Japanese HWR

Steven B. Poggel
steven.poggel@gmail.com

December 19, 2009

Contents

Chapter 1

On-Line Handwriting Recognition

Contents

1.1 Introduction

Handwriting is a very personal skill to individuals. It consists of graphical marks on a surface and can be used to identify a person. Communication is the main purpose and this is achieved by drawing letters or other *graphemes*, which in turn represent parts of a language. The characters have a certain basic shape, which must be recognisable for a human in order for the communication process to function. There are rules for the combination of letters, which have the ability - if known to the reader - to help recognise a character or word.

Handwriting was developed as a means of communication and to expand one's own memory. With the advent of each new technologies the question arose, if handwriting was going to survive. However, the opposite seems to be the truth: For example, the printing press increased the number of documents available and therefore increased the number of people who learnt to read and write. Through the increased rate of alphabetisation, naturally there was an increased use of handwriting as a means of communication.

In various situations handwriting seems much more practical than typing on a keyboard. For instance children at school are using notepads and pencils or ink pens, which are regarded as a better tool to teach writing by German teachers. Therefore it can be concluded that there is little danger of the extinction of handwriting as a communication tool. In fact, as the length of handwritten messages decreases, the number of people using handwriting increases (Plamondon and Srihari 2000).

1.2 Handwriting Features

Any script of any language has a number of features. The fundamental characteristic of a script is that the differences between the features of different characters are more decisive than the different features of drawing variants of the same letter in individual handwriting styles. There might be exceptions, because *0* and *O* or *1* and *I* respectively, can be written alike. However, in those cases, context makes clear which one was intended by the writer. Despite the exception, written communication can only work with that fundamental quality (Tappert et al. 1990).

1.2.1 Handwriting Properties of Latin Script

In the Latin script we have 26 letters, each of which has two variants, a capital and a lowercase variant. When writing a character in the Latin script, there are four main areas, in which the character can reside. All characters have their main part between a top line and a ground line. There is also a middle line. Capital characters stretch out to use the full space between the ground line and the top line, whereas lowercase characters usually use the space between the ground line and the middle line. Some lowercase characters (like lowercase *b, d, f, h, k, l, t*) have an ascender and use the area above the middle line as well, some lowercase characters have a descender and use the area below the ground line (like lowercase *g, j, p, q, y*). In handwritten cursive script, there are writing variants where also some lowercase letters (*f, z*) and certain uppercase characters (*G, J*) expand below the ground line. For all latin-based alphabets, usually one character is finished before the next one starts, however, there are exceptions: In cursive handwriting, the dots on *i* and *j* and the crosses of *t* might be delayed until the underlying portions of all the characters of the word are completed.

XXX Graphic with example of expanding cursive letters here.

1.2.2 Handwriting Properties of East Asian Scripts

Generally, a handwriting is a formed of a number of strokes, that are drawn in a time sequence. Opposed to the latin-based alphabets, consider Chinese and Japanese script. Chinese has a larger alphabet, up to 50000 characters, 3000-5000 of which are in active use. There are also two writing styles, block style - which corresponds to printed characters in Latin alphabets, even if handwritten. The other style is cursive style. In block style the individual parts of the character are usually written in proper stroke order, and abide by the proper stroke number. In cursive style the characters are written faster, with less care and don't necessarily abide to stroke number or order. In fact, they are usually written with fewer strokes, connecting some block-style strokes by using simpler radical shapes (Tappert et al. 1990).

In Japanese, three different scripts are in active use at the same time, mixed and next to each other. They are called *Hiragana* (ひらがな),

Katakana (カタカナ) and *Kanji* (漢字). Hiragana and Katakana are syllabic alphabets, each containing 46 characters (see section **??**), whereas Kanji are essentially the Chinese *Hànzì* characters (汉字) as they were imported into the Japanese language (see section **??**).

The different scripts can even be blended with each other within one word. Take for instance the verb 食べる (pron. *taberu*, Eng. *to eat*). The first character is a Kanji character, pronounced */ta/*, which also bears the meaning of the word. The second and third characters are the Hiragana characters *be* and *ru* which are there for conjugation only as well as for phonetic reasons. Without them, the character 食 still bears the meaning of the concept *eat*, but the character alone does not result in the verb *taberu*.

1.3 Automated Recognition of Handwriting

1.3.1 Short History of Handwriting Recognition

Handwriting recognition (HWR) as a technological discipline performed by machines has been around for many years. The quality of the systems recognising handwriting has improved over the decades. It is the key technology to pen-based computer systems. The first research papers concerned with *pattern recognition* on computers were published in the late 1950'ies, *Handwriting recognition* as an individual subject in the early 1960'ies. (Goldberg 1915) describes in a US Patent a machine that can recognise alphanumeric characters as early as 1915. However, despite the surprise of how early such a device was invented, it should be taken into consideration that that was before the times of modern computers, therefore the methods he employs are quite different from the algorithms used after the advent of computers, more concretely, computers with screens.

(Tappert et al. 1990) describe in their review the development of handwriting recognition, which was a popular research topic in the early 1970'ies and then again in the 1980'ies, due to the increased availability of pen-input devices. Generally speaking, handwriting recognition (HWR) involves automatic conversion of handwritten text into a machine readable character encoding like ASCII or UTF-8. Typical HWR-environments include a pen or stylus that is used for the handwriting, a touch-sensitive surface, which the user writes on and a an application that interprets the strokes of the stylus on the surface and converts them into digital text. Usually, the writing surface captures the x-y coordinates of the stylus movement.

1.3.2 Pattern Recognition Problems

The general problem of *pattern recognition* is to take a non-symbolic representation of some pattern, like mouse or pen coordinates and transform it into a symbolic representation like a *rectangle* with its coordinates, or in the case of handwriting recognition, a character. Pattern recognition is a symbol manipulation procedure, that tries to generate discrete structures or sub-sets of discrete structures. Some see it as a game theory problem, where machine 'players' try to match the interpretation of an input produced by machine 'experts' (Zanibbi et al. 2005).

1.3.2.1 Related Problems

There are several related problems, the recognition of equations, line drawings and gestures symbols. The recognition of language symbols includes the different large alphabets of Chinese, the different scripts of Japanese, alphabetic scripts like Greek, or Arabic and other non-alphabetic scripts like the Korean Hangul, but also various writing styles of the latin-based alphabets, and diacritics that are used to denote pronunciation variants in different languages using Latin script, like Turkish or Vietnamese.

Other Problems that are related to handwriting recognition include for example mathematical formula recognition, where mathematical formulae are analysed and put into a computable format (Chan and Yeung 2001). In diagram recognition both the characters and the diagram layout are recognised (Blostein and Haken 1999).

1.3.2.2 Problem of Similar Characters

XXX remake fig 2. in Tappert1990 which he had stolen from his reference 240

There are several subproblems to the task of pattern recognition of a character. Different styles of handwriting can be seen in xxx LABELOFGRAPHICABOVE. The scripts toward the bottom of LABELOFGRAPHICABOVE are harder to recognise. In the case of boxed discrete characters, segmentation is done for the machine by the user. Run-on discrete characters are easier to recognise than pure cursive handwriting, because there is a pen-up and pen-down between each character. In cursive handwriting, segmentation between the characters becomes a more difficult task. Some parts of the writing may be delayed, like the crosses of *t* or the dots on *i* or *j*. Besides the segmentation, the discrimination of shapes is often not trivial. Humans may or may not be able to decipher somebody else's handwriting and clearly distinguish between, say *U*-*V*, but most of the times, context helps with that task. Other characters have similar shapes, too, like *C-L*, *a-d* and *n-h*. Confusion can arise between characters and numbers like *O-0*, *I-1*, *l-1*, *Z-2*, *S-5*, *G-6*, *I-1*, *Z-2*. Lowercase and uppercase are hard to distinguish in the cases of *C-c*, *K-k*, *O-o*, others are mainly distinguished by their position relative to the base line of the rest of the text: *P-p*, *Y-y*. Therefore, context helps the human reader to identify the correct character. This could be used as an advantage in automated pattern recognition, as well. However, the other characters nearby would have to be recognised first, which creates a (solvable) hen and egg problem. In the Japanese script the problem is taken to another dimension. Consider 本 (root) and 木 (tree) those two characters are only a very basic sample of characters that might lead to confusion of the different symbols. However, due to the hierarchical organisation of the characters with their radicals (see section **??**) there are many more shapes that look very much alike. From the shape recognition perspective, minor changes to the shape of a character can change its meaning drastically. Compare 嘸 (how, indeed), 撫 (stroke, pat), 蕪 (turnip) and 憮 (disappointment). They all contain the radical 無 (nothingness, none), which doesn't seem to have any semantic connection with the characters, however, it can be seen as the main radical in those characters.

1.3.3 Hardware Requirements

In order to perform on-line handwriting recognition, the handwriting needs to be captured in some way. Special hardware is necessary to perform the task of capturing both the x-y coordinates and the time information of the handwriting input device.

Several different hardware commercial products are available in order to capture the x-y coordinates of a stylus or pen. Graphics tablet like the products of the Wacom Co., Ltd.[1] are popular input devices for hand motions and hand gestures. The use of pen-like input devices has also been recommended, since 42% of mouse users report feelings of weakness, stiffness and general discomfort in the wrist and hand when using the mouse for long periods (Woods et al. 2002). The sampling rates of digital pens are usually around 50-200 Hz. The higher the sampling rate the finer the resolution of the resulting co-ordinates, which leads to more accurate measurement of fast strokes.

Moreover there are PDAs and Tablet PCs, where the writing surface serves as an output device, i.e. an display at the same time. If the pen capturing area is transparent and has an instantaneous display behind that shows immediately whatever input the user drew with the stylus, a high level of interactivity can be reached (Santosh and Nattee 2009).

[1] www.wacom.com

These displays include touch screens, which are a newer development. New generation mobile phones like the notorious iPhone from Apple Inc. also contain touch-displays, but for those it is more common to be operated without a stylus. For the task of handwriting recognition, a stylus can be regarded as the more natural device, since people usually write with pens on paper, therefore a stylus on a display seems more natural than using a finger on a display for writing. In order to interpret user gestures, an input given directly with the fingers is a more natural option. Gestures for zooming into digital pictures, or turning to the next page of a document are interpreted on these devices.

Another rather new development are real-ink digital pens. With those, a user can write on paper with real ink, and the pen stores the movements of the pen-tip on the paper. The movements are transferred to a computer later. It can be expected that with technologies like Bluetooth it may be possible to transfer those data in real-time, not delayed. In a fairly new development accelerometer technology has been used for handwriting recognition, using a mobile phone as a device to write in the air (Agrawal et al. 2009). That approach can be regarded as an area that is only loosely related to classical handwriting recognition, as the phone stores an image of the strokes in the air that were measured by the accelerator device, but does not transform the strokes into characters.

xxx: if you want to add more technical details here, see Santosh2009:Digitizer Technology. :xxx

1.3.4 Recognition vs Identification

Handwriting recognition is the task of transforming a spatial language representation into a symbolic representation. In the English language (and many others) the symbolic representation is typically 8-bit ASCII. However, with *Unicode* being around for more than a decade now, storage space on hard disks not being as much of an issue any more and *RAM* being readily available to the Gigabytes, it has become more common to use a *UTF-8* encoding, which is a variable-length character encoding for Unicode (The Unicode Consortium 2000). Akin disciplines to handwriting recognition are *handwriting identification*, which is the task of identifying the author of a handwritten text sample from a set of writers, assuming that each handwriting style can be seen as individual to the person who wrote it. The task of *signature verification* is to determine if a given signature stems from the person who's name is given in the signature. Thus, handwriting identification and verification can be used for analysis in the field of jurisdiction. They determine the individual features of a handwritten sample of a specific writer and compare those to samples given by a different or the same writer. By analysing those features one can find out if a piece of handwritten text is authentic or not.

1.3.5 Interpretation of Handwriting

Handwriting recognition and interpretation are trying to filter out the writer-specific variations and extract the text message only. This conversion process can be a hard task, even for a human. Humans use context knowledge in order to determine the likeliness of a certain message in a certain context. For instance, a handwritten message on a shopping list that could be read as *bread* or *broad* due to the similarities of the characters for 'e' and 'o' in some cursive handwriting styles, will be interpreted as *bread*, since it is a much more likely interpretation in the shopping list domain. However, if the next word on the shopping list is *beans*, the likelihood for the interpretation of the first word as *broad* rises, because the collocation *broad beans* is a sequence that is likely on a shopping list, at least more likely than having the interpretation *bread* and then *beans* without a clear separation between the two. Even with non-handwritten, but printed characters, the human mind can be tricked because of the brain's ability to perform these interpretations within milliseconds without conscious thinking. An example of that are modern T-Shirt inscriptions that state things like *Pozilei* in a white font on a green ground (the German police colours in most federal states are green and white), which German native speakers usually read as *Polizei* (police), because that is the most likely interpretation.

1.3.6 On-Line vs. Off-Line Recognition

1.3.6.1 Basic Features of On-Line Recognition

On-line HWR means that the input is converted in *real-time*, *dynamically*, while the user is writing. This recognition can lag behind the user's writing speed. (Tappert et al. 1990) report average writing rates of 1.5-2.5 characters/s for English alphanumerics or 0.2-2.5 characters/s for Chinese characters. In on-line systems, the data usually comes in as a sequence of coordinate points. Essentially, an on-line system accepts as input a stream of x-y coordinates from an input device that captures those data combined with the appropriate measuring times of those points.

1.3.6.2 Basic Features of Off-Line Recognition

Off-line HWR is the application of a HWR algorithm after the writing. It can be performed at any time after the writing has been completed.

That includes recognition of data transferred from the real-ink pens (see section 1.3.3) to a computing device after the writing has been completed. The standard case of off-line HWR, however, is a subset of optical character recognition (OCR). An scanner transfers the physical image on paper into a bitmap, the character recognition is performed on the bitmap. An OCR system can recognise several hundred characters per second. Images are usually binarised by a threshold of its colour pattern, such that the image pixels are either 1 or 0 (Santosh and Nattee 2009).

1.3.6.3 Similarities and Differences of On-Line and Off-Line Recognition

There are two main differences between on-line and off-line handwriting recognition. *a)* Off-line recognition happens, hence the name, after the time of writing. Therefore, a complete piece of writing can be expected as an input by the machine. *b)*On-line devices also get the dynamic information of the writing as input, since each point coordinate is captured at a specific point of time, which can be provided to the handwriting recogniser along with the point coordinates by the operating system. In addition, the recogniser has information about the input stroke sequence, the stroke direction and the speed of the writing. In the off-line case these pieces of information are not readily available, but can be partially reconstructed from the off-line data (Santosh and Nattee 2009).

All these information can be an advantage for an on-line system, however, off-line systems have used algorithms of line-thinning, such that the data consists of point coordinates, similar to the input of on-line systems (Tappert et al. 1990). When line thinning has been applied, an off-line system could estimate the trajectory of the writing and then use the same algorithm as an on-line system (Plamondon and Srihari 2000). Vice versa, an on-line system can employ algorithms of off-line systems, since it is possible to construct a binary image from mouse coordinates of points. However, only few systems of that kind have been developed. A promising approach was developed in the middle of the 1990'ies by (Nishida 1995), where an on-line and and off-line system are fully integrated with each other as a *blend system*.

On-line systems can refer interactively to the user in case of an unsuccessful or uncertain recognition. Along these lines, an on-line system can adapt to the way a specific user draws certain characters and a user can adapt to the way a system expects characters to be written distinctively.

1.4 A Typical On-Line HWR Application

A typical HWR application has several parts that follow up on each other in a procedural fashion. The main parts of such an application are the following:

- **Data capturing**: The data is captured through an input device like a writing surface and a stylus. Data capturing is described in section 1.4.1.
- **Preprocessing**: The data is segmented, noise reduction like smoothing and filtering are applied. Preprocessing is described in section 1.5.2.1.
- **Character Recognition**: Feature analysis, stroke matching, time, direction and curve matching. A description of character recognition can be found in section 1.4.3.

In a typical HWR application there are some intermediate steps that can be regarded as partial processes of the ones mentioned above. In preprocessing, there is often a segmentation step (see 1.4.2.1), most systems perform one or several methods of noise reduction (see 1.4.2.2) and it is common to employ a data normalisation step (see 1.4.2.3). In character recognition the subprocesses are xxx: see sections that are not subsections of 1.4.3, but fall under character recognition conceptually and post them here. :xxx Additionally, there are some systems that employ a postprocessing step (see 1.4.4).

1.4.1 Data Capturing

Special hardware is needed in order to capture the data necessary for on-line HWR. It is possible to input data with a regular mouse on a regular computer, however the data will be less *noisy*, if a stylus is used for input of the handwriting. See section 1.3.3 for more information on the hardware requirements of HWR. The data that is served by a device driver or the operating system is structured as mouse coordinates. In case of pressure-sensitive input devices the device driver also returns the pressure intensity. However, a typical on-line HWR application uses only the mouse coordinates, not the pressure intensity. Generally, HWR applications work with mouse coordinates, thus the trajectory of the stylus, in accordance with the appropriate time information for each of the sampling points.

xxx: if you want to add more here, see (Santosh and Nattee 2009): *Digitizer Technology* as well as (Tappert et al. 1990): *Digitizer Technology* but don't repeat to much stuff from 1.3.3 :xxx

xxx: The reason for not using pressure intensity as a recognition feature might be that it is plausible to assume that pressure intensity is even more individual to the writer than the shape of the drawn characters. Therefore, featuring pressure intensity in a HWR application could add more noise to the data. However, the differences in pressure intensity between different characters drawn by the same writer should be analysed. [xxx put the speculation about pressure intensity into outlook chapter? something along the lines of progress in device technology, subsection pressure intensity of tablets? Or maybe refer from here to outlook chapter?] :xxx

1.4.1.1 Sampling

The pen-up and pen-down information is a central part in data capturing. Depending on the sampling rate, there will be between 50 and 200 points per second. Those can be viewed as a function of time. With theses pieces of information, it is possible to track the number and order of strokes drawn by the user. According to the writing speed, the number of coordinates per distance can vary. A fast stroke will have fewer point samples, even if the same distance was covered on the writing surface. Some see this as a problem and *re-sample* the points to be of equal distance to each other and to have a constant number of sample points in every stroke (Santosh and Nattee 2009). (Joshi et al. 2005) re-sample the strokes in a way that they yield a constant number of point samples in space, rather than in time. They propose an approach for Devanāgarī [2] characters, where each of the characters is represented by 60 sample points.

1.4.2 Preprocessing

Most On-Line HWR systems apply some kind of preprocessing. Generally, preprocessing serves to smoothen the data, eliminate noise and normalise the data, in order to retrieve a more accurate recognition in the next step. (Tappert et al. 1990) distinguish between the phases *external* and *internal segmentation*, a *noise reduction* step that has several sub steps:
Smoothing, *filtering*, *wild point correction*, *dehooking*, *dot reduction* and *stroke connection*. Some systems also employ a *normalisation* step, that can include *deskewing*, *basline drift correction* and *size normalisation*. (Santosh and Nattee 2009) employ a similar perception in their review about several on-line HWR systems. Their *noise elimination* corresponds to *noise reduction* and the aforementioned *normalisation* describes roughly

[2] Devanāgarī is a script used to write the Sanskrit, Prākrit, Hindi, Marathi, and Nepali languages, developed from the North Indian monumental script known as Gupta and ultimately from the Brāhmī alphabet, from which all modern Indian writing systems are derived (Encyclopædia Britannica Inc. 2009).

the same techniques that (Tappert et al.) depict. (Santosh and Nattee) also present the additional processing step of *repetition removal*, is comparable to the *dot reduction* mentioned above. In the remainder of this section the typical preprocessing steps are presented in more detail.

1.4.2.1 Segmentation

The term *Segmentation* describes the processing step of segmenting the individual characters or strokes from each other.

External Segmentation *External segmentation* is the step of isolating writing units. That can be single strokes, characters or words respectively. In alphabetic scripts it is sometimes difficult to segment each character externally. In CJK cursive script it is difficult to distinguish individual strokes, however, usually each character can be isolated without much processing. The earliest method of external segmentation is an explicit signal from the user, other work used the projection of the trajectories on the X-axis for spatial segmentation. Another method uses temporal information and employs a time-out between the end of one and the beginning of another character. A fairly easy possibility is of course the use of boxed characters, where the user is required to write each character in a separate box. Taking the box idea to the next level of abstraction, some systems provide different boxes on a single screen for different types of characters (Tappert et al. 1990).

Internal Segmentation In cursive script, several strokes are connected together, even several characters can be written without a pen-up movement. Therefore some partial recognition is needed in order to perform the segmentation. In *overlaid handwriting recognition* a sequence of characters is written on the same area, e.g. on a very small display like a wrist watch with a touch screen. In that case, segmentation becomes a problem even for OLCCR systems. (Shimodaira et al. 2003) employ substroke HMMs in a wearable computing environment. They achieved performances of 69.2% with free stroke order and 88% with fixed stroke order characters.

1.4.2.2 Noise Reduction

Any stroke drawn on a device with a pen input contains noise. Digitising errors like limited accuracy of the tablet or erratic pen-down indication or hand fluctuation are sources of noise in the input data. There are different types of noise and different types of filters for elimination of the noise from the data (Santosh and Nattee 2009).

Smoothing *Smoothing* is usually a technique that averages a point with its neighbour point. A common variant of that is only averaging a point with its previous point. That ensures that the recognition can proceed as the input is received (Tappert et al. 1990). (Joshi et al. 2005) employ a 5-tap low pass Gaussian filter in order to smoothen the data. Today, Gaussian filters are often used as a means of smoothing data, despite the fact that *filtering* usually refers to reducing the number of samples in a sequence of points.

Filtering *Filtering* describes a technique that thins out the number of samples in a point sequence. It is sometimes referred to as *thinning* but should not be confused with *line thinning* in off-line HWR. Here, thinning means reducing the number of data samples. Also, duplicate points can be detected and eliminated through filters. The ideal filtering method depends on the recognition method. There are filtering techniques that enforce a minimum distance between points. When a filtering method of that type is used, points tend to be equally spaced. This filtering technique assumes many samples per distance. When a stroke is drawn quickly on a tablet, it can happen, that the distance of the actual sample points is larger than the minimum distance the filter permits. In that case, interpolation can be used in order to introduce new data points in between and achieve equidistant points. Smoothing and filtering can also be performed in one operation for time optimisation (Tappert et al. 1990).

Wild Point Correction *Wild Point Correction* is used for elimination of spurious points. Those points occur usually as a result of errors in data capturing. Acceleration and generally any change of velocity of the hand movement can cause the digitiser tablet to detect wild points (Tappert et al. 1990).

Dehooking Hooks occur at the beginning and more frequently at the end of a stroke. They are recognised by the data capturing device because of inaccurate pen-down detection or random motion when lifting the stylus off the tablet. *Dehooking* is a technique to remove the hooks at the end and at the beginning of a stroke (Tappert et al. 1990).

Repetition Removal *Repetition removal* or *dot reduction* reduces dots to single points (Tappert et al. 1990). Points that are accumulated on a very close area and form a dot, do not improve recognition quality, but rather distort the trajectory. There can even be co-occurring points. Slow handwriting will generate repeating points, and those are generally at the dominant points, such as corners (Santosh and Nattee 2009).

Stroke Connection A *stroke connection* preprocessing method can eliminate pen-up movements that are suspected to be accidental or inaccurately measured. There are different methods to do this, one of which connects strokes if the distance between pen-up and the next pen-down is short in comparison to the size of the character (Tappert et al. 1990).

1.4.2.3 Normalisation

Data *normalisation* is a preprocessing step that prepares data to better match the gold standard and the lexicon entries. It can correct slants or size.

Deskewing is a technique that modifies the slant of an input. Deskewing algorithms can be applied to individual characters or whole words (Tappert et al. 1990). In Chinese and Japanese character recognition, slant correction can be applied in a similar fashion like with latin-based alphabets.

Baseline Drift Correction detects a baseline or horizontal line relative to a word or a set of characters. After detection, that line is corrected to be horizontal (Tappert et al. 1990).

Size Normalisation Adjustment to a standard size can be done with *size normalisation.* This process also normalises for location by relocating the centre of a character (Tappert et al. 1990).

Stroke Length Normalisation sometime called *resampling* normalises the number of points in a stroke or character. Dealing with strokes that were normalised in such a way provides for easy alignment and classification (Tappert et al. 1990).

1.4.3 Character Recognition

xxx: see Plamondon2000: 3.1.1 different models xxx: see all the substroke stuff, Santosh, shimodaira2003, nakai2003: very short, properly in OLCCR

Character recognition is a special case of shape recognition. It is the recognition of the shape of writing units. Character recognition falls into different categories, there are methods for the recognition of characters, cursive script, words, gestures, equations, line drawings. Also, there are several different methods for the recognition of the shape of a character. The most common methods include

- **Feature Analysis**, where the distinctive features of characters are taken into account (see 1.4.3.1).
- **Zone Sequences** that code zones on the writing surface (see 1.4.3.2).
- **Stroke Direction** - the individual strokes or substrokes of a character are identified and matched against a stroke sequence database (see 1.4.3.3).
- **Curve Matching** is a method that comes from signal processing and is essentially independent of the alphabet (see 1.4.3.4).
- **HMM Analysis** is independent of the other methods. Basically HMM analysis performs a statistical analysis of the features used, regardless of what they are, i.e. curves, time sequences or writing features of the characters themselves (see 1.5.2.2).

1.4.3.1 Feature Analysis

A character or shape can be represented by a set of features. These features are dependent on the alphabet. Therefore, systems can employ a pre-made analysis of the individual characters of the alphabet. For the Latin alphabet, the systems use features like descender or not descender, dot or no dot (Tappert et al. 1990). Some systems use a decision tree for **binary features**. If there is a descender, for lowercase characters, the recognition choice is reduced to *f, g, j, p, q, y, z*. If an ascender is present, for lowercase characters, the choice is reduced to *b, d, f, h, k, l, t*. In case there is an ascender and a descender, the intersection of the two sets leaves only one element, namely the character *f*. A HWR system can come to similar conclusions by checking if a dot is present *i, j*, if the character has an open line *c, u, v, w, y* or a closed line *a, b, d, e, g, o, p, q*. Systems that perform feature analysis have the potential to use psycholinguistic knowledge and drive the recognition process along the lines humans distinguish characters from each other. The feature analysis method has the disadvantage that it may not produce alternative recognition choices.

There are also systems that use **non-binary features**. It is a very common technique in pattern recognition to use a fixed number of non-binary features. A feature space of that kind can be further divided into decision regions (Tappert et al. 1990).

1.4.3.2 Zone Sequences

There are systems that use *zone sequences* in order to represent characters. The zones can be specified by sub-dividing the bounding box of a character. The trajectory of the pen that is drawn through the zones is then transformed into a time sequence of zones. That time sequence can be matched against a dictionary of sequences, in which each character is represented (Tappert et al. 1990).

1.4.3.3 Stroke Direction

Stroke direction is another popular feature among HWR systems. The pen-tip motion is captured and each stroke is categorised as a primitive direction (up, down, left, right, other systems also use the diagonal directions and a pen-down, pen-up). (Nakai et al. 2001) and (Tokuno et al. 2002) use the stroke direction feature combined with a pen-coordinate feature. (Okumura et al. 2005) and others follow a similar approach. The substroke approach is very popular in OLCCR, it is often combined with HMMs (see 1.5.2.2).

1.4.3.4 Curve Matching

Curve matching is a set of methods that has been popularised in signal processing. Prototype characters are stored in a data base as descriptions of curves. During the recognition process the input curves are matched against the prototypes. It is common to use curves that are functions of time, the direction of the tangent to the stroke or both. Characters have been encoded as a time sequence of eight stroke directions, and as time regions (Tappert et al. 1990). Chinese characters have been encoded as a small number of fixed points, since they consist of mainly straight strokes (Nakagawa et al. 2008).

Elastic matching Curve matching and pattern matching become equivalent in their feature space when there is a constant number of points in a curve and in one-to-one correspondence. However, since many strokes are non-linear the best fit is often not a linear alignment of the points. In order to solve point sequence comparison problem, elastic matching has been used successfully (Tappert et al. 1990). (Joshi et al. 2004) compare different elastic matching methods for Tamil handwriting recognition. *Dynamic Time Warping* (DTW) is an elastic matching technique that has been applied to handwriting recognition by a number of systems e.g. (VuoriADDPROPERREFERENCESTOVUORI 2525), (Niels 2004), (Joshi et al. 2005) and (Bashir and Kempf 2008). The DTW algorithm yielded promising results, especially for the different scripts of the Indian subcontinent, however, due to its complexity, (Keogh and Pazzani 2002), (Chu et al. 2002) as well as (Bashir and Kempf 2008) proposed improvements to the algorithm, mainly through pruning.

1.4.4 Postprocessing

When the recognition process is finished, some systems start another process that takes as input the output of the character recognition. In this postprocessing step, language information can be used to improve the recognition accuracy. Postprocessing is mainly used for continuous character recognition, where more than a single charaters is recgonised. Since some systems yield a single string of characters others yield a number of alternative recognitions, often in addition with a certainty measure. A postprocessing unit can use the information to calculate estimates for words or even sentences. When the character recogniser yields single choices for characters, the postprocessor can apply a correction algorithm, based on a language model (Tappert et al. 1990). (Shimodaira et al. 2003) proposed a method for overlaid handwriting recognition, where the segmentation problem (see 1.4.2.1) is resolved at the end of the recognition process: Alternative choices of characters and character boundaries are generated and probability estimations calculate the moste likely sequence of characters, using both the alterative recognition results and a language model for Japanese.

1.5 HWR of Hànzì (汉字) and Kanji (漢字)

The HWR of the Chinese Hànzì (汉字) and the Japanese Kanji (漢字) merged as Online Chinese Character Recognition (OLCCR). The techniqes are essentially the same, only the language models differ. Hereafter, I'm going to use the term OLCCR for online character recognition of both Chinese and Japanese characters.

From the 1990'ies, On-Line Japanese and Chinese Character Recognition (OLCCR) systems have been aiming at loosening the restrictions imposed on the writer when using an OLCCR system. Their focus shifted from recognition of block style script ('regular' script) to fluent style script, which is also called 'cursive' style. Accuracies of up to about 95% are achieved in the different systems.

(Nakagawa et al. 2008) report their recent results of on-line Japanese handwriting recognition and its applications. Their article gives important insights into character modelling, which are employed in this application. Multiple subpatterns of characters are detected and used for character modelling.