Chapter 10

Stroke-Based Cursive Character Recognition

K.C. Santosh*

INRIA Nancy Grand Est Research Centre

France

Eizaburo Iwata

Universal Robot Co. Ltd.

Japan

1. Introduction

Human eye can see and read what is written or displayed either in natural handwriting or in printed format. The same work in case the machine does is called handwriting recognition. Handwriting recognition can be broken down into two categories: off-line and on-line.

Off-line character recognition – Off-line character recognition takes a raster image from a scanner (scanned images of the paper documents), digital camera or other digital input sources. The image is binarised based on for instance, color pattern (color or gray scale) so that the image pixels are either 1 or 0.

On-line character recognition – In on-line, the current information is presented to the system and recognition (of character or word) is carried out at the same time. Basically, it accepts a string of (x, y) coordinate pairs from an electronic pen touching a pressure sensitive digital tablet.

In this chapter, we keep focusing on on-line writer independent cursive character recognition engine. In what follows, we explain the importance of on-line handwriting recognition over off-line, the necessity of writer independent system and the importance as well as scope of cursive scripts like Devanagari. Devanagari is considered as one of the known cursive scripts Jayadevan et al. (2011); Pal & Chaudhuri (2004). However, we aim to include other scripts related to the current study.

*Corresponding author: Santosh.KC@inria.fr

Authors' copy, 2012

1.1 Why On-line?

With the advent of handwriting recognition technology since a few decades Arica & Yarman-Vural (2001); Plamondon & Srihari (2000), applications are challenging. For example, OCR is becoming an integral part of document scanners, and is used in many applications such as postal processing, script recognition, banking, security (signature verification, for instance) and language identification. In handwriting recognition, feature selection has been an important issue ∅ivind Due Trier et al. (1996). Both structural and statistical features as well as their combination have been widely used Foggia et al. (1999); Heutte et al. (1998). These features tend to vary since characters' shapes vary widely. As a consequence, local structural properties like intersection of lines, number of holes, concave arcs, end points and junctions change time to time. These are mainly due to

- *deformations* can be from any range of shape variations including geometric transformation such as translation, rotation, scaling and even stretching; and
- *defects* yield imperfections due to printing, optics, scanning, binarisation as well as poor segmentation.

In the state-of-the-art of handwritten character recognition, several different studies have shown that off-line handwriting recognition offers less classification rate compared to on-line Plamondon & Srihari (2000); Tappert et al. (1990). Furthermore, on-line data offers significant reduction in memory and therefore space complexity. Another advantage is that the digital pen or a digital form on a tablet device immediately transforms your handwriting into a digital representation that can be reused later without having any risk of degradation usually associated with ancient handwriting. Based on all these reasons, one can cite a few examples Boccignone et al. (1993); Doermann & Rosenfeld (1995); Qiao et al. (2006); Viard-Gaudin et al. (2005) where they mainly focus on temporal information as well as writing order recovery from static handwriting image. On-line handwriting recognition systems provide interesting results.

On-line character recognition involves the automatic conversion of stroke as it is written on a special digitizer or PDA, where a sensor picks up the pen-tip movements as well as pen-up/pen-down switching. Such data is known as digital ink and can be regarded as a dynamic representation of handwriting. The obtained signal is converted into letter codes which are usable within computer and character-processing applications.

The elements of an on-line handwriting recognition interface typically include:

1. a pen or stylus for the user to write with, and a touch sensitive surface, which may be integrated with, or adjacent to, an output display.
2. a software application i.e., a recogniser which interprets the movements of the stylus across the writing surface, translating the resulting strokes into digital character.

Globally, it resembles one of the applications of pen computing i.e., computer user-interface using a pen (or stylus) and tablet, rather than devices such as a keyboard, joysticks or a mouse. Pen computing can be extended to the usage of mobile devices such as wireless tablet personal computers, PDAs and GPS receivers.

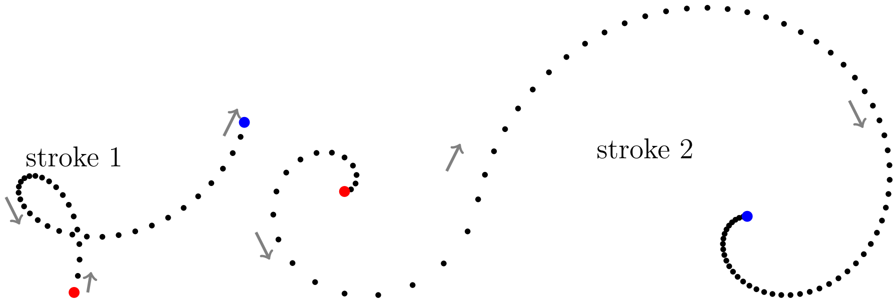


Fig. 1: On-line stroke sequences in the form of 2D (x, y) coordinates. In this illustration, initial pen-tip position is coloured with red and pen-up (final point) is coloured with blue.

Historically, pen computing (defined as a computer system employing a user-interface using a pointing device plus handwriting recognition as the primary means for interactive user input) predates the use of a mouse and graphical display by at least two decades, starting with the Stylator Dimond (1957) and RAND tablet Groner (1966) systems of the 1950s and early 1960s.

1.2 Why Writer Independent?

As mentioned before, on-line handwriting recognition systems provide interesting results almost over all types scripts. The recognition systems vary widely which can be due to nature of the scripts employed along with the associated particular difficulties including the intended applications. The performance of the application-based (commercial) recogniser is used to determine by its speed in addition to accuracy.

Among many, more specifically, template based approaches have a long standing record Bahlmann & Burkhardt (2004); Connell & Jain (1999); Hu et al. (1996); Santosh & Nattee (2006a); Schenkel et al. (1995). In many of the cases, writer independent recogniser has been made since every new user does not require training – which is widely acceptable. In such a context, the expected recognition system should automatically update or adapt the new users once they provide input or previously trained recogniser should be able to discriminate new users.

1.3 Why Devanagari?

In a few points, interesting scope will be summarised.

1. Pencil and paper can be preferable for anyone during a first draft preparation instead of using keyboard and other computer input interfaces, especially when writing in languages and scripts for which keyboards are cumbersome. Devanagari keyboards for instance, are quite difficult to use. Devanagari characters follow a complex structure and may count up to more than 500 symbols Jayadevan et al. (2011); Pal & Chaudhuri (2004).
2. Devanagari is a script used to write several Indian languages, including Nepali, Sanskrit, Hindi, Marathi, Pali, Kashmiri, Sindhi, and sometimes Punjabi. According to the 2001 Indian census, 258 million people in India used Devanagari.

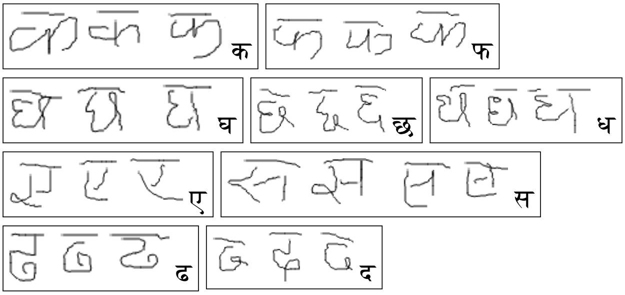


Fig. 2: A few samples of several different similar classes from Devanagari script.

3. Writing one's own style brings unevenness in writing units, which is the most difficult part to recognise. Variation in basic writing units such as number of strokes, their order, shapes and sizes, tilting angles and similarities among classes of characters are considered as the important issues. In contrast to Roman script, it happens more in cursive scripts like Devanagari.

   Devanagari is written from left to right with a horizontal line on the *top* which is the *shirorekha*. Every character requires one *shirorekha* from which text(s) is(are) suspended. The way of writing Devanagari has its own particularities. In what follows, in particular, we shortly explain a few major points associated difficulties.

   - Many of the characters are similar to each other in structure. Visually very similar *symbols* – even from the same writer – may represent different *characters*. While it might seem quite obvious in the following examples to distinguish the first from the second, it can easily be seen that confusion is likely to occur for their handwritten *symbol* counterparts (क, फ), (य, प), (ढ, द), etc.). Fig. 2 shows a few examples of it.
   - The number of strokes, their order, shapes and sizes, directions, skew angle etc. are writing units that are important for symbol recognition and classification. However, these writing units most often vary from one user to another and there is even no guarantee that a same user always writes in a same way. Proposed methods should take this into account.

Based on those major aforementioned reasons, there exists clear motivation to pursue research on Devanagari handwritten character recognition.

1.4 Structure of the Chapter

The remaining of the paper is organised as follows. In Section 2, we start with detailing the basic concept of character recognition framework in addition to the major highlights on important issues: feature selection, matching and recognition. Section 3 gives a complete outline of how we can efficiently handle optimal recognition performance over cursive scripts like Devangari. In this section, we first provide the complete and then validate the whole process step by step with genuine reasoning and a series of experimental tests over our own dataset but, publicly available. We conclude the chapter in Section 4.

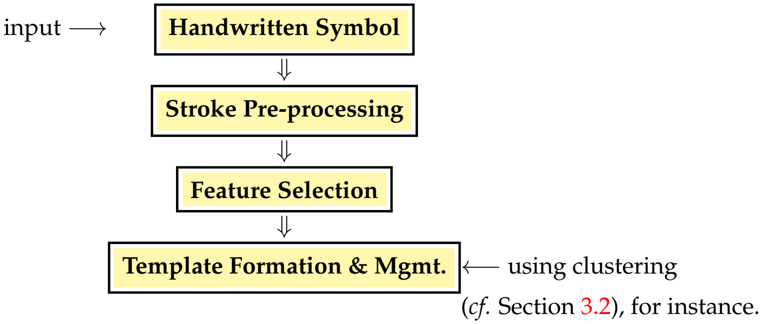


Fig. 3: Learning strokes from the handwritten symbols. In this illustration, we present a basic concept to form template via clustering of features of the strokes immediately after they are pre-processed.

2. Character Recognition Framework

Basically, we can categorise character recognition system into two modules: learning and testing. In learning or training module, following Fig. 3, handwritten strokes are learnt or stored. Testing module follows the former one. The performance of the recognition system is depends on how well handwritten strokes are learnt. It eventually refers to the techniques we employ.

Basically, learning module employs stroke pre-processing, feature selection and clustering to form template to be stored. Pre-processing and feature selection techniques can be varied from one application to another. For example, noisy stroke elimination or deletion in Roman cannot be directly extended to the cursive scripts like Urdu and Devanagari. In other words, these techniques are found to be application dependent due to their different writing styles. However, they are basically adapted to each other and mostly ad-hoc techniques are built so that optimal recognition performance is possible. In the framework of stroke-based feature extraction and recognition, one can refer to Chiu & Tseng (1999); Zhou et al. (2007), for example. It is important to notice that feature selection usually drives the way we match them. As an example, fixed size feature vectors can be straightforwardly matched while for non-linear feature vector sequences, dynamic programming (elastic matching) has been basically used Keogh & Pazzani (1999); Kruskall & Liberman (1983); Myers & Rabiner. (1981); Sakoe (1978). The concept was first introduced in the 60’s Bellman & Kalaba (1959). Once we have an idea to find the similarity between the strokes’ features, we follow clustering technique based on their similarity values. The clustering technique will generate templates as the representative of the similar strokes provided. These stored templates will be used for testing in the testing module. Fig. 4 provides a comprehensive idea of it (testing module). More specifically, in this module, every test stroke will be matched with the templates (learnt in training module) so that we can find the most similar one. This procedure will be repeated for all available test strokes. At the end, aggregating all matching scores provides an idea of the test character closer to which one in the template.

2.1 Preprocessing

Strokes directly collected from users are often incomplete and noisy. Different systems use a variety of different pre-processing techniques before feature extraction Alginahi (2010);

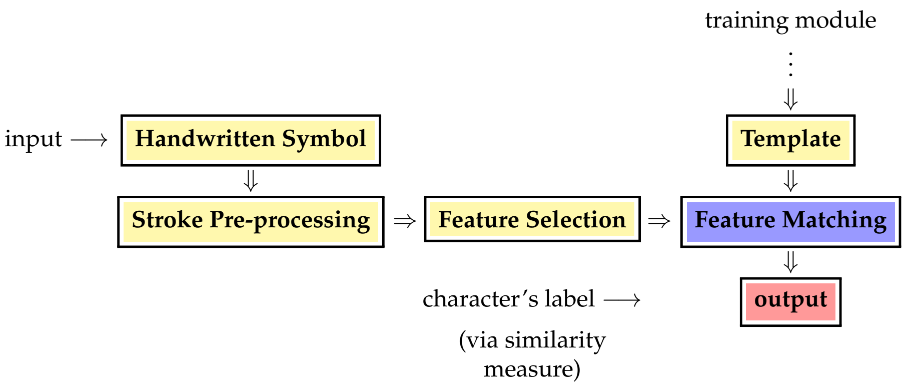


Fig. 4: An illustration of testing module. As in learning module, test characters are pre-processed and we present a basic concept to form template via clustering of features of the strokes immediately after they are pre-processed.

Blumenstein et al. (2003); Verma et al. (2004). The techniques used in one system may not exactly fit into the other because of different writing styles and nature of the scripts. Very common issues are repeated coordinates deletion Bahlmann & Burkhardt (2004), noise elimination and normalisation Chun et al. (2005); Guerfali & Plamondon (1993).

Besides pre-processing, in this chapter, we mainly focus on feature selection and matching techniques.

2.2 Feature Selection

If you have complete address of your friend then you can easily find him/her without an additional help from other people on the way. The similar case is happened in character recognition. Here, an address refers to a feature selection. Therefore, the complete or sufficient feature selection from the provided input is the crucial point. In other words, appropriate feature selection can greatly decrease the workload and simplify the subsequent design process of the classifier.

In what follows, we discuss a few but major issues associated with feature selection.

- Pen-flow i.e., speed while writing determines how well the coordinates along the pen trajectory are captured. Speed writing and writing with shivering hands, do not provide complete shape information of the strokes.
- Ratios of the relative height, width and size of letters are not always consistent - which is obvious in natural handwriting.
- Pen-down and pen-up events provide stroke segmentation. But, we do not know which and where the strokes are rewritten or overwritten.
- Slant writing style or writing with some angles to the left or right makes feature selection difficult. For example, in those cases, zoning information using orthogonal projection does not carry consistent information. This means that the zoning features will vary widely as soon as we have different writing styles.

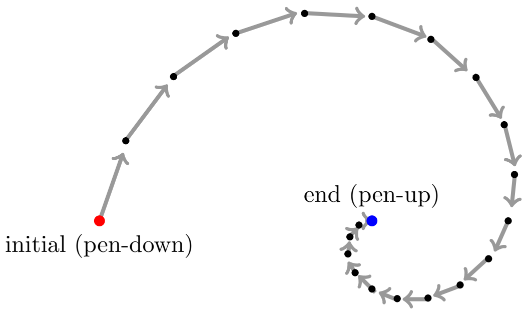


Fig. 5: An illustration of feature selection: pen-tip position and tangent at every pen-tip position along the pen trajectory.

We repeat, features should contain sufficient information to distinguish between classes, be insensitive to irrelevant variability of the input, allow efficient computation of discriminant functions and be able to limit the amount of training data required Lippmann (1989). However, they vary from one script to another Blumenstein et al. (2003); Namboodiri & Jain (2004); Okumura et al. (2005); Verma et al. (2004).

Feature selection is always application dependent i.e., it relies on what type of scripts (their characteristics and difficulties) used. In our case, we use a feature vector sequence of any stroke is expressed as in Okumura et al. (2005); Santosh & Nattee (2006a); Santosh, Nattee & Lamiroy (2012):

$$\mathbf{F} = \left[\left(\mathbf{p}_1, \alpha_{\mathbf{p}_1,\mathbf{p}_2}\right), \left(\mathbf{p}_2, \alpha_{\mathbf{p}_2,\mathbf{p}_3}\right), \ldots, \left(\mathbf{p}_{l-1}, \alpha_{\mathbf{p}_{l-1},\mathbf{p}_l}\right) \right] \quad (1)$$

where, $\alpha_{\mathbf{p}_{l-1},\mathbf{p}_l} = \arctan\left(\frac{y_l - y_{l-1}}{x_l - x_{l-1}}\right)$. Fig. 5 shows a complete illustration.

Our feature includes a sequence of both pen-tip position and tangent angles sampled from the trajectory of the pen-tip, preserving the directional property of the trajectory path. It is important to remind that stroke direction (either left – right or right – left) leads to very different features although they are geometrically similar. To efficiently handle it, we need both kinds of strokes or samples for training and testing. This does not mean that same writer must be used.

The idea is somehow similar to the directional arrows that are composed of eight types, coded from $0 - 7$. This can be expressed as, $\begin{array}{ccc} \nwarrow & \uparrow & \nearrow \\ \leftarrow & \circ & \rightarrow \\ \swarrow & \downarrow & \searrow \end{array}$.

However, these directional arrows provide only the directional feature of the strokes or line segments. Therefore, more information can be integrated if the relative length of the standard strokes is taken into account Cha et al. (1999).

2.3 Feature Matching

Besides, discussing on classifiers, we explain how features can be matched to obtain similarity or dissimilarity values between them.

Matching techniques are often induced by how features are taken or strokes are represented. For instance, normalising the feature vector sequence into a fixed size vector provides an immediate matching. On the other hand, features having different lengths or non-linear features need dynamic programming for approximate matching, for instance. Considering the latter situation, we explain how dynamic programming is employed.

Dynamic time warping (DTW) allows us to find the dissimilarity between two non-linear sequences potentially having different lengths Keogh & Pazzani (1999); Kruskall & Liberman (1983); Myers & Rabiner. (1981); Sakoe (1978). It is an algorithm particularly suited to matching sequences with missing information, provided there are long enough segments for matching to occur.

Let us consider two feature sequences

$$\mathbf{X} = \{\mathbf{x}_k\}_{k=1,\ldots,K} \text{ and}$$
$$\mathbf{Y} = \{\mathbf{y}_l\}_{l=1,\ldots,L}$$

of size K and L, respectively. The aim of the algorithm is to provide the optimal alignment between both sequences. At first, a matrix M of size $K \times L$ is constructed. Then for each element in matrix M, local distance metric $\delta(k,l)$ between the events e_k and e_l is computed i.e., $\delta(k,l) = (e_k - e_l)^2$. Let $D(k,l)$ be the global distance up to (k,l),

$$D(k,l) = \min \begin{bmatrix} D(k-1,l-1), \\ D(k-1,l), \\ D(k,l-1) \end{bmatrix} + \delta(k,l)$$

with an initial condition $D(1,1) = \delta(1,1)$ such that it allows warping path going diagonally from starting node $(1,1)$ to end (K,L). The main aim is to find the path for which the least cost is associated. The warping path therefore provides the difference cost between the compared signatures. Formally, the warping path is,

$$\mathcal{W} = \{w_t\}_{t=1\ldots T},$$

where $\max(k,l) \leq T < k+l-1$ and t^{th} element of $\mathcal{W}$ is $w(k,l)_t \in [1:K] \times [1:L]$ for $t \in [1:T]$. The optimised warping path $\mathcal{W}$ satisfies the following three conditions.

c1. boundary condition:

$$w_1 = (1,1) \text{ and } w_T = (K,L).$$

c2. monotonicity condition:

$$k_1 \leq k_2 \leq \cdots \leq k_K \text{ and } l_1 \leq l_2 \leq \cdots \leq l_L.$$

c3. continuity condition:

$$w_{t+1} - w_t \in \{(1,1)(0,1),(1,0)\} \text{ for } t \in [1:T-1].$$

c1 conveys that the path starts from $(1,1)$ to (K,L), aligning all elements to each other. **c2** forces the path advances one step at a time. **c3** restricts allowable steps in the warping path to adjacent cells, never be back. Note that **c3** implies **c2**.

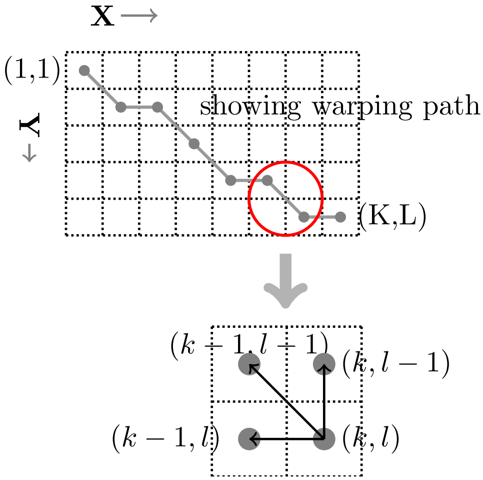


Fig. 6: Classical DTW algorithm – an alignment illustration between two non-linear sequences **X** and **Y**. In this illustration, diagonal DTW-matrix is shown including how back-tracking has been employed.

We then define the global distance between **X** and **Y** as,

$$\Delta\left(\mathbf{X}, \mathbf{Y}\right) = \frac{D(K, L)}{T}.$$

The last element of the $K \times L$ matrix gives the DTW-distance between **X** and **Y**, which is normalised by T i.e., the number of discrete warping steps along the diagonal DTW-matrix. The overall process is illustrated in Fig. 6.

Until now, we provide a global concept of using DTW distance for non-linear sequences alignment. In order to provide faster matching, we have used local constraint on time warping proposed in Keogh (2002). We have $w(k, l)_t$ such that $l - r \leq k \leq l + r$ where r is a term defining a reach i.e., allowed range of warping for a given event in a sequence. With r, upper and lower bounding measures can be expressed as,

$$\begin{aligned} \text{Upper bound } U_k &= \max(\mathbf{x}_{k-r} : \mathbf{x}_{k+r}) \\ \text{Lower bound } L_k &= \min(\mathbf{x}_{k-r} : \mathbf{x}_{k+r}). \end{aligned}$$

Therefore, for all i, an obvious property of U and L is $U_k \geq \mathbf{x}_k \geq L_k$. With this, we can define a lower bounding measure for DTW:

$$\text{LB\_Keogh}(\mathbf{X}, \mathbf{Y}) = \sqrt{\sum_{k=1}^{K} \begin{cases} (\mathbf{y}_k - U_k)^2 & \text{if } \mathbf{y}_k > U_k \\ (\mathbf{y}_k - L_k)^2 & \text{if } \mathbf{y}_k < L_k \\ 0 & \text{otherwise.} \end{cases}}$$

Since this provides a quick introduction of local constraint for lower bounding measure, we refer to Keogh (2002) for more clarification.

2.4 Recognition

From a purely combinatorial point of view, measuring the similarity or dissimilarity between two symbols

$$\mathbf{S}_1 = \left\{\mathbf{s}_1^i\right\}_{i=1\dots n} \text{ and } \mathbf{S}_2 = \left\{\mathbf{s}_2^j\right\}_{j=1\dots m}$$

composed, respectively, of n and m strokes, requires a one by one matching score computation of all strokes $\mathbf{s}_1^i$ with all $\mathbf{s}_2^j$. This means that we align individual test strokes of an unknown symbols with the learnt strokes. As soon as we determine the test strokes associated with the known class, the complete symbol can be compared by the fusion of matching information from all test strokes. Such a concept is fundamental under the purview of stroke-based character recognition.

Overall, the concept may not always be sufficient, and these approaches generally need a final, global coherence check to avoid matching of strokes that shows visual similarity but do not respect overall geometric coherence within the complete handwritten character. In other words, matching strategy that happens between test stroke and templates of course, should be intelligent rather than straightforward one-to-many matching concepts. However, it in fact, depends on how template management has been made. In this chapter, this is one of the primary concerns. We highlight the use of relative positioning of the strokes within the handwritten symbol and its direct impact to the performance Santosh, Nattee & Lamiroy (2012).

3. Recognition Engine

To make the chapter coherence as well as consistent (to Devanagari character recognition), it refers to the recognition engine which is entirely based on previous studies or works Santosh & Nattee (2006a;b; 2007); Santosh et al. (2010); Santosh, Nattee & Lamiroy (2012). Especially because of the structure of Devanagari, it is necessary to pay attention to the appropriate structuring of the strokes to ease and speed up comparison between the symbols, rather than just relying on global recognition techniques that would be based on a collection of strokes Santosh & Nattee (2006a). Therefore, Santosh et al. (2010); Santosh, Nattee & Lamiroy (2012) develop a method for analysing handwritten characters based on both the number of strokes and the their spatial information. It consists in four main phases.

step 1. Organise the symbols representing the same character into different groups based on the number of strokes.

step 2. Find the spatial relation between strokes.

step 3. Agglomerate similar strokes from a specific location in a group.

step 4. Stroke-wise matching for recognition.

For more clear understanding, we explain the aforementioned steps as follows. For a specific class of character, it is interesting to notice that writing symbols with the equal number of strokes, generally produce visually similar structure and is easier to compare.

In every group within a particular class of character, a representative symbol is synthetically generated from pairwise similar strokes merging, which are positioned identically with respect to the *shirorekha*. It uses DTW algorithm. The learnt strokes are then stored accordingly. It is mainly focused on stroke clustering and management of the learnt strokes.

We align individual test strokes of an unknown symbols with the learnt strokes having both same number of strokes and spatial properties. Overall, symbols can be compared by the fusion of matching information from all test strokes. This eventually build a complete recognition process.

3.1 Stroke Spatial Description and its Need

The importance of the location of the strokes is best observed by taking a few pairs of characters that often lead to confusion:

(भ ↔ म), (ध ↔ घ), (थ ↔ य) etc.

The first character in every pair has visually two distinguishing features: its particular location of the *shirorekha* (more to the right) and a small curve in the text. There is no doubt that one of the two features is sufficient to automatically distinguish both characters. However, small curves are usually not robust feature in natural handwriting, finding the location of the *shirorekha* only can avoid possible confusion. Our stroke based spatial relation technique is explained further in the following.

To handle relative positioning of strokes, we use six spatial predicates i.e., 2×3 relational regions:

$$\mathcal{R} = \begin{bmatrix} \textit{top-left} \text{ (T–L)} & \textit{top} \text{ (T)} & \textit{top-right} \text{ (T–R)} \\ \textit{bottom-left} \text{ (B–L)} & \textit{bottom} \text{ (B)} & \textit{bottom-right} \text{ (B–L)} \end{bmatrix}.$$

For easier understanding, iconic representation of the aforementioned relational matrix $\mathcal{R}$ can be expressed as,

$$\begin{bmatrix} \circ & \circ & \circ \\ \circ & \circ & \bullet \end{bmatrix}$$

where black-dot represents the presence i.e., stroke is found to be in the provided *bottom-right* region.

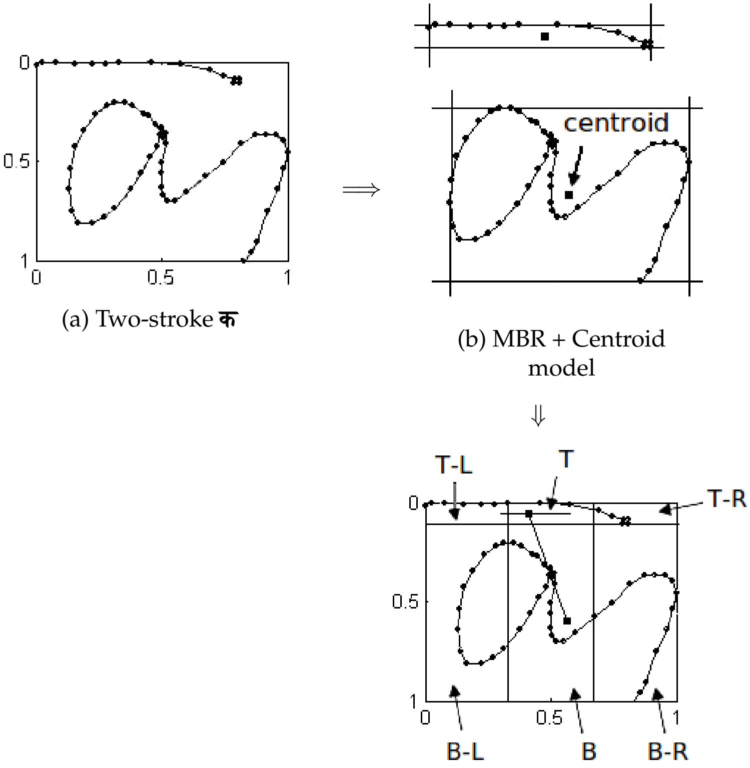
To confirm the location of the stroke, we use the projection theory: minimum boundary rectangle (MBR) Papadias & Sellis (1994) model combined with the stroke's centroid.

Based on Egenhofer & Herring (1991), we start with checking fundamental topological relations such as *disconnected* (DC), *externally connected* (EC) and *overlap/intersect* (O/I) by considering two strokes $\mathbf{s^j}$ and $\mathbf{s^{j'}}$:

$$\mathbf{s^j} = \left\{\mathbf{p}_k^j\right\}_{k=1\ldots l} \text{ and } \mathbf{s^{j'}} = \left\{\mathbf{p}_{k'}^{j'}\right\}_{k'=1\ldots l'}$$

as follows,

$$\mathbf{s^j} \cap \mathbf{s^{j'}} = \begin{cases} 1 \text{ if } (\mathbf{p}_k^j \cap \mathbf{p}_{k'}^{j'} \neq \emptyset) \Rightarrow \text{EC, O/I} \\ 0 \text{ otherwise} \Rightarrow \text{DC.} \end{cases}$$



(a) Two-stroke क

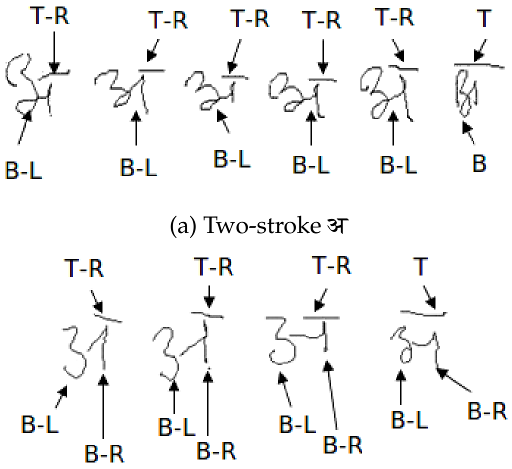
(b) MBR + Centroid model

(c) Model realisation

Fig. 7: Pairwise spatial relation for a two-stroke क.

We then use the border condition from the geometry of the MBR. It is straightforward for *disconnected* strokes while, is not for *externally connected* and *overlap/intersect* configurations. In the latter case, we check the level of the centroid with respect to the boundary of the MBR. For example, if a boundary of the *shirorekha* is above the centroid level of the *text* stroke, then it is confirmed that the *shirorekha* is on the *top*. This procedure is applied to all of the six previously mentioned spatial predicates. Note that use of angle-based model like bi-centre Miyajima & Ralescu (1994) and angle histogram Wang & Keller (1999) are not the appropriate choice due to the cursive nature of writing.

On the whole, assuming that the *shirorekha* is on the *top*, the locations of the *text* strokes are estimated. This eventually allows to cross-validate the location of the *shirorekha* along with its size, once *texts*' locations are determined. Fig. 7 shows a real example demonstrating relative positioning between the strokes for a two-stroke symbol क. Besides, symbols with two *shirorekha*s are also possible to treat. In such a situation, the first *shirorekha* according to the order of strokes is taken as reference.



(a) Two-stroke अ

(b) Three-stroke अ

Fig. 8: Relative positions of strokes for a class अ in two different groups i.e., two-stroke and three-stroke symbols.

3.2 Spatial Similarity based Clustering

Basically, clustering is a technique for collecting items which are similar in some way. Items of one group are dissimilar with other items belonging to other groups. Consequently, it makes the recognition system compact. To handle this, we present spatial similarity based stroke clustering.

As mentioned in previous work Santosh et al. (2010); Santosh, Nattee & Lamiroy (2012), the clustering scheme is a two-step process.

- The first step is to organise symbols representing a same character into different groups, based on the number of strokes used to complete the symbol. Fig. 8 shows an example of it for a class of character अ.
- In the second step, strokes from the specific location are agglomerated hierarchically within the particular group. Once relative position for every stroke is determined as shown in Fig. 8, single-linkage agglomerative hierarchical clustering is used (*cf.* Fig. 10). This means that only strokes which are at a specific location are taken for clustering. As an example, we illustrate it in Fig. 9. This applies to all groups within a class.

In agglomerative hierarchical clustering (*cf.* Fig. 10), we merge two similar strokes and find a new cluster. The distance computation between two strokes follows Section 2.3. The new cluster is computed by averaging both strokes via the use of the discrete warping path along the diagonal DTW-matrix. This process is repeated until it reaches the cluster threshold. The threshold value yields the number of cluster representatives i.e., learnt templates.

$$\underbrace{\underbrace{\begin{bmatrix}\circ&\circ&\circ\\\bullet&\circ&\circ\end{bmatrix}}_{1},\underbrace{\begin{bmatrix}\circ&\circ&\circ\\\circ&\bullet&\circ\end{bmatrix}}_{2},\underbrace{\begin{bmatrix}\circ&\circ&\circ\\\circ&\bullet&\circ\end{bmatrix}}_{3}}_{\textit{text}\text{ clustering}}\text{ and }\underbrace{\underbrace{\begin{bmatrix}\bullet&\circ&\circ\\\circ&\circ&\circ\end{bmatrix}}_{1},\underbrace{\begin{bmatrix}\circ&\bullet&\circ\\\circ&\circ&\circ\end{bmatrix}}_{2},\underbrace{\begin{bmatrix}\circ&\circ&\bullet\\\circ&\circ&\circ\end{bmatrix}}_{3}}_{\textit{shirorekha}\text{ clustering}}$$

Fig. 9: Clustering technique for each class. Stroke clustering is based on the relative positioning. As a consequence, we have three clustering blocks for *text* strokes and remaining three for *shirorekha*.

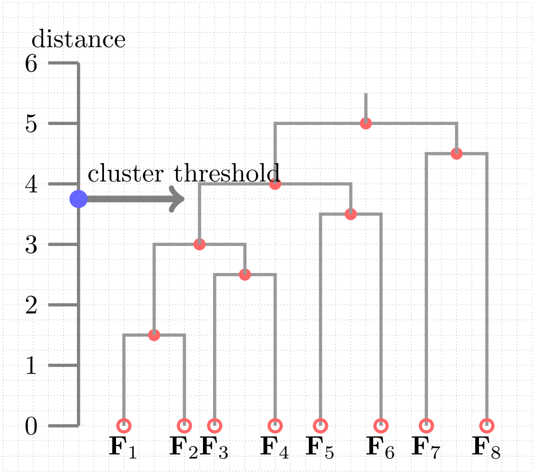


Fig. 10: Hierarchical stroke clustering concept. At every step, features are merged according to their similarity up to the provided threshold level.

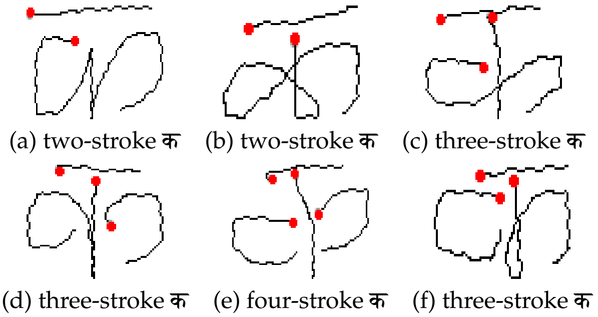
3.3 Stroke Number and Order Free Recognition

In natural handwriting, number of strokes as well as their order vary widely. This happens from one writing to another, even from the same user – which of course exits from different users. Fig. 11 shows the large variation of stroke numbers as well as the orders.

Once we have organised the symbols (from the particular class) into groups based on the number of strokes used, our stroke clustering has been made according to the relative positioning. As a consequence, while doing recognition, one can write symbol with any numbers and orders because stroke matching is based on relative positioning of the strokes in which group while it does not need to care about the strokes order.

3.4 Dataset

In this work, as before, publicly available dataset has been employed (*cf.* Table 1) where a Graphite tablet (WCACOM Co. Ltd.), model ET0405A-U, was used to capture the pen-tip position in the form of $2D$ coordinates at the sampling rate of 20 Hz. The data set is composed of 1800 symbols representing 36 characters, coming from 25 native speakers. Each writer



(a) two-stroke क (b) two-stroke क (c) three-stroke क

(d) three-stroke क (e) four-stroke क (f) three-stroke क

Fig. 11: Different number of strokes and order for a class क. In this illustration, red-dot refers to the initial pen-tip position so that it makes easy to realise how many number of strokes to make a complete symbol. In addition, stroke ordering is different from one to another.

Table 1: Dataset formation and its availability.

Item	Description
Classes of character	36
Users	25
Dataset size	1800
Visibility	IAPR tc–11 http://www.iapr-tc11.org

was given the opportunity to write each character twice. No other directions, constraints, or instructions were given to the users.

3.5 Recognition Performance Evaluation

While experimenting, every test sample is matched with training candidates and the closest one is reported. The closest candidate corresponds to the labelled class, which we call 'character recognition'. Formally, recognition rate can be defined as the number of correctly recognised candidates to the total number of test candidates.

To evaluate the recognition performance, two different protocols can be employed:

1. dichotomous classification and
2. $\mathbb{K}$-fold cross-validation (CV).

In case of dichotomous classification, 15 writers are used for training and the remaining 10 are for testing. On the other hand, $\mathbb{K}$-fold CV has been implemented. Since we have 25 users for data collection, we employ $\mathbb{K} = 5$ in order to make recognition engine writer independent.

In $\mathbb{K}$-fold CV, the original sample for every class is randomly partitioned into $\mathbb{K}$ sub-samples. Of the $\mathbb{K}$ sub-samples, a single sub-sample is used for validation, and the remaining $\mathbb{K} - 1$ sub-samples are used for training. This process is then repeated for $\mathbb{K}$ folds, with each of the $\mathbb{K}$ sub-samples used exactly once. Finally, a single value results from averaging all. The aim of the use of such a series of rigorous tests is to avoid the biasing of the samples

Table 2: Error rates (in %) and running time (in sec. per character). The methods can be differentiated by the additional use of L_B Keogh tool Keogh (2002) and the evaluation protocol employed.

Method	# of Mis-recognition	# of Rejection	Avg. Error %	Time sec.
M1.	33	08	05.0	04
M2.	24	08	03.5	02

Index:
M1. Santosh, Nattee & Lamiroy (2012).
M2. Santosh, Nattee & Lamiroy (2012) + Keogh (2002) and 5-fold CV.

that can be possible in conventional dichotomous classification. In contrast to the previous studies Santosh, Nattee & Lamiroy (2012), this will be an interesting evaluation protocol.

3.6 Results and Discussions

Following evaluation protocols we have mentioned before, Table 2 provides average recognition error rates. In the tests, we have found that the recognition performance has been advanced by approximately more than 2%.

Based on results (*cf.* Table 2), we investigate the recognition performance based on the observed errors. We categorise the origin of the errors that are occurred in our experiments. As said in Section 1.3, these are mainly due to

1. structure similarity,
2. reduced and/or very long ascender and/or descender stroke, and
3. others such as re-writing strokes and mis-writing.

Compared to previous work Santosh, Nattee & Lamiroy (2012), number of rejection does not change while confusions due to structure similarity has been reduced. This is mainly because of the 5-fold CV evaluation protocol. Besides, running time has been reduced by more than a factor of two i.e., 2 seconds per character, thanks to LB_Keogh tool Keogh (2002).

4. Conclusions

In this chapter, an established as well as validated approach (based on previous studies Santosh & Nattee (2006a;b; 2007); Santosh et al. (2010); Santosh, Nattee & Lamiroy (2012)) has been presented for on-line natural handwritten Devanagari character recognition. It uses the number of strokes used to complete a symbol and their spatial relations[1]. Besides, we have provided the dataset publicly available for research purpose. Considering such a dataset, the success rate is approximately 97% in less than 2 seconds per character on average. In this chapter, note that the new evaluation protocol reduces the errors (mainly due to multi-class similarity) and the optimised DTW reduces the delay in processing – which has been new attestation in comparison to the previous studies.

[1] A comprehensive work based on relative positioning of the handwritten strokes, is presented in Santosh, Nattee & Lamiroy (2012). Once again, to avoid contradictions, this chapter aims to provide coherence as well as consistent studies on Devanagari character recognition.

The proposed approach is able to handle handwritten symbols of any stroke and order. Moreover, the stroke-matching technique is interesting and completely controllable. It is primarily due to our symbol categorisation and the use of stroke spatial information in template management. To handle spatial relation efficiently (rather than not just based on orthogonal projection i.e., MBR), more elaborative spatial relation model can be used Santosh, Lamiroy & Wendling (2012), for instance. In addition, use of machine learning techniques like inductive logic programming (ILP) Amin (2000); Santosh et al. (2009) to exploit the complete structural properties in terms of first order logic (FOL) description.

Acknowledgements

Since the chapter is based on the previous studies, thanks to researchers Cholwich Nattee, School of ICT, SIIT, Thammasat University, Thailand and Bart Lamiroy, Université de Lorraine – Loria Campus Scientifique, France for their efforts. Besides, the dataset is partially based on master thesis: TC-MS-2006-01, conducted in Knowledge Information & Data Management Laboratory, School of ICT, SIIT, Thammasat University under Asian Development Bank – Japan Scholarship Program (ADB-JSP).

5. References

Alginahi, Y. (2010). *Preprocessing Techniques in Character Recognition*, intech.

Amin, A. (2000). Prototyping structural description using an inductive learning program, *International Journal of Intelligent Systems* **15**(12): 1103–1123.

Arica, N. & Yarman-Vural, F. (2001). An overview of character recognition focused on off-line handwriting, *IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews* **31**(2): 216 –233.

Bahlmann, C. & Burkhardt, H. (2004). The writer independent online handwriting recognition system frog on hand and cluster generative statistical dynamic time warping, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(3): 299–310.

Bellman, R. & Kalaba, R. (1959). On adaptive control processes, *Automatic Control* **4**(2): 1–9.

Blumenstein, M., Verma, B. & Basli, H. (2003). A novel feature extraction technique for the recognition of segmented handwritten characters, *Proceedings of International Conference on Document Analysis and Recognition*, p. 137.

Boccignone, G., Chianese, A., Cordella, L. & Marcelli, A. (1993). Recovering dynamic information from static handwriting, *Pattern Recognition* **26**(3): 409 – 418.

Cha, S.-H., Shin, Y.-C. & Srihari, S. N. (1999). Approximate stroke sequence string matching algorithm for character recognition and analysis, *Proceedings of International Conference on Document Analysis and Recognition*, pp. 53–56.

Chiu, H.-P. & Tseng, D.-C. (1999). A novel stroke-based feature extraction for handwritten chinese character recognition, *Pattern Recognition* **32**(12): 1947–1959.

Chun, L. H., Zhang, P., Dong, X. J., Suen, C. Y. & Bui, T. D. (2005). The role of size normalization on the recognition rate of handwritten numerals, *IAPR TC3 Workshop of Neural Networks and Learning in Document Analysis and Recognition*, pp. 8–12.

Connell, S. D. & Jain, A. K. (1999). Template-based online character recognition, *Pattern Recognition* **34**: 1–14.

Dimond, T. (1957). Devices for reading handwritten characters, *Proceedings of the Eastern Joint Computer Conference*, pp. 232–237.

Doermann, D. S. & Rosenfeld, A. (1995). Recovery of temporal information from static images of handwriting, *International Journal of Computer Vision* **15**(1-2): 143–164.

Egenhofer, M. & Herring, J. R. (1991). Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases, *Univ. of Maine, Research Report*.

Foggia, P., Sansone, C., Tortorella, F. & Vento, M. (1999). Combining statistical and structural approaches for handwritten character description, *Image and Vision Computing* **17**(9): 701–711.

Groner, G. (1966). Real-time recognition of handprinted text, Memorandum RM-5016-ARPA, The Rand Corporation.

Guerfali, W. & Plamondon, R. (1993). Normalizing and restoring on-line handwriting, *Pattern Recognition* **26**(3): 419–431.

Heutte, L., Paquet, T., Moreau, J.-V., Lecourtier, Y. & Olivier, C. (1998). A structural/statistical feature based vector for handwritten character recognition, *Pattern Recognition Letters* **19**(7): 629–641.

Hu, J., Brown, M. K. & Turin, W. (1996). Hmm based on-line handwriting recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **18**: 1039–1045.

Jayadevan, R., Kolhe, S. R., Patil, P. M. & Pal, U. (2011). Offline recognition of devanagari script: A survey, *IEEE Transactions on Systems, Man, and Cybernetics, Part C* **41**(6): 782–796.

Keogh, E. J. (2002). Exact indexing of dynamic time warping, *Proceedings of 28th International Conference on Very Large Data Bases*, Morgan Kaufmann, pp. 406–417.

Keogh, E. J. & Pazzani, M. J. (1999). Scaling up dynamic time warping to massive dataset, *European PKDD*, pp. 1–11.

Kruskall, J. B. & Liberman, M. (1983). The symmetric time warping algorithm: From continuous to discrete, *Time Warps, String Edits and Macromolecules: The Theory and Practice of String Comparison*, Addison-Wesley, pp. 125–161.

Lippmann, R. P. (1989). Pattern classification using neural networks, *IEEE Comm. Magazine* **27**(11): 47–50.

Miyajima, K. & Ralescu, A. (1994). Spatial organization in 2D segmented images: representation and recognition of primitive spatial relations, *Fuzzy Sets Systems* **65**(2-3): 225–236.

Myers, C. S. & Rabiner., L. R. (1981). A comparative study of several dynamic time-warping algorithms for connected word recognition, *The Bell System Technical Journal* **60**(7): 1389–1409.

Namboodiri, A. M. & Jain, A. K. (2004). Online handwritten script recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **26**(1): 124–130.

Okumura, D., Uchida, S. & Sakoe, H. (2005). An hmm implementation for on-line handwriting recognition - based on pen-coordinate feature and pen-direction feature, *Proceedings of International Conference on Document Analysis and Recognition*, pp. 26–30.

Pal, U. & Chaudhuri, B. B. (2004). Indian script character recognition: a survey, *Pattern Recognition* **37**(9): 1887–1899.

Papadias, D. & Sellis, T. (1994). *Relation Based Representations for Spatial Knowledge*, PhD Thesis, National Technical Univ. of Athens.

Plamondon, R. & Srihari, S. (2000). Online and off-line handwriting recognition: a comprehensive survey, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **22**(1): 63 –84.

Qiao, Y., Nishiara, M. & Yasuhara, M. (2006). A framework toward restoration of writing order from single-stroked handwriting image, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **28**(11): 1724–1737.

Sakoe, H. (1978). Dynamic programming algorithm optimization for spoken word recognition, *IEEE Transactions on Acoustics, Speech, and Signal Processing* **26**: 43–49.

Santosh, K. C., Lamiroy, B. & Ropers, J.-P. (2009). Inductive logic programming for symbol recognition, *Proceedings of International Conference on Document Analysis and Recognition*, pp. 1330–1334.

Santosh, K. C., Lamiroy, B. & Wendling, L. (2012). Symbol recognition using spatial relations, *Pattern Recognition Letters* **33**(3): 331–341.

Santosh, K. C. & Nattee, C. (2006a). Stroke number and order free handwriting recognition for nepali, *in* Q. Yang & G. I. Webb (eds), *Proceedings of the Pacific Rim International Conferences on Artificial Intelligence*, Vol. 4099 of *Lecture Notes in Computer Science*, Springer-Verlag, pp. 990–994.

Santosh, K. C. & Nattee, C. (2006b). Structural approach on writer independent nepalese natural handwriting recognition, *Proceedings of the International Conference on Cybernetics and Intelligent Systems*, pp. 1–6.

Santosh, K. C. & Nattee, C. (2007). Template-based nepali natural handwritten alphanumeric character recognition, *Thammasat International Journal of Science and Technology* **12**(1): 20–30.

Santosh, K. C., Nattee, C. & Lamiroy, B. (2010). Spatial similarity based stroke number and order free clustering, *Proceedings of IEEE International Conference on Frontiers in Handwriting Recognition*, pp. 652–657.

Santosh, K. C., Nattee, C. & Lamiroy, B. (2012). Relative positioning of stroke based clustering: A new approach to on-line handwritten devanagari character recognition, *International Journal of Image and Graphics* **12**(2): 25 pages.

Schenkel, M., Guyon, I. & Henderson, D. (1995). On-line cursive script recognition using time delay neural networks and hidden markov models, *Machine Vision and Applications* **8**(4): 215–223.

Tappert, C. C., Suen, C. Y. & Wakahara, T. (1990). The state of the art in online handwriting recognition, *IEEE Transactions on Pattern Analysis and Machine Intelligence* **12**(8): 787–808.

∅ivind Due Trier, Jain, A. K. & Taxt, T. (1996). Feature extraction methods for character recognition – a survey, *Pattern Recognition* **29**(4): 641 – 662.

Verma, B., Lu, J., Ghosh, M. & R., G. (2004). A feature extraction technique for on-line handwriting recognition, *Proceedings of IEEE International Joint Conference on Neural Networks*, pp. 1337–1341.

Viard-Gaudin, C., Lallican, P. M. & Knerr, S. (2005). Recognition-directed recovering of temporal information from handwriting images, *Pattern Recognition Letters* **26**(16): 2537–2548.

Wang, X. & Keller, J. M. (1999). Human-based spatial relationship generalization through neural/fuzzy approaches, *Fuzzy Sets Systems* **101**(1): 5–20.

Zhou, X.-D., Liu, C.-L., Quiniou, S. & Anquetil, E. (2007). Text/non-text ink stroke classification in japanese handwriting based on markov random fields, *Proceedings of International Conference on Document Analysis and Recognition*, pp. 377–381.