Journal of Universal Computer Science, vol. 17, no. 1 (2011), 81-93
submitted: 8/4/10, accepted: 12/12/10, appeared: 1/1/11 © J.UCS

Fusion of Complementary Online and Offline Strategies for Recognition of Handwritten Kannada Characters

Rakesh Rampalli
(Indian Institute of Science, Bangalore, India
rrakesh100@gmail.com)

Angarai Ganesan Ramakrishnan
(Medical Intelligence and Language Engineering Laboratory,
Department of Electrical Engineering, Indian Institute of Science
Bangalore, India 560012.
ramkiag@ee.iisc.ernet.in)

Abstract This work describes an online handwritten character recognition system working in combination with an offline recognition system. The online input data is also converted into an offline image, and parallely recognized by both online and offline strategies. Features are proposed for offline recognition and a disambiguation step is employed in the offline system for the samples for which the confidence level of the classifier is low. The outputs are then combined probabilistically resulting in a classifier out-performing both individual systems. Experiments are performed for Kannada, a South Indian Language, over a database of 295 classes. The accuracy of the online recognizer improves by 11% when the combination with offline system is used.

Key Words: Online handwriting recognition, Offline handwriting recognition, Classifier fusion, Kannada script, Re-sampling, Pen direction angle, Support vector machine, Spline curve, Directional distance distribution, Nearest stroke pixel, Transition count, Projection profiles, Principal component analysis, Mahalanobis distance.

Categories: J.6, I.2.1, I.4.9, I.5.4

1 Introduction

Online handwriting recognition systems employ an active tablet, on which the user writes the text. The tablet captures the movement of pen tip on its screen in terms of PEN_DOWN, PEN_UP information and x,y co-ordinates sampled at equal intervals of time. This transducer is connected to a computer online and the data captured by the tablet is sent to the recognition system. Like any other pattern recognition system, the conventional handwriting recognition system fundamentally consists of basic building blocks like segmentation, pre-processing, feature extraction and classification. On-line handwriting recognition methods using stroke-information are generally expected to achieve higher accuracy than the off-line methods. However, the variations in the number of strokes,

stroke direction and stroke order impede the increase in accuracy.
On the other hand, off-line recognition methods do not depend on the number and the order of strokes. The number of strokes decreases when words are cursively written and increases due to irregular PEN_UP/PEN_DOWN. So in our current work, online data is converted to an offline image to explore the advantages of parallel offline recognition. The online input data is independently processed by the online recognizer, while the converted image is recognized by the offline system[Hamanaka et al., 1993]. The outputs of the two recognizers are then combined probabilistically resulting in a classifier performing better than both online and offline systems.
Classifier combination, if judiciously used, is well known to improve the accuracy of individual recognizers. Although many techniques have been proposed to effectively combine different systems, the most important aspect is the diversity of classifiers. An important aspect of diversity is that misclassifications that are frequent for offline system are rare for online one and vice versa. i.e. the errors of the two classifiers should be uncorrelated. Since we expect many of the errors of online and offline classifiers to be uncorrelated, we propose that by combining the two classifiers we can improve the recognition accuracy [Vinciarelli et al., 2003] . Accordingly, we propose a novel method to enhance the accuracy of online handwritten recognition using offline image-based features.

2 Kannada script and database

Kannada is the official language of the South Indian state of Karnataka. Modern Kannada alphabet has a base set of 52 characters, comprising 16 vowels and 36 consonants. There are 2 more consonants used in old Kannada, namely the retroflex /La/ and the stressed /Ra/, taking the total number of consonants to 38 [Prasad et al., 2009] . Further, there are consonant modifiers (conjuncts) and vowel modifiers. The number of these modifiers is the same as that of base characters, namely 52. Compound characters called aksharas are formed by graphically combining the symbols corresponding to consonants, consonant modifiers and/or vowel modifiers using well defined rules of combination. Thus, the number of possible consonant-vowel combination aksharas is 38 x 16 = 608. Similarly, the number of possible consonant-consonant-vowel aksharas is 38 x 38 x 16 =23104. The script has its own numerals too. In addition, the script includes special symbols used in poetry, shlokas (prayer chants) and Kannada grammar.
While designing a character recognition system, if we consider each akshara as a separate class, the number of classes becomes prohibitively high. However, in Kannada, consonant-modifiers and some of the vowel modifiers are mostly written separately from the base character. So, if we treat each connected component as a different class, the number of classes in recognition can be reduced

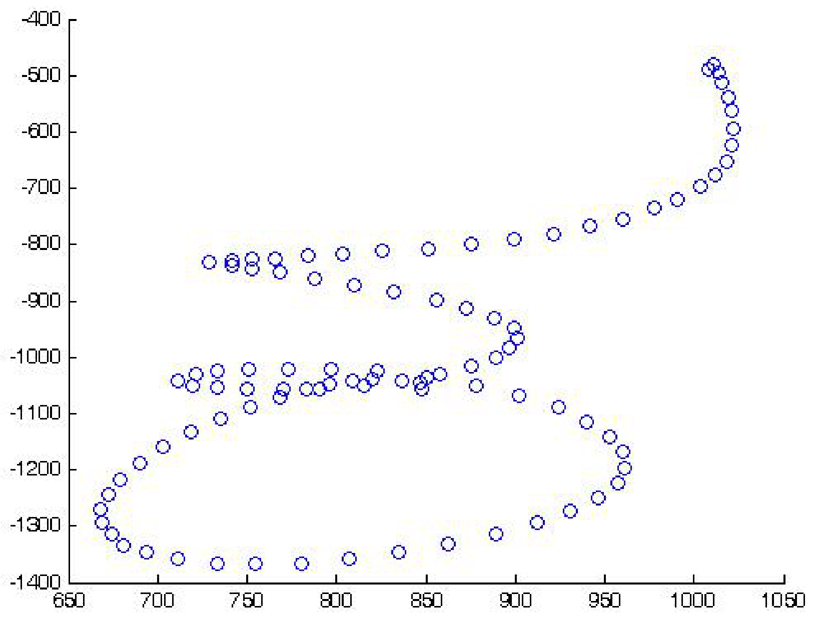


Figure 1: Raw data of a Kannada character sample.

by a great extent [Prasad et al., 2009]. MILE lab database has been created by collecting data from 69 different writers so that the recognition engine could be trained with different styles of handwriting in turn making it writer independent [Kunwar et al., 2010].

3 Online recognition

The data obtained from the tablet PC is pre-processed, where the noise is removed and the size of the character normalized. The data is of variable length and may contain duplicate points since they are sampled at equal intervals of time. Figure 1 shows the raw data of a sample of the Kannada consonant, /ka/. To make the data of equal length independent of the writing speed, we obtain new sample points regularly spaced with respect to arc length by simple linear interpolation. Thus the entire character is resampled to a fixed number of points with all the x and y coordinates normalized between 0 and 1. Figure 2 shows the normalized and resampled version of the character data given in Fig. 1. The following features are derived from the online trace.

1) *x-y co-ordinates*: The samples of the preprocessed symbols are used as features.

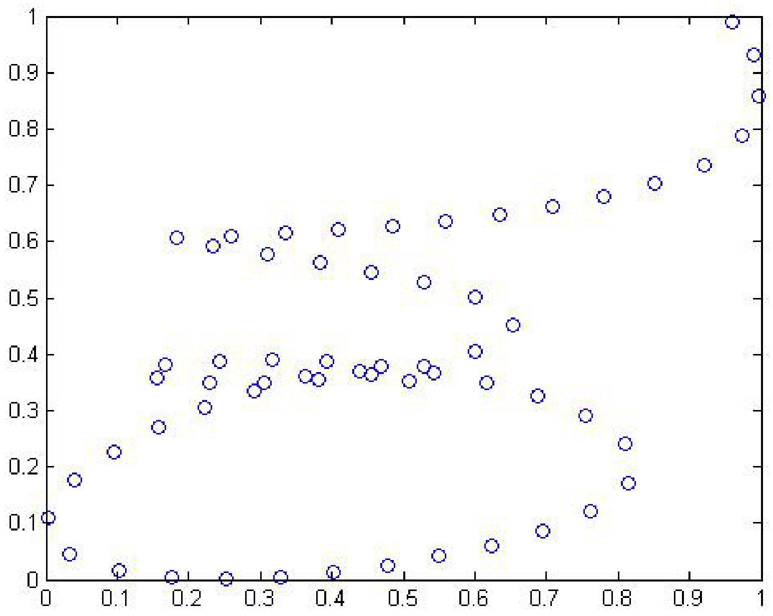


Figure 2: The character shown in Fig. 1, resampled along the arc length.

2) *Pen direction angle*: At each point, the direction of pen tip movement from that point to the next point is used as a feature.
3) *First and second derivatives of x and y co-ordinates*: At any point of pre processed character, first and second derivative of x and y are calculated as,

$$x'_i = \frac{\sum_{t=1}^{w} t(x_{i+t} - x_{i-t})}{2\sum_{t=1}^{w} t^2} \quad (1)$$

$$y'_i = \frac{\sum_{t=1}^{w} t(y_{i+t} - y_{i-t})}{2\sum_{t=1}^{w} t^2} \quad (2)$$

where $2w$ is the number of neighboring points involved in computation. Here w is taken as 2. The second derivatives are calculated by replacing x_i, y_i with x'_i, y'_i in the same equations.
All the above features are calculated at each of the 60 resampled points.

3.1 SVM1 for online classification

The above features are calculated for all the training samples of different classes and are used to train a SVM classifier. The same features are computed for the test samples. Let F_1 be the vector output from this classifier, containing the posterior probabilities of the test sample belonging to each of the classes.

4 Converting online data into an image

The normalized data with all the x-y co-ordinates between 0 and 1 is converted into an image by fitting a spline curve of degree 3 along the points. The curve fitting is carried out stroke by stroke. The individual images so obtained are added to get the final image of the character or symbol. A binary image is obtained with all the stroke pixels at intensity level 1 and all the background pixels at intensity level 0. The output image is resized to 64x64 pixels. Figure 3 shows the image obtained by performing the above procedure on the character sample shown in Fig. 2.

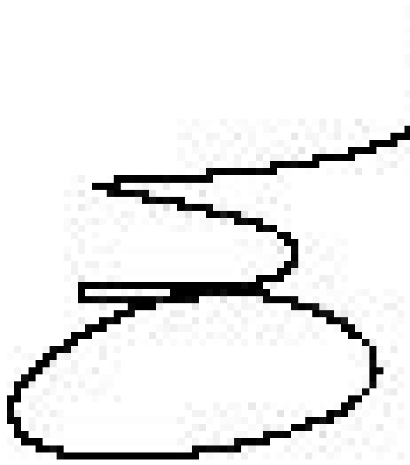


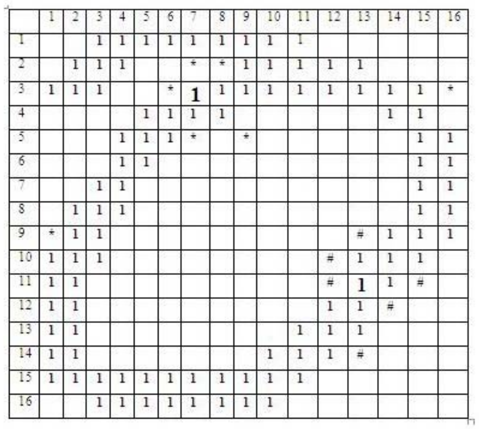
Fig. 3. Binary image of the character sample in Fig. 2, obtained by spline fitting.

5 Offline recognition

The offline handwriting recognition makes use of a composite feature vector obtained by the concatenation of four different feature vectors, namely 1) directional distance distribution 2) distances of nearest stroke pixels 3) transition count and 4) projection profiles.

5.1 Directional distance distribution (DDD)

This feature is based on the distance of each pixel to its nearest neighboring pixel of opposite kind, in each of the 8 directions around it. Two sets of 8 bytes



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1			1	1	1	1	1	1	1	1	1					
2		1	1	1			*	*	1	1	1	1	1			
3	1	1	1			*	**1**	1	1	1	1	1	1	1	1	*
4					1	1	1	1						1	1	
5				1	1	1	*		*						1	1
6				1	1										1	1
7			1	1											1	1
8		1	1	1											1	1
9	*	1	1										#	1	1	1
10	1	1	1									#	1	1	1	
11	1	1										#	**1**	1	#	
12	1	1										1	1	#		
13	1	1									1	1	1			
14	1	1								1	1	1	#			
15	1	1	1	1	1	1	1	1	1	1	1					
16			1	1	1	1	1	1	1	1						

Figure 4. Obtaining the directional distance distribution from the binary pattern of a character.

which we call the W set and the B set are allocated. For a white pixel, the set W is used to encode the distances to the nearest black pixels in 8 directions and the set B is filled with zeros. For a black pixel, the set B is used to encode the distances to the nearest white pixels in 8 directions and the W set is filled with zeros. By regarding the image as periodic in both x and y directions, we get a better discriminating power of the feature [Oh et al., 1998] .

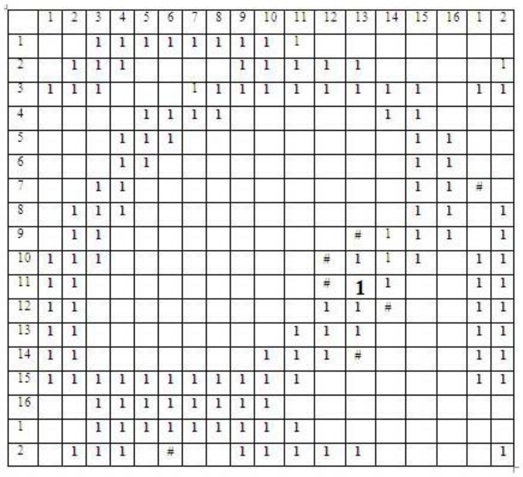
As an example, consider the foreground pixel at location (3,7) in Fig. 4. Since it is a white (foreground) pixel, all the B entries are zeros. The '*' entries in the matrix indicate the nearest black pixels in the different directions, and their distances from the pixel under consideration form the vector W. Accordingly, the WB encoding is given by,

9	1	1	1	1	6	2	2	0	0	0	0	0	0	0	0

5.1.1 *Special cases*

1) A case arises when we reach the array boundary without encountering any opposite pixel. This is resolved by creating periodic extension of the image.
2) Another case arises when the distance is higher than the maximum of the

horizontal and vertical dimensions of the array before or when meeting the opposite pixel. In this case, we consider the maximum of the horizontal and vertical dimensions of the array as the distance in that direction.



	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	1	2
1			1	1	1	1	1	1	1	1	1							
2		1	1	1					1	1	1	1	1					1
3	1	1	1				1	1	1	1	1	1	1	1	1		1	1
4					1	1	1	1						1	1			
5				1	1	1									1	1		
6				1	1										1	1		
7			1	1											1	1	#	
8		1	1	1											1	1		1
9		1	1										#	1	1	1		1
10	1	1	1									#	1	1	1		1	1
11	1	1										#	**1**	1			1	1
12	1	1										1	1	#			1	1
13	1	1									1	1	1				1	1
14	1	1								1	1	1	#				1	1
15	1	1	1	1	1	1	1	1	1	1	1						1	1
16			1	1	1	1	1	1	1	1								
1			1	1	1	1	1	1	1	1	1							
2		1	1	1		#			1	1	1	1	1					1

Figure 5. A part of the periodic version of the binary pattern

Consider Fig. 5, where partial periodic extension of the binary image of the character is shown in both x and y directions. The 'hash' signs in the matrix indicate the nearest pixel of opposite kind in each direction for the pixel at (11,13). To calculate $w(1)$, we move as follows: (11,13)->(10,14)->(9,15)->(8,16)->(7,1). At (8,16), we hit the boundary without having encountered any opposite pixel. By treating the image as periodic, we meet the opposite pixel at (7,1). The WB encoding is hence,

2	4	2	1	1	7	3	1	0	0	0	0	0	0	0	0

5.1.2 *Obtaining the final DDD vector*

After computing the WB encodings of all the pixels in the image, we divide the character image matrix of size NxN into sub-images, each of size (N/4)x(N/4).

For each of the resulting 16 grids, the average of the WB encodings is computed. So we get a 16 (bytes per average WB) * 16 (No. of grids) feature vector which is normalized to (0 1) before feeding it into the second SVM, used as one of the offline classifiers. The vector so formed is resized to 1 * 256.

5.1.3 *Properties of the DDD feature*

Two major properties contribute to the discriminating power of this feature:
1) The array is being regarded as periodic. Consider the characters 3 and 8 which have a common pattern to the right. If the array wasn't regarded as periodic, the feature values would have been same for both classes in those common regions. So the discriminating power of the feature deteriorates to a certain degree. In contrast, the DDD feature values of the right part of this example are also affected by pixel distributions at the left area because the ray passes the array boundary and proceeds to the left area.
2) The second property is that it contains rich information of the distribution of both black and white pixels over the whole area of the pattern.

5.2 Distances to nearest stroke pixels (NSP)

This feature is based on the distance of the nearest stroke pixel in each row and column from each of the boundaries. The distances in a row are counted both by moving from left to right and right to left till we hit a stroke pixel. Let r_1 be the vector of counts obtained by starting at the left boundary. Since this is done for all the rows, r_1 is of size N for an NxN image. Let r_2 be the vector of counts obtained by starting at the right boundary and moving towards left. The same procedure is performed on each column to compute c_1 and c_2 by starting from the bottom and the top, respectively. The entries in the feature vector vary from 0 to N-1; The value of '0' is obtained when the starting element in a row is a stroke pixel and N-1, when we hit the boundary or a stroke pixel at the boundary. All the feature vectors r_1, c_1, r_2 and c_2 are normalized before concatenating them to form a feature vector of size 4*N. From the 16x16 image in figure 4, r_1 =[2 1 0 4...0 2] ; r_2 =[5 3 1 1...5 6] ; c_1=[1 1 0 0...6 8] ; c_2=[2 1 0 0...2 4];
The nearest stroke pixel (NSP) feature vector = $[r_1\ c_1\ r_2\ c_2]$

5.3 Transition count (TC)

The number of transitions in any row or column also proves to be a distinguishing feature. The count in a particular row is obtained by counting the number of transitions from 1 to 0 and 0 to 1 from the start to the end of the row. The same is repeated for all the rows. Thus we obtain a vector h_1 by moving from

left to right along each row. Similarly the count in a particular column v_1 can be obtained by moving from top to bottom along the column. Both h_1 and v_1 are normalized and concatenated to form a feature vector of length 2*N for an image of size NxN [Toraichi et al. 1996]. For the image in Fig. 4, h_1 = [2 4.....1 2] ; v_1 =[4 4.....2 2];
The transition count feature vector =[h_1 v_1]

5.4 Projection profiles (PP)

The horizontal projection profile (HPP), computed by adding the total number of 1's along each row and the vertical projection profile (VPP), computed by adding the total number of 1's along each column (VPP) are concatenated to form a vector of size 2*N for a NxN image. For the image in figure 4, VPP = [7 10 9........8 4] ; HPP =[9 8 12........11 8]. The HPP and the VPP are normalized before concatenation.
The projection profile (PP) feature vector = [HPP VPP].

5.5 Offline feature vector

This is obtained by the concatenation of all the above four feature vectors, namely directional distance distribution, nearest stroke pixel, transition count and projection profiles. For an image of size NxN, the offline feature vector is given by,

$$[DDD(1*256); NSP(4*N); TC(2*N); PP(2*N)]$$

5.6 SVM2 for offline classification

The offline feature vector is computed for the training samples of all the classes. A second SVM is used for offline classification. The output of the SVM would be the posterior probability that the test sample belongs to a particular class. Let F_2 be the vector output from SVM2 which contains the posteriors of the test sample belonging to each of the classes. Let p_1, p_2 be the likelihoods of the first and the second top most choices of SVM classifier 2, respectively and L_1, L_2 be their respective class labels. If the SVM classifies correctly, in most cases the likelihood for the correct class would be high ($p_1 > 0.5$). If the classifier gets confused, then it would not be so and there is said to be certain amount of ambiguity. To resolve this ambiguity, we employ a principal component analysis (PCA) based approach on the offline images of the data. It is observed that the two offline recognition systems, using PCA and SVM2, are complementary in most cases. Hence the two recognition systems are combined appropriately by invoking the PCA based system only when the confidence level of the output from

SVM2 is less. The confidence of the SVM classifier 2 is checked by picking the top two posteriors of SVM outputs namely p_1 and p_2. If the difference between the two is less than a fixed threshold ($T_p = 0.11$ in our case), then the module which uses PCA for images is invoked.

5.7 PCA based disambiguating offline classifier

PCA involves a mathematical procedure that transforms a number of possibly correlated variables into a smaller number of uncorrelated variables called principal components. The first principal component accounts for as much of the variability in the data as possible, and each succeeding component accounts for as much of the remaining variability as possible, in mutually orthogonal directions. This requires the computation of the covariance matrix of each class and then the eigen values and eigen vectors of the covariance matrix. We rearrange the eigen values in decreasing order of magnitude and pick the top 20 eigen values and their corresponding eigen vectors. These top 20 eigen values and eigen vectors are stored, along with the mean feature vector $\bar{\nu_c}$ for each class.

The test sample is reshaped to a column vector and the differences between the test vector and the mean vector of each of the classes c ($c = 1, 2....C$) are computed. For each class c, we compute the Mahalanobis distance given by the following expression.

$$d_c = \sum_{m=1}^{20} \frac{\langle \nu - \bar{\nu_c}, u_{c,m} \rangle^2}{\lambda_{c,m}} \tag{3}$$

where, 'ν' is the test vector, '$\bar{\nu_c}$' is the mean vector of class 'c', $\lambda_{c,m}$ is the m^{th} highest eigen value of class 'c', and $u_{c,m}$ is the m^{th} eigen vector of class c ($m = 1, 2....20$). The test sample is assigned the label of the class, for which the distance is minimum. The confidence level of SVM2 corresponding to the class label output by PCA is increased by a value ($2{*}T_p$) in the vector F_2. Thus, the output of classifier2 is weighted based on the output of the PCA classifier.

6 Fusion of online and offline classifications

Figure 6 gives the block schematic of the complete system, illustrating the fusion of the online and offline recognizers. The SVM1 of the online system outputs P_1(c|d) as the posterior probability of class 'c', given the data 'd' and online features. Similarly, the SVM2 of the offline recognizer outputs P_2(c|d) as the likelihood of class 'c', given the data 'd' and offline features. Whenever the likelihoods of the top two choices of the offline SVM are very close, the PCA classifier is invoked, which outputs the class label c_3. Fusion consists of combining the outputs of online and offline systems to obtain a score robust with respect to eventual

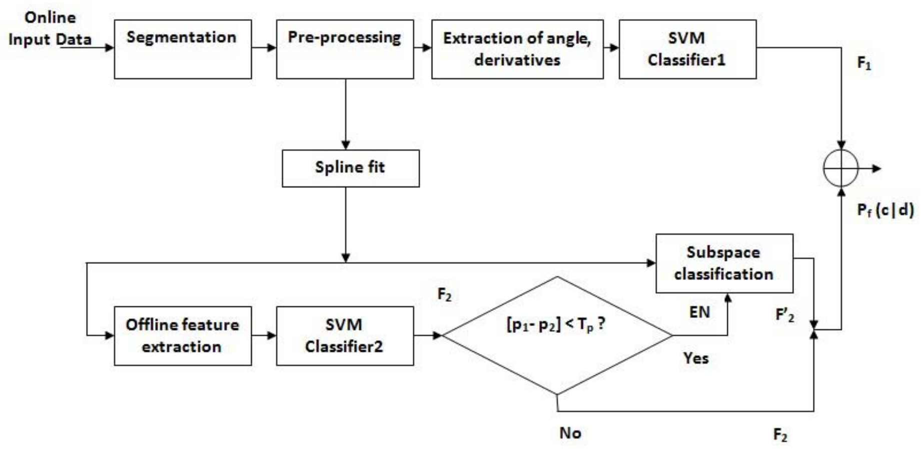


Figure 6. Block diagram of the proposed fusion of online and offline classifiers. F_1 and F_2 are the output vectors of the online and offline SVM classifiers, respectively. P_f(c|d) is the posterior probability of the test sample belonging to class c, given by the fusion of the classifiers. p_1, p_2 are the likelihoods given by offline SVM2 for the two topmost classes. T_p is the threshold that determines whether or not the offline PCA classifier is invoked.

errors of one or both the classifiers. We adopt the sum rule, wherein importance is given to both the classifiers.
The fused likelihood, P_f(c|d) $= P_1$(c|d)*w_{on} $+P_2$(c|d)*w_{off}
where w_{on} and w_{off} are the weights given to the online and offline classifiers, respectively and w_{on} $+w_{off} = 1$. Here we assume that w_{on} and w_{off} are independent of 'd', and hence, w_{on} is optimized from the data. In fusion, $P_2(c_3|\mathrm{d})$ is replaced by $P'_2(c_3|\mathrm{d})$ $=P_2(c_3|\mathrm{d})$ + 2 T_p to give weightage to the class c_3 output by PCA, whenever the latter is invoked for disambiguation. When w_{on} is varied from 0 to 1, the fusion system shifts from a purely offline one to a purely online recognizer. The final values used in the experiments are $w_{on} = 0.6$ and $w_{off} = 0.4$.

7 Results

SVM2, used as the primary offline classifier, is trained using 40 samples of each class and tested on 21 samples of the same. The numbers in Table 1 indicate the accuracies obtained using (i) online recognition alone using SVM1, (ii) only offline recognition using SVM2, (iii) offline recognition using PCA alone, and (iv)

Table 1: Performance of online, offline and fusion classifiers.

No. of classes	Online recognition: angles, derivatives & SVM1	Offline recognition: DDD, TC, NSP, PP & SVM2	Offline recognition: DDD, TC, NSP, PP & PCA	Classifier fusion
200	81.2%	77.4%	84.9%	92.3%
295	79.3%	75.2%	81.8%	89.7%

the combination of online and offline recognition systems. Experiments are performed over two sets; one set comprises all the Kannada characters (200 classes) and the other set includes all the special characters, numbers and symbols along with Kannada characters (200+95 classes).

8 Conclusions

In this paper, we have presented a recognition method for online handwritten characters by combining offline image based recognition. The ambiguous cases are resolved using a third (subspace based) classifier, which uses PCA on the features obtained from the images of the data. In the literature, some researchers have demonstrated elastic matching for handwriting recognition in which eigen deformations are estimated by the principal component analysis of actual deformations [Uchida et al., 2002] . Our future work will focus on exploiting features with more discriminating power to improve the accuracy of online recognition and using the elastic matching for images.

Acknowledgements

We thank Mr. Mohan P, Mr. Mahadeva Prasad, Mr. Venkatesh N, Mr. Thotreingam Kasar, Mr. Deepak Kumar, Mr. Suresh Sundaram and Mr. Rituraj Kunwar for their inputs which made these experiments possible. Special thanks to Technology Development for Indian Languages (TDIL), DIT, Government of India for sponsoring the data collection used in this work.

References

[Hamanaka et al., 1993] Hamanaka, M., Yammada, K., Tsukumo, J.: "Online Japanese character recognition experiments by an offline method based on normalization-cooperated feature extraction," Proc ICDAR, 1993, pages 204-207.

[Kunwar et al., 2010] Kunwar, R., Shashikiran, K., Ramakrishnan, A.: "Online handwritten Kannada word recognizer with unrestricted vocabulary," Proc. International Conf on Frontiers of Handwriting Recognition (ICFHR) 2010, pages 611-616.

[Oh et al., 1998] Oh, S., Suen, C.: "Distance features for neural network based recognition of handwritten characters," International Journal of Document Analysis and Recognition, Vol. 1, No. 2, July 1998, pages 73-88.

[Prasad et al., 2009] Prasad, M., Sukumar, M., Ramakrishnan, A.: "Divide and conquer technique in online handwritten Kannada character recognition," Proc International Workshop on Multilingual OCR, 2009.

[Toraichi et al. 1996] Toraichi, K., Kumamoto, T., Tamamoto, K., Yamada, H.: "Feature analysis of hand printed Chinese characters," Pattern Recognition Letters, Vol. 17, Issue 7, 1996, pages 795-800.

[Uchida et al., 2002] Uchida, S., Sakoe, H.: "A handwritten character recognition method based on unconstrained elastic matching and eigen-deformations," Proc. Eighth International Workshop on Frontiers of Handwriting Recognition, 2002, pages 72-77.

[Vinciarelli et al., 2003] Vinciarelli, A., Perrone, M.: "Combining online and offline handwriting recognition," Proc. ICDAR, Vol. 2, 03-06 August 2003, pages 844-848.