DYNAMIC SPACE WARPING OF STROKES FOR RECOGNITION OF ONLINE HANDWRITTEN CHARACTERS

AMRIK SEN

Department of Applied Mathematics
School of Engineering and Applied Sciences
University of Colorado, Boulder, CO 80309-0526, USA
Amrik.Sen@Colorado.edu

G. ANANTHAKRISHNAN

Department of Speech Music and Hearing
Royal Institute of Technology (KTH)
Lindstedtsvägen 24, SE-100 44 Stockholm, Sweden
agopal@kth.se

SURESH SUNDARAM[*] and A. G. RAMAKRISHNAN[†]

Department of Electrical Engineering, Indian Institute of Science
Bangalore 560012, India
[*]*suresh@ee.iisc.ernet.in*
[†]*ramkiag@ee.iisc.ernet.in*

This paper suggests a scheme for classifying online handwritten characters, based on dynamic space warping of strokes within the characters. A method for segmenting components into strokes using velocity profiles is proposed. Each stroke is a simple arbitrary shape and is encoded using three attributes. Correspondence between various strokes is established using Dynamic Space Warping. A distance measure which reliably differentiates between two corresponding simple shapes (strokes) has been formulated thus obtaining a perceptual distance measure between any two characters. Tests indicate an accuracy of over 85% on two different datasets of characters.

Keywords: Online character recognition; dynamic space warping; perceptual shape.

1. Introduction

Online Handwritten Character recognition is the recognition of characters written on an electronic page, using a pen or stylus based interface. It is called "online", because of the availability of temporal information about the trace of the pen or stylus in this method of digitized handwriting. Robust features for recognition can be extracted from this temporal information, giving online handwriting recognition an edge over its offline counterpart. Online recognition is especially useful for signature verification and for scripts like English cursive writing and Kanji script where symbols are written in a prescribed order and a prescribed style.

Several methods of feature extraction and classification have been proposed for component based, character based and word based recognition engines. Connel[4] has presented an extensive survey on the various methods. Online recognizers can be broadly classified as follows:

- based on the geometrical and structural models,[1]
- based on statistical models of the contours,[12,2]
- based on elastic matching of shapes.[3]

With the spread of the use of online handwriting input devices, the user base includes people with disparate writing styles or lexemes. The number of components, order of components or the direction of writing a particular component may vary from user to user. Figure 1 shows the English uppercase letter "B" written in different styles.

It is therefore necessary to build lexemic variability into the handwriting recognition engines. Online recognition algorithms are a bane in such situations, because different lexemes have completely different temporal information. Connel[1] has proposed detailed schemes in order to adapt existing techniques to different lexemes.

It is interesting to note that in spite of the different lexemes and component orders, the final appearance of the character is similar. Prevost and Milgram[16] have suggested both offline and online techniques for classifying online data. However, the robustness of the features obtained in online data is unavailable when made into an offline problem.

The cursive script for English often contains words written with a single component. Nathan *et al.*[14] showed that word level recognition (by considering each word as a symbol) puts a limit on the lexicon size. Writer independent, unconstrained vocabulary, word recognition engines give poor results. Similarly, in some Asian languages like Telugu, Hindi and Kannada, the unique disjoint symbol set is unmanageably large. In both the above cases, there is a strong motivation for component segmentation and stroke based classification.

Dynamic programming has been used for elastic matching of shapes by many researchers in this area. Among the first ones was Tappert[19] who proposed a dynamic two-dimensional elastic matching technique for recognizing online handwritten characters. Extensive work on this area has also been done by Uchida *et al.*[22]

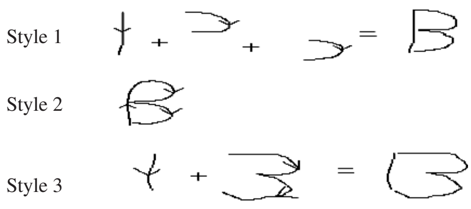

Fig. 1. Demonstration of various writing styles for the English uppercase letter "B".

and Vuori.[24] Uchida *et al.* had suggested a two-dimension warping (2DW) which maps the pixel values of one handwritten character with another with a monotonicity constraint. The advantage of the algorithm is an improvement in computation time along with global optimality. Vuori used a similar elastic matching algorithm which matched points on two curves, with continuity restrictions. It was called the Dynamic Time Warping (DTW) algorithm similar to the dynamic programming algorithms used for speech processing. In both these algorithms, it is considered that the features to be matched have a specified sequence and the matching must be done in the same sequence such that, the distance between the two handwritten curves is minimized. The number of features in each curve may differ.

The aim of this paper is to suggest an algorithm which uses robust features available in online data and then employ another dynamic programming technique called the Dynamic Space Warping (DSW) (essentially an offline method) for finding the correspondence between the strokes, and finally suggest a perceptual distance measure between any two characters. The difference between the proposed algorithm and the other dynamic programming algorithms commonly used in handwriting recognition, is that no restriction is placed on the sequence of features in time. However, the restriction is only on the spatial location, i.e. the parts of the character (strokes) should be as close to each other as possible in space. Thus, it is an offline algorithm in the sense that all the features of the character must be obtained, before the matching can be started. The proposed algorithm thus tries to combine the advantages in the offline and online methods in order to take care of various writing styles. The block diagram of the proposed algorithm is shown in Fig. 2 and the individual blocks are explained in the subsequent sections.

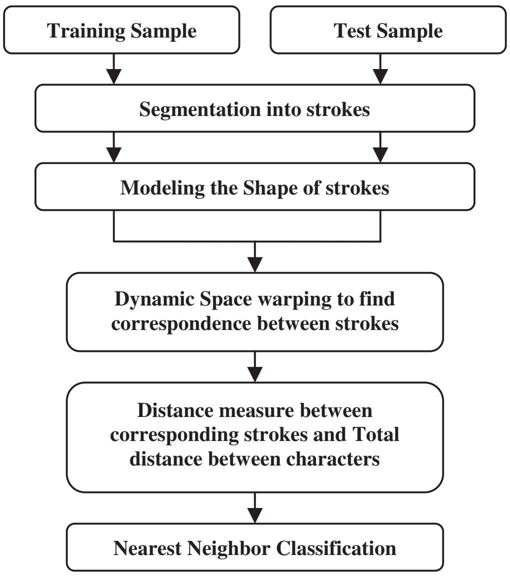
Training Sample
Test Sample
Segmentation into strokes
Modeling the Shape of strokes
Dynamic Space warping to find correspondence between strokes
Distance measure between corresponding strokes and Total distance between characters
Nearest Neighbor Classification

Fig. 2. The block-diagram for the proposed algorithm.

2. Segmentation into Strokes

The definition of a component was given by Plamondon[15] as the writing from pen down to pen up as recorded by the digital pen device. A stroke is a part of a component, written with a single motor movement.

There are several methods of segmenting components into strokes such as in Refs. 11 and 18. Stefano *et al.*[18] used the curvature at a point on the component in order to segment it into several strokes, while Li *et al.*[11] used angular velocity and momentum to obtain the strokes. The technique proposed in this paper is a modification of the method proposed by Teulings *et al.*[20] which exploits the velocity profile of a component to extract strokes.

The trace of every component starts with a low velocity, accelerates to a maximum somewhere around the middle of the stroke, and deccelarates to zero at the end of the stroke. If a person wants to change the direction of the trace in a stroke, it is natural for him/her to reduce the speed of the motion, and accelerate in a different direction. We define a stroke as the part of a component demarcated by an intentional direction change, each component having one or many strokes. Thus our segmentation algorithm reduces to finding out local minima in the instantaneous velocity profile of a component.

Instantaneous velocity is the distance traversed by the trace between two time instants divided by the difference in the time instants. Since the trace is sampled at frequency F_S, the instantaneous velocity profile is nothing but the Euclidean distance traversed between consecutive sample points as shown in Eq. (1).

$$\begin{aligned}&\forall t \text{ from } 2 \text{ to } T\\ &V_U(t) = F_S\sqrt{(X(t)-X(t-1))^2 + (Y(t)-Y(t-1))^2}\end{aligned} \tag{1}$$

where $X(t)$ and $Y(t)$ are the X and Y co-ordinates of the tth sample (coordinates are as in the tablet device), T is the total number of samples in a component and $V_U(t)$ is the instantaneous velocity profile. However, the velocity profile obtained is not smooth because the digital surface which records the trace, consists of discrete spatial locations. Figure 3 shows the unsmoothed velocity profile $V_U(t)$.

Smoothing is performed on the obtained velocity profile by an averaging filter at two levels, one with a smaller window W_S as in Eq. (2) which captures local variations, and the other with a larger window W_L as in Eq. (3) which gives a more general trend of the velocity of writing.

$$V_S(t) = \frac{1}{2W_S}\sum_{k=t-W_S}^{t+W_S} V_U(k) \tag{2}$$

$$V_L(t) = \frac{1}{2W_L}\sum_{k=t-W_L}^{t+W_L} V_U(k) \tag{3}$$

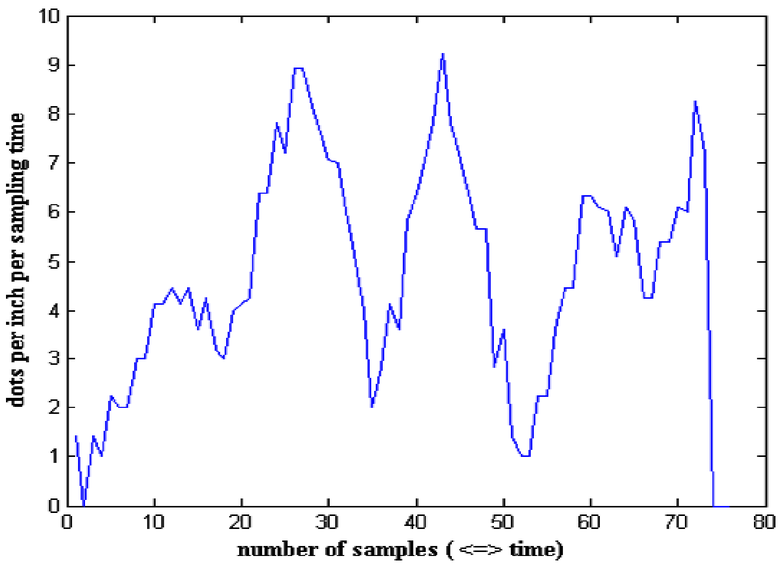

Fig. 3. The unsmoothed velocity profile for a component $V_U(t)$.

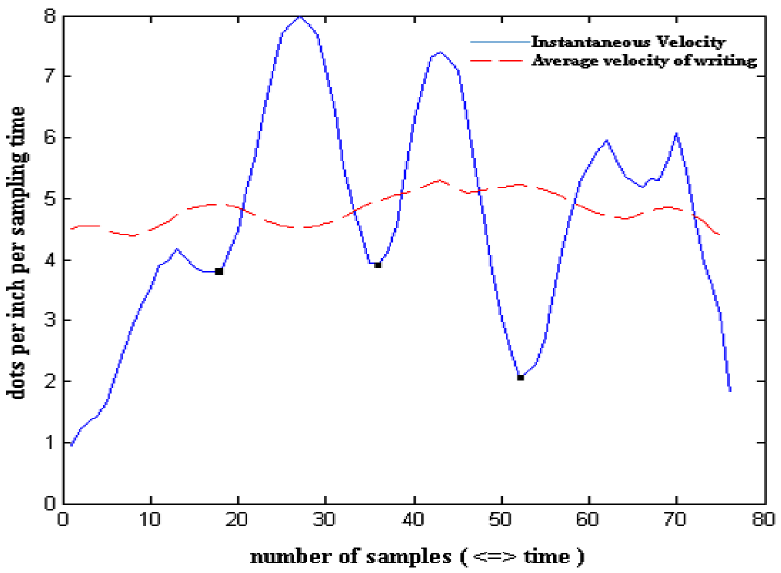

Fig. 4. The smoothed velocity profiles of instantaneous velocity $V_s(t)$ and dynamic average velocity $V_U(t)$. The segmentation points are marked in black. The beginning and the end are also stroke boundaries.

where $V_S(t)$ and $V_L(t)$ are the smoothed velocity profiles as shown in Fig. 4. We consider $V_S(t)$ as the instantaneous velocity and $V_L(t)$ as the dynamic average velocity of writing which serves as a threshold for finding local minima.

Thus a stroke is obtained as in Eq. (4)

$$\begin{aligned}&\forall t \text{ from } 2 \text{ to } T\\ &t \text{ is a stroke boundary } \tau \text{ if}\\ &V_S(t) \leq V_L(t) \ \&\ V_S(t) = \min_{k=t-W_s:t+W_s} (V_S(k))\end{aligned} \tag{4}$$

The first and last sample points of the stroke are also considered as stroke boundaries.

The selection of W_S and W_L is crucial and depends on the time sampling rate F_S of the input device. The value used in our experiments is $F_s * 0.06$ for the smaller window, W_S, which is similar to the value taken in Ref. 20. It should be the smallest time taken to make a direction change by a person. The larger window, W_L, should be the average time taken for a stroke to be completed. This value was estimated to be around $F_s * 0.5$.

3. Modeling of the Shape of a Stroke

The strokes obtained from the segmentation algorithm in Sec. 2 are *simple shapes.* By a *simple shape*, we mean that there is no significant change in direction of the trace of the stroke. We model only simple shapes, because a complex shape with a change in direction would have already been segmented by the algorithm in Sec. 2. Figure 5 shows how some complex components are segmented into simple shapes (strokes). There are many ways a simple shape can be modeled, some of which are discussed in Ref. 17. We have considered each simple shape (stroke) to be characterized by three attributes. Before describing the attributes, a few terms need to be defined.

Total stroke length (S_τ) — is defined as in Eq. (5) where τ_1 and τ_2 are the stroke boundaries

$$S_\tau = \sum_{k=\tau_1+1}^{\tau_2} V_U(k) \tag{5}$$

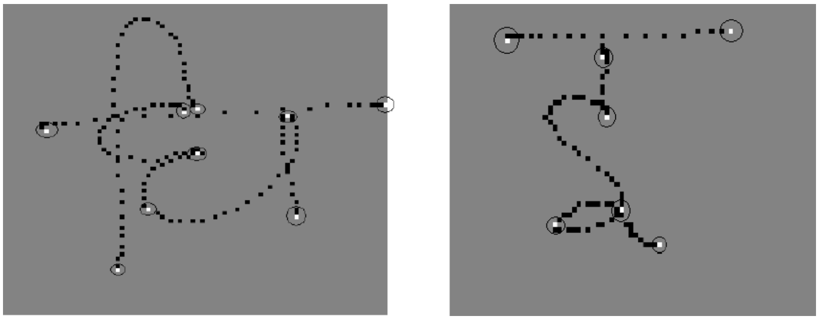

Fig. 5. Segmentation of some complex shapes. The white spots are substroke boundaries.

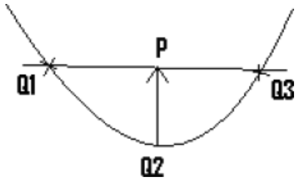

Fig. 6. A typical substroke. The line segment joining Q_1 and Q_3 is the *chord* and the line joining Q_2 and P is the *mid-point segment vector.*

Quartiles (Q_1, Q_2 and Q_3) — Every stroke is divided into four equal parts based on the distance traversed by the trace to give the three connecting points, the quartiles. The second quartile is the same as the mid-point of the stroke. Figure 6 shows a typical stroke with the three quartiles.

Chord — is the line-segment joining the two quartiles Q_1 and Q_3, as shown in Fig. 6.

Mid-point segment vector — is the vector joining the second quartile (the midpoint of the stroke) with the midpoint of the *chord* as shown in Fig. 6.

Inclination (A_1), *Proclivity* (A_2), *Curvature* (A_3) — are the three attributes which characterize an arbitrary simple shape.

3.1. *Orientation (attributes A_1 and A_2)*

Since we have to deal with a variety of shapes, defining orientation is quite a difficult problem. Our perception of orientation is very typical and depends on the symmetry of the shape. Lines and circles are mirror-image symmetric, while curves such as semi-circles are not mirror symmetric. In order to achieve this kind of a perceptual effect, we define two attributes which specify orientation, namely Inclination and Proclivity.

Inclination (A_1) is defined as in Eq. (6)

$$\mathrm{A}_1 = \frac{4 * \left(\sin^{-1}\left(\sin\left(\frac{\theta}{2}\right)\right)\right)}{\pi} \tag{6}$$

where $\theta \in [0,\ 2 * \pi]$ is the counter clockwise angle in radians made by the *chord* of a stroke with positive X-axis. Thus A_1 is a value between 0 and 2 and is periodic. It must be noted that with respect to θ, it is periodic with a period of π.

Proclivity (A_2) is defined as in Eq. (7)

$$\mathrm{A}_2 = \frac{\phi}{\pi} \tag{7}$$

where $\phi \in [0,\ 2 * \pi]$ is the counter clockwise angle in radians made by the *mid-point segment vector* with positive X-axis. Therefore, A_2 is a value between 0 and 2 and is periodic. It must be noted that with respect to θ, it is periodic with a period of $2 * \pi$.

The above described features are simple and help in distinguishing simple shapes. More features may be added to make it robust and tackle more complex shapes.

3.2. *Curvature (attribute A_3)*

Curvature is a very important property and needs to be addressed carefully. A simple shape covers a wide range of curves from lines to conic sections to circles. Most importantly, an attribute defining Curvature should be size invariant.

First, the radius of curvature r is found by fitting a circle through the three quartiles of the stroke. The radius of this circle is taken as an approximation of the radius of curvature r of the whole stroke. It is normalized against the length of the stroke S_τ as defined in Eq. (5). The attribute Curvature (A_3) is inversely proportional to the normalized radius of curvature. Thus A_3 takes the form

$$\mathrm{A}_3 = k * \left(\frac{S_\tau}{r}\right)^n \tag{8}$$

where k and n are constants to be determined from the following constraints.

- $\mathrm{A}_3 = 0$ for a straight line, since $r = \infty$.
- $\mathrm{A}_3 = 1$ for a perfect circle where S_τ would be equal to the circumference of the circle. For all the other conic section curves, A_3 should lie between 0 and 1.
- $\mathrm{A}_3 = 0.5$ for a perfect semicircle.

Solving for these constraints, the constants are $k = 1/(2^*\pi)$ and $n = 1$

Curvature (A_3) is thus defined as in Eq. (9)

$$\mathrm{A}_3 = \frac{S_\tau}{2 * r * \pi} \tag{9}$$

The ath stroke of a character P is characterized by the attributes $\{\mathrm{A}_1^a, \mathrm{A}_2^a, \mathrm{A}_3^a\}$. A segmented character is illustrated in Fig. 7 and the corresponding attributes of the numbered strokes are shown in Table 1.

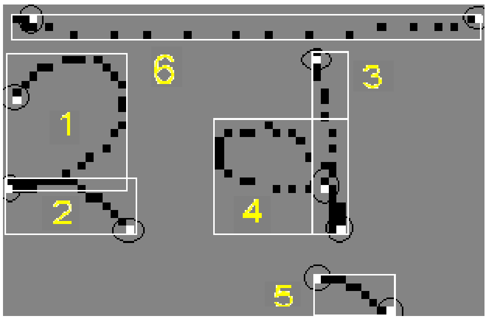

Fig. 7. Bounding boxes of normalized characters.

Table 1. Attributes of the strokes in Fig. 7.

Stroke No. Attributes	1	2	3	4	5	6
A_1	0.89	0.15	0.89	0.05	0.29	0
A_2	0.95	0.58	0.95	0.47	1.64	1.5
A_3	0.75	0.27	0.13	0.95	0.1	0

4. Dynamic Space Warping

Dynamic Space Warping deals with finding corresponding pairs between the different strokes of any two characters. This is necessary because of inversion of the component order and also distortion in the way a character is written. It is formulated as follows.

Let us assume that character P has N strokes $\{P_1, P_2, \ldots, P_N\}$ and character Q has M strokes $\{Q_1, Q_2, \ldots, Q_M\}$. For convenience, let $N \leq M$. First, the characters are normalized to a fixed range. Then the bounding box of each stroke is found as shown in Fig. 7. For normalized stroke P_a, let $\{X_{bb}(a), Y_{bb}(a)\}$ be its center and $\Lambda_{bb}(a)$ the area of its bounding box. The comparison between the various strokes is only based on the distance between the bounding boxes. The DSW algorithm tries to find a spatial fit between the strokes of two characters. It does not consider the shape of the strokes, but only the position.

The Manhattan distance between strokes P_a and Q_b is defined below

$$\delta(a,b) = |X_{bb}(a) - X_{bb}(b)| + |Y_{bb}(a) - Y_{bb}(b)| \tag{10}$$

For every stroke in character P, we have to find a corresponding stroke in character Q. The correspondence is found based on minimizing the total cost in terms of Manhattan distance, for finding a unique pairing between the strokes in character P and character Q. It is a dynamic programming algorithm and is formulated in three steps as follows. It must be noted that the stroke order is irrelevant for this algorithm, unlike other dynamic programming approaches which use time sequence information. For Dynamic Space Warping, only the proximity is important, neither the shape, nor the sequence.

DSW Algorithm. —

```
Initialization
  for $m = 1 : M$
    $E_1(m) = \delta(1, m)$
    $\Im_1(m) = \{m\}$
    $\varepsilon_1(m) = m$
  end for
```

E is the cost variable, $\Im$ is the memory variable for remembering paths. ε is the variable for referencing the paths.

```
Recursion
  for $n = 2 : N$
    for $m = 1 : M$
      $i \in \{1, 2....M\}$
      $E_n(m) = \min_{\forall i | m \notin \Im_{n-1}(i)} (E_{n-1}(i)) + \delta(n, m)$
      $\varepsilon_n(m) = \arg\min_{\forall i | m \notin \Im_{n-1}(i)} (E_{n-1}(i))$
      $\Im_n(m) = \{\Im_{n-1}(\varepsilon_n(m)), m\}$
    end for
  end for
```

$E_n(m)$ is the cost of taking the path containing m in the nth stage, $\Im_n(m)$ shows the path followed till stage $n-1$, assuming that m is selected for the nth stage and $\varepsilon_n(m)$ is the index of the path with the lowest cost. At every stage, the path with the lowest cost is taken. However, this is done in such a way that the pairing between n and m are unique, i.e. if a particular combination is selected at stage n, then the same combination should not occur in any of the previous stages. Thus, a one-one combination with the lowest cost, in terms of Manhattan distance, is selected.

```
Termination
  $\lambda_{opt} = \arg\min_{m=1:M} E_N(m)$
  $\Im_{opt} = \Im_N(\lambda_{opt})$
  $\lambda_j = k \ \forall k \in \{1, 2....M\} | k \notin \Im_{opt}$
  for $j = 1 : M - N$
    $\Upsilon_j = \arg\min_{1<n<N} (\delta(n, \lambda_j))$
  end for
```

Although, it is suboptimum in terms of the cost function because it does not consider all possible combinations, a local minima is good enough for most occasions. This recursive method, however, reduces the computation time. The computations are of the order $O(M^2N)$.

Thus $\Im_{\text{opt}}$ is a set of N elements which is the order of the strokes in character Q, that correspond with the N strokes of character P.

i.e. $\forall i$ from 1 to N; $Q_{\Im_{\text{opt}}(i)} \Leftrightarrow$ with P_i (corresponds with)

λ_j are the remaining $M - N$ strokes of the character Q which do not have a corresponding pair. Υ_j is the list of strokes in character P which is closest to the λ_j in terms of Manhattan distance.

Thus the DSW algorithm warps the spatial co-ordinates to find the best possible correspondence between the strokes in two characters as shown in Fig. 8.

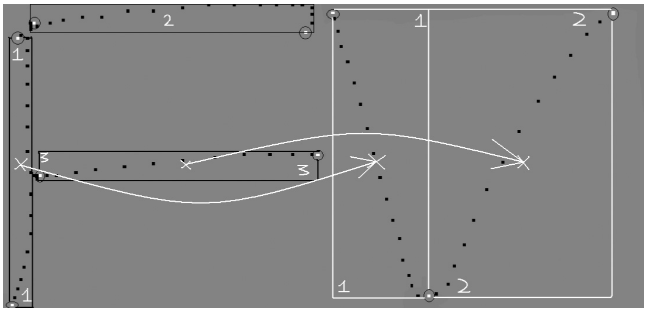

Fig. 8. Correspondence between strokes established by DSW.

5. The Distance Measure

5.1. *Distance between any two simple shapes*

The formulation of the distance measure is as tricky as the selection of the attributes for the following reasons. Each of the attributes must be assigned suitable weights. Orientation is a relative attribute and is periodic. For example, a line which makes 0° with the positive X-axis, is perceptually considered farther away from a line which makes an angle of 90° with the positive X-axis, than with a line which makes an angle of 170°. Besides, the effect of orientation depends on the *curvature*. For example, rotating a perfect circle by any angle has simply no effect on its perception. However, a line rotated by 90° has maximum perceptual distance from the original line while, the maximum difference for a curve is when rotated by 180°. Thus *Inclination* and *Proclivity* have to be weighted by a function of *Curvature*.

Considering all the above factors, the distance function $D(a,b)$ for strokes P_a and Q_b with attributes $\{\mathrm{A}_1^a, \mathrm{A}_2^a, \mathrm{A}_3^a\}$ and $\{\mathrm{A}_1^b, \mathrm{A}_2^b, \mathrm{A}_3^b\}$ respectively is formulated as in Eq. (11)

$$D(a,b) = \sqrt{(C_1 * d_m(\mathrm{A}_1^a, \mathrm{A}_1^b))^2 + (C_2 * d_m(\mathrm{A}_2^a, A_2^b))^2 + (A_3^a - A_3^b)^2} \tag{11}$$

The selection of weighting factors C_1 and C_2 are very critical. The effect of Inclination (A_1) is maximum for a line and keeps reducing as the Curvature (A_3) increases. The effect is irrelevant for any shape with a Curvature greater than 0.5. The effect of Proclivity (A_2) is minimum for a line and a circle, but maximum for a shape similar to a semi-circle. Lastly, the sum of squares of C_1 and C_2 should not be greater than 1. That is why C_1 and C_2 are selected as in Eqs. (12) and (13).

The proper selection of C_1 and C_2 allow the rotation of a line or a curve giving suitable distance measures as shown in Figs. 9 and 10. We can see that, if a line is rotated by 90°, the distance between the line and the rotated line is 1, but nearly zero when rotated by 180°. On the other hand, a curve needs to be rotated by an angle of 180° for the distance to be equal to 1. There is no effect on the distance if a circle is rotated. Thus the perceptual effect described in Sec. 3 has been captured by suitable selection of C_1 and C_2.

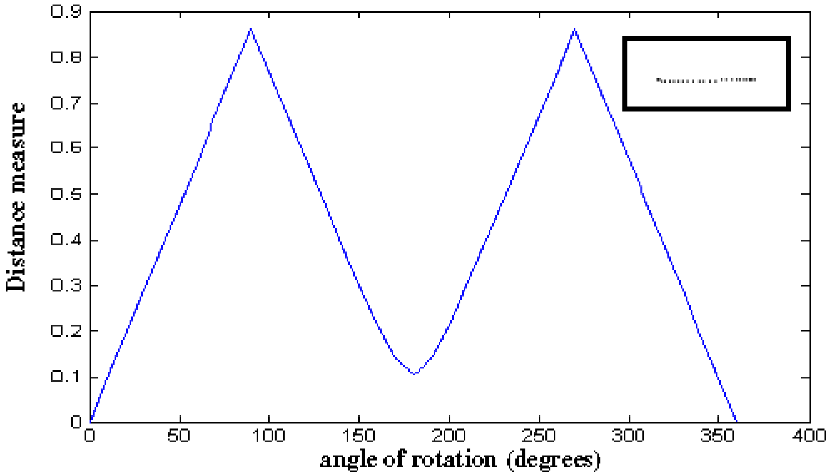

Fig. 9. Distance measure $D(a, b)$ for the line (shown inset) rotated by various angles.

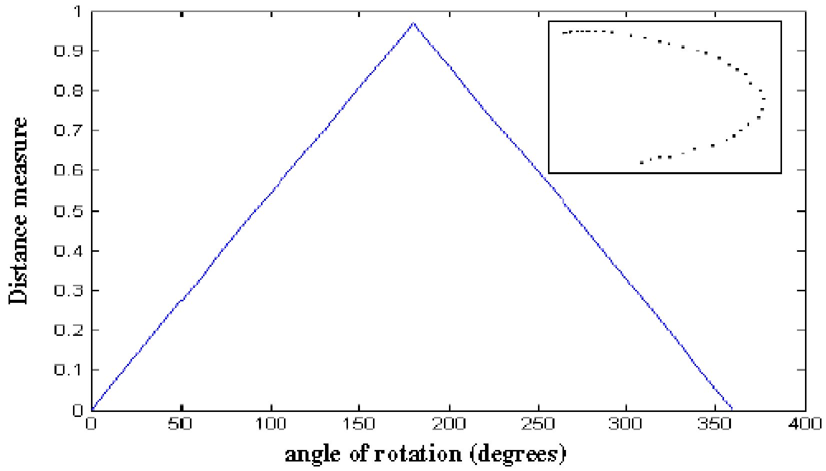

Fig. 10. Distance measure $D(a, b)$ for the curve (shown inset) rotated by various angles.

$$\begin{array}{ll} \text{if } \mathrm{A}_3^a \leq 0.5 & C_1^a = |\cos(\pi * \mathrm{A}_3^a)| \quad \text{and} \\ \text{else} & C_1^a = 0 \\ \text{if } \mathrm{A}_3^b \leq 0.5 & C_1^b = |\cos(\pi * \mathrm{A}_3^b)| \\ \text{else} & C_1^b = 0 \end{array}$$

Thus we get the weighting constants

$$C_1 = (C_1^a + C_1^b)/2 \quad \text{and} \tag{12}$$

$$C_2 = \frac{\sin(\pi * A_3^a) + \sin(\pi * A_3^b)}{2} \tag{13}$$

The function $d_m(\alpha, \beta)$ is defined as in Eq. (14) for some α and $\beta \in [0, 2]$. This is done, because the attributes A_1 and A_2 are periodic with a period 2.

$$d_m(\alpha, \beta) = (\min\{|\alpha - \beta|, 2 - |\alpha - \beta|\}) \tag{14}$$

Figures 9 and 10 show the distance measure as a function of angle of rotation (from $0°$ to $360°$) for a line and curve, respectively. One can realize why the constants have been selected as above.

$D(a, b)$ exhibits three properties of a metric[5] namely reflexivity ($D(a, a) = 0$), non-negativity ($D(a, b) \geq 0$) and symmetry ($D(a,b) = D(b,a)$). $D(a, b)$ does not exhibit the triangular inequality property. However, since this distance function tries to mimic perceptual distance, exhibiting the triangular inequality property is not a must as explained in Ref. 9.

5.2. *Distance between any two characters*

The final distance is calculated in Eq. (15) between two characters P with strokes $\{P_1, P_2, \ldots, P_N\}$ and Q with strokes $\{Q_1, Q_2, \ldots, Q_M\}$

$$\begin{aligned} D_{ch}(P, Q) = \Bigg(\frac{1}{N^2} \Bigg(\sum_{k=1}^{N} D\left(k, \Im_{opt}(k)\right) * \delta\left(k, \Im_{opt}(k)\right) \Bigg) \\ + \Bigg(\sum_{j=1}^{M-N} D(\Upsilon_j, \lambda_j) * \delta(\Upsilon_j, \lambda_j) * \Lambda_{bb}(j) \Bigg) \Bigg) \end{aligned} \tag{15}$$

There are two terms in Eq. (15). The first one represents the average distance between best corresponding strokes. The second term represents penalization of all the unpaired strokes present in character Q. The penalization is weighted by the area of the bounding box. The larger the bounding box is, the larger the penalization. Each term is further weighted by the Manhattan distance between the bounding boxes of the strokes. More importance is given to the strokes which lie close to each other in space.

It is easy to observe that the function $D_{ch}(P, Q)$ exhibits the same three properties of a metric, which the function $D(a, b)$ exhibits.

6. Results and Discussion

As a preliminary testing scheme, classification of characters has been done using k-nearest neighbor classifier (k-NNC) based on the distance measure (D_{ch}) computed in Sec. 5. Testing has been done on two datasets, one of Tamil characters (156 class) collected from 168 users with an average of five trials per user and the second dataset for English uppercase and lowercase characters (26 classes each) has been collected from 35 users with an average of eight trials per user. Thus, the Tamil dataset has a total of 843 trials and the English dataset has a total of 270 trials. Both the datasets have been collected using the Compaq tc 1100 tablet PC. The English data was collected specifically for users with different styles of writing and contains characters written in various styles, number of components, order of components and direction of components. The data contained both left-handed writers as well as right-handed writers.

The experiments were performed using a jack-knife principle where the training and validation data samples were rotated through the database. In each cross-validation, the test dataset was always around 10% of the number of samples per class in the dataset and selected randomly. The remaining samples were the training and validation sets. The optimum value of k (for the k-NNC) was selected based on the best accuracy obtained on this test set. The number of trials per character used for training was varied in different experiments. The remaining trials were used for validation. For example, in the Tamil dataset, for the first experiment, only ten samples per class (character) were used for training. 84 trials per character were used for testing and the remaining 749 trials per character were used for validation. In each experiment, the choice of the number of training samples being 10, 50, 100 for English and Tamil, and 500 for the Tamil dataset, were chosen randomly. The choice of the training and validation trial sets were rotated ten times as a jack-knife to give ten cross-validations. The proposed algorithm is also tested against a standard DTW implementation as in Ref. 10 and with an SVM implementation of resampled characters as in Ref. 7. The results, presented in Table 2, are the average of all the cross-validation sets.

We can see that an advantage of the proposed algorithm is in the higher accuracy for a small number of training samples. The reason for that could be attributed to the abstraction obtained by breaking the complex characters into simple shapes and describing them by simple features. At the same time, the proposed algorithm combines the use of online features for stroke identification and modeling, but uses an inherently offline method, dynamic space warping, to compare the characters. As the number of training samples increases, the various methods perform almost

Table 2. % Error of the proposed algorithm using different number of training samples per class. 10% of the samples per class randomly chosen from the dataset were used for testing. The number in the brackets is the optimum value of K obtained from the test set.

Dataset	Algorithm	Number of Samples Per Class Used in Training			
		10	50	100	500
Tamil	DSW	**67.7% (K = 3)**	**43.6% (K = 9)**	21.3% (K = 11)	13.3% (K = 11)
	DTW	82.8% (K = 4)	64.7% (K = 9)	21.4% (K = 10)	15.1% (K = 12)
	SVM	86.6%	68.8%	**20.28%**	**12.0%**
English Lowercase	DSW	**62.7% (K = 3)**	**33.6% (K = 5)**	14.2% (K = 7)	—
	DTW	82.3% (K = 3)	49.0% (K = 6)	15.9% (K = 7)	—
	SVM	91.9%	46.5%	**13.8%**	—
English Uppercase	DSW	**63.4% (K = 3)**	**33.7% (K = 6)**	15.0% (K = 6)	—
	DTW	84.0% (K = 2)	51.3% (K = 6)	15.4% (K = 7)	—
	SVM	88.8%	48.7%	**13.9%**	—

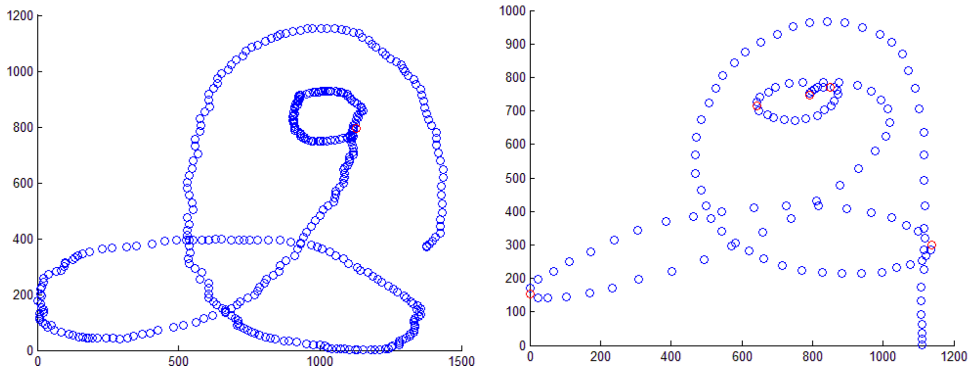
Fig. 11. A Tamil character written with different speeds. The character to the left does not show the typical velocity pattern as in the character to the right. This results in segmentation errors.

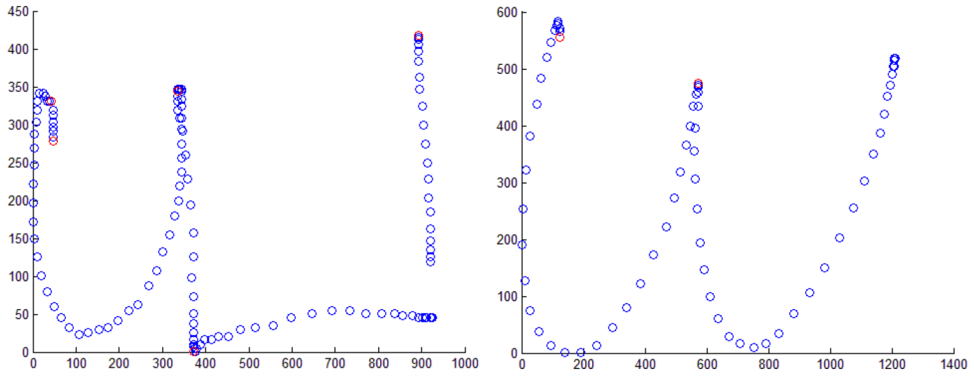
Fig. 12. The two lines in the Tamil character to the left have been replaced by a curve in the character to the right. This leads to recognition errors due to incorrect calculation of the distance measure.

similarly. This demonstrates the robustness of the algorithm in a way, because it deals well with distortions of a few template characters.

For larger databases, which contain several styles of writing in the training set, DSW may not be the best method, because of the higher time complexity. However, for larger databases, the time complexity can be reduced by selecting a few prototypes from the entire data. Methods used by Gowda and Krishna[6] who performed clustering on quasi-metrics can be used effectively for this prototype selection.

The most common errors, using the proposed algorithm, occur when the handwriting is either too slow or too fast. Figure 11 shows the same character written by two writers. The one on the left is written very slowly and there is hardly any difference in the relative velocity throughout the character. On the right-hand side, we see the same character written with a reasonable average velocity.

Table 3. Accuracy of the proposed algorithm on different standard databases and a comparison with other competitive algorithms.

Database	Number of Training Samples	Number of Testing Samples	%Error	Comparing Method	% Error for Comparing Method
Tamil Online	50683	26926	9.7%	HMM[22]	9.3%
				Neural Networks with Online and Offline features[23]	6.5%
				KNN-DTW[17]	8.8%
Unipen 1a	15953	8598	3.5%	OnSNT[23]	2.6 %
				KP-NN[8]	3.7%
Unipen 1b	28069	16401	9.3%	OnSNT	8.7%
Unipen 1c	61360	37122	13.2%	OnSNT	11.5%
Unipen 2	122668	122083	29.9%	OnSNT	27.4%

The other sources of errors are, when a curve is replaced by a pair of lines or vice-versa. In Fig. 12, the two lines in the character to the left are replaced by a curve in the character to the right.

In order to test the algorithm against other state of the art algorithms, tests were performed using two standard datasets. The first one is the Online Tamil Handwriting Recognition competition[13] and the second one is the Unipen Database.[23] The results are presented in Table 3.

We can see the method presented is comparable with the state-of-the-art, although not better. Secondly, it is quite time consuming because no prototype selection has been employed. Both the accuracy and time consumed may be improved by employing pruning and prototype selection techniques.

7. Conclusion and Future Work

An algorithm has been proposed which uses online features to segment components into simple shapes (strokes). Simple shapes have been characterized using three attributes. Correspondence between the strokes is obtained using a DSW formulation. The DSW performs a spatial warping of one character so that the distance between the strokes of one character and the other are minimized. A distance function has been proposed to find the perceptual distance between any two handwritten characters. A maximum average accuracy of 86% and 85% for Tamil and English datasets has been obtained by using k-NNC. The accuracy on the Tamil Online dataset is 90.3% and for Unipen 1a, it is 96.5%.

Future work includes applying this algorithm for different datasets such as English words and Hindi and Kannada databases. It should also be able to address segmentation of running handwriting in English (cursive writing). Using the distance measure for clustering and prototype selection is also a useful direction. A more robust segmentation algorithm is necessary to avoid either over segmentation or under segmentation. Several shape description features could be added to make the features more robust. The features must be able to address a little more complexity, in order to compensate for inaccuracies in the segmentation algorithm. Prototyping and pruning larger databases can be used to reduce the time complexity.

Acknowledgment

The authors would like to acknowledge Sriganesh Madhvanath (SriG), Senior Research Scientist at HP Labs India for providing help and support with the DTW toolkit.

References

1. K. F. Chan and D. Y. Yeung, Elastic structural matching for on-line handwritten alphanumeric character recognition, *Proc. 14th ICPR*, Brisbane, Australia (1998), pp. 1508–1511.
2. M.-Y. Chen and A. Kundu, Multi level HMM for handwritten word recognition, *Proc. ICASSP*, Detroit, USA, **4**(1) (1995) 2623–2626.
3. K.-W. Cheung, D.-Y. Yeung, and R. T. Chin, Bidirectional deformable matching with application to handwritten character extraction, *IEEE Trans. PAMI* **24** (2002) 1133–1139.
4. S. D. Connel, Online handwriting recognition using multiple pattern class models, Doctor of Philosophy Thesis, Department of Computer Science and Engineering, Michigan State University, 2000.
5. R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification* (John Wiley and Sons, NY, 2001).
6. K. C. Gowda and G. Krishna, Agglomerative clustering using the concept of mutual nearest neighbourhood, *Patt. Recogn.* **10** (1977) 105–112.
7. S. Hariharan, A. Jayaraman, V. S. Chakravarthy and C. Sekhar, Online handwritten character recognition of devanagiri and telugu characters using support vector machines, *Proc. IWFHR* (2006).
8. J. Hébert, M. Parizeau and N. Ghazzali, A new fuzzy geometric representation for on-line isolated character recognition, *Proc. 14th ICPR* (1998) 1121–1123.
9. D. Jacobs, D. Weinshall and Y. Gdalyahu, Condensing image databases when retrieval is based on non-metric distances, *Proc. 6th ICCV* (1998), pp. 596–601.
10. N. Joshi, G. Sita, A. G. Ramakrishnan and S. Madhavnath, Comparison of elastic matching algorithms for online Tamil handwritten character recognition, *Proc. IWFHR* (2004), pp. 444–449.
11. X. Li, M. Parizeau and R. Plamondon, Segmentation and reconstruction of online handwritten scripts, *Patt. Recogn.* **31** (1998) 675–684.
12. X. Li, R. Plamondon and M. Parizeau, Model based on-line handwritten digits, *Proc. 14th ICPR*, Brisbane, Australia (1998), pp. 1134–1136.
13. S. Madhvanath and S. M. Lucas, IWFHR 2006 online tamil handwritten character recognition competition. In Suvisoft LTD., *Proc. 10th IWFHR*, La Baule (France) (2006), pp. 239–242.
14. K. S. Nathan, H. S. Beigi, J. S. Clary and H. Maruyama, Real-time on-line unconstrained handwriting recognition using statistical methods, *Proc. ICASSP*, Detroit, USA, **4**(1) (1995) 2619–2622.
15. R. Plamondon, A model-based segmentation framework for computer processing of handwriting, *Proc. ICPR* **2** (1992) 303–307.

16. L. Prevost and M. Milgram, Automatic allograph selection and multiple expert classification for totally unconstrained handwritten character recognition, *Proc. 14th ICPR*, Brisbane, Australia (1998), pp. 381–383.
17. E. H. Ratzlaff, Methods, report and survey for the comparison of diverse isolated character recognition results on the UNIPEN database, *Proc. 7th Int. Conf. Document Analysis and Recognition (ICDAR)* (2003), pp. 623–628.
18. C. D. Stefano, A saliency based segmentation method for online cursive handwriting, *Int. J. Patt. Recogn. Artific. Intell.* **18** (2004) 1139–1156.
19. C. C. Tappert, Cursive script recognition by elastic matching, *IBM J. Res. Devel.* **26** (1982) 765–771.
20. H. L. Teulings, Invariant handwriting features useful in cursive script recognition, *Fundamentals of Handwriting Recognition*, ed. S. Impedovo (1994), pp. 178–189.
21. A. H. Toselli, M. Pastor and E. Vidal, On-line handwriting recognition system for tamil handwritten characters, *Proc. IbPRIA*, Part 1, LNCS 4477 (2007), pp. 370–377.
22. S. Uchida and H. Sakoe, A monotonic and continuous two-dimensional warping based on dynamic programming, *Proc. 14th ICPR* **1**(2) (1998) 521–524.
23. Unipen Project, http://www.visionbib.com/bibliography/char997.html#TT84973 (1994).
24. V. Vuori, J. Laaksonen, E. Oja and J. Kangas, On-line adaptation in recognition of handwritten alphanumeric characters, *Proc. ICDAR* (1999) 792–795.