

Accepted in International Journal of Image and Graphics
© World Scientific Publishing Company

RELATIVE POSITIONING OF STROKE BASED CLUSTERING: A NEW APPROACH TO ON-LINE HANDWRITTEN DEVANAGARI CHARACTER RECOGNITION

K.C. SANTOSH*

*INRIA Nancy – Grand Est
615 rue du Jardin Botanique, 54600 Villers-lès-Nancy, France
Santosh.KC@inria.fr*

CHOLWICH NATTEE

*School of ICT, SIIT, Thammasat University,
Bangkadi Campus, Pathumthani 12000, Thailand
cholwich@siit.tu.ac.th*

BART LAMIROY

*Université de Lorraine, LORIA - Campus Scientifique,
BP 239 - 54506 Vandoeuvre-lès-Nancy Cedex, France
Bart.Lamiroy@loria.fr*

In this paper, we propose a new scheme for Devanagari natural handwritten character recognition. It is primarily based on spatial similarity based stroke clustering. A feature of a stroke consists of a string of pen-tip positions and directions at every pen-tip position along the trajectory. It uses the dynamic time warping (DTW) algorithm to align handwritten strokes with stored stroke templates and determine their similarity. Experiments are carried out with the help of 25 native writers and a recognition rate of approximately 95% is achieved.

Our recogniser is robust to a large range of writing style and handles variation in the number of strokes, their order, shapes and sizes and similarities among classes.

Keywords: Stroke Number and Order Free; Spatial Relations; DTW; Clustering; On-line Devanagari Character Recognition.

1. Introduction

1.1. Motivation

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.

*Corresponding author

Devanagari is derived from two roots: ‘*deva*’ and ‘*nagari*’. ‘*deva*’ means ‘deity’ and ‘*nagari*’ means ‘city’. Together, it implies both religion and urbane. Devanagari is a script used to write several Indian languages, including Nepali, Sanskrit, Hindi, Marathi, Pali, Kashmiri, Sindhi, and sometimes Punjabi. Devanagari characters follow a complex structure and may count up to more than 500 symbols. In our experiments, we take the most common type of discrete characters that are frequently used. Our dataset is composed of 36 classes of consonants and vowels. Fig. 1 shows a few examples of it where a few samples are provided from each class of a few Devanagari handwritten characters.

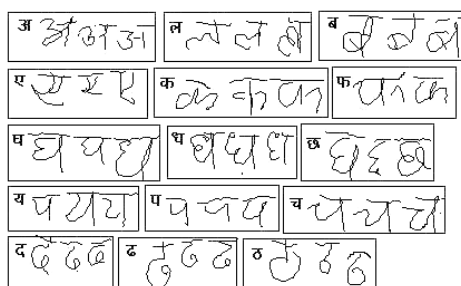


Fig. 1. A few sample images from each class of a few Devanagari natural handwritten characters.

In this paper, we develop a technique to employ strokes spatial relations to build a writer independent on-line Devanagari handwritten character recognition system. The recogniser yields satisfactory results.

1.2. Vocabulary Issue

In order to clearly establish the semantics of our reasoning, we first introduce the following vocabulary distinction: the term “*character*” refers to a token having a lexicographic meaning, independently of its graphic representation. We shall represent it by its printed form, as in the upper-left corners in each box of Fig. 1. “*Symbols*” are graphical representations of “*characters*”, most often handwritten. A *symbol* is an visual instance of a specific *character*, and visually quite different *symbols* might represent the same *character*, as shown in Fig. 1. In this illustration, each box contains three symbols representing the same character. When addressing recognition in Section 3, each character will represent a “*class*”.

1.3. Structure of Devanagari Characters

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.

- 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. or Fig. 1)
- 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.

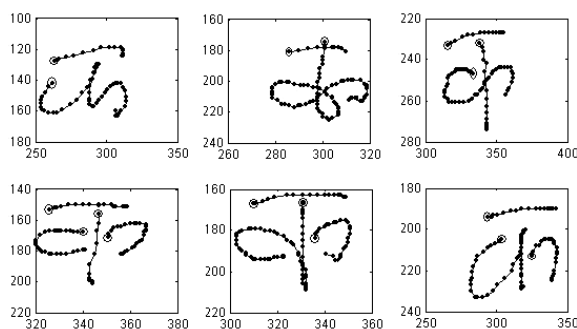


Fig. 2. A few possible variations in writing units for a consonant क.

As a consequence, significant shape variations may occur between symbols representing the same character. Fig. 2 illustrates this. Since we are operating in an on-line framework, pen strokes are the sequences of sampled points between ‘pen-down’ to ‘pen-up’ events. In this illustration, every initial pen-tip position in each stroke is encircled to show the number of strokes used to complete a symbol. Stroke extraction and the complete handwritten character representation will be explained in Section 2.1.

1.4. Related Work

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^{23,31}. 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^{3,8,24,33} 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. More specifically, template based approaches have a long standing record^{1,6,12,26,28}. Our approach is inspired by the cluster generative statistical dynamic time warping (CS-DTW) technique¹ but varies distinctly where the clustering technique itself is concerned. As mentioned in Section 1.3, Connell *et al.*, 2000⁷ also highlighted the difficulties associated with the cursive nature of Devanagari writing. Due to its structural complexity, Niranjana *et al.*, 2005¹³ focussed on structural properties and separated *shirorekha* based on directional code such as *east* or *west*. Their work is writer dependent and is limited to natural handwriting where curve sequences can be *shirorekha*. In the literature, there has been an extensive use of specific unique features but varying according to the nature of the script. Two concrete examples are the use of *shirorekha* in²⁰ to make Devanagari difference with other scripts, and *hat* feature¹⁷ in Urdu handwriting for recognition confidence. Our claim is that integrating relative positioning of strokes by referencing such a unique feature leads to better recognition.

Handling spatial relations between strokes, in on-line handwriting is not a completely new approach^{4,16,18,29}. But, these existing approaches are not very generic. Swethalakshmi *et al.*, 2007³⁰ use spatio-structural feature based on zoning information (as explained in⁹) to recognise Devanagari and Tamil scripts, and provides 95% and 92% recognition rates respectively. The approach however, is very sensitive to handwriting strokes with asymmetric structure and symbols having very long ascender and/or descender, for instance. Our approach is different and uses pairwise spatial relations between the strokes by fixing *shirorekha* as a reference within a symbol as in²⁰.

1.5. Outline of the Proposed Method

This paper is an extension of previous works^{26,27}. 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 strokes²⁶. Therefore, we develop a method for analysing handwritten characters based on both the number of strokes and the their spatial information²⁷. This work is essentially identical to previous work²⁷ but extends it by providing an in-depth description of the method and a more developed experimental validation framework. It consists in four main phases.

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

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. Stroke representation within the

complete symbol will be explained in Section 2.1.

step 2. Find the spatial relation between strokes.

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 Section 2.3.2.

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

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. In Section 2.4, we explain in detail about stroke clustering and management of the learnt strokes.

step 4. Stroke-wise matching for recognition.

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. We derive the complete recognition process in Section 3.

1.6. Structure of the Paper

Until now, we have mentioned our motivations in Section 1.1, defined vocabulary in Section 1.2, explained Devanagari graphical structure in Section 1.3, reviewed literature in Section 1.4 and briefly outlined the proposed method in Section 1.5. The remaining of the paper is organised as follows. Based on the outline presented in Section 1.5, our learning module is developed in Section 2: it includes handwritten character representation, feature selection and dissimilarity measure definition as well as template management. Section 3 covers our recognition module while Section 4 gives an overview of the obtained results and provides an error analysis. The paper is concluded with a brief generalisation of the method in Section 5 along with a few steps to go further.

2. Learning

2.1. Handwritten Character Representation

The digitiser captures a series of strokes during pen movement as soon as it starts to move over the tablet. A string of coordinates (pen-tip positions) from pen-down to pen-up events represents a stroke. Along the pen trajectory, the initial pen-tip

position represents the first coordinate of the particular stroke. Similarly, the last point is taken as soon as pen-up event takes place. Depending on the user and/or writing style, a same symbol can be written with a varying number of strokes.

Since the digitiser capture a string of coordinates on a temporal basis, there is no segmentation problem. A symbol \mathbf{S} is composed of a set of m strokes, in which each stroke \mathbf{s}^j consists of a string of coordinates \mathbf{p}_k^j ,

$$\mathbf{S} = [\mathbf{s}^1, \mathbf{s}^2, \dots, \mathbf{s}^m] \quad (1)$$

$$\mathbf{s}^j = [\mathbf{p}_1^j, \mathbf{p}_2^j, \dots, \mathbf{p}_l^j] \quad (2)$$

where $\mathbf{p}_k^j = (x_k^j, y_k^j)$ for $j \in [1 : m]$ and $k \in [1 : l]$.

Strokes directly collected from users are often incomplete and noisy. Different systems use a variety of different pre-processing techniques before feature extraction^{2,32}. The techniques used in one system may not exactly fit into the other because of different writing styles and nature of the scripts. We utilise repeated coordinates deletion, noise elimination and normalisation.

2.1.1. Co-occurrence Coordinates Deletion

As in¹, we delete co-occurrence of coordinates $\mathbf{p}_k = \mathbf{p}_{k+1}$ when they occur for consecutive coordinates of a particular stroke, along the pen trajectory.

2.1.2. Noise Elimination

Sequences with no information about the symbol need to be eliminated in order to enhance recognition accuracy and speed. Elimination of noisy sequences in cursive writing is a difficult task and there exist no robust noise elimination algorithms that can be applied in all situations. Besides, they vary from one script to another. In our case, the following filters have been applied.

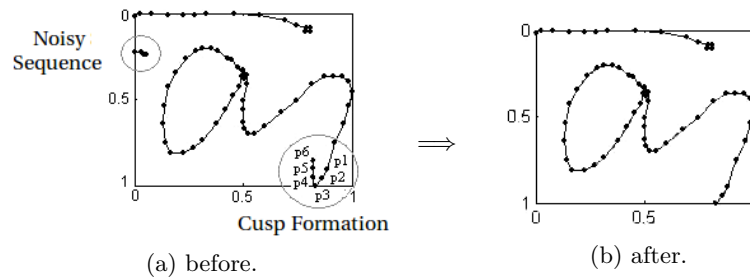


Fig. 3. Stroke pre-processing: noise and cusp elimination, and normalisation.

Deletion of short sequences. We identify unnecessarily written sequences for deletion. We delete short sequences (≤ 5 coordinates) as they do not give any

information about the character. It is based on empirical knowledge. If we change it to 10 coordinates, for instance, there may be a chance to delete stroke with important information. Considering the employed dataset, such sequences are often written at the end as re-writing strokes. An example is shown in Fig. 3.

Chopping of undesirable hook or cusp. We chop undesirable hooks or cusps at both the ascender and descender of sequences. This is done by the use of tangent angles along the trajectory path of 5 – 10 2D coordinates related to the ascender and descender. The sequence is chopped from the point where the trajectory path turns sharply (angle changes drastically i.e., $80^\circ - 100^\circ$) as shown in Fig. 3.

2.1.3. Size Normalisation

Many authors demonstrate the use and importance of normalisation^{5,11}. Because of variable size of writing strokes, it is necessary to transform a complete symbol into a standard window. In our case, we map the symbol S into a $[0, 1] \times [0, 1]$ unit square as shown in Fig. 3 before computing the strokes' spatial relations.

2.2. Feature Selection and Dissimilarity Measure

2.2.1. Feature Selection

Appropriate feature selection can greatly decrease the workload and simplify the subsequent design process of the classifier. 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 required¹⁵. However, they vary from one script to another^{2,20,21,32}.

In this work, a feature vector sequence of every j^{th} stroke is expressed as in^{21,26}:

$$\mathbf{F}^j = \left[\left(\mathbf{P}_1^j, \alpha_{\mathbf{P}_2^j, \mathbf{P}_3^j} \right), \left(\mathbf{P}_2^j, \alpha_{\mathbf{P}_2^j, \mathbf{P}_3^j} \right), \dots, \left(\mathbf{P}_l^j, \alpha_{\mathbf{P}_{l-1}^j, \mathbf{P}_l^j} \right) \right] \quad (3)$$

where, $\alpha_{\mathbf{P}_{l-1}^j, \mathbf{P}_l^j} = \arctan \left(\frac{y_l^j - y_{l-1}^j}{x_l^j - x_{l-1}^j} \right)$. 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.

2.2.2. Dissimilarity Measure – Matching Score using DTW

Distance computation between the symbols expresses how similar or dissimilar they are. Given representation explained in Section 2.2.1, how can distance between

two feature sequences be measured? Since our features are non-linear sequences of potentially different lengths, the use of a simple distance metric is not appropriate. Following the idea presented in¹⁴, we employ DTW.

DTW Algorithm. Let us consider two feature sequences $\mathbf{F}_A = \{\mathbf{f}_a\}_{a=1,\dots,\mathcal{A}}$ and $\mathbf{F}_B = \{\mathbf{f}_b\}_{b=1,\dots,\mathcal{B}}$ of size \mathcal{A} and \mathcal{B} respectively. The aim of the algorithm is to provide the optimal alignment between both sequences. At first, a matrix \mathcal{M} of size $\mathcal{A} \times \mathcal{B}$ is constructed. Then for each element in matrix \mathcal{M} , local distance metric $\delta(a, b)$ between the events e_a and e_b is computed. $\delta(a, b)$ can be expressed as,

$$\delta(a, b) = \sqrt{(e_a - e_b)^2}, \quad (4)$$

where $e_a = \mathbf{f}_a$ and $e_b = \mathbf{f}_b$ for $a \in [1 : \mathcal{A}]$ and $b \in [1 : \mathcal{B}]$. Let $D(a, b)$ be the global distance up to (a, b) ,

$$D(a, b) = \min [D(a-1, b-1), D(a-1, b), D(a, b-1)] + \delta(a, b) \quad (5)$$

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 $(\mathcal{A}, \mathcal{B})$. 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,

$$\mathcal{W} = \{w_q\}_{q=1\dots Q}$$

where $\max(a, b) \leq Q < a+b-1$ and q^{th} element of \mathcal{W} is $w(a, b)_q \in [1 : \mathcal{A}] \times [1 : \mathcal{B}]$ for $q \in [1 : Q]$. The optimised warping path \mathcal{W} satisfies the following three conditions:

- c1.** boundary condition: $w_1 = (1, 1)$ and $w_Q = (\mathcal{A}, \mathcal{B})$,
- c2.** monotonicity condition: $a_1 \leq a_2 \leq \dots \leq a_{\mathcal{A}}$ and $b_1 \leq b_2 \leq \dots \leq b_{\mathcal{B}}$ and
- c3.** continuity condition: $w_{q+1} - w_q \in \{(1, 1)(0, 1), (1, 0)\}$ for $q \in [1 : Q-1]$.

c1 conveys that the path starts from $(1, 1)$ to $(\mathcal{A}, \mathcal{B})$, 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**.

We then define the global distance between \mathbf{F}_A and \mathbf{F}_B as

$$\Delta(\mathbf{F}_A, \mathbf{F}_B) = \frac{D(\mathcal{A}, \mathcal{B})}{Q}. \quad (6)$$

This means that the last element of the $\mathcal{A} \times \mathcal{B}$ matrix gives the DTW-distance between two vector sequences. It can be normalised by the Q , the number of discrete warping steps along the diagonal DTW-matrix. This is called the DTW-matching score in our recognition processes, and measures similarity between shapes. It also allows ranking between stroke sequences with respect to a reference sequence.

The warping path can be computed with the help of back-tracking along the minimum cost index pairs (a, b) starting from $(\mathcal{A}, \mathcal{B})$. As shown in Fig. 4, the back-tracking procedure following the optimal warping path is handled with the help of

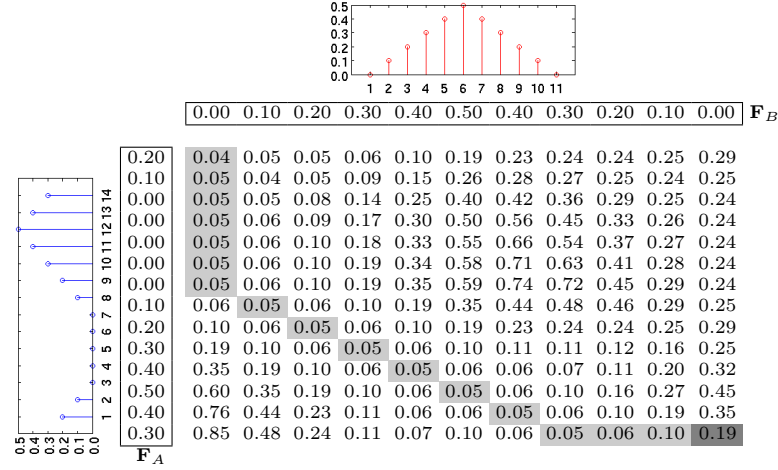


Fig. 4. DTW distance between two non-linear sequences – an example.

Dynamic Programming (DP).

$$w_{q-1} = \begin{cases} (1, b-1) & \text{if } a = 1 \\ (a-1, 1) & \text{if } b = 1 \\ \operatorname{argmin}\{D(a-1, b-1), D(a-1, b), D(a, b-1)\} & \text{otherwise,} \end{cases} \quad (7)$$

where we take the lexicographically smallest pair in case ‘argmin’ is not unique.

Fig. 4 shows a DTW-matrix between two continuous time series signatures having different lengths. In this illustration, note that the warping path makes easier to take the average similarity of two non-linear feature sequences. The process of averaging is explained in Section 2.4.1.

2.3. Recognition

From a purely combinatorial point of view, measuring the similarity or dissimilarity between two symbols $\mathbf{S}_1 = \{\mathbf{s}_1^i\}_{i=1\dots n}$ and $\mathbf{S}_2 = \{\mathbf{s}_2^j\}_{j=1\dots m}$ 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 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 order to reduce both combinatorial complexity and enhance global coherence, we introduce a stroke representation, based on the *shirorekha* and relative positioning of the strokes with respect to the former. To handle this however, we need to identify the *shirorekha*.

2.3.1. *Shirorekha and its identification*

As said in Section 1.3, 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. Sometimes there may be two (Fig. 5 – ध) but this usually depends on users' writing style. In natural handwriting, the *shirorekha* may not always be strictly horizontal and exactly on the *top*. It may be a small curve and may intersect *text*. Since locations of both *shirorekha* and *text* vary, it is difficult to build a general rule to compute spatial relations.



Fig. 5. Straight sequence(s) plus curve sequence(s) in a symbol.

In general, the location of the *shirorekha* is assumed to be on the *top* and the *text(s)* is(are) determined with respect to the *shirorekha*. In Fig. 5, it is clearly seen that every symbol has at least two kinds of strokes: straight and curve. Naturally, straight strokes are more likely to represent the *shirorekha*. To identify whether a stroke is a straight one, we check the straightness property S_t for every j^{th} stroke,

$$S_t = \frac{\delta(\mathbf{p}_1^j, \mathbf{p}_l^j)}{\sum_1^{l-1} \delta(\mathbf{p}_k^j, \mathbf{p}_{k+1}^j)} = \begin{cases} 0.8 \leq S_t \leq 1 & \text{:straight} \\ \text{otherwise} & \text{:curve} \end{cases} \quad (8)$$

where $\delta(\mathbf{p}_k^j, \mathbf{p}_{k+1}^j) = \sqrt{(x_k^j - x_{k+1}^j)^2 + (y_k^j - y_{k+1}^j)^2}$. It is important to notice that $S_t = 1$ for completely straight sequences. This is however, not possible in case of natural handwriting. Now, let us check whether the provided S_t range in equation (8) can classify straight sequences by taking real-world samples. To handle this, we have measured S_t over handwritten *shirorekha*. We found that 98% of *shirorekha* from all training symbols are lying within the range. While, remaining 2% of the *shirorekha* are still curved sequences. The distribution of S_t values is shown in Fig. 6. If we increase its range (i.e., reduces the lower limit), remaining 2% can be included with an increased risk of detecting real curve sequences. But, note that these remaining strokes are still located on the *top* with respect to other strokes. Thus, there is still a chance to correctly identify them as the *shirorekhas* with the help of their location.

Finding straight strokes does not necessarily mean that *shirorekhas* are detected. For the two-stroke symbols, it is easy to separate the straight and curve sequence. But, ambiguity occurs when the symbol is composed of more than two strokes, since there may exist many straight sequences. In such a case, we need to determine the *shirorekha* from these straight stroke candidates. Therefore, we check the following conditions one after another:

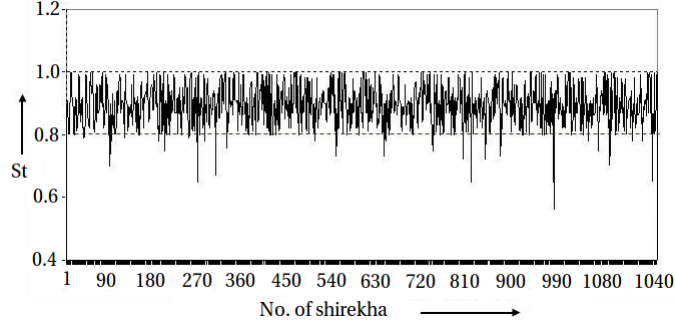


Fig. 6. Straightness calculation for all *shirorekhas* in training dataset.

- spatial relation between the straight sequences,
- width of the sequence i.e., $w = x_{max} - x_{min}$, and
- global stroke direction from initial to the end coordinate i.e., $\theta_{\mathbf{p}_1, \mathbf{p}_l} = \arctan\left(\frac{y_l - y_1}{x_l - x_1}\right) \times \left(\frac{180^\circ}{\pi}\right)$.

We select the stroke as the *shirorekha* which is located on the *top* with respect to others. We may find a few cases having two strokes on the *top* due to re-writing *shirorekha*, for instance. In the latter case, we take the one which has largest width. It is quite rare to get first two conditions identical. If it happens, then we check minimum global stroke direction.

Once the *shirorekha* is identified, we use it as reference stroke to compute spatial relations between remaining strokes within the symbol. This is explained in the following section.

2.3.2. Relative Positioning of Strokes

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

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

To confirm the location of the stroke, we use the projection theory: minimum boundary rectangle (MBR)²² model combined with the stroke's centroid.

Based on¹⁰, we start with checking fundamental topological relations such as *disconnected* (DC), *externally connected* (EC) and *overlap/intersect* (O/I). Considering two strokes $\mathbf{s}^j = \{\mathbf{p}_k^j\}_{k=1\dots l}$ and $\mathbf{s}^{j'} = \{\mathbf{p}_{k'}^{j'}\}_{k'=1\dots l'}$ as follows,

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

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-centre¹⁹ and angle histogram³⁴ are not the appropriate choice due to the cursive nature of writing.

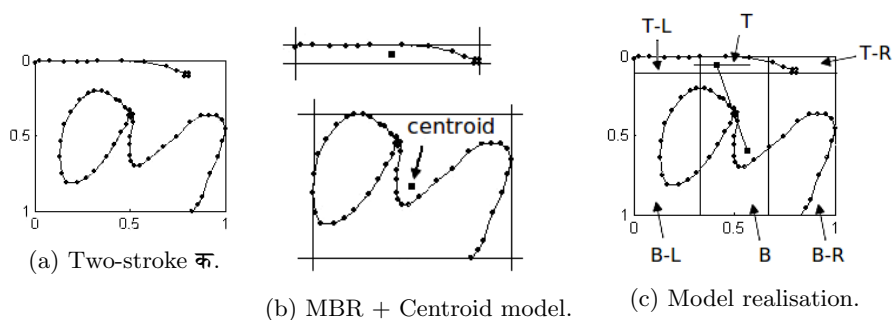


Fig. 7. Pairwise spatial relation for a two-stroke क taken from Fig. 3.

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 *shirorekhas* are also possible to treat. In such a situation, the first *shirorekha* according to the order of strokes is taken as reference.

2.4. Templates

In this section, we explain how we represent handwritten characters using stroke templates. These known representative templates are used for unknown stroke alignment to do recognition. To obtain them, stroke clustering is used.

2.4.1. Stroke 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.

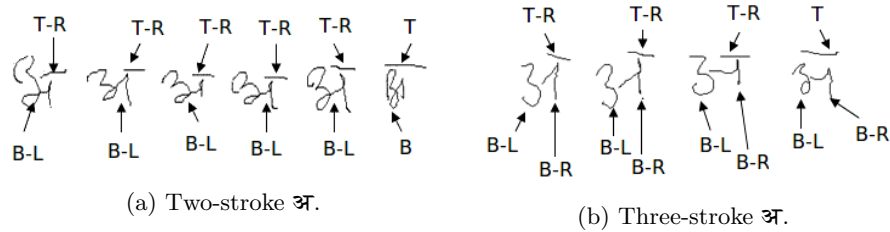


Fig. 8. Relative positions of strokes for a class अ in two different groups i.e., two-stroke and three-stroke symbols.

As mentioned in Section 1.5, our 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. This means that only strokes which are at a specific location are taken for clustering. This applies to all groups within a class. Fig. 9 illustrates a simple example of stroke clustering.

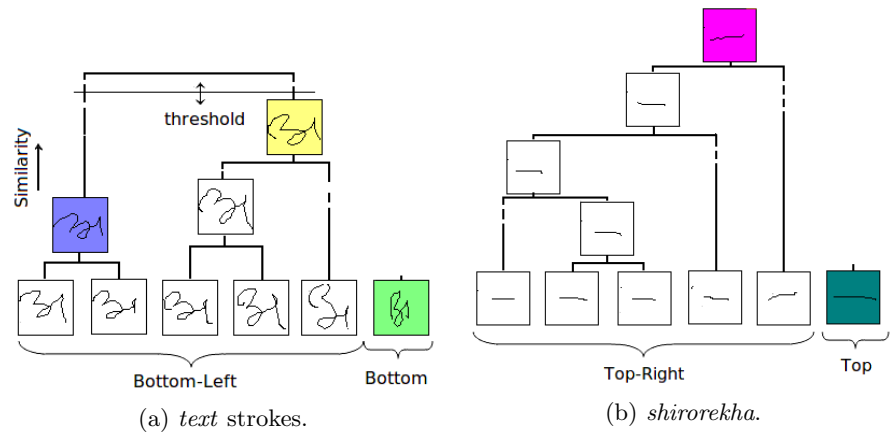


Fig. 9. Stroke clustering for two-strokes vowel अ based on relative positions. These strokes are taken from Fig. 8 (a) where six users are employed.

In agglomerative hierarchical clustering, we merge two similar strokes and find a new cluster. The distance computation between two strokes follows Section 2.2. 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. Consider $\mathcal{T}_{\mathcal{G}}$ is the total number of number of templates (learnt strokes) in one group from a class of character. It consists of a templates in all relative positions described in Section 2.3.2 i.e.,

$$\mathcal{T}_{\mathcal{G}} = \underbrace{\mathcal{T}_{top-left} + \mathcal{T}_{top} + \mathcal{T}_{top-right}}_{\mathcal{T}_{shirorekha}} + \underbrace{\mathcal{T}_{bottom-left} + \mathcal{T}_{bottom} + \mathcal{T}_{bottom-right}}_{\mathcal{T}_{text}}.$$

Our clustering threshold is based on the number of strokes in the particular relative positions. For instance, if there exists ‘shirorekhas’ in three different positions such as *top-left*, *top* and *top-right*, clustering happens individually. For ‘shirorekha’, we develop a single template for each location. In case of *text* strokes, we still keep importance of relative position while threshold is determined based on the number of strokes. Now, we have

$$\begin{aligned} \mathcal{T}_{bottom-left} &= (\mathcal{T}_{\mathcal{G}} - \mathcal{T}_{shirorekha}) \times \left(\frac{\mathcal{N}_{bottom-left}}{\mathcal{N}_{total}} \right), \\ \mathcal{T}_{bottom} &= (\mathcal{T}_{\mathcal{G}} - \mathcal{T}_{shirorekha}) \times \left(\frac{\mathcal{N}_{bottom}}{\mathcal{N}_{total}} \right) \text{ and} \\ \mathcal{T}_{bottom-right} &= (\mathcal{T}_{\mathcal{G}} - \mathcal{T}_{shirorekha}) \times \left(\frac{\mathcal{N}_{bottom-right}}{\mathcal{N}_{total}} \right), \end{aligned}$$

where $\mathcal{N}_{\mathcal{R}}$ represents the number of strokes in the particular relative position and $\mathcal{N}_{total} = \mathcal{N}_{bottom-left} + \mathcal{N}_{bottom} + \mathcal{N}_{bottom-right}$. Since number of templates are always in fixed number, we round-up (and round-down) by giving importance to stroke relative positioning. Following Fig. 9, overall concept can be provided when $\mathcal{T}_{\mathcal{G}} = 5$.

Overall, let us emphasise the spatial similarity in stroke clustering. Since clustering is applied for those strokes which have identical relative positions, a single *shirorekha* which is on the *top* never merges with others. This brings an impression that the location of the *shirorekha* is important otherwise ambiguities may be introduced for very similar pairs of characters such as (ध, घ), (भ, म), (थ, य) etc. as mentioned in Section 1.5.

2.4.2. Template Management

Following Section 2.4.1, stroke templates are taken by averaging similar strokes. In every class of character, there are variable groups of clusters according to the number of strokes used to complete symbol. Each of them is representative of other strokes in every particular location. For instance, the stroke representatives from two-stroke symbols, three-stroke symbols and four-stroke symbols are respectively labelled as first, second, and third group. Fig. 10 shows a sample of our *template management* technique by taking a few groups for a vowel अ. This procedure is applied for all classes.

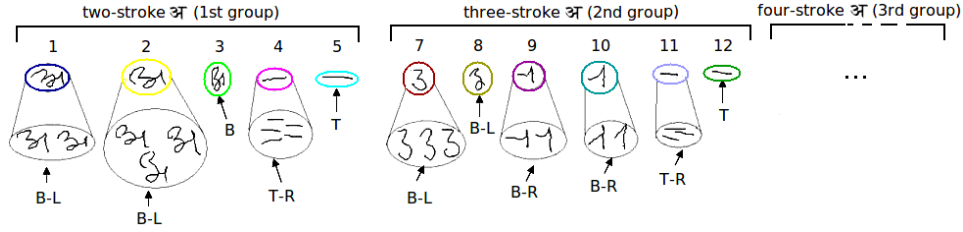


Fig. 10. Cluster management for a class अ, following Fig. 8 and 9.

2.4.3. Weight Determination

Our approach allows for the determination of the probability that strokes are at specific location in a specific group. The probability of occurrence of a template (stroke representative) in every specific location can be obtained by using the proportion of strokes in the training set. Consider the first group having two-strokes अ only in Fig. 10 to realise the weights for all possible locations^a.

- for *shirorekha*:
 - $\mapsto \text{top-left} = \frac{0}{6} = 0\%$
 - $\mapsto \text{top} = \frac{1}{6} = 16.7\%$ and
 - $\mapsto \text{top-right} = \frac{5}{6} = 83.4\%$.
- for *text*:
 - $\mapsto \text{bottom-left} = \frac{5}{6} = 83.4\%$
 - $\mapsto \text{bottom} = \frac{1}{6} = 16.7\%$ and
 - $\mapsto \text{bottom-right} = \frac{0}{6} = 0\%$.

We shall use this probability information as a weight in Section 3 for template matching in order to find the test strokes for recognition. It gives a trend of the global writing style in the training set by showing, for example, how often the *shirorekha* is exactly on the top position.

3. Classification

In the previous sections we have described how to represent handwritten characters based on templates, and how to compare individual strokes. Combining those together, we can now proceed to developing our recognition method. The aim is to verify a given unknown handwritten test symbol with respect to the corresponding character class label.

Let \mathbf{S} be an unknown test symbol composed of n strokes; $\mathbf{S} = \{\mathbf{s}^i\}_{i=1\dots n}$. Let \mathcal{T} be the set of template strokes coming from the cluster representing the symbols

^aFor clarity, the presented weights correspond to the stroke example shown in Fig. 10, is a representative but small illustration. In our real dataset, the sample set is large.

with same number of strokes as \mathbf{S} . We use simple template matching on a stroke-by-stroke basis via DTW. Each stroke \mathbf{s}^i from symbol \mathbf{S} is matched with the templates independently. The stroke is said to be similar with the template for which the lowest distance is produced. The following algorithm gives a complete idea of how test strokes are matched with the templates.

- a. Find the number of strokes n in the test symbol \mathbf{S} . Separate the *shirorekha* from the text parts and determine each stroke's spatial relation with respect to it.
- b. For each known character c , the strokes $\{\mathbf{s}^i\}$ are matched with the corresponding ones (i.e., those positioned similarly with respect to their *shirorekha*) from the templates in \mathcal{T}_c . Thus the set of corresponding templates is $\mathcal{T}_c|_{\mathbf{s}^i} = \{\tau_{c,i}^t\}$.

Every matching produces a matching score that is represented as,

$$\mathbf{M} = \begin{pmatrix} \mathcal{D}_1 \\ \mathcal{D}_2 \\ \vdots \\ \mathcal{D}_n \end{pmatrix}$$

where $\mathcal{D}_i = \{\Delta_i^t\}$, Δ_i^t is the matching score between \mathbf{s}^i and the appropriate templates $\tau_{c,i}^t$ of $\mathcal{T}_c|_{\mathbf{s}^i}$. These matching scores Δ_i^t are obtained via DTW as explained in equation (6).

- c. Each matching score is divided by the weight of the corresponding template $\tau_{c,i}^t$ as described in Section 2.4.3. Finally, the weighted matching score is,

$$\overline{\Delta_{c,i}^t} = \Delta_{c,i}^t / \text{weight}(c, i). \quad (9)$$

Once we have obtained these weighted matching scores, we can compute an overall matching score for each character c by computing $\sum_{i=1}^n \min_t (\overline{\Delta_{c,i}^t})$. The classification candidate is the character having obtained the lowest value. The template matching is straightforward when spatial information is not considered.

The stability of the overall matching score can be significantly improved by taking into account the potential risk of having mismatched individual strokes. Indeed, the value of $\min_t (\overline{\Delta_{c,i}^t})$ can be weighted by the number of other strokes having a very similar matching score. In that case,

- e. Count the number of matching scores close to the minimum, upto a threshold $\epsilon_{c,i} = \min_t (\overline{\Delta_{c,i}^t}) + \epsilon_i$, for every stroke s_i and every character c . This number determines how many similar strokes are available as templates:

$$W_{c,i} = \sum_{l=1}^t \mathcal{C}(\overline{\Delta_{c,i}^l}, \epsilon_{c,i}), \quad (10)$$

$$\text{where } \mathcal{C}(z_1, z_2) = \begin{cases} 1 & \text{if } z_1 < z_2 \\ 0 & \text{otherwise.} \end{cases}$$

f. Finally, the test symbol is classified by the character's label,

$$L = \underset{c}{\operatorname{argmin}} \sum_{i=1}^n \frac{\min_t (\overline{\Delta}_{c,i}^t)}{W_{c,i}} \text{ for } c = 1, \dots, \kappa. \quad (11)$$

4. Experiments

4.1. Dataset

In this work, we have used a Graphite tablet (WCACOM Co. Ltd.), model ET0405A-U, which captures 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 was given the opportunity to write each character twice. No other directions, constraints, or instructions were given to the users. Our dataset can be downloaded from the IAPR tc-11 website <http://www.iapr-tc11.org>.

In our test, 15 writers are used for training and the remaining 10 are for testing. For our experiment, we have used MATLAB 7.0.4 on a 1.81 GHz, 1.00 GB RAM, PC running Microsoft Windows XP professional.

4.2. Experimental Results

We compare our approach to two different methods: one does not use spatial information of the strokes¹ and another uses a spatio-structural feature based on zoning information³⁰. It is important to notice that the method mentioned in²⁶ is an extension of the method presented in¹.

Table 1. Error rates for both training and test data and running time (per character) from different methods.

Method	Dataset	# of Misrecog.	# of Reject	Avg. Error %	Time sec.
M1. Swethalakshmi <i>et al.</i> , 2007 ³⁰	Training	16	13	03.0	16
	Test	91	26	16.0	
M2. CSDTW ¹	Training	19	07	02.0	32
	Test	71	19	12.5	
M3. Our Method	Training	10	03	01.0	04
	Test	33	08	05.0	

Table 1 shows the experimental results for both training and test datasets and recognition speed. In order to check the training quality, it makes sense to confront training symbols to the recognition system. In case of the training set, recognition rates for all methods do not provide notable difference. For the test set, our method provides error rate of approximately 5% which is less than by more than 7% from

the method where spatial information is not used¹ and by 11% from the method that uses spatio-structural information of the strokes³⁰.

Besides the recognition rate, performance also depends on recognition speed. In average, recognition speed of our method is less than 4 seconds per character, compared to approximately 32 seconds from¹ and 17 seconds from³⁰. The huge difference between our method and¹ is due to the fact that the latter does not include spatial information and the categorisation of the symbols while learning. Besides, we have controllable stroke matching, thanks to our template management, due to which, stroke matching is made only with those which have identical relative position from the same group (*cf.* Section 3).

Overall, our method provides significant improvement in recognition performance (both recognition rate and speed) over provided benchmarking methods.

4.3. *Experimental Error Analysis*

This section investigates the recognition performance based on the observed errors. We first study the origin of errors and then analyse them one by one. Table 2 shows 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.

In our experiment (see Table 1 and 2), we observe that spatio-structural feature based on zoning information³⁰ simply does not provide complete shape information of the strokes when their sizes vary widely – which is obvious in natural handwriting. This means that the approach is very sensitive to handwriting strokes with asymmetric structure and symbols having very long ascender and/or descender, for instance. In such a situation, DTW (used in¹) can absorb the varying strokes' features. Additional use of spatial relations information in our method, provides better recognition performance. In what follows, we discuss origin, cause and effect of different error types, one by one.

Table 2. Error types (test data).

Error type	# of Chars.			<i>Index</i>
	M1	M2	M3	
1. Structure similarity	52	43	19	M1. Swethalakshmi <i>et al.</i> , 2007 ³⁰
2. Reduced and/or very long ascender and/or descender stroke	18	13	09	M2. CSDTW ¹
3. Others	47	42	28	M3. Our Method

4.3.1. Structure similarity

As discussed in Section 1.3, Devanagari has a structural tendency to produce confusions between the characters. However, most of the similar character pairs such as (भ,म), (थ,य), and (ध,घ) are correctly recognised, thanks to stroke spatial information. Fig. 11 gives an example between म and भ. In this illustration, cusp elimination described in Section 2.1, has been integrated with spatial relation information.

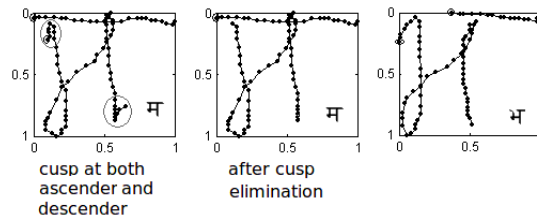
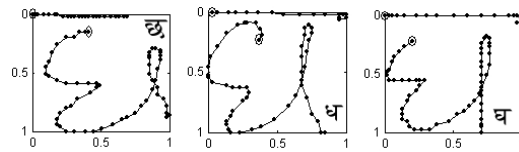
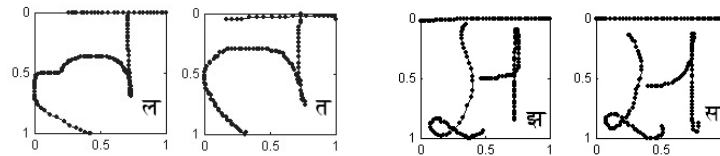


Fig. 11. Effect of stroke pre-processing and spatial information in recognition. The location of the *shirorekha* after cusp elimination can distinguish म and भ even though *texts* are similar.

However, some of the characters do not have such distinguishing features. Symbols representing them cannot be correctly classified, as is shown in Fig. 12 (a) for छ, ध and घ. Sometimes, we have observed that user introduces one-way confusions between classes of characters as shown in Fig. 12 (b). One-way confusion refers to confusion of symbol S_1 to symbol S_2 i.e., $S_1 \rightarrow S_2$ but not vice versa. In a similar manner, there are two-way confusions ($S_1 \leftrightarrow S_2$) as shown in Fig. 12 (c).



(a) Three characters: छ, ध and घ with similar handwritten representations.



(b) One-way confusion: (ल \rightarrow त). (c) Two-way confusion: (झ \leftrightarrow स).

Fig. 12. Two examples of confusion pairs – (ल \rightarrow त) and (झ \leftrightarrow स).

4.3.2. *Reduced and/or elongated ascenders and/or descenders*

Due to lack of skill in writing with a digitiser, some strokes are too long and sometimes too short. Having very long or small stroke sequences causes problems in classifying the symbol as shown in Fig. 13. However, most of these cases are correctly recognised in our method as well as in CSDTW¹, while more mis-recognitions are found to be in³⁰.

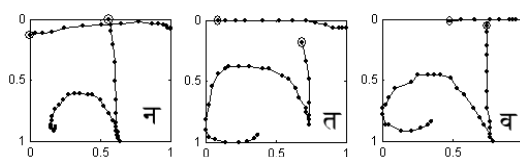


Fig. 13. Confusion between characters (even for human eyes) due to reduced descenders.

4.3.3. *Others*

Symbols with re-writing strokes as well as mis-writing symbols come under this.

Re-writing strokes. Users often add corrections to their writings by adding strokes to complete the previous ones. In an off-line context, such re-written strokes are helpful in giving a complete graphical interpretation, but they are close to nothing in case of on-line context. We have not implemented an explicit method to eliminate or treat these extra re-written strokes, since they are extremely delicate to identify or distinguish from regular strokes. Usually, however, re-writing strokes are small, and are eliminated during pre-processing. Fig. 14 shows two samples of characters having re-writing strokes, which are mis-recognised.

Mis-writing and Miscellaneous. A collection of strokes giving no supplementary information about the characters is grouped under mis-writing. These symbols are often rejected. These are usually occurred in cursive handwriting. The worst drawings are found in the miscellaneous category.

In both cases, in contrast, our method does not suffer much. Overall, compared to all methods mentioned in Table 1, our method provides interesting results. Following Table 2, Table 3 presents class-wise experimental results from our method in order to provide closer analyse.

5. Conclusion and Extensions

In this paper, we have established an approach, and validated its efficacy for on-line natural handwritten Devanagari character recognition. It uses the number of strokes used to complete a symbol and their spatial relations. Considering such a

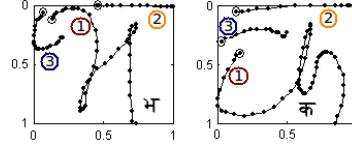


Fig. 14. Two examples of re-writing strokes from class भ and क – every third stroke is used to complete the first stroke.

Table 3. Class-wise experimental results (test data) for both consonant and vowel.

Class	# of Recog.	# of Confusion	# of Rejection	Class	# of Recog.	# of Confusion	# of Rejection
क	17	2 (फ)	2	ख	20	0	0
ग	18	2 (स)	0	घ	20	0	0
च	19	1 (य)	0	छ	20	0	0
ज	20	0	0	झ	13	5(स)	2
ट	20	0	0	ठ	20	0	0
ड	20	0	0	ढ	20	0	0
त	18	1(व), 2(न)	0	थ	18	1(य), 1(च)	0
द	16	2(ढ), 1(ट)	1	ध	19	1(घ)	0
न	18	1(म), 1(त)	0	प	20	0	0
फ	19	2(क)	0	ब	19	0	1
भ	19	2(म)	0	म	19	2(भ)	0
य	18	1(थ), 1(प)	0	र	20	0	0
ल	19	1(त)	0	व	19	1(त)	0
स	18	1(झ), 1(ग)	0	श	20	0	0
ह	20	0	0	क्ष	20	0	0
ज्ञ	20	0	0	अ	18	1(भ)	1
ए	18	1(ग)	1	उ	19	0	1
इ	18	2(ड)	0	ऊ	20	0	0

dataset, our approach outperforms state-of-the-art character recognition systems, by providing notable difference in recognition performance. Overall, the success rate is approximately 95% in less than 4 seconds per character on average.

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. However, the relational model based on MBR cannot handle huge skew-angle orientation since it is simply based on the orthogonal projection. In this situation, more elaborative spatial relation model can be used²⁵. In our dataset, we do not have strokes of more than 30°. On the other hand, size variation of the strokes does not affect. However, it is sensitive to tremor

handwriting, symbols with re-writing strokes and mis-writing.



Fig. 15. A few samples of Devanagari syllables.

To extend the work, we are going to increase the size of the dataset so that we can adequately apply cross-validation evaluation protocol. This can avoid possible confusions between the characters having structure similarity (more specifically one-way confusions) happened in dichotomous classification of dataset. On the other hand, in our current method, we do not yet integrate the recognition of Devanagari at syllable level (see Fig. 15). Syllable composition accounts for the major part of the 500 symbols used in Devanagari. Therefore, we are inspired to extend our approach up to that level.

Acknowledgement

Authors would like to thank to the anonymous reviewers for their clear guidance, comments and suggestions to improve manuscript.

References

1. Bahlmann, C., Burkhardt, H.: 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 (2004)
2. Blumenstein, M., Verma, B., Basli, H.: A novel feature extraction technique for the recognition of segmented handwritten characters. In: *Proceedings of International Conference on Document Analysis and Recognition*. p. 137 (2003)
3. Boccignone, G., Chianese, A., Cordella, L., Marcelli, A.: Recovering dynamic information from static handwriting. *Pattern Recognition* 26(3), 409 – 418 (1993)
4. Bouteruche, F., Anquetil, É., Ragot, N.: Handwritten gesture recognition driven by the spatial context of strokes. In: *Proceedings of International Conference on Document Analysis and Recognition*. pp. 1221–1225 (2005)
5. Chun, L.H., Zhang, P., Dong, X.J., Suen, C.Y., Bui, T.D.: The role of size normalization on the recognition rate of handwritten numerals. In: *IAPR TC3 Workshop of Neural Networks and Learning in Document Analysis and Recognition*. pp. 8–12 (2005)
6. Connell, S.D., Jain, A.K.: Template-based online character recognition. *Pattern Recognition* 34, 1–14 (1999)
7. Connell, S.D., Sinha, R.M.K., Jain, A.K.: Recognition of unconstrained on-line devanagari characters. In: *Proceedings of International Conference on Pattern Recognition*. pp. 368–371 (2000)
8. Doermann, D.S., Rosenfeld, A.: Recovery of temporal information from static images of handwriting. *International Journal of Computer Vision* 15(1-2), 143–164 (1995)

9. Øivind Due Trier, Jain, A.K., Taxt, T.: Feature extraction methods for character recognition – a survey. *Pattern Recognition* 29(4), 641 – 662 (1996)
10. Egenhofer, M., Herring, J.R.: Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. In: Univ. of Maine, Research Report (1991)
11. Guerfali, W., Plamondon, R.: Normalizing and restoring on-line handwriting. *Pattern Recognition* 26(3), 419–431 (1993)
12. Hu, J., Brown, M.K., Turin, W.: Hmm based on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 18, 1039–1045 (1996)
13. Joshi, N., Sita, G., Ramakrishnan, A.G., Deepu, V., Madhvanath, S.: Machine recognition of online handwritten devanagari characters. In: Proceedings of International Conference on Document Analysis and Recognition. pp. 1156–1160 (2005)
14. Kruskal, J.B., Liberman, M.: The symmetric time warping algorithm: From continuous to discrete. In: *Time Warps, String Edits and Macromolecules: The Theory and Practice of String Comparison*. pp. 125–161. Addison-Wesley (1983)
15. Lippmann, R.P.: Pattern classification using neural networks. *IEEE Comm. Magazine* 27(11), 47–50, 59–64 (1989)
16. Luo, Z., Wu, C.H.: A unit decomposition technique using fuzzy logic for real-time handwritten chinese character recognition. *IEEE Transactions on Industrial Electronics* 44(6), 840–847 (1999)
17. Malik, S., Khan, S.A.: Urdu online handwriting recognition. In: *IEEE Symposium on Emerging Technology*. pp. 27–31 (2005)
18. Marukatat, S., Artieres, T.: Handling spatial information in on-line handwriting recognition. In: *Proceedings of IEEE International Workshop on Frontiers in Handwriting Recognition*. pp. 14–19 (2004)
19. Miyajima, K., Ralescu, A.: Spatial organization in 2D segmented images: representation and recognition of primitive spatial relations. *Fuzzy Sets Systems* 65(2-3), 225–236 (1994)
20. Namboodiri, A.M., Jain, A.K.: Online handwritten script recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 26(1), 124–130 (2004)
21. Okumura, D., Uchida, S., Sakoe, H.: An hmm implementation for on-line handwriting recognition - based on pen-coordinate feature and pen-direction feature. In: *Proceedings of International Conference on Document Analysis and Recognition*. pp. 26–30 (2005)
22. Papadias, D., Sellis, T.: *Relation Based Representations for Spatial Knowledge*. PhD Thesis, National Technical Univ. of Athens (1994)
23. Plamondon, R., Srihari, S.: Online and off-line handwriting recognition: a comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 22(1), 63–84 (2000)
24. Qiao, Y., Nishiara, M., Yasuhara, M.: A framework toward restoration of writing order from single-stroked handwriting image. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28(11), 1724–1737 (2006)
25. Santosh, K.C., Lamiroy, B., Wendling, L.: Symbol recognition using spatial relations. *Pattern Recognition Letters* 33(3), 331–341 (2011)
26. Santosh, K.C., Nattee, C.: Stroke number and order free handwriting recognition for nepali. In: Yang, Q., Webb, G.I. (eds.) *Proceedings of the Pacific Rim International Conferences on Artificial Intelligence*. Lecture Notes in Computer Science, vol. 4099, pp. 990–994. Springer-Verlag (2006)
27. Santosh, K.C., Nattee, C., Lamiroy, B.: Spatial similarity based stroke number and order free clustering. In: *Proceedings of IEEE International Conference on Frontiers*

24 *K.C. Santosh et al.*

- in Handwriting Recognition. pp. 652–657 (2010)
28. Schenkel, M., Guyon, I., Henderson, D.: On-line cursive script recognition using time delay neural networks and hidden markov models. *Machine Vision and Applications* 8(4), 215–223 (1995)
 29. Sinha, R.M.K., Mahabala, H.N.: Machine recognition of devanagari script. *IEEE Transactions on Systems, Man, and Cybernetics* 9(8), 435–441 (1979)
 30. Swethalakshmi, H., Sekhar, C.C., Chakravarthy, V.S.: Spatiostructural features for recognition of online handwritten characters in devanagari and tamil scripts. In: de Sá, J.M., Alexandre, L.A., Duch, W., Mandic, D.P. (eds.) *International Conference on Artificial Neural Networks*. Lecture Notes in Computer Science, vol. 4669, pp. 230–239. Springer-Verlag (2007)
 31. Tappert, C.C., Suen, C.Y., Wakahara, T.: The state of the art in online handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 12(8), 787–808 (1990)
 32. Verma, B., Lu, J., Ghosh, M., R., G.: A feature extraction technique for on-line handwriting recognition. In: *Proceedings of IEEE International Joint Conference on Neural Networks*. pp. 1337–1341 (2004)
 33. Viard-Gaudin, C., Lallican, P.M., Knerr, S.: Recognition-directed recovering of temporal information from handwriting images. *Pattern Recognition Letters* 26(16), 2537–2548 (2005)
 34. Wang, X., Keller, J.M.: Human-based spatial relationship generalization through neural/fuzzy approaches. *Fuzzy Sets Systems* 101(1), 5–20 (1999)



K.C. Santosh is graduated from Pokhara University, Nepal in 2003 with an Engineering degree in Electronics and Communication. He received an M.S. degree by research and thesis in Information Technology from SIIT, Thammasat University, Thailand in 2007 under ADB-JSP program. In 2011, he received a Ph.D. in Computer Science from Institut National Polytechnique de Lorraine (INPL), France under INRIA-CORDI research grant.

Currently, he is working as a post-doctoral researcher at LORIA, INRIA Nancy Grand Est Research Centre, France. His research interests include handwriting analysis and recognition, graphics recognition, table extraction, sketch recognition and forensic image analysis like footwear impression evidence verification. He received several awards including ‘best student paper’ from International Conference on Knowledge Information and Creativity Support Systems (KICSS) in 2006.



Cholwich Nattee received a B.Eng. degree in Computer Engineering from Chulalongkorn University, Thailand in 1998. He received M.Eng. and D.Eng. degrees in Computer Science from Tokyo Institute of Technology, Japan in 2001 and 2004, respectively. His research areas include Machine Learning, Pattern Recognition, Natural Language Processing, and Artificial Intelligence.

He is currently an assistant professor at the School of Information, Computer, and Communication Technology, Sirindhorn International Institute of Technology,

Thammasat University, Thailand.



Bart Lamiroy is a permanent faculty member at the Université de Lorraine, in Nancy, France, and member of the associated LORIA research lab since 2000. He was a visiting scientist at Lehigh University from January 2010 to July 2011. He has a broad experience in Machine Perception. Over the years, his research topics have ranged from Content Based Image Retrieval over Visual Servoing to Document Analysis.

From 2007 to 2009, he was head of the Computer Science and IT Department at the École des Mines de Nancy, France, and has been a permanent faculty member there since 2000. Before that he was a research contractor at INRIA, after having obtained his Ph.D. in computer vision at the Institut National Polytechnique de Grenoble, France in 1998. He received his bachelor's degree in applied mathematics in 1993. He also serves on the International Association for Pattern Recognition TC-10 Committee as Dataset Curator, and on the Publicity and Publications Committee.