentation may take place at four conceptual levels:

1. Text line segmentation—divides the input ink data into individual lines;
2. Word segmentation—breaks each line into separate word units;
3. Character segmentation—if the classifier is character-based, the words are further segmented into character candidates;
4. Stroke segmentation—the segmentation of the pen-tip trajectory into small segments (e.g., representation velocity-based strokes or strokes bounded by points of high curvature).

Noise reduction involves the following steps:

1. Smoothing—eliminates noise introduced by the tablet or shaky writing;
2. Filtering—reduces the number of data points and eliminates “wild” points;
3. Dehooking—removes artifacts (“hooks”) at the beginnings and ends of strokes;
4. Break correction—eliminates unintended (short) breaks between strokes.

Common normalization procedures include:

1. Baseline drift—corrects for the tendency of the character baseline to rise or fall as writing progresses from left to right;
2. Writing slant—compensates for the natural slant that can vary widely from writer to writer;
3. Size normalization—adjusts the characters to be a standard size.

Lastly, the preprocessor must generate a representation appropriate for input to the classifier used by unspecific system. The range of possible representations is enormous and highly dependent on the classifier. A common operation at this point, however, is to segment the input further into subcharacter units like strokes (a step also known as oversegmentation).

Each of these stages is now discussed in more detail.

Segmentation

Segmentation is the process of breaking the input data into smaller and smaller logical units (a divide-and-conquer approach). These basic units can either be disjoint, or allowed to overlap each other (see Fig. 5). In many cases, the segmentation is only tentative and may be corrected later during classification. Figure 7(a) illustrates several of the various problems that arise during segmentation, whereas Fig. 7(b) shows their solutions.

The earliest systems achieved segmentation at the character level through either an explicit signal from the user (81), or by requiring the user to write in predefined boxes (82). Both of these assumptions can be burdensome, however. A

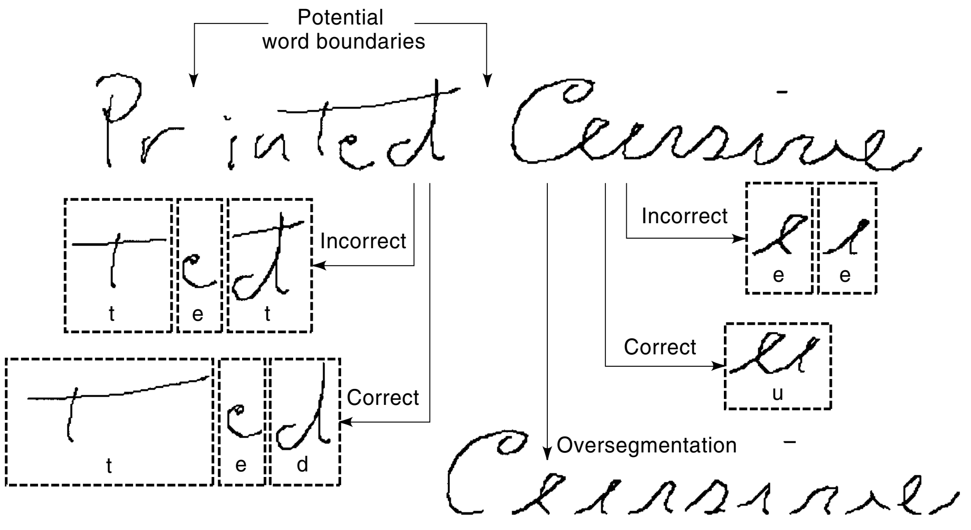


Figure 7. Illustration of the various segmentation problems and their solutions.

more accommodating policy is to allow the user's input to be completely free-form, even though this makes the segmentation problem more difficult.

Text Line Segmentation. If the handwriting is not being recognized real-time but is being saved for later processing (as in an electronic reproduction of a paper notepad), the issue of identifying the lines of text comes first. An approach for text line segmentation (zoning) for unconstrained handwritten cursive script based on approximating spline functions is discussed in Hennig et al. (83).

Word Segmentation. Word segmentation can be performed in either the temporal or spatial domain or through a combination of both (77). For example, a new word can be said to begin with a pen-down event if the time since the last pen-up exceeds a given threshold and/or if the pen tip has moved more than a certain minimum distance to the right of the previous stroke. In the case of word-based classifiers (sometimes called holistic approaches), the segmentation process is complete at this point.

Character Segmentation. Attempting to segment the input into individual characters can be extremely challenging. In the case of cursive writing, ligatures make it difficult to tell where one character ends and the next begins. For printed text, overlapping characters can still cause problems: the cross of a *t*, for example, may interfere with adjacent characters on either side. Temporal approaches (i.e., timeouts) have been applied to the segmentation of carefully handprinted characters (77,78). Even so, a *t* may be crossed right away, or perhaps not until after the rest of the word is finished.

An approach for segmenting cursive script into allographs based on unsupervised learning appears in Mackowiak, Schomaker, and Vuurpijl (84). The problem of segmenting numerals and capital letters in mixed (cursive/printed) writing is described in Lee et al. (85).

The difficulties of character segmentation can be sidestepped by changing the input alphabet to a unicomponent (86) [see also Fig. 3(a)]. Because each character is represented by exactly one component (pen-down to pen-up), there is no ambiguity as to character boundaries. This is a radical assumption, however, because it requires the user to learn a new way of writing.

When character segmentation is treated as a preprocessing step, as described here, it is known as "external segmentation." A different strategy is to delay segmentation and handle it within the recognition process; this is known as "internal segmentation" or "segmentation by recognition."

Noise Reduction

Noise reduction is necessary to address limitations in the tablet hardware (finite resolution, digitization errors, mistaken components), as well problems the user might cause by dwelling with the pen too long at a single location. Figure 8(a) shows a contrived example illustrating all of these problems, whereas Fig. 8(b) shows the results of the corrective procedures described in this section.

Smoothing. The purpose of smoothing is to eliminate noise introduced by the tablet or shaky writing. This is accomplished by averaging a point with certain of its neighbors, usually those recorded prior to the point in question (so that processing can proceed forward in time as each point is received). Possible approaches include using a mobile average filter (78,79).

Filtering. Filtering reduces the number of data points and in particular eliminates duplicates (caused when the user lets the pen tip linger too long at one spot). This could involve, for example, resampling based on a minimum distance between points, or a minimum change in direction of the tangent to the writing. The latter allows for more points in regions of greater curvature and is generally preferable. The filtering procedure also corrects for wild points (hardware-induced outliers far away from the actual trajectory of the pen tip) (78,79).

Dehooking. Hooks are artifacts at the beginnings and ends of strokes caused by the inaccurate detection of pen-down/pen-up events. They can be found and removed by searching for a significant direction change near the edge of a component and, if the size of the hook is small, eliminating it (79).

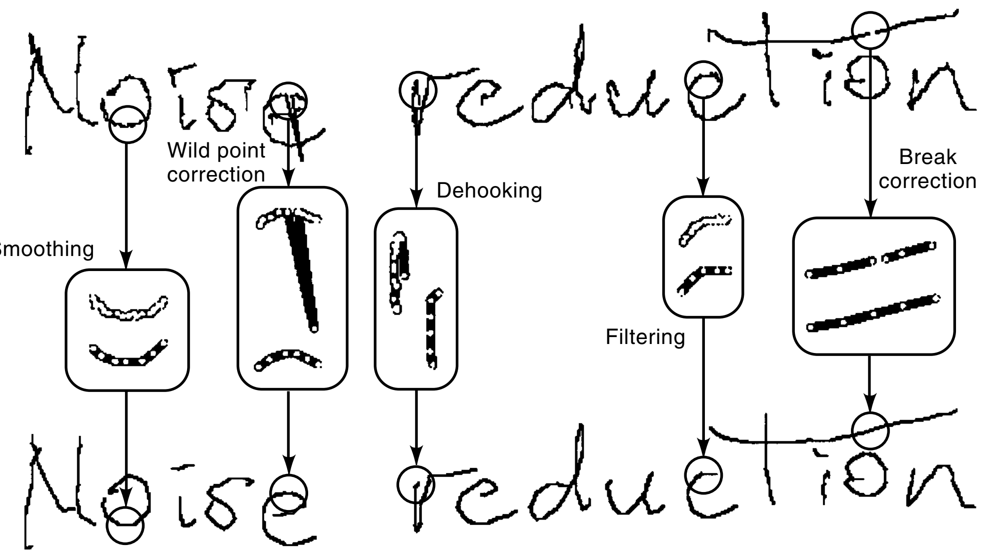


Figure 8. Illustration of the various noise conditions and their solutions.

Break Correction. Inadvertent breaks between components can be corrected by connecting adjacent components when the distance between the pen-up and pen-down is small relative to the size of the writing (77).

Normalization

In practice, there is a great deal of variability in the way people write a particular allograph (87). The goal of geometric normalization is to minimize (or at least reduce) these instance- and writer-dependent variations. An example illustrating these problems is shown in Fig. 9(a), whereas their solution appears in Fig. 9(b).

Baseline Drift. As the user writes across a tablet, it is not unusual for the text to drift upward or downward (this is especially true if the tablet has no visible guidelines). The later determination of certain key features such as ascenders and descenders depends on having an accurate estimate of the character baseline. Hence, drift must be compensated for (88).

Writing Slant. Cursively written text can exhibit significant slant. This is estimated and corrected using techniques such as those described in Brown and Ganapathy (88) (on a per-word basis) or Burr (89) (on a per-character basis). A good estimate of the writing slant angle is the direction of a vector which is tangential to the writing trace at the points of maximum velocity (and minimum curvature), in downward strokes of robust length (90).

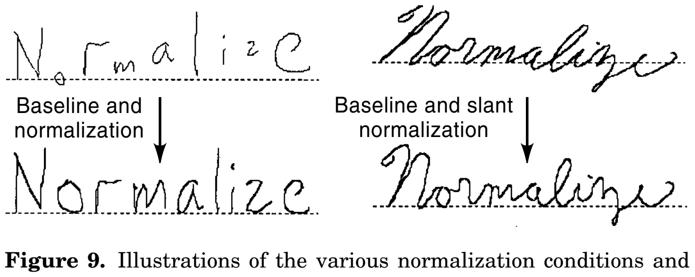


Figure 9. Illustrations of the various normalization conditions and their solutions.

Size Normalization. In the case of character-based recognition, the characters may be normalized by mapping them into a box of a fixed size (78). In addition, their coordinates may be translated so that the origin lies at a specific location relative to the bounding box (e.g., the lower left corner) (77). Component-based schemes may normalize each component to a specific number of points (77).

Subcharacter Segmentation

As was remarked earlier, segmenting cursive handwriting into characters on the basis of bottom-up, data-driven knowledge can be extremely difficult. Instead, a more promising strategy is to segment the input into strokes (see Fig. 5), which are then interpreted and combined in various ways during the recognition process (oversegmentation). For most of the systems based on Hidden Markov Models (HMMs), segmentation and recognition naturally take place at the same time as part of the optimization performed using for example the Viterbi algorithm. Also in other approaches (91,92), the segmentation into characters takes place because of a segmentation through classification approach. In fact, such methods utilize higher-level information of a lexical or statistical nature to solve the otherwise ill-posed segmentation problem for mixed and cursive script types (e.g., the word minimum).

A component-based approach to stroke segmentation is described in Fujisaki et al. (93). A method based on attempting to locate three distinct shapes in the writing—cusps, intersections, and inflection points—appears in Higgins and Ford (94). Another approach based on extracting shape primitives is described by Bontempi and Marcelli (95). Lower-level schemes for segmenting strokes based on local extrema of curvature points is Li et al. (96,97). Schomaker (98) uses a velocity-based segmentation into strokes, which also yields stroke boundaries at points of maximum curvature. Identifying perceptually important (i.e., robust) strokes is the focus of Anquetil and Lorette (99).

A scale-space approach—segmentation points obtained at a range of scales are combined to form the final segmentation—appears in Kadirkamanathan and Rayner (100). Oversegmentation based on horizontal and vertical constraints between allographs is discussed in Parizeau and Plamondon (101). Segmentation at vertical velocity zero crossings is the subject of Karls et al. (102). The notion of dual strokes (two successive down-up or up-down strokes that form certain shapes) as the basis for segmentation is discussed in Abbink et al. (103).

Segmentation using indefinite models learned through training is discussed in Hu et al. (104,105). Cusps, loops, and humps form the basic units of segmentation in Bercu and Lorette (106). Segmentation into elemental "frames" prior to HMM recognition is described in Bellegarda et al. (107). In Lee et al. (108), letter spotting is performed by a Hidden Markov Model yielding a word hypothesis lattice that is then searched to find the optimal path.

Handwriting Variability

The recognition of online handwriting was originally coined as an easy first step toward the more difficult recognition of speech, as was expected at the time. However, the problem of translating an input stream of pen-tip coordinates into an ASCII string of characters that correctly represents the text the writer intended to produce appears to be a very difficult task. Why is handwriting recognition so difficult? There are four basic and independent sources of variability and noise in handwriting patterns. The four factors all exert their influence on the quality of the written shapes:

1. Geometric variations: affine transforms;
2. Neuro-biomechanical noise: sloppiness space;
3. Allographic variation: letter-shape variants;
4. Sequencing problems: stroke order, spelling problems, corrective strokes.

Geometric Variations. Position, size, baseline orientation, and slant are all under voluntary control by the writer and must be normalized away. This can be done using an affine transform based on parameter estimation, as explained in the preprocessing section. It is at present unclear, however, whether the human writer just applies an affine transform when asked to write at a different slant, or whether more fundamental changes in the stroke production process also take place (109).

Neuro-Biomechanical Noise. Depending on the writer's mental concentration and the writing speed, stroke formation may be accurate or deformed. Sometimes the term sloppiness space (86) is used to describe this factor. The control of pen-tip movement is an articulation process that is under the influence of the recent past in the handwriting trajectory as well as under the influence of shapes that are being planned to be produced by the human motor system. Under conditions of suboptimal cognitive resource distribution, a number of effects may be observed at the microlevel of detailed trajectory control. Usually sharp stroke endings become obtuse or looped, and consecutive individual (velocity-based) strokes may fuse into a single stroke, barely representing the taught letter characteristics (109). However, human readers are often very well able to read sloppily written words, provided that a minimum number of legible characters are present. In many languages, the writer anticipates this ability of the reader and sloppily rushes over the lexically redundant tails of the words, similar to a damped oscillator.

Allographic Variation. An allograph is letter-shape variant (Fig. 4). A capital letter *A* and a lowercase letter *a* are both allographs of the letter identity *a*. Allographs are produced by different writers, on the basis of their writing education and personal preferences. But also within a writer, multiple variants of a letter shape may be used. Examples are different forms of *s* and *t*, depending on the position in the word. The number of allographs is very high in Western handwriting. Even within a limited category, like digits, a large amount of training data must be present to cover all shapes likely to be encountered by an online handwriting recognizer in a practical application. For the letters of the alphabet, allographic variation is virtually unlimited, because of the freedom for individual expression, which is given to writers in many Western cultures. It is useful to make a distinction between the kernel of a character shapes, and the initial and final ligatures and embellishments that may be encountered, especially in connected-cursive handwriting styles.

Sequencing Problems. As discussed previously, handwriting is a form of human motor output and suffers from the internal noise present in the central nervous system. Apart from the consequences at the micro level (see the section on modeling), neural variability also expresses itself in the form of errors and variations in the sequence of produced strokes. As an example, the capital letter *E* written as an isolated handprint character, may be produced in $4!(2^4) = 384$ different stroke orders (four strokes with two starting points per stroke). Although there is a preferred stroking order, there are individual differences, and the process is stochastic rather than deterministic. Spelling errors, letter omissions, letter insertions, and post-hoc editing with the pen all fall within this category. Also here, the handwritten product may very well be legible by humans, who are not aware of the underlying sequence variability. The last type of problem in this category comes from delayed dotting on the *i*s and *j*s and the crossing of the *t*. This process is also of a nondeterministic nature, and it cannot be predicted exactly when the dots and bars will be produced. In the worst case, the small dots and bars may be entered after finishing a whole sentence or paragraph.

Although some of the problems mentioned can be normalized away through a proper preprocessing method, as explained in the previous section, there will always remain a considerable variation and variability in the shapes that must be classified during the recognition process of online handwriting. The most difficult problem to handle is that of sloppiness space. Whereas the problem of allographic variation can be solved by using larger training sets of handwritten samples to develop online handwriting recognizers, it is difficult to deconvolve the fused and sloppy pen traces back into their clean, intended form (109).

Furthermore, it is evident that handwritten shapes isolated from their context cannot be recognized correctly by humans (110,111). Thus, linguistic context must be taken into

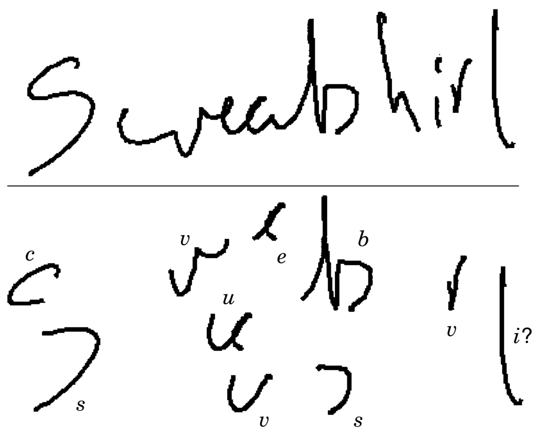


Figure 10. An arbitrary sample of natural handwriting obtained from a large database.

Table 2. UNIPEN Benchmark Overview: Categories of Handwritten Input

Benchmark	Description
1a	Isolated digits
1b	Isolated uppercase
1c	Isolated lowercase
1d	Isolated symbols (punctuations, etc.)
2	Isolated characters, mixed case
3	Isolated characters in the context of words or texts
4	Isolated printed words, not mixed with digits and symbols
5	Isolated printed words, full character set
6	Isolated cursive or mixed-style words (without digits and symbols)
7	Isolated words, any style, full character set
8	Text (minimally two words of) free text, full character set

account, as will be described in the section on postprocessing techniques. Figure 10 displays many problems of handwriting recognition. For a human writer, the word is fairly easy to decipher. Below the dotted line, some actual segments of handwriting are cut out from the local context. For each segment in the pen-tip trajectory, many reasonable character hypotheses can be generated, only one of them being the correct letter. Here is a small list of problems in this example: (1) The *t*s are not crossed, (2) the initial letter *s* may be uppercase or not, (3) the *ts* looks like a *b*, (4) several letters (*w*, *r*, *h*) are ill formed, and (5) the base line is not horizontal. Furthermore, this writer produces a mixed style: some components are in hand print style, whereas other components are written in cursive style. A good handwriting recognition system must solve all these problems. The intended word was *sweatshirt*.

RECOGNITION

Before proceeding with the description of the actual recognition methods, the object of the recognition must be defined. It is important to note that a number of categories in handwriting recognition are meaningless to the user of a pen computer but are extremely important to the developer of a recognition system. In recognition practice, there are classifiers of isolated characters, and there are classifiers of whole words, or even of sentences or paragraphs. In the UNIPEN project (33) for benchmarking handwriting recognition systems, the following categories have been defined in Table 2.

Benchmark number 8 refers to the most difficult case of unsegmented free text. The category of punctuations is much neglected in current research and not trivial. Similarly, the diacritics that are very important in some languages are an underexposed research topic. The categories described earlier fit very well in the framework of handwriting recognition research. On the other hand, consider the taxonomy from the point of view of the pen user interface (PUI) design (112). The following taxonomy has been proposed in Table 3.

Current efforts in online handwriting recognition research are largely focused on the handwriting shape categories, without taking into consideration the problems encountered in user-interface design. One of the most difficult issues appears to be the disambiguation between the categories: Does a small vertical line refer to a letter *i*, a letter *l*, a digit *1*? Is it the beginning of a line which should be beautified into a straight line? Or is it just intended to be sketched ink? It is very difficult to detect these categories automatically, and it is very cumbersome for the user to identify the categories, because users are now generally accustomed to modeless user interfaces. The use of gestures does not automatically solve the problem fundamentally because gestures themselves may have an ambiguous shape when compared with other shape categories present in a system. Also, it should be noted that

Table 3. A Taxonomy of Pen-Based Input

- [1] Textual data input
  - [1.1] Conversion to ASCII
    - [1.1.1] Free text entry
      - [1.1.1.1] Fully unconstrained (size, orientation, styles) (e.g., PostIts)
      - [1.1.1.2] Lineated form, no prompting, free order of actions
      - [1.1.1.3] Prompted
        - [1.1.1.3.1] Acknowledge by “OK” dialog box
        - [1.1.1.3.2] Acknowledge by Time-out (e.g., 800 ms)
        - [1.1.1.3.3] Acknowledge by Gesture (see item 2.2)
    - [1.1.2] Boxed forms
    - [1.1.3] Virtual keyboard
  - [1.2] Graphical text storage (plain handwritten ink)
- [2] Command entry
  - [2.1] Widget selection
  - [2.2] Drag-and-drop operations
  - [2.3] Pen gestures
    - [2.3.1] Position-independent gestures
    - [2.3.2] Position-dependent context gestures
  - [2.4] Continuous control (e.g., sliders, ink thickness by pressure)
- [3] Graphical pattern input
  - [3.1] Free-style drawings
  - [3.2] Flow charts and schematics
  - [3.3] Mathematical symbols
  - [3.4] Music scores
- [4] Signature verification

[a]From Ref. 112.

the recognition of handwritten text [item 1.1] is just one of the forms of pen-based input, and not the necessarily the essence of all pen-based applications.

Classifiers

Having noted this, we now identify the most common types of handwriting classifiers:

1. Isolated-characters recognizers
2. Word recognizers
   a. Character-based word recognizers
   b. Whole-word classifiers

Isolated Character Classifiers. In the character classifier, it is assumed that the basic input ink object is a single character or gesture. Note that, for the word category, we have used the term *recognizer* instead of *classifier,* because a word-based system will generally be much more complex than a system for the classification of isolated characters. A typical word-based recognizer also has to solve the problem of segmentation into characters (it will often have more than one embedded character classifier), and it will have modules that address the incorporation of linguistic, usually lexical top-down information. Within the class of word recognizers, we can make a distinction between systems that try to identify the individual characters within a word and the holistic word classifiers, which do not try to find characters but are based on word-level shape features, such as word contour.

Shape Classification. Shape classification is the process by which given raw samples of a handwritten trajectory are identified as belonging to either a known or a more abstract class of similar shapes. In this respect, shape can refer to the shape of whole words, of characters, of glyphs (multicharacter shapes, possibly consisting of fused characters), or even of the shape of individual strokes. Therefore, the design of an online handwriting classifier starts with the object definition: what is the target object of the classification process? As described earlier, this object is currently either the individual character or the whole word in most existing recognizers of online handwriting. Regardless of the object definition, there are two distinct classification methods in online handwriting recognition, which can be characterized as

1. (I) formal structural and rule-based methods and
2. (II) statistical classification methods.

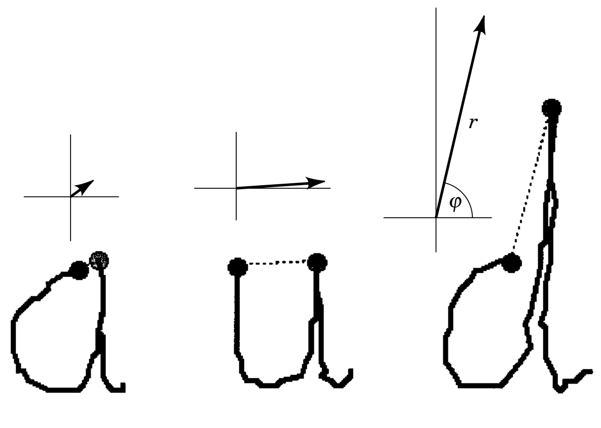
The statistical classification methods can be further divided into explicit-statistical and implicit-statistical methods. In explicit-statistical methods, knowledge of the statistical properties of the data is modeled and used, under a number of assumptions such as distribution shapes and independence of the variables involved. The implicit-statistical classifiers are the artificial neural networks (ANN) for pattern classification. The performance of these latter classifiers is strongly influenced by the statistical properties of the training data, but no explicit statistical modeling is being used in ANNs. A separate and special variant of the statistical classifiers is the Hidden Markov Modeling method.

Structural, Rule-Based Classification Methods

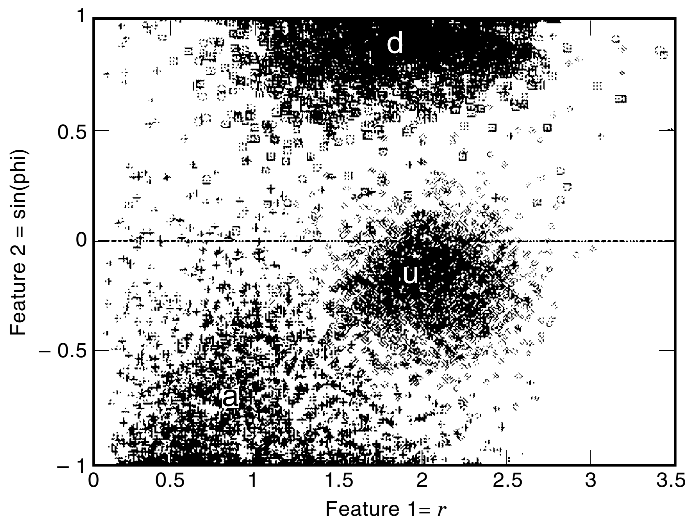
Structural rule-based or shape-syntactical approaches are based on the idea that character shapes can be formally described in an abstract fashion, not paying attention to irrelevant variations in shape. This rule-based approach originates in the 1960s and has been abandoned to a large extent because it has appeared to be extremely difficult to formulate general rules that can be determined in a reliably way. As an example consider the statement "all lowercase *t*s consist of a vertical elongated ink component that is crossed by a horizontal but shorter elongated ink component." We need the attribute "shorter" to disambiguate between + (plus) and *t*. There may be writers who violate this rule in a number of ways. The horizontal bar may be slanted, or it may even be located above the vertical bar for a group of writers. Additionally, the attributes that are of a symbolic nature must be accompanied by metric or geometric constraints, usually in the form of one or more scalars. For example, the symbolic attribute "shorter" from the preceding example may require an additional scalar constraint: "the length of the horizontal bar of a *t* is no longer than 60% of the length of the vertical bar." Using this approach leads to brittle recognition systems with a large number of parameters and thresholds that are difficult to determine by means of automatic methods. More recent approaches within this area of handwriting recognition classification methods therefore incorporates fuzzy rules and grammars (113,114) that use statistical information on the frequency of occurrence of particular features. Although the rule-based approach often has not produced very impressive results, when applied in isolation, the fact remains that if solid rules can be defined and if their conditions can be determined reliably, then useful decisions can be taken by such a recognizer. A particular advantage of a rule-based approach is that it does not require large amounts of training data. However, a more conspicuous advantage of the rule-based approach is that the number of features used to describe a class of patterns may be variable for each class, a property lacking in the statistical approaches presented in the next section. And, what is more, a single particular feature may be used to disambiguate between two particular classes. As an example, the feature top-openness [feature *r* in Fig. 11(a)] is noninformative in most contexts along the handwriting trace, but may be useful to disambiguate between the cursive *a* and cursive *u* without clouding the classification process in cases where this feature is irrelevant.

Statistical Classification Methods

In statistical approaches, the handwriting shapes (stroke, character, or word, usually) are described by a fixed and a limited number of features, which are designed to create a multidimensional space, in which the different classes are present in the form of a class vector or centroid, with a multidimensional probability distribution around it. Observation of instances of a class must be close to the centroid of that class and as far as possible away from the other classes. Another way of looking at this problem is to realize that there must be a watershed boundary (or separatrix) between two classes in feature space. In Fig. 11, for example, two features are calculated for three letters (*a*, *u*, and *d*). Feature 1 reflects the distance (*r*) between the first point of the letter and the last vertical maximum produced in the pen-tip trajectory. Feature



(a)



(b)

Figure 11. Illustrations of the statistical approach to letter classification in online handwriting recognition.

2 is the sine of the angle (phi) of the same vector [Fig. 11(a)]. There were 2000 instances of each letter, taken from the UNIPEN database. A scatter plot of the two features for the three letters (*a*, *u*, and *d*) shows that simple boundary functions may be found in planes, separating the classes. However, at the same time, it is evident that there will be some overlap such that a perfect classification will not be possible using these two features alone. A limiting characteristic of this approach is the fact that the dimensionality of the feature vector is fixed, for all classes. This means that a feature that is helpful in discriminating between two classes *a* and *u* in Fig. 11(b) may act as noise in the comparison of class *d* versus *u*.

The following three variants of statistical approaches exist:

1. Explicit statistical approaches,
2. Implicit statistical approaches (neural networks),
3. Markov modeling and Hidden Markov Modeling.

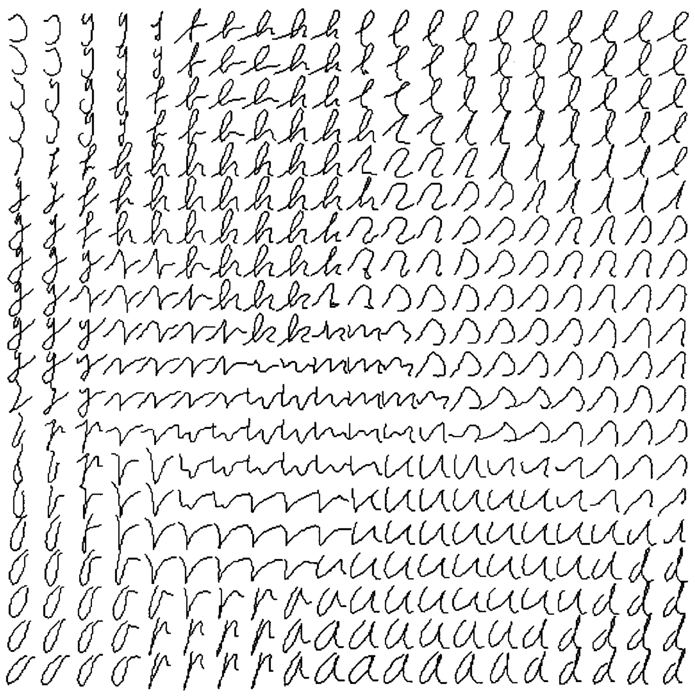
Explicit Statistical Approaches. Explicit statistical approaches in online handwriting recognition use the knowledge and tools which have been developed in the research field of Data Analysis and Methods and in theoretical Pattern Recognition. There are many commercial software packages that allow for the analysis and classification of high-dimensional stochastic data, such as feature vectors derived from online handwriting signals. Examples are Linear Discriminant Analysis, which allows for finding linear separatrices between two classes (of, e.g., characters); Hierarchical Cluster Analysis, which allows for a tree-oriented representation of the data structure; and Principal Components Analysis, which allows for the detection of a number of orthogonal axes in the high-dimensional feature space which are sufficient to describe the variance in the data. The problem with the explicit statistical approaches is that there are often assumptions about the statistical distribution of the features that cannot be held, either because the corresponding theoretical distribution is not known or because it is known but is not allowed in a given classification approach. Most approaches assume Gaussian distribution of the features. The relative importance of different features is determined by their average values, for which reason scaling is needed. There are well-founded approaches that try to transform the features in such a way that the class distributions are spherical, using the inverse of the covariance matrix (115). A practical problem in online handwriting may be that the available software cannot handle the high dimensionality of the feature vectors and the enormous size of the data sets. The advantage of the explicit statistical approaches is that they have a theoretical basis. The method and its results can be explained easily to colleagues in the field or to the users of the recognition system (for example, because performance aspects can be reported in the form of real probabilities). A disadvantage of the explicit statistical approach is that it is sometimes difficult to translate the method to real working practical systems, which are often small hand-held computers with limited computing and memory resources.

Implicit Statistical Approaches. The term Artificial Neural Networks refers to a class of algorithms for which the design of the computational architecture and the process involved is inspired by knowledge of information processing in the physiological brain. Information is processed by a large number of simple processing elements that exchange information in the form of activation levels. An activation level is usually a real-valued parameter, reflecting the state of a neuron or unit, similar to the physiological firing rate of a biological neuron measured at a given time instance. Usually, the artificial neural networks are not considered to be statistical classifiers. This is only partly correct. The ANN method is indeed not statistical in the sense that it is based on theoretical probability density functions. However, the ANN method is statistical in the sense that the classification behavior of an ANN system will be fully determined by the statistical characteristics of the data set that has been used to train the system. The relationship between ANNs and statistical approaches is described in detail by Bishop (116).

A well-known class of ANN approaches consists of the Multi-Layer Perceptron (MLP) usually trained by the method of error Back Propagation (BP) (117). The training of such networks is supervised, meaning that the classes of shapes (e.g., all digits 0–9) are well defined. The input features are mapped onto the input units of the MLP, and the output units of the network are defined to represent a particular class of characters (e.g., the digit 4). A training set is presented several times to the network until a satisfying rms error between target and obtained output activations has been achieved. Subsequently, the classification performance of the MLP is tested against a new and unseen test set. For a limited symbol set as in the case of digits, this approach may be useful, provided that suitable features are being used in the input representation. The MLP is able to find a nonlinear separatrix between classes, if one or more hidden layers are used. A straightforward approach, for example, would entail the use of an $N \times M \times 26$ unit architecture with N input units for the online handwriting features, M hidden units, and 26 output units for the letters of the alphabet. However, there are a number of problems with this "naive" approach to handwriting recognition.

1. Symbolic categories such as the letters in the Western alphabet may encompass a very large range of shape classes, especially if a large population of writers has contributed to the training set. What works for digits does not by necessity deliver good results for other character classes where the number of shape variants is large.
2. Problems of geometric invariance are still present. Even if a geometric normalization stage is present in the preprocessing of the characters, a substantial amount of variation will still be present.
3. The generalization from the recognition of isolated shapes to the recognition of connected-cursive script is difficult.

A number of solutions has been proposed to solve these problems. The problem of allographic variation (many shapes refer to a single letter symbol) can be solved by using techniques that are based on automatized detection of prototypical shapes in a large training set of character samples. A common "neural" technique for automatic allograph class identification is the Kohonen (118) Self-Organized Feature Map (SOFM). Similar approaches make use of k-means clustering or hierarchical clustering. The result of the application of such techniques is a finite list of allographs (character prototypes), that can be used in several ways as the basis for a classification system, such as the MLP. Figure 12 shows an example of a Kohonen SOFM. In Fig. 12(a), within a predetermined planar map of 20×20 cells, the algorithms has found a place for prototypical character shapes, already at the initial stages of training, as shown here. More elaborate training on a larger dataset will ultimately reveal detailed class shapes for all the characters, provided that their distribution in the training set is equalized. Figure 12(b) depicts a SOFM in which the individual cells represent velocity-based strokes. Stroke features mainly consisted of angles along the trace. The heavy black dots denote stroke endings. White strokes are pen-up movements, which are also recorded and presented to the Kohonen network. The vector-quantization properties of the Kohonen SOFM can be used in subsequent character recognition stages by mapping the shape codes to their possible interpretations in the language. For example, in characters, the cursive and looped e shape may be interpreted as representing both the 1 and the e interpretation. In the case of strokes, which may occur in different characters, vari-


(a)

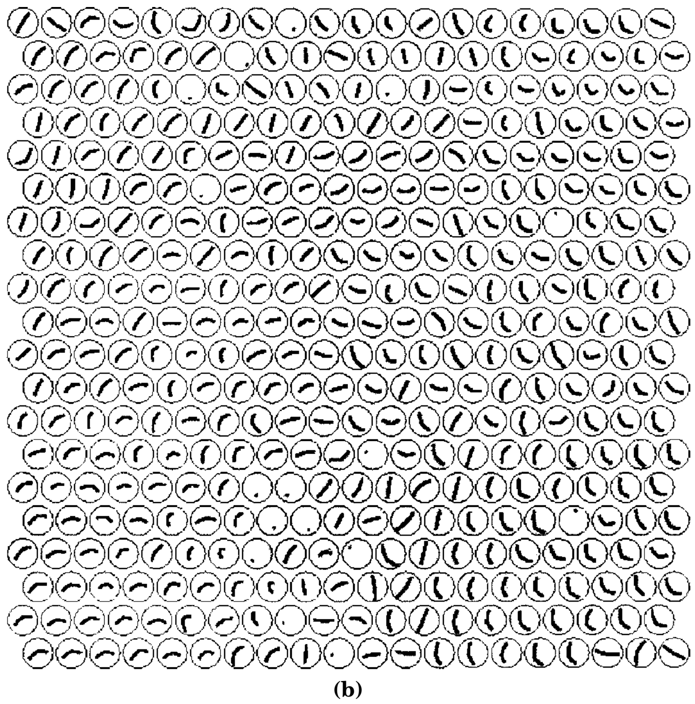

(b)

Figure 12. (a) A Kohonen SOFM with characters after unsupervised training on 4000 characters. Features consisted of 30 (x, y, z) coordinates per character. (b) A SOFM in which the individuals cells (20×20) represent velocity-based strokes and are based on about 40,000 strokes of 40 writers.

ous interpretations are possible for a single stroke out of context.

Problems of sensitivity to scale and translation dependence in space and time were solved by a particular type of MLP, which exploits the notion of convolution kernels for digital filtering. This technique (i.e., convolutional Time-Delay Neu-

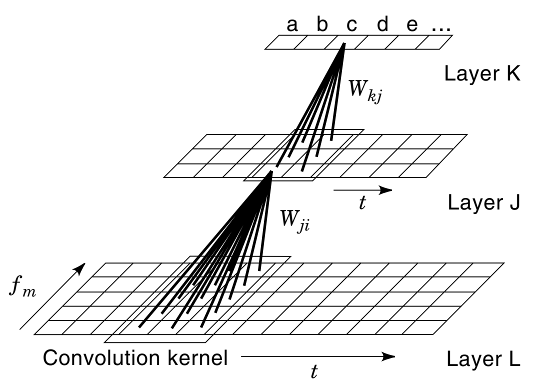


Figure 13. An example of a convolutional Time-Delay Neural Network (TDNN) architecture, as applied to the recognition of online handwriting.

ral Networks), is not based on fully interconnected layers as is the standard MLP but uses many fewer connections (W_{ij}) by defining a limited-size receptive field for units in hidden layers. Figure 13 gives a schematic overview of this architecture. The convolutional TDNN (cTDNN) can be used both in spatial representations for optical character recognition, as in space–time representations for the online recorded handwriting signal (119,120). In Fig. 13, the input buffer (I) contains the time-varying features values f_m. Each hidden unit in the hidden layer (J) can be viewed as a convolution kernel, with a receptive field that is swept over the complete input layer during the training of an individual character. The same convolution approach may be used for the following layers. In this example, layer K constitutes the final character classification stage. The convolutional TDNN approach allows for the automatic detection of time- and position-invariant features, whereas saving a considerable number of parameters (weights W) as compared to a static and fully interconnected multilayer perceptron.

The strength of the cTDNN approach is mainly due to the fact that a useful degree of invariance to spatial and temporal variations in input can be obtained. At the end of a training session, the hidden units often will represent meaningful position- and time-independent features. If the position of features in spatiotemporal zones reflects information that is useful to the classification, the coverage area of the convolution kernel of a hidden or output unit may be constrained within the total input field in order to retain this translation dependency of some features. The use of the cTDNN architecture entails a substantial reduction in the number of free parameters as compared to a standard MLP. However, the number of hidden layers in a cTDNN will be usually somewhat larger than in the case of a standard MLP. At the level of single characters, the cTDNN is able to compensate for temporal and spatial variabilities in the pen-tip trajectory. However, the cTDNN approach does not solve an inherent problem of all types of TDNNs, which is the fixed time-window. Especially when making the step from single characters to whole words, it becomes evident that the problem of varying-duration patterns has not been solved completely.

As in the automatic recognition of speech, an automatic recognizer of online handwriting must be able to deal with patterns of broadly varying duration. As an example, a word-based recognizer may have to represent the English word "a," as well as the word "dermatomucosomyositides." Such a wide range of sequence lengths cannot be handled gracefully by a cTDNN as such, and methods involving some form of state recursion will provide a solution. In the area of online handwriting recognition, Recurrent Neural Networks (RNN) have been applied occasionally, as well as in speech recognition. Recurrent Neural Networks are a special form of multilayer perceptron, in which a portion of the system state at processing time step k is reentered into the network at processing $k + 1$. Recursive connections may run from output units to hidden or input units, and/or individual neurons may possess self-recurrent connections. The RNN approach requires a large amount of training and training data. The following class of statistical classifiers yields considerable success in speech recognition and is currently successfully applied to the recognition of online handwriting: the Hidden Markov Models (HMM), which appear to be a better solution to the problem of variable sequence duration.

Hidden Markov Models. Hidden Markov Models refer to a particular method of Markov modeling (121–125) where the idea of transition probabilities between symbols in a language is abstracted from the observation of symbols as such. Instead, a number of states is defined for the handwriting object of interest (character, word, sentence), in which either the occurrence of a set of symbols is lumped (discrete HMM), or the occurrence of particular real-valued feature values is lumped together as being characteristic for a particular state. Thus, instead of a Markov model for observable stochastic symbol strings, HMM assumes several unobservable hidden states. The first stage in designing a HMM recognizer of handwriting is the design of the state machine that captures all possible and relevant transitions between the assumed hidden states. It has been shown that the HMM is a more general approach for an early dynamic programming approach (Dynamic Time Warping, DTW). Instead of different templates for a particular class, as in the DTW approach, a HMM aims at representing all stochastic variations within a class within a single model per class.

Therefore, the model must represent all possible forms of state transition: Simple continuation from state S_i to state S_{i+1}, dwelling (from S_i to S_i itself), and skipping (from state S_i to S_{i+k}). A particular example of an HMM transition model is given in Fig. 14. The model is designed to capture all sto-

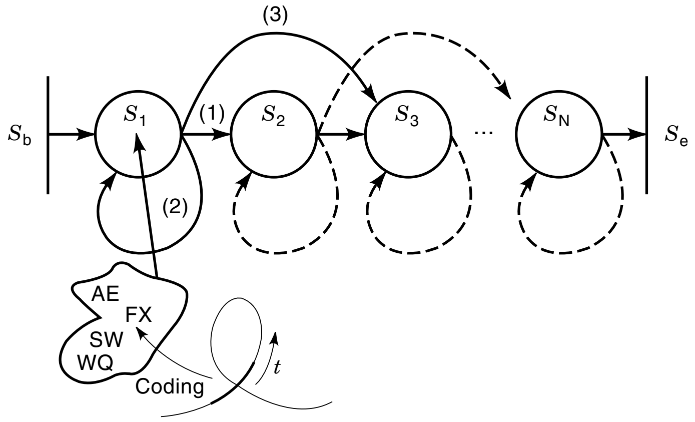


Figure 14. An example of a Hidden Markov Model describing the character *e*.

chastic variation of a particular class of words or characters. A distinction is made between a number of states S_i occurring between an initial state S_b and a final absorbing state S_e. This particular model example allows for the following transition types between states: Type (1) concerns the normal time progression of states, type (2) is a self-recurrent transition that models durational variability of a particular state, type (3) in this example allows for the occasional omission of one single state, during the sequence. This is an example of a discrete HMM of the character *e*. Feature vectors are obtained from a sliding time window and subsequently quantized and represented by discrete codes of type LL in this example. These codes will have a probability distribution over the whole letter, differing for the different states.

Contrary to explicit (surface) Markov models of transition probabilities in symbol strings, the states in HMMs are containers for sets of observable symbols or events. For the modeling of online handwriting, usually a sliding time window along the pen-tip trajectory is used as the basis for the input to a HMM recognizer. From the *XY* coordinates of a time window *W* of duration *T* occurring at time *t*, feature values are calculated. A successful approach uses a cTDNN for feature vector calculation (126).

The HMM approach can be based on two different event models: discrete symbol observations (and their transition statistics) or continuous models. The application of the discrete HMM approach requires that the input feature vectors are converted into discrete symbols by using a vector quantization (VQ) algorithm. The occurrence probabilities of these symbols for the stroke shapes in sliding window *W* are then used as the basis for the HMM algorithm (Fig. 14). The second approach, continuous HMM modeling, uses the variances and covariances of the features to estimate the probability of occurrence of an observed feature vector (e.g., under the assumption of Gaussian feature distribution). Using Bayes' theorem, the goal of the HMM algorithm then is to find for each word *C* in a lexicon (or for each character in an alphabet, depending on the type of classifier), the probability that class *C* is the most likely class, given a sequence of observations $O_1, O_2, \ldots, O_n$:

$$P(C|O) = P(O|C) \cdot P(C)/P(O) \tag{5}$$

where $P(C|O)$ is the probability that *C* is correct, given sequence of observations *O*, $P(O|C)$ is the probability that class *C* elicits sequence *O*, $P(C)$ is the probability of occurrence of class *C* in the real world (e.g., in particular language), and $P(O)$ is the probability that any class *C* elicits sequence *O* to be emitted from a combined system that consists of the writer and the feature computation algorithm. This process is called the a posteriori probability calculation. The essence of the HMM approach is thus to determine the a posteriori probability for a class, given an observed sequence, by using the probability that the HMM state machine jumps to a particular state S_j at a given time *t*, when S_i was the previous state and observations $O_1, O_2, \ldots, O_t$ have just been made. The transitions from the beginning S_b to S_1 and the transition from the last state S_N to the end S_e are treated slightly differently, but the core of the so-called forward algorithm consists of the following recursive equation:

$$P(S_j, t) = \sum_{i=1}^{N} P(S_i, t-1) \cdot p(i, O_t) \cdot a_{ij} \tag{6}$$

where $P(S_j, t)$ will correspond to $P(C, O)$ when *t* is at the end of the sequence $P(S_i, t-1)$ has a similar meaning and corresponds to the previous time step, $p(i, O_t)$ is the probability of O_t being produced in state S_i, and finally and most importantly a_{ij} is the probability of transition from state S_i to state S_j. The estimation of a_{ij} constitutes a central and often difficult aspect of the HMM approach. The advantage of the HMM approach is that the number of states can be much smaller than the number of symbols in the observed sequences, as long as sufficient information is retained to identify the model corresponding to the correct class. Instead of calculating the state probabilities for all classes (e.g., words) separately and then finding the model with the highest probability, a best-first search strategy can be used, saving considerable time. A disadvantage of the HMM approach is that it requires a large amount of training data to ensure reliable estimates of the probabilities involved. Historically, the success of HMM in speech recognition could not be replicated quickly in the area of online handwriting recognition until good models and feature representations were developed.

No single classifier approach will be suitable to all classes of online handwritten input. Therefore, classifier combination schemes are needed in a realistic high-performance system. But also within a class of input shapes (digits, words) there will be subclasses for which a particular recognizer design is more suitable than another. Multiple-expert combination approaches have been developed, using a multitude of information integration algorithms including such techniques as Bayes, Dempster-Shafer, Borda count, and linear combination. Some authors conceptualize a system consisting of several expert classifier algorithms as a new meta classifier (127–129).

For any multiple classifier scheme, the performance improvement will be best if the classifiers are independent (i.e., based on different signal models, feature schemes, and classification algorithms). Finally, it should be noted that the recognition of isolated handwritten words is not perfect in human reading (110,111,130).

In Table 4, results of four small human reading experiments are shown. Experiment A required single and isolated word recognition of neatly written and well-known words. In experiments B–D, three-word sequences were presented, the middle word had to be recognized, and flanking words were linguistically related or unrelated (C) or linguistically unrelated but from the same or a different writer. From these findings, it can be observed that human recognition rate is sensitive to handwriting style and context. This context can be linguistic (Table 4, conditions B and C) or based on neighboring shapes (Table 4, condition D). If linguistic context and shape context are absent or noninformative, human recognition performance can be rather low. In the next section, we will address methods in online handwriting recognition that try to include the use of higher-level information in the recognition process, in order to improve on the raw, shape-based classifier performances.

POSTPROCESSING

Because of the variability in handwriting styles and distortions caused by the digitizing process, even the best handwritten word recognizer is unreliable when the number of

Table 4. Human Recognition Rates of Handwritten Words on Paper

Experiment	Context	Style	Writers	Target Words	Readers	Words Recognized
A	Frequently used words	Hand print	1	30	12	N = 360 98%
	Frequently used words	Neat cursive	1	30	12	N = 360 88%
B	Sentence fragments, same writer	Cursive	3	15	12	N = 180 85%
C	Sentence fragments, same writer	Cursive	4	13	20	N = 260 85%
	Unrelated words, same writer	Cursive	4	13	20	N = 260 77%
D	Unrelated words, same writer	Fast cursive	12	12	15	N = 180 72%
	Unrelated words, different writers	Fast cursive	12	12	15	N = 180 54%

word choices is large. Systems that perform word-based recognition require a predefined lexicon. Differenti sets of features may be extracted from the segmented word unit to reduce the size of this lexicon (to speed the later recognition process). Word shapes are often used to serve this purpose (131–134). Generally speaking, it is clear that the key to improving raw handwriting recognizer (HR) output is to use contextual knowledge such as linguistic constraints (which employ phrase and sentence-level context) to achieve a performance level comparable to that of humans (135–137). This process is referred to as *language modeling.*

Language modeling for handwriting recognizer systems borrows heavily from the techniques used to postprocess speech recognition and optical character recognition (OCR) systems. However, some adaptation of these models to the handwriting medium is required. Furthermore, effective design and use of training corpora and lexicons is necessary because language employed in handwritten text tends to be informal and ungrammatical.

As an example, consider Fig. 15, which shows the digitized image of the sentence "My alarm clock did not wake me up this morning" along with the output of the handwriting recognizer. If we restrict the recognizer to return only the top choice for each input word, the sentence will be erroneously read as "my alarm code soil rout wake me up thai moving." Although more correct words can be found in the top n choices, it requires the use of contextual constraints in order to override the (sometimes erroneous) top choice. This example also illustrates the tendency of word recognizers to misread short (three characters or fewer) words more frequently than longer words. Furthermore, short words tend to be pronouns, prepositions, and determiners causing frequent word confusion between very different syntactic categories (e.g., as, an). The handwriting recognition problem is further compounded by the possible absence of the correct word in the top n choices.

Language modeling for handwriting recognition systems can consist of several phases. The process of matching character recognizer output against a lexicon of words, a process referred to as *string matching,* could be considered the first phase. Following this, various linguistic models, ranging from simple word n-grams to syntax and deeper semantic models may be employed (138,139). Hybrid models that combine both linguistic knowledge with recognizer confidences have also been proposed. Performance metrics show the utility of accurate language models by measuring recognition accuracy as a function of the amount of processing required. Srihari (135,140) reports on earlier efforts toward achieving these goals.

String Matching

Since the candidate words produced by the word recognition phase may not be legal words, a *string matching* procedure between candidate words and the lexicon is required. Kukich (141) contains a detailed discussion of isolated-word error correction techniques. String matching algorithms rank the lexicon according to a distance metric between a candidate word and each word in the lexicon. The ranked lexicon consists of sets of words $n_1, n_2, \ldots, n_t$ such that all the words in a set n_i have the same value for the distance metric. From the ranked lexicon, word recognition systems typically select a neighbourhood N of words that (hopefully) contain the intended word.

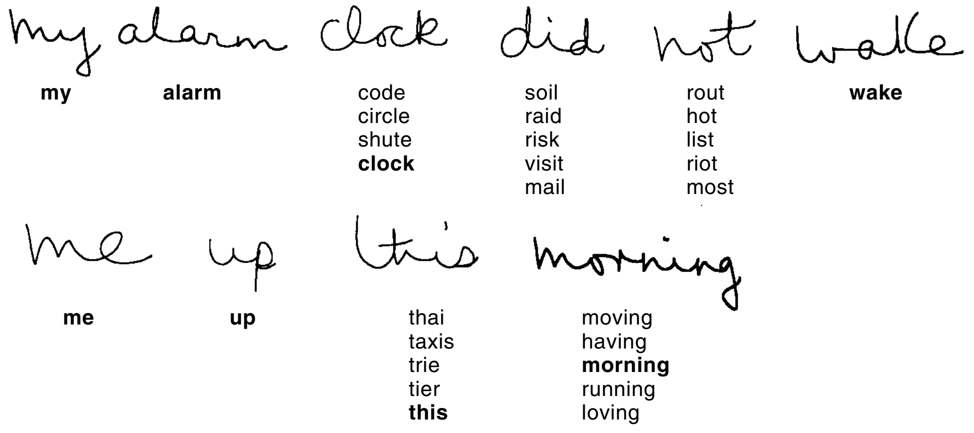


Figure 15. Isolated word recognition output. Correct words are shown in bold; italicized lists indicate that correct word is not among top choices.

Typically, string-matching algorithms use for string edit distance some variation of the Levenshtein distance metric, which incorporates edit costs in ranking the lexicon. Traditionally, edit costs have been associated with the most common edit errors encountered in typographical systems, namely, substitution, insertion, and deletion. Wagner and Fischer (142) discuss a dynamic programming algorithm to compute string edit distances.

Edit costs in handwriting are based on the probabilities of substitutions, merges, and splits occurring. Although it is possible to model these as combinations of substitution, insertion, and deletion operations, some researchers have attempted to increase the accuracy of string matching by explicitly modeling splitting (e.g., *u* being recognized as *ii*) and merging of characters (e.g., *cl* being recognized as *d*) as distinct edit operations (143). Furthermore, the errors have been quantized into one of three categories, and weights have been determined for each type of error. Errors associated with misinterpretation of ascenders or descenders are given the highest weight (i.e., most costly). The modifications do not adversely affect the complexity of the algorithm. Finally, recent work (144) has also focused on learning weights for various edit costs based on machine learning techniques; given a corpus of examples reflecting output from a recognizer, such methods learn the optimal string edit distance function.

These methods take into account channel information only. String matching can be improved by incorporating dictionary statistics such as the frequency of letter transitions, words, in the training data. Good (145) as well as Gale and Church (146) discuss some techniques (and possible pitfalls) for accomplishing this task. String matching produces a ranking of the lexicon for each word position; thus, an input handwritten sentence or phrase results in a lattice of word choices. After completing the string matching process, language modeling can be applied to select the best sentence or phrase among the lattice of word choices.

Language Models and Perplexity

Language models are used in recovering strings of words after they have been passed through a noisy channel. The "goodness" of a language model as applied to a given corpus can be measured in terms of the corresponding *perplexity* of the text. Intuitively, perplexity measures the average number of successor words that can be predicted for each word in the text (based on a language model). As an example, if the language model utilizes only first-order transitions between successive words (bigrams), the perplexity P of a corpus with T words is

$$P = \exp\left(-\frac{1}{T}\sum_{t=1}^{T}\ln(p(w_t|w_{t-1}))\right) \tag{7}$$

where $p(w_t|w_{t-1})$ is the probability to get the word w_t given the word w_{t-1} (based on this corpus). In the worst case, the perplexity is the same as the number of words in the corpus. Using higher values of n will lower the perplexity; however, estimating the conditional probabilities $P(w_k|w_{k-1})$ becomes a problem. Brown et al. (147) describe a language model as a computational mechanism for obtaining these conditional probabilities. In general, the more powerful the language model is, the more computationally intensive it will be. It is crucial that the text used in training is representative of what the model is intended for.

Statistical models are the most common language model used in postprocessing the output of noisy channels (speech, OCR, handwriting). Such models reflect both syntactic and semantic knowledge, albeit, at a very local level (e.g., a neighborhood of three words). The advantage of such models is their robustness, as well as the ability to train them on specific corpora. Because these are typically modeled as Hidden Markov Models, depending on the recognition technique being employed, it is possible to combine postprocessing with recognition; Schenkel et al. (148) describe such work.

Context-free grammars model language by capturing the syntax of all possible input sentences and reject those sentence possibilities not accepted by the grammar. The problems with such a method are (1) the inability of the grammar to cover all possibilities (especially because informal language is frequently ungrammatical) and (2) the computational expense involved in parsing (increased due to several word choices). For these reasons, context-free grammars have not seen any significant use in postprocessing handwriting recognition output. Stochastic context-free grammar's (149) alleviate the first problem somewhat because techniques have been developed for automatically constructing context-free grammar's from large training corpora.

N-Gram Word Models. The performance of a handwriting recognition system can be improved by incorporating statistical information at the word sequence level. The performance improvement derives from selection of lower-rank words from the handwriting recognition output when the surrounding context indicates such selection makes the entire sentence more probable. Given a set of output words $\overline{X}$ that emanate from a noisy channel (such as a handwriting recognition), n-gram word models (150,151) seek to determine the string of words $\overline{W}$ that most probably gave rise to it. This amounts to finding the string $\overline{W}$ for which the a posteriori probability

$$P = (\overline{W}|\overline{X}) = \frac{P(\overline{W})^{*}P(\overline{X}|\overline{W})}{P(\overline{X})} \tag{8}$$

is maximum, where $P(\overline{X}|\overline{W})$ is the probability of observing $\overline{X}$ when $\overline{W}$ is the true word sequence, $P(\overline{W})$ is the a priori probability of $\overline{W}$, and $P(\overline{X})$ is the probability of string $\overline{X}$. The values for each of the $P(X_i|W_i)$ are known as the channel (or confusion) probabilities and can be estimated empirically. If we assume that words are generated by an nth order Markov source, then the a priori probability $P(\overline{W})$ can be estimated as

$$P = (\overline{W}) = P(W_{m+1}|W_{m+1-n})\ldots P(W_1|W_0)^{*}P(W_0) \tag{9}$$

where $P(W_n|W_{k-n}\ .\ .\ .\ W_{k-1})$ is called the *nth-order transitional probability.* The *Viterbi algorithm* (152) is a dynamic method of finding optimal solutions to the quantity in Eq. (9).

Guyon and Pereira (136) discuss a technique for effectively using character n-grams. It is interesting to note that in the early days of OCR systems, character n-grams were the only viable technique for postprocessing because memory and speed restrictions did not permit the use of higher-level word models. As these restrictions relaxed, word n-grams have seen widespread use. However, Guyon and Pereira (136) report on techniques for combining several character n-grams effectively; the resulting model is almost as powerful as a word n-gram model. More importantly, the composite model satisfies various requirements of online handwriting recognition systems such as (1) ability to predict the next character, thus providing user with immediate feedback; (2) compactness; (3) rapid customization; and (4) ability to generalize to words not present in the training data. The composite consists of a generic module representing statistics about general English; this is combined with specialized models for capitalization, number strings, symbols, and proper nouns.

The problem with word n-gram models is that as the number of words grows in the vocabulary, estimating the parameters reliably becomes difficult (151). More specifically, the number of low or zero-valued entries in the transition matrix starts to rise exponentially.

Rather than using raw frequency data as estimates of probabilities, several alternatives that improve the approximation have been proposed. This includes back-off models as well as maximum entropy techniques.

Smoothing, Back-Off Models. Most standard language models are based on the bigram or trigram model. Even if very large corpora are used, it will contain only a fraction of the possible trigrams; hence, it is necessary to *smooth* the statistical data in order to provide better estimates of the unseen events. The *maximum likelihood* estimate for the probability of an event E that occurs r times out of a possible R is $P(E) = r/R$. Smoothing techniques involve distributing the probability mass so that unseen events are also covered. Typically, this is achieved by various *discounting* strategies. Instead of using r, r^* is used, where $r^* = rd_r$ and d_r is a chosen discounting function. The Good–Turing discounting strategy (145) is frequently used.

Back-off models refer to the process of using lower-order statistical models to estimate higher-order models. For example, if $P(w_1, w_2, \ldots, w_n)$ has not been observed, the conditional probability $P(w_n|w_1, w_2, \ldots w_{n-1})$ can be estimated from $P(w_n|w_2, w_3, \ldots w_{n-1})$. Very often, smoothing and back-off techniques are combined. Clarkson and Rosenfeld (153) describe a toolkit for statistical language modeling that implements various smoothing and back-off strategies.

Maximum Entropy Methods. The Maximum Entropy (ME) principle has been successfully employed as a powerful technique for combining statistical estimates from various sources (154). The ME principle calls for the reformulation of the various statistical estimates (e.g., bigrams, trigrams, quadgrams) as constraints; among all the probability distributions that satisfy the constraints, the one with the highest entropy is chosen. A unique ME solution is guaranteed to exist and takes the form

$$P(x) = \prod_i u_i^{f_i(x)} \tag{10}$$

where the u_is are some unknown constants (to be found), and the f_is are constraint functions associated with each constraint i. An iterative algorithm for searching the solution space for Eq. (10), known as Generalized Iterative Scaling, exists; this is guaranteed to converge to a solution if it exists. Language models based on Maximum Entropy have been demonstrated to outperform language models using traditional back-off techniques.

***N*-Gram Class Models.** In n-gram class models, words are mapped into syntactic (155,156) (or semantic classes, discussed in the next section). In this situation, $p(w_t|w_{t-1})$ becomes

$$p(w_t|w_{t-1}) = p[w_t|C(w_t)]p[C(w_t)|C(w_{t-1})] \tag{11}$$

where $p[C(w_t)|C(w_{t-1})]$ is the probability to get to the class $C(w_t)$ following the class $C(w_{t-1})$ and $p[w_t|C(w_t)]$ is the probability to get the word w_t among the words of the class $C(w_t)$.

The following example illustrates the use of n-gram class models by using part-of-speech (POS) tags as the method of classifying words. The notation $A:B$ is used to indicate the case where word A has been assigned the tag B. For each sentence analyzed, a word:tag lattice which represents all possible sentences for the set of word choices output by string matching (see Fig. 16) is formed. The problem is to find the best path(s) through this lattice. Computation of the best path requires the following information: (1) tag transition statistics and (2) word probabilities.

Word Lattice

Transition probabilities describe the likelihood of a tag following some preceding (sequence of) tag(s). These statistics are calculated during training as

$$P(tag_B|tag_A) = \frac{\#(tag_A \to tag_B)}{\#(tag_A)} \tag{12}$$

Beginning- and end-of-sentence markers are incorporated as tags themselves to obtain necessary sentence-level information.

Word probabilities are defined (and calculated during training) as

$$P(Word|Tag) = \frac{\#(Word : Tag)}{\#(AnyWord : Tag)} \tag{13}$$

The Viterbi algorithm is used to find the best $Word:Tag$ sequence through the lattice, that is, the maximal value of the following quantity:

$$\prod_{i=1}^{n} P(Word_i|Tag_i)P(Tag_i|Tag_{i-1}) \tag{14}$$

over all possible tag sequences $T = Tag_0, Tag_1, \ldots, Tag_{n+1}$ where Tag_0 and Tag_{n+1} are the beginning- and end-of-sentence tags, respectively. The Viterbi algorithm allows the best path to be selected without explicitly enumerating all possible tag sequences. A modification to this algorithm produces the best n sequences.

The lattice of Fig. 16 demonstrates this procedure being used to derive the correct tag sequence even when the correct word ("the") was not output by the handwriting recognition. The chosen path is illustrated in boldface. The values on the edges represent tag transition probabilities, and the node values represent word probabilities. Analysis showed that the correct tag most frequently missing from the lattice was the

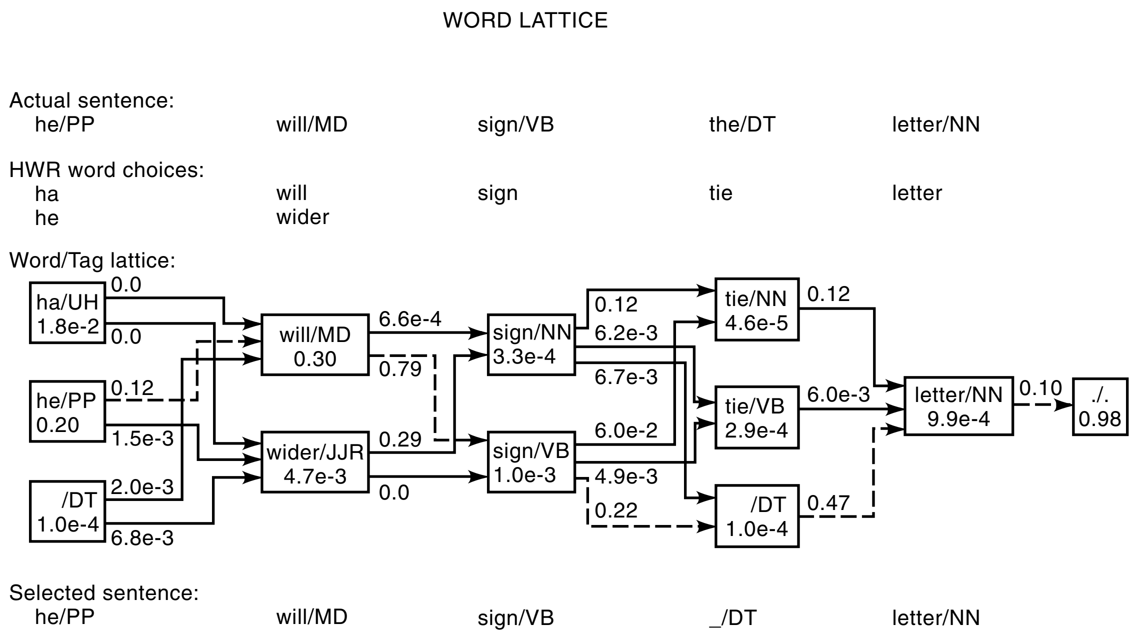


Figure 16. Sample word : tag lattice for analysis of HR choices.

determiner (DT) tag. Thus, the DT tag is automatically included in the lattice in all cases of short words (fewer than four characters) where it was not otherwise a candidate.

The percentage of words correctly recognized as the top choice increased from 51% to 61% using this method; the ceiling is 70% because correct word choice is absent in handwriting recognition output. Furthermore, by eliminating all word choices that were not part of the top 20 sequences output by the Viterbi, a reduction in the average word neighborhood of 56% (from 4.4 to 1.64 choices/word) was obtained with an error rate of only 3%. The latter is useful if an additional language model is to be applied (e.g., semantic analysis) because fewer word choices and, therefore, far fewer sentence possibilities remain.

Incorporating Signal Information

The Word–Tag model in the previous section is a pure language model because it does not deal with the recognizer score-vector $S = (s_1, s_2, \ldots, s_n)$ associated with W. The score s_i provides signal-level information about the input and is useful in discriminating among word-tag paths that are equally likely. Postprocessing models of handwriting recognition systems could incorporate both language models as well as recognizer confidences. It should be pointed out that confidences associated with recognizers are not probabilities and, hence, should be interpreted with caution.

Incorporating word score into the word-tag model would thus involve determining the word-tag path (W, T) which maximizes the following quantity:

$$\prod_{i=1}^{n} P(s_i|w_i t_i) P(w_i|t_i) P(t_i|t_{i-1}) \tag{15}$$

We make the valid assumptions that a score value s_i depends only on word w_i and not on the other word $w_{j \neq i}$ and that s_i is independent of the tag t_i. Thus,

$$P(S|W, T) = \prod_{i=1}^{n} P(s_i|w_i t_i) = \prod_{i=1}^{n} P(s_i|w_i) \tag{16}$$

Bouchaffra et al. (157) report on preliminary research toward calculating probabilities $P(s_i|w_i)$ from a training corpus annotated with scores.

Semantics

There have been attempts (137) to use semantic knowledge for improving recognition performance. Previously, semantic information had to be handcrafted for specific applications. Nowadays, semantic information is available through electronic resources such as machine-readable dictionaries and can also be computed from electronic corpora. One such technique involves the use of dictionary definition overlaps between competing word candidates in neighboring positions. The candidates with the highest overlap in each word position are selected as the correct words. For example, if the word choices for neighboring positions include (*savings swings*), and (*accept account*), respectively, then the obvious correct choice would be the words *savings account*. Another technique involves the use of *collocations* (158). Collocations reflect strong co-occurrence relations and account for a large percentage of English word combinations. Examples include *ice cream* and *computer science*. Word neighborhoods at distances of up to four word positions are compared with a list of potential collocates; each candidate is assigned a score according to the overlap between its neighborhood and its list of likely collocates. Finally, attempts have been made to use *semantic codes* that are available in certain dictionaries such as the *Long-*

man's Dictionary of Contemporary English (LDOCE) (159). A predefined set of domains has been established by LDOCE; content words such as nouns, verbs, adjectives, and adverbs are associated with one or more of these domain codes. These codes are then applied to select word candidates whose senses conform to the same domain. Although specific working examples of these techniques have been illustrated, there are no large-scale test results demonstrating the effectiveness of using semantic information in postprocessing the results of an handwriting recognition system. Furthermore, phenomena such as *collocations* can be efficiently handled by variations of *n*-gram techniques.

Combining Language Models. Maximum Entropy on its own is a very general framework and can be used for modeling various linguistic phenomena. Rosenfeld (160) discusses an approach that combines various statistical models, including a static trigram model and a Maximum Entropy model. At any given time, the various language models are combined linearly, using a dynamic weighting system that is determined empirically. It has been shown that such a combination leads to improved performance.

CONCLUSION

The ultimate goal of online handwriting recognition is to design a computer that can read a message as it is written on an electronic pen-pad. In other words, one would like to develop a system that picks up an idea, a sketch, or a note as it is written by someone and converts this message into a printed format that is easier to read by someone else.

In this perspective, the goal is not to mimic any human reader but to mimic a specific one: the one that is reading what he is writing, as he is writing. This is made possible because the data collected by the pen-pad include the time sequence of the pen-tip position, that is information about the sequence of human movements involved in the generation of the message. As we have shown in this survey, online handwriting recognition stands among the most difficult problems in pattern recognition and artificial intelligence. It relies upon our basic understanding of the generation of human movements and the perception of line images. It exploits numerous key concepts and principles in signal processing, algorithm design, system development, and testing. Moreover, the overall approach has to be driven by language modeling.

Although many interesting methods have been proposed for specific pen-pad applications, the recognition of unconstrained handwritten text is still an open problem. Its solution will require several breakthroughs in the basic domains involved: handwriting generation and reading processes on the one hand and language modeling and understanding on the other hand.

Indeed, handwriting generation models are not fully integrated into these systems nowadays. They are either simplistic, unrealistic, and not reliable or complex and not efficient. The artificial modeling of reading processes mostly concentrates on general algorithms for signal processing and understanding, classification schemes, and learning methodologies. A few systems take advantage of some key features of the reading processes, but most of them ignore fundamental phenomena like eyes movements, focus of attention, and disambiguation processes. Similarly, linguistic processing is a critical phase required in most of these systems. Language models for handwriting recognition are still too slow; they are not robust enough nor adaptable to new domains.

As long as these issues are not addressed and their solutions are not integrated into the different design phases of a system, it will be difficult to develop a pen-pad that consistently recognize general handwritten text with a level of performance that is required to make these tools a viable market product for a horizontal market. Meanwhile, it is expected that the development of specific pen-pads for some specific applications will continue and that several systems will become successful in many vertical markets.

ACKNOWLEDGMENTS

This work was supported by grant OGP0000915 from NSERC Canada, grant ER-1220 from FCAR, Québec to R. Plamondon. R. Srihari was jointly supported by NSF (National Science Foundation) and ARPA (Advanced Research Projects Agency) under the Human Language Technology program (grant IRI-9315006) and L. Schomaker by a grant from NICI (Nijmegen Institute for Cognition and Information). The authors wish to thank Mrs Hélène Dallaire, secretary, for her kind help with the various versions of this manuscript as well as Dr. Wacef Guerfali for his contribution to the preparation of Figs. 1–5.

BIBLIOGRAPHY

1. M. Eden, Handwriting and pattern recognition, *IRE Trans. Inf. Theory,* **IT-8**: 160–166, 1962.
2. C. Y. Suen, M. Berthod, and S. Mori, Automatic recognition of handprinted characters: The state of the art, *Proc. IEEE,* **68**: 469–487, 1980.
3. S. Mori, K. Yamamoto, and M. Yasuda, Research on machine recognition of handprinted characters, *IEEE Trans. Pattern Anal. Mach. Intell.,* **6**: 386–405, 1984.
4. S. Mori, C. Y. Suen, and K. Yamamoto, Historical review of OCR research and development, *Proc. IEEE,* **80**: 1029–1058, 1992.
5. S. Impedovo, L. Ottaviano, and S. Occhinero, Optical character recognition: A survey, *Int. J. Pattern Recognition Artif. Intell.,* **5** (2): 1–24, 1991.
6. A. W. Senior, *Off-Line Handwriting Recognition: A Review and Experiments,* Tech. Rep. CUED/F-INFENG/TRI05, Cambridge, UK: Engineering Dept., Cambridge Univ., 1992.
7. V. K. Govindan and A. P. Shivaprasad, Character recognition: A review, *Pattern Recognition,* **23** (7): 671–683, 1990.
8. G. Nagy, At the frontier of OCR, *Proc. IEEE,* **80**: 1093–1100, 1992.
9. P. Crettez and G. Lorette, Reconnaissance de l'écriture manuscrite, *Traité informatique, Techniques de l'ingénieur,* H1358, 1997.
10. C. C. Tappert, C. Y. Suen, and T. Wakahara, The state of the art in online handwriting recognition, *IEEE Trans. Pattern Anal. Mach. Intell.,* **12**: 787–808, 1990.
11. F. Nouboud and R. Plamondon, Online recognition of handprinted characters: Survey and beta tests, *Pattern Recognition,* **25** (9): 1031–1044, 1990.
12. A. Amin, Offline character recognition: A survey, *Proc. 4th Int. Conf. Doc. Anal. Recognition,* 1997, pp. 596–599.
13. E. C. W. Tse and H. W. Tan, *Proc. 17th Int. Conf. Comput. Process. Orient. Lang.,* Hong Kong, 1997, Vols. 1 and 2.

14. D. Barba, M. Gilloux, and C. Viard-Gaudin, *C. R. 4th Colloq. Natl. Écrit Doc.,* Nantes, 1996, pp. 1–308.

15. R. Plamondon and R. Sabourin (eds.), *1er Colloq. Francophone Écrit Doc., Cah. Sci. Association Canadienne Française pour l'Avancement des Sciences,* Québec, 1998.

16. R. Plamondon and G. Lorette, Automatic signature verification and writer identification: The state of the art, *Pattern Recognition,* **22** (2): 107–131, 1989.

17. F. Leclerc and R. Plamondon, Signature verification: The state of the art, 1989–1994, *Int. J. Pattern Recognition Artif. Intell., Spec. Issue Signature Verif.,* **8** (3): 643–660, 1994.

18. G. Leedham, Automatic recognition and transcription of Pitman's handwritten shorthand, *Computer Processing of Handwriting,* Singapore: World Scientific, pp. 235–269, 1990.

19. Lempesis Research, The time for pen computing is at-hand, *Pen-Vision News,* **2** (1): 1, January, 1992.

20. A. Kay and A. Goldberg, Personal dynamic media, *Computer,* **10** (3): 31–41, 1977.

21. B. Ryan, Dynabook revisited with Alan Key, *Byte,* **16** (2): 203–208, 1991.

22. L. Vuurpijl and L. R. B. Schomaker, Coarse writing-style clustering based on simple stroke-related features, *Proc. 5th Int. Workshop Front. Handwriting Recognition,* Colchester, UK, 1996, pp. 29–32.

23. Y. A. Dimitriadis and J. L. Coronado, Towards an art based mathematical editor, that uses online handwritten symbol recognition, *Pattern Recognition,* **28** (6): 807–822, 1995.

24. H. J. Winkler and M. Lang, Symbol segmentation and recognition for understanding handwritten mathematical expressions, *Proc. 5th Int. Workshop Front. Handwriting Recognition,* Colchester, UK, 1996, pp. 465–469.

25. C. Faure and Z. X. Wang, Automatic perception of the structure of handwritten mathematical expressions, in R. Plamondon and C. G. Leedham (eds.), *Computer Processing of Handwriting,* Singapore: World Scientific, 1990, pp. 337–361.

26. L. K. Welbourn and R. J. Whitrow, A gesture based text and diagram editor, in R. Plamodnon and C. G. Leedham (eds.), *Computer Processing of Handwriting,* Singapore: World Scientific, 1990, pp. 221–234.

27. L. Julia and C. Faure, Pattern recognition and beautification for a pen-based interface, *Proc. 3rd Int. Conf. Doc. Anal. Recognition,* Montreal, Can., 1995, pp. 58–63.

28. D. Galindo and C. Faure, Perceptually-based representation of network diagrams, *Proc. 4th Int. Conf. Doc. Anal. Recognition,* Ulm, Germany, 1997, pp. 352–356.

29. G. Utton et al., A strategy for online interpretation of sketched engineering drawings, *Proc. 4th Int. Conf. Doc. Anal. Recognition,* Ulm, Germany, 1997, pp. 771–775.

30. *JOT: A Specification for an Ink Storage and Interchange Format,* unpublished standards document, 1993; http://hwr.nici.kun.nl/unipen/jot.html.

31. I. Guyon and L. R. B. Schomaker, *UNIPEN Project of Data Exchange and Recognizer Benchmarks,* unpublished standards document, 1993; http://hwr.nici.kun.nl/unipen/unipen.def.

32. H. Chen, O. E. Agazzi, and C. Y. Suen, Piecewise linear modulation model of handwriting, *Proc. 4th Int. Conf. Doc. Anal. Recognition,* Ulm, Germany, 1997, pp. 363–367.

33. I. Guyon et al., UNIPEN project of online data exchange and recognizer benchmarks, *Proc. 12th Int. Conf. Pattern Recognition,* 1994, pp. 29–33.

34. A. J. W. M. Thomassen, P. J. G. Keuss, and G. P. van Galen (eds.), *Motor Aspects of Handwriting: Approaches to Movement in Graphic Behavior,* Amsterdam: North-Holland, 1984.

35. H. S. R. Kao, G. P. van Galen, and R. Hoosain (eds.), *Graphonomics: Contemporary Research in Handwriting,* Amsterdam: North-Holland, 1986.

36. R. Plamondon, C. Y. Suen, and M. L. Simner (eds.), *Computer Recognition and Human Production of Handwriting,* Singapore: World Scientific, 1989.

37. G. P. van Galen, A. J. W. M. Thomassen, and A. M. Wing (eds.), *Handwriting. Special Double Issue of Human Movement Sciences,* Amsterdam: Elsevier, 1992, vol. 11.

38. J. Wann, A. M. Wing, and N. Sovik (eds.), *Development of Graphic Skills: Research, Perspectives and Educational Implications,* London: Academic Press, 1991.

39. G. P. van Galen and G. E. Stelmach (eds.), *Handwriting: Issues of Psychomotor Control and Cognitive Models,* Special volume of Acta Psychologica, Amsterdam: North-Holland, 1993.

40. R. Plamondon (ed.), *Pattern Recognition,* Special Issue on Handwriting Processing and Recognition, Oxford: Pergamon Press, Vol. 26, No. 3, pp. 379–460.

41. R. Plamondon, *Progress in Signature Verification,* Singapore: World Scientific, 1994.

42. R. Plamondon (ed.), *Machine Vision and Applications,* Special Issue on Cursive Script Recognition, vol. 8, 195–259, 1995.

43. C. Faure et al. (eds.), *Advances in Handwriting and Drawing: A Multidisciplinary Approach,* Paris: Europia, 1994.

44. M. L. Simner, W. Hulstijn, and P. Girouard (eds.), *Forensic, Developmental and Neuropsychological Aspects of Handwriting,* Special Issue on the Journal of Forensic Document Examination, 1994.

45. M. L. Simner, C. G. Leedham, and A. J. W. M. Thomassen (eds.), *Handwriting and Drawing Research: Basic and Applied Issues,* Amsterdam: IOS Press, 1996.

46. A. Goshtasby and R. W. Ehrich, Contextual word recognition using probabilistic relaxation labelling, *Pattern Recognition,* **21**: 455–462, 1988.

47. J. J. Hull and S. N. Srihari, Experiments in text recognition with binary *n*-grams and viterbi algorithms, *IEEE Trans. Pattern Anal. Mach. Intell.,* **4**: 520–530, 1982.

48. A. Jones, A. Story, and W. Ballard, Integrating multiple knowledge sources in a bayesian OCR postprocessor, *Int. Conf. Doc. Anal. Recognition,* St. Malo, France, 1991, pp. 925–933.

49. J. R. Ullmann, A binary *n*-gram technique for automatic correction of substitution, deletion, insertion and reversal errors in words, *Comput. J.,* **20** (2): 141–147, 1975.

50. D. M. Ford and C. A. Higgins, A tree-based dictionary search technique and comparison with n-*gram* letter graph reduction. In R. Plamondon and G. Leedham (eds.), *Computer Processing of Handwriting,* Singapore: World Scientific, 1990, pp. 291–312.

51. C. J. Wells et al., Word look-up for script recognition—choosing a candidate, *Int. Conf. Doc. Anal. Recognition,* St. Malo, France, 1991, pp. 620–628.

52. A. Aho and J. Ullman, *The Theory of Parsing, Translation and Compiling,* Parsing, Series in Automatic Computation, Englewood Cliffs, NJ: Prentice-Hall, 1972, vol. 1.

53. P. Dumouchel et al., Three probabilistic language models for a large-vocabulary speech recognizer, *Int. Conf. Acoust, Speech Signal Process. '88,* 1988, pp. 513–516.

54. F. Jelinek, *Up from Trigrams! The Struggle for Improved Language Models,* Eurospeech 1991, Continuous Speech Recognition Group IBM Research T. J. Watson Research Center, Yorktown Heights, NY, 1991.

55. K. W. Church, Stochastic parts program and noun phrase parser for unrestricted text, *Int. Conf. Acoust., Speech Signal Process. '89,* 1989, pp. 695–698.

56. A. M. Derouault and B. Merialdo, Natural language modelling for phoneme-to-text transcription, *IEEE Trans. Pattern Anal. Mach. Intell.,* **8**: 742–749, 1984.

57. F. G. Keenan, L. J. Evett, and R. J. Whitrow, A large vocabulary stochastic syntax analyzer for handwriting recognition, *Int. Conf. Doc. Anal. Recognition,* St. Malo, France, 1991, pp. 794–802.

58. P. Viviani and N. Stucchi, Motor perceptual interactions, in G. E. Stelmach and J. Requin (eds.), *Tutorials in Motor Behavior II,* Amsterdam: Elsevier, 1992, pp. 229–248.

59. R. Plamondon and F. J. Maarse, An evaluation of motor models of handwriting, *IEEE Trans. Syst. Man Cybern.,* **19**: 1060–1072, 1989.

60. R. Plamondon et al., Modelling velocity profiles of rapid movements: A comparative study, *Biol. Cybern.,* **69** (2): 119–128, 1993.

61. R. Plamondon and W. Guerfali, The generation of handwriting with delta-lognormal synergies, *Biol. Cybern.,* **78**: 119–132, 1998.

62. D. Bullock, S. Grossberg, and C. Mannes, A neural network model for cursive script recognition, *Biol. Cybern.,* **70**: 15–28, 1993.

63. P. Morasso and F. A. Mussa-Ivaldi, Trajectory formation and handwriting: a computational model, *Biol. Cybern.,* **45**: 131–142, 1982.

64. R. Plamondon and F. Lamarche, Modelization of handwriting: A system approach, in H. S. R. Kao, G. P. van Galen, and R. Hoosain (eds.), *Graphonomics: Contemporary Research in Handwriting,* Amsterdam: Elsevier, North-Holland, 1986, pp. 169–183.

65. J. M. Hollerbach, An oscillation theory of handwriting, *Biol. Cybern.,* **39**: 139–156, 1981.

66. O. Stettiner and D. Chazan, A statistical parametric model for recognition and synthesis of handwriting, *Proc. 12th Int. Conf. Pattern Recognition,* 1994, vol. 2, pp. 34–38.

67. Y. Singer and N. Tishby, Dynamical encoding of cursive handwriting, *Biol. Cybern.,* **71**: 227–237, 1994.

68. H. Chen, O. E. Agazzi, and C. Y. Suen, Piece linear modulation model of handwriting, *Proc. 4th Int. Conf. Doc. Anal. Recognition,* Ulm, Germany, 1997, pp. 363–366.

69. L. R. B. Schomaker, A. J. W. M. Thomassen, and H. L. Teulings, A computational model of cursive handwriting, in R. Plamondon, C. Y. Suen, and M. L. Simner (eds.), *Computer Recognition and Human Production of Handwriting,* Singapore: World Scientific, 1989, pp. 153–177.

70. P. Morasso, V. Sanguineti, and T. Tsuji, A model for the generation of virtual targets in trajectory formation, in C. Faure et al. (eds), *Advances in Handwriting and Drawing: A Multidisciplinary Approach,* Europia, Paris, 1994, pp. 333–348.

71. T. Flash and N. Hogan, The coordination of arm movements: An experimentally confirmed mathematical model, *Neuroscience,* **7**: 1688–1703, 1985.

72. S. Edelman and T. Flash, A model of handwriting, *Biol. Cybern.,* **57**: 25–36, 1987.

73. Y. Uno, M. R. Kawato, and R. Suzuki, Formation and control of optimal trajectory in human multi-joint arm movement, *Biol. Cybern.,* **61**: 89–101, 1989.

74. G. Lorette, *Méthode de traitement automatique d'images de points et de lignes en reconnaissance de formes,* Thèse de doctorat d'états, Université Paris Val-de-Marne, 1984.

75. R. Plamondon, A kinematic theory of rapid human movements: part I: Movement representation and generation, *Biol. Cybern.,* **72** (4): 295–307, 1995.

76. R. Plamondon, A kinematic theory of rapid human movements: part II: Movement time and control, *Biol. Cybern.,* **72** (4): 309–320, 1995.

77. C. C. Tappert, C. Y. Suen, and T. Wakahara, The state of the art in online handwriting recognition, *IEEE Trans. Pattern Anal. Mach. Intell.,* **12**: 179–190, 1990.

78. F. Nouboud and R. Plamondon, A structural approach to online character recognition: System design and applications, in P. S. P. Wang (ed.), *Character and Handwriting Recognition: Expanding Frontiers,* Singapore: World Scientific, 1991, pp. 311–335.

79. C. C. Tappert, Speed, accuracy, and flexibility trade-offs in online character recognition, in P. S. P. Wang (ed.), *Character and Handwriting Recognition: Expanding Frontiers,* Singapore: World Scientific, 1991, pp. 79–95.

80. W. Guerfali and R. Plamondon, Normalizing and restoring online handwriting, *Pattern Recognition,* **26** (3): 419–431, 1993.

81. R. M. Brown, Online computer recognition of handprinted characters, *IEEE Trans. Electron. Comput.,* **EC-13**: 750–752, 1964.

82. K. Yoshida and H. Sakoe, Online handwritten character recognition for a personal computer system, *IEEE Trans. Consumer Electron.,* **28**: 202–209, 1982.

83. A. Hennig, N. Sherkat, and R. J. Whitrow, Zone-estimation for multiple lines of handwriting using approximate spline functions, *Proc. 5th Int. Workshop Front. Handwriting Recognition,* Colchester, UK, 1996, pp. 325–328.

84. J. Mackowiak, L. R. B. Schomaker, and L. Vuurpijl, Semi-automatic determination of allograph duration and position in online handwriting words based on the expected number of strokes, in A. C. Downton and S. Impedovo (eds.), *Progress in Handwriting Recognition,* Singapore: World Scientific, pp. 69–74, 1997.

85. S. H. Lee, H. K. Lee, and J. H. Kim, Numeral characters and capital letters segmentation recognition in mixed handwriting context, *Proc. 3rd Int. Conf. Doc. Anal. Recognition,* Montreal, Can., 1995, pp. 878–881.

86. D. Goldberg and C. Richardson, Touch-typing with a stylus, *Proc. Int. Conf. Hum. Factors Comput. Syst.,* Amsterdam, 1993, pp. 80–87.

87. L. Vuurpijl and L. R. B. Schomaker, Finding structure in diversity: A hierarchical clustering method for the categorization of allographs in handwriting, *Proc. 4th Int. Conf. Doc. Anal. Recognition,* Ulm, Germany, 1997, pp. 387–393.

88. M. K. Brown and S. Ganapathy, Preprocessing techniques for cursive script word recognition, *Pattern Recognition,* **16**: 447–458, 1983.

89. D. J. Burr, Designing a handwriting reader, *IEEE Trans. Pattern Anal. Mach. Intell.,* **PAMI-5**: 554–559, 1983.

90. F. J. Maarse and A. J. W. M. Thomassen, Produced and perceived writing slant: Difference between up and down strokes, *Acta Psychol.,* **54**: 131–147, 1983.

91. C. J. Wells et al., Fast dictionary look-up for contextual word recognition, *Pattern Recognition,* **23**: 501–508, 1990.

92. L. R. B. Schomaker and H. L. Teulings, A handwriting recognition system based on the properties and architectures of the human motor system, *Proc. Int. Workshop Front. Handwriting Recognition,* Montreal, Can., 1990, pp. 195–211.

93. T. Fujisaki et al., Online run-on character recognizer: Design and performance, in P. S. P. Wang (ed.), *Character and Handwriting Recognition: Expanding Frontiers,* Singapore: World Scientific, 1991, pp. 123–137.

94. C. A. Higgins and D. M. Ford, A new segmentation method for cursive script recognition, in S. Impedovo and J. C. Simon (eds.), *From Pixels to Features III: Frontiers in Handwriting Recognition,* Amsterdam: Elsevier, 1992, pp. 75–86.

95. B. Bontempi and A. Marcelli, Online segmentation of cursive script using an arclength representation, in M. L. Simner, C. G. Leedham, and A. J. W. M. Thomassen (eds.), *Handwriting and Drawing Research: Basic and Applied Issues,* Amsterdam: IOS Press, 1996, pp. 315–327.

96. X. Li, M. Parizeau, and R. Plamondon, Detection of extreme points of online handwritten scripts, *Proc. 5th Int. Workshop Front. Handwriting Recognition,* Colchester, UK, 1996, pp. 67–72.

97. X. Li, M. Parizeau, and R. Plamondon, Segmentation and reconstruction of online handwritten scripts, *Pattern Recognition,* **31** (6): 675–684, 1998.

98. L. R. B. Schomaker, Using stroke- or character-based self-organizing maps in the recognition of online, connected cursive script, *Pattern Recognition,* **26** (3): 443–450, 1993.

99. E. Anquetil and G. Lorette, Perceptual model of handwriting drawing application to the handwriting segmentation problem, *Proc. 4th Int. Conf. Doc. Anal. Recognition,* Ulm, Germany, 1997, pp. 112–117.

100. M. Kadirkamanathan and P. J. W. Rayner, A scale-space filtering approach to stroke segmentation of cursive script, in R. Plamondon and C. G. Leedham (eds.), *Computer Processing of Handwriting,* Singapore: World Scientific, 1990, pp. 133–166.

101. M. Parizeau and R. Plamondon, Allograph adjacency constraints for cursive script recognition, *Proc. 3rd Int. Workshop Front. Handwriting Recognition,* Buffalo, NY, 1993, pp. 252–261.

102. I. Karls et al., Segmentation and recognition of cursive handwriting and improved structured lexical, *Proc. 3rd Int. Workshop Front. Handwriting Recognition,* Buffalo, NY, 1993, pp. 437–442.

103. G H. Abbink, H. L. Teulings, and L. R. B. Schomaker, Description of online script using Hollerbach's generation model, *Proc. 3rd Int. Workshop Front. Handwriting Recognition,* Buffalo, NY, 1993, pp. 217–224.

104. J. Hu, M. K. Brown, and W. Turin, Handwriting recognition with hidden Markov models and grammatical constraints, *Proc. 4th Int. Workshop Front. Handwriting Recognition,* Taipei, Taiwan, 1994, pp. 195–205.

105. J. Hu, M. K. Brown, and W. Turin, HMM based online handwriting recognition, *IEEE Trans. Pattern Anal. Mach., Intell.,* **18**: 1039–1045, 1996.

106. S. Bercu and G. Lorette, Online handwritten word recognition: An approach based on hidden Markov models, *Proc. 3rd Int. Workshop Front. Handwriting Recognition,* Buffalo, NY, 1993, pp. 385–390.

107. E. J. Bellegarda et al., Segmentation and recognition of cursive handwriting with improved structured lexica, *Proc. 3rd Int. Workshop Front. Handwriting Recognition,* Buffalo, NY, 1993, pp. 225–234.

108. S. H. Lee, H. K. Lee, and J. H. Kim, Online cursive script recognition using an island-driven search technique, *Proc. 3rd Int. Conf. Doc. Anal. Recognition,* Montreal, Can., 1995, pp. 886–889.

109. W. Guerfali and R., Plamondon, Effect of variability on letter generation with the vectorial delta-lognormal model, invited paper, *1st Brazilian Symp. Doc. Image Anal.,* 1997, pp. 1–10.

110. C. Barrière and R. Plamondon, Human recognition of letters in mixed script handwriting: An upper bound on recognition rate, *IEEE Trans. Syst. Man Cybern.,* **28** (1): 78–82, 1998.

111. M. Parizeau and R. Plamondon, Machine vs. human in a cursive script reading experiment without linguistic knowledge, *Proc. 12th Int. Conf. Pattern Recognition,* 1994, pp. 93–98.

112. L. R. B. Schomaker, S. Muench, and K. Hartung, A taxonomy of multimodal interaction in the human information processing system, *Rep. Esprit Proj.,* **8579**: 1–187, 1995.

113. M. Parizeau and R. Plamondon, A fuzzy syntactical approach to allograph modelling for cursive script recognition, *IEEE Trans. Pattern Anal. Mach. Intell.,* **17** (7): 702–712, 1995.

114. E. Anquetil and G. Lorette, Online handwriting character recognition system based on hierarchical qualitative fuzzy modeling, *Progress in Handwriting Recognition,* Singapore: World Scientific, 1997, pp. 109–116.

115. P. C. Mahalonobis, On tests and measures of group divergence. I, *J. Proc. Asiat. Soc. Bengal.,* **26**: 541–588, 1930.

116. C. M. Bishop, *Neural Networks for Pattern Recognition,* Oxford: Clarendon Press, 1995.

117. J. L. McClelland and D. E. Rumelhart, *Parallel Distributed Processing: Explorations in the Microstructure of Cognition,* Cambridge, MA: MIT Press, 1986, vol. 1.

118. T. Kohonen, *Self-organizing Maps,* Springer Ser. Inf. Sci., Heidelberg: Springer, 1995, vol. 30.

119. Y. LeCun and Y. Bengio, Convolutional networks for images, speech, and time-series, in M. A. Arbib (ed.), *The Handbook of Brain Theory and Neural Networks,* Cambridge, MA: MIT Press, pp. 255–258, 1995.

120. I. Guyon et al., Design of a neural network character recognizer for a touch terminal, *Pattern Recognition,* **24** (2): 105–119, 1991.

121. L. E. Baum, An inequality and associated maximization technique in statistical estimation of probabilistic functions of Markov processes, *Inequalities,* **3**: 1–8, 1972.

122. S. E. Levison, L. R. Rabiner, and M. M. Sondhi, An introduction to the application of the theory of probabilistic functions of Markov process to automatic speech recognition, *Bell System Tech. J.,* **62** (4): 1035–1074, 1983.

123. L. R. Rabiner and B. H. Juang, An introduction to hidden Markov models, *IEEE ASSP Mag.,* **3** (1): 4–16, 1986.

124. A. B. Poritz, Hidden Markov models: A guided tour, *Proc. Int. Conf. ASSP,* **1**: 7–13, 1998.

125. L. R. Rabiner, A tutorial on hidden Markov models and selected application in speech recognition, *Proc. IEEE,* **77**: 257–286, 1989.

126. S. Manke, M. Finke, and A. Waibel, NPen++: A writer independent, large vocabulary online cursive handwriting recognition system, *Proc. 3rd Int. Conf. Doc. Anal. Recognition (ICDAR'95),* 1995, pp. 403–408.

127. Y. S. Huang and C. Y. Suen, The behavior-knowledge space method for the combination of multiple classifiers, *Proc. IEEE Conf. Comput. Vision Pattern Recognition,* 1993, pp. 347–352.

128. R. K. Powalka, N. Sherkat, and R. J. Whitrow, Multiple recognizer combination topologies, *Handwriting and Drawing Research: Basic and Applied Issues,* IOS Press, pp. 329–342, 1996.

129. D. Gader, M. A. Mohamd, and J. M. Keller, Fusion of handwritten word classifiers, *Pattern Recognition Lett.,* **17**: 577–584, 1996.

130. C. Y. Suen, Handwriting generation, perception and recognition, *Acta Psychologica,* **54**: 295–312, 1983.

131. A. Leroy, Lexicon reduction based on global features for online handwriting, *Proc. 4th Int. Workshop Front. Handwriting Recognition,* Taipei, Taiwan, 1994, pp. 431–440.

132. A. Leroy, Progressive lexicon reduction for online handwriting, *Proc. 5th Int. Workshop Front. Handwriting Recognition,* Colchester, UK, 1996, pp. 399–404.

133. R. K. Powalka, N. Sherkat, and R. J. Whitrow, The use of word shape information for cursive script recognition, *Proc. 4th Int. Workshop Front. Handwriting Recognition,* Taipei, Taiwan, 1994, pp. 67–76.

134. G. Seni and R. K. Srihari, A hierarchical approach to online script recognition using a large vocabulary, *Proc. 4th Int. Work-*

shop Front. Handwriting Recognition, Taipei, Taiwan, 1994, pp. 472–479.

135. K. Rohini and C. M. Baltus, Incorporating syntactic constraints in recognizing handwriting sentences, *Proc. Int. Jt. Conf. Artif. Intell. (IJCAI-93),* 1993, pp. 1262–1267.

136. I. Guyon and F. Pereira, Design of a linguistic postprocessor using variable memory length Markov models, *Proc. Int. Conf. Doc. Anal. Recognition (ICDAR '95),* 1995, pp. 454–457.

137. T. G. Rose and L. G. Evett, Semantic analysis for large vocabulary cursive script recognition, *Proc. Int. Conf. Doc. Anal. Recognition (ICDAR '93),* 1993, p. 236–239.

138. S. Clergeau-de-Tournemire and R. Plamondon, Integration of lexical and syntactical knowledge in a handwriting recognition system, *Mach. Vision Appl., Spec. Issue Cursive Script,* **8** (4): 249–260, 1995.

139. R. Plamondon, S. Clergeau-de-Tournemire, and C. Barrière, Handwritten sentence recognition: From signal to syntax, *Proc. 12th Int. Conf. Pattern Recognition,* 1994, pp. 117–122.

140. R. K. Srihari, Use of lexical and syntactic techniques in recognizing handwriting text, *Proc. ARPA Hum. Lang. Technol. Workshop,* 1994, pp. 427–431.

141. K. Kukich, Techniques for automatically correcting words in text, *ACM Comput. Surv.,* **24** (4): 377–439, 1992.

142. R. A. Wagner and M. J. Fischer, the string-to-string correction problem, *J. Assoc. Comput. Mach.,* **21** (1): 168–173, 1974.

143. G. Seni, K. Sundar, and R. K. Srihari, Generalizing edit distance for handwritten text recognition, *Pattern Recognition,* **29** (3): 405–414, 1996.

144. E. S. Ristad and P. N. Yianilos, Larning string edit distance, *IEEE Trans. Pattern Anal. Mach. Intell.,* **20** (5): 522–532, 1998.

145. I. J. Good, The population frequencies of species and the estimation of population parameters, *Biometrika,* **40**: 237–264, 1953.

146. W. A. Gale and K. W. Church, Poor estimates of context are worse than none, *Proc. DARPA Speech Nat. Lang. Workshop,* 1990, pp. 283–287.

147. P. F. Brown et al., Class-based *n*-gram models of natural language, *Comput. Ling.,* **18** (4): 467–479, 1992.

148. M. Schenkel, I. Guyon, and D. Henderson, Online cursive script recognition using time delay neural networks and hidden Markov models, *Mach. Vision Appl.,* **8** (4): 215–223, 1995.

149. K. Lari and S. J. Young, The estimation of stochastic context-free grammars using the inside-outside algorithm, *Comput. Speech Lang.,* **4** (1): 35–56, 1990.

150. L. R. Bahl, F. Jelinek, and R. L. Mercer, a maximum likelihood approach to continuous speech recognition, *IEEE Trans. Pattern Recognition Mach. Intell.* **PAMI-5** (2): 179–190, 1983.

151. C. A. Higgins and R. Whitrow, Online cursive script recognition, in B. Shakel (ed.), *Human-Computer Interaction-INTERACT'84,* vol. 2, pp. 140–144, 1985.

152. G. E. Forney, Jr., The Viterbi algorithm, *Proc. IEEE,* **61**: 268–278, 1973.

153. P. Clarkson and R. Rosenfeld, Statistical language modeling using the CMU-Cambridge tookit, *Proc. Eurospeech'97* ; [online], available http://www.cs.cmu.edu/afs/cs/user/roni/WWW/HomePage.html.

154. R. Rosenfeld, A Maximum Entropy approach to adaptive statistical language modeling, *Comput. Speech Lang.,* **10**: 187–228, 1996.

155. F. G. Keenan, L. J. Evett, and R. J. Whitrow, A large vocabulary stochastic syntax analyser for handwriting recognition, *Proc. 1st Int. Conf. Doc. Anal. Recognition (ICDAR'91),* 1991, pp. 794–802.

156. J. J. Hull, Incorporation of a Markov model of syntax in a text recognition algorithm, *Proc. Symp. Doc. Anal. Inf. Retrieval,* 1992, pp. 174–183.

157. D. Bouchaffra et al., Incorporating diverse information sources in handwriting recognition postprocessing, *Int. J. Imag. Syst. Technol.,* **7**: 320–329, 1996.

158. F. Smadja, Macrocoding the lexicon with co-occurrence knowledge, in U. Zernik (ed.), *Lexical Acquisition: Exploiting On-Line Resources to Build a Lexicon,* Hillsdale, NJ: Erlbaum, 1991, pp. 427–431.

159. P. Proctor (ed.), *Longman Dictionary of Contemporary English,* London: Longman, 1978.

160. R. Rosenfeld, A hybrid approach to adaptive statistical language modeling, *Proc. Adv. Res. Projects Agency Hum. Lang. Technol. Workshop,* 1994, pp. 76–81.

RÉJEAN PLAMONDON
École Polytechnique de Montréal
DANIEL P. LOPRESTI
Lucent Technologies, Inc.
LAMBERT R. B. SCHOMAKER
Nijmegen Institute for Cognition and Information
ROHINI SRIHARI
CEDAR, UB Commons