A Hybrid Feature Extraction Algorithm for Devanagari Script

DEEPTI KHANDUJA, NEETA NAIN, and SUBHASH PANWAR,
Department of Computer Science and Engineering, MNIT Jaipur

The efficiency of any character recognition technique is directly dependent on the accuracy of the generated feature set that could uniquely represent a character and hence correctly recognize it. This article proposes a hybrid approach combining the structural features of the character and a mathematical model of curve fitting to simulate the best features of a character. As a preprocessing step, skeletonization of the character is performed using an iterative thinning algorithm based on Raster scan of the character image. Then, a combination of structural features of the character like number of endpoints, loops, and intersection points is calculated. Further, the thinned character image is statistically zoned into partitions, and a quadratic curve-fitting model is applied on each partition forming a feature vector of the coefficients of the optimally fitted curve. This vector is combined with the spatial distribution of the foreground pixels for each zone and hence script-independent feature representation. The approach has been evaluated experimentally on Devanagari scripts. The algorithm achieves an average recognition accuracy of 93.4%.

Categories and Subject Descriptors: I.5.2 [**Pattern Recognition**]: Feature Evaluation and Selection

General Terms: Feature Extraction, Neural Network, Performance

Additional Key Words and Phrases: Thinning, zoning, raster scan, curve fitting

ACM Reference Format:
Deepti Khanduja, Neeta Nain, and Subhash Panwar. 2015. A hybrid feature extraction algorithm for Devanagari script. ACM Trans. Asian Low-Resour. Lang. Inf. Process. 15, 1, Article 2 (November 2015), 10 pages.
DOI: http://dx.doi.org/10.1145/2710018

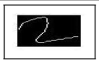
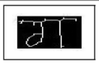
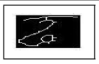
1. INTRODUCTION

In the field of pattern recognition, character recognition is regarded as one of the most interesting and challenging steps for script processing. The handwritten character recognizer (*HCR*), a tool to recognize and convert scanned text script images into digitized form, broadly consists of three major stages: preprocessing, segmentation, and recognition. *HCR* initializes with preprocessing that includes binarization, skew, and slant normalization of the text document. In further steps, the preprocessed text image is divided into lines, lines are divided into words, and finally each word is partitioned into individual characters. In the simplest case, this character is fed to the recognition algorithm; however, to avoid complexity of the recognition system, a set of features is extracted for each character in order to reduce its dimensionality, and this reduced representation of character is fed into the classifier to classify the character into its appropriate class. To make this feature set extraction task more effective and refined, a new preprocessing step applied on individual characters, named thinning, has been

Authors' addresses: D. Khanduja, N. Nain, and S. Panwar, Department of Computer Science and Engineering, Malaviya National Institute of Technology, Jaipur, India; emails: deepti.khanduja89@gmail.com, nnain.cse@mnit.ac.in, panwar.subhash@gmail.com.

DOI: http://dx.doi.org/10.1145/2710018

Table I. Result of Thinning Algorithm on Some Typical Devanagari Characters

Input Character	Skeleton	Thinned Character	Input Character	Skeleton	Thinned Character
					
					
					

introduced in this article. Since feature vector is the direct input to the recognition step, accuracy of the system depends on the effectiveness of the vector obtained. A number of approaches are introduced earlier to simulate the feature extraction, like character zone based [Rajashekararadhya and Ranjan 2009], direction based [Basli 2003], neural network based [Pradeep 2011], and transform based [Mowlaei 2002]. In most of the approaches, either structural features are estimated or spacial and statistical features are calculated. In this article, we have proposed an approach based on both structure and spacial distribution.

The article gives the details of the proposed approaches in Section 2. Section 4 defines the neural network classifier applied. Section 3 provides justification of the various heuristics used in the approach. Section 6 gives a performance analysis with existing techniques. Results and conclusions are appended in Section 5 and 7, respectively.

2. PROPOSED APPROACH

The proposed approach after preprocessing and noise cleaning [Khanduja 2013] has been prorated in four steps: thinning, structural feature extraction, statistical feature extraction, and the complete approach.

2.1. THINNING: A Step Further in Preprocessing, a Refinement

Before applying any feature extraction mechanism, as a preprocessing step, thinning is introduced. Thinning is a form of data compression technique that helps in reducing the amount of information input to the pattern recognition algorithm while maintaining the configuration of the pixels and topology of the image. Character recognition is a major research area that utilizes thinning to expedite the extraction of critical structural features of a character. Another crucial aspect of thinning is that the output should be located approximately to the medial axis of the character stroke. This step reduces chaos in the structural elements of the character and retains their generality more effectively. The proposed thinning algorithm is based on raster scan of the character image, considering one pixel at a time and detecting the subsequent pixels by raster scanning. The algorithm for thinning is summarized in Algorithm 1.

The algorithm for thinning has been extensively tested on several Devanagari script characters, with the summarized results presented in Table I.

2.2. Proposed Algorithm

Feature set generation is the method of converting highly redundant, variable, and diverse data to a small-in-size, robust, abstract, and complete set that conveys all

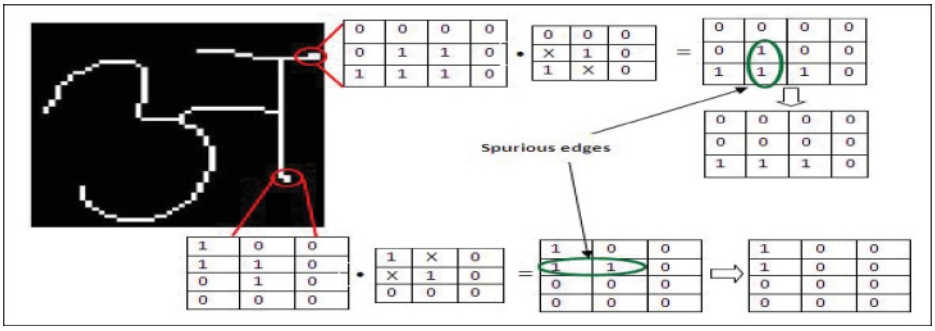


Fig. 1. Step-wise illustration of the thinning algorithm 1 on a sample image.

ALGORITHM 1: Algorithm for thinning a character image

```
Input: 2-Dimensional binary segmented character I.
Output: Thinned character image.
1 Apply closing operation on $I$ to complete the distorted images, i.e., images disconnected due
  to writer or writing tool variation. Calculate the minimum stroke width $W$ of character to be
  used as threshold of nonspurious edges.
2 Compute skeleton of the input character image using standard mechanism defined in terms
  of the hit-or-miss transform as described in Equation (1):
      $I \oslash B = I - (I \otimes B) = I \cap (I \otimes B)^c$        (1)
      $B = \begin{bmatrix} 1 & 1 & 1 \\ 1 & x & 1 \\ 1 & 1 & 1 \end{bmatrix},$
  where B is the structuring element.
3 The resulting character matrix consists of single-element edge irregularities leading to
  distortion and spurious edges. Repeat steps 4-7 until no change appears, to remove such
  edges.
4 Perform thinning operation on the resultant image using the following masks:
      $M_1 = \begin{bmatrix} 1 & X & 0 \\ X & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$  $M_2 = \begin{bmatrix} 0 & X & 1 \\ 0 & 1 & X \\ 0 & 0 & 0 \end{bmatrix}$  $M_3 = \begin{bmatrix} 0 & 0 & 0 \\ X & 1 & 0 \\ 1 & X & 0 \end{bmatrix}$  $M_4 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & X \\ 0 & X & 1 \end{bmatrix}$
  Figure 1 shows the application of masks $M_1$ and $M_3$ on a sample character image. The
  result is then filtered by removing spurious edges from the resultant image.
5 Find the endpoint pixels with 1-neighborhood depicted by $N_1$.
6 Traverse the connecting pixels in clockwise direction using 8-neighborhood until the pixel
  with 3 or more neighborhood is found depicted by $N_3$.
7 Calculate the distance $D$ between $N_1$ and $N_3$.
8 If $D < W$ remove the edge as it is a spurious edge.
```

features of the original data. Generally, data with greater size increases complexity and becomes time consuming for applications to process. Hence, mapping to a corresponding small set of data without loss of significant information makes the task easier.

For character recognition, extracting those features that are essential to differentiate among characters is a tedious task. A simple method that generates the most appropriate and complete set of features is always needed. In this article, we have tried to fabricate a simple mathematical approach in Algorithm 2 based on a combination of statistical and structural feature extraction techniques.

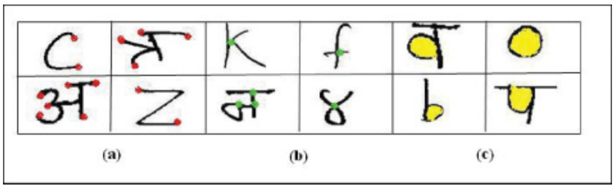


Fig. 2. (a) Endpoints. (b) Intersection points. (c) Loop.

ALGORITHM 2: Algorithm to extract feature vector

```
Input: 2-Dimensional binary segmented character.
Output: Feature vector consisting of 125 features to be fed to the neural network for
        further classification.
1 Apply thinning on the input image using Algorithm 1 to deduce a 1-pixel-wide character
  skeleton.
2 Call Procedure StructuralFeatures() to compute structural features of Input Character
  resulting in a vector of 3 features. Resize input image Img to dimensions 100x100 and
  partition Img in 5x5 block using equal partitioning, thus resulting in 25 blocks each of size
  20x20.
3 Repeat Procedure StatisticalFeatures() for each zone.
4 Hence, a complete feature vector consists of ((1*25) +(3*25)+3), i.e., 103 entries that give a
  wide range of features to differentiate among characters.
5 Finally, this feature vector can be fed as input to the neutral network for training and for
  testing as well.
```

2.3. Structural Feature Set Extraction

Structural features are used to acquire the topological information of the characters, ensuring minimal effect of shape/font/size variation on the feature set. The proposed technique captures the endpoint (EndPts), intersection point (IntersectPts), and presence of hole (HasLoop) in the character image as shown in Figures 2(a) through 2(c). EndPts are composed of the number of single neighborhood pixels, whereas IntersectPts are composed of three or more neighborhood pixels. The HasLoop feature is true if the character contains one/more closed loop or hole is/are present in the character image.

The procedure for computing structural features is illustrated in Procedure 3.

Procedure StructuralFeatures(Image).

```
1 Compute number of neighbors, $N$, in Image by using 8-connectivity mechanism.
2 For each foreground pixel (i,j) do
```

$$\text{if} N_{(i,j)} = \begin{cases} 1 & EndPts = EndPts + 1, \\ >2 & IntersectPts = IntersectPts + 1. \end{cases}$$

```
3 Set binary variable HasLoop = 1 if holes exist in the image using Flood Fill Algorithm
  [Hearn and Baker 1997].
4 return [EndPts IntersectPts HasLoop]
```

Figures 3(a) through 3(d) present the results of implementing Algorithm 2 on sample character images.

2.4. Statistical Feature Set Extraction

To derive the enhanced features from the character image that can significantly capture the trend of foreground pixel distribution, as a statistical measure in a processed image,

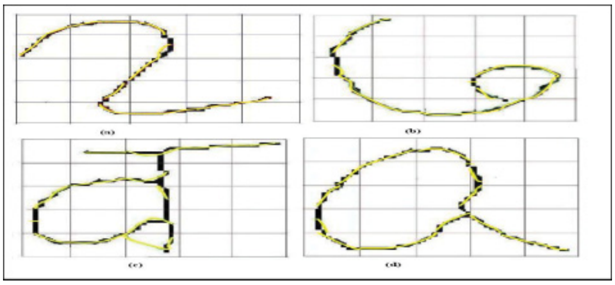


Fig. 3. Quadratic curve fitted on the character image using *MSE* approximation in Algorithm 2.

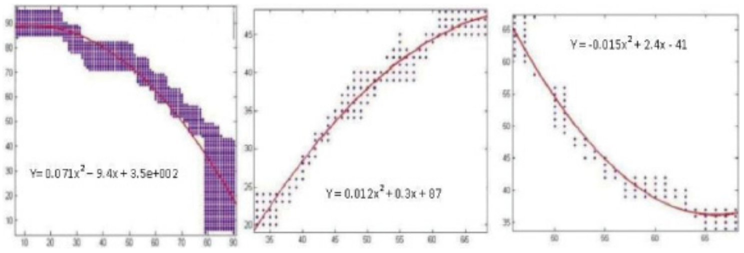


Fig. 4. Quadratic curve fitting along with the resulting polynomial equation.

Procedure StaisticalFeatures(Zone$_i$).

```
1 Find Sum$_i$, using Equation (2), the number of dark pixels for Zone$_i$. It constitutes 1 feature
  for each input zone:
      $Sum_i = Sum_i + Zone_{ij}$   for all $\mathbf{Zone_{ij} = 1}$   (2)
2 Using curve-fitting mechanism over the skeleton of the character, generate a quadratic
  polynomial QuadPolyCoeff$_i$ using Equation (3):
      $f(x) = a_0 + a_1 x + a_2 x^2.$   (3)
3 Finally, using $a_0, a_1$, and $a_2$, compute standard error(6) and iterate until MSE$_j \approx MSE_{j+1}$.
4 The polynomial generated in previous step gives 3 values of corresponding coefficients
  ($a_0, a_1$, and $a_2$), constituting 3 entries in the feature vector.
5 return [$Sum_i$ QuadPolyCoeff$_i$]
```

a two-way method is applied. First, a curve is fit on the character and the coefficients of the curve equation as features are determined; second, a count of the number of foreground pixels is used as a direct measure of distribution. To further improve the resolution of process output, we have segmented the character image into fixed small blocks of size 5×5. Each such block will derive a curve and count of foreground pixels.

Curve fitting is defined as a way to capture the trend of spatial distribution of pixels by assigning a single function across the entire range possibly subject to constraints. Each constraint can be a point, angle, or curvature. To fit a curve on the specified skeleton pixels, we postulate a function form f(x) as shown in Equation (4), a polynomial function of second order to describe the general trend in the data. Figure 4 shows the results of quadratic curve fitting with the coefficients labeled:

$$f(x) = ax^2 + bx + c. \quad (4)$$

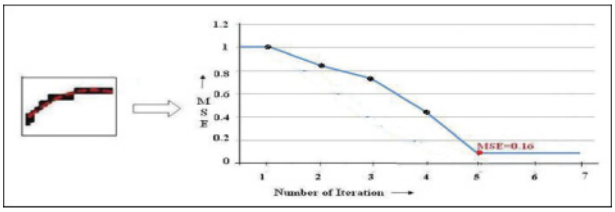


Fig. 5. Evolution of *MSE* for a sample curve.

Let $IR^n \rightarrow IR^k$ be a smooth mapping from a set of n-dimensional data points $D = p_1, \ldots, p_q$ to curve fit with the stationary points (i.e., maxima and minima) defined as $Z(f) = x : f(x) = 0$.

Let Z(f) be the set of zeros of function $f = \phi_\alpha : IR^n \rightarrow IR^k$. As α is known and also $\delta(\alpha, p_1)^2$, $\delta(\alpha, p_2)^2, \ldots . \delta(\alpha, p_q)^2$ are independent and uniformly distributed, let $\delta(\alpha, x)$ denote the distance from some x to $Z(\phi_\alpha)$. This results in χ^2 distribution as defined by Equation (5) with variance σ^2:

$$\chi^2 = \frac{1}{\sigma^2} \sum_{i=1}^{q} \delta(\alpha, p_i)^2. \tag{5}$$

Curve fitting corresponds to the minimization of Equation (5) with respect to parameters $\alpha_1, \alpha_2 \ldots . . \alpha_q$. Assuming that variance σ^2 is known, the problem is equivalent to minimizing the approximate Mean Square Error (*MSE*) in Equation (6) from the dataset D to the set of zeros of $f = \phi_\alpha$ as shown in Figure 5:

$$\Delta_D^2(\alpha) = \frac{1}{q} \sum_{i=1}^{q} \delta(\alpha, p_i)^2. \tag{6}$$

3. JUSTIFICATION OF THE HEURISTICS

All types of curves can be represented with the help of the n^{th} order polynomial of the form $a_n X^n + a_{n-1} X^{n-1} + a_{n-2} X^{n-2} \cdots + a_1 X^1 + a_0 X^0 = 0$. The basic question that arises while fitting a curve over some structure is what order (n) will be sufficient and optimally fit the structure. This depends both on the structure of the curve and on the number of data points available.

Let us understand these two aspects one by one. A close watch over the structure of Devanagari vowels, consonants, and in turn words gives a glimpse that almost all structures are of the nature of either straight lines, loops (circles), or half curves, which could be approximated using polynomials. For example, choosing an $(n-1)^{th}$ order polynomial to fit n data points forces the curve through every point. However, by doing this, it has to be decided that no random variation exists in any data point. In such a case, interpolation, extrapolation, and differentiation of the resulting polynomial are extremely unreliable, leading to oscillations and erratic behavior. To minimize this risk, it is better that a lower-order polynomial be chosen to fit the general tendency of the data. And the lowest nonlinear polynomial that can represent curves is the quadratic polynomial. Compared to linear curves, quadratic polynomials are more flexible, and compared to higher-degree polynomials, they are more stable and less complex. A regression analysis of various degrees of polynomials is illustrated in Table II. This further asserts the usage of quadratic polynomials as a feature vector. The basic property of polynomials is that they can possess one less peak and valley than their orders. Maxima or minima representing peak and valley respectively can be found by finding

Table II. Comparison of Various Feature Vector Extraction Techniques

Degree of Polynomial for Curve Fit	Size of Feature Vector	Regression Curve Value
Linear	(25+50+3)=78	.97
Quadratic	(25+75+3)=103	.98
Cubic	(25+100+3)=128	.98
Quartic	(25+125+3)=153	.96

the values of x, where the derivative df/dx is zero using Equation (8):

$$\left.\frac{\partial f}{\partial x}\right|_x = 0. \tag{7}$$

Assuming a dataset, that is, zoned skeleton data points $(X, Y) = (x_1, y_1), ((x_2, y_2) \cdots (x_N, y_N)$, the goal is to optimize a polynomial function $f(x)$ that will correctly fit the data points using some measure of performance. A measure of performance that we have used is the empirical risk [Perez-Cruz et al. 2003]:

$$R_{emp}(f]) = \frac{1}{N}\sum_{i=1}^{N} C(f(x_i), y_i), \tag{8}$$

where $C(f(x), y)$ is the cost residual function $f(x) - y$.

It can be said the we need to find the function $f(x)$ that minimizes the average risk in terms of fitting the curve on the dataset. On the contrary, the law of large numbers ensures that the empirical risk asymptotically converges to the expected risk for $f(x) - > \infty$, and for small samples, it cannot always be guaranteed that Empirical Risk Minimization (*ERM*) will also minimize the expected risk. Thus, we combine the approach of *VC* dimension [Hush and Scovel 2001] to deduce the class of functions $f(\alpha)$ that shatters the dataset; that is, for every possible dichotomy, there is a function in $f(\alpha)$ that models it. The *VC* dimension $VC(f)$ is the size of the largest dataset that can be shattered by the set of functions $f(\alpha)$. If the *VC* dimension of $f(\alpha)$ is $\hbar$, then there exists at least one set of $\hbar$ points that can be shattered by $f(\alpha)$. *VC* dimensions provide a bound on the expected risk as a function of the empirical risk.

For our experiments, the expected risk, that is estimated error on future data trend, is computed by using the empirical risk and *VC* confidence as shown in Equation (9):

$$R(f) = R_{emp}(f) \leq \sqrt{\frac{h(ln(\frac{2N}{h}) + 1) - ln(\frac{n}{4})}{N}}, \tag{9}$$

where h is the *VC* dimension of $f(\alpha)$ and N is the number of data points; also, $N > h$. As the ratio N/h increases, the *VC* confidence decreases and the actual risk gets closer to empirical risk. A function $f(\alpha)$ with h points is said to generalize the set of data points if $R(f(\alpha))$ is approximately equal to 0.

In the proposed approach, experiments have been done by zoning the character into 4×4, 5×5, and 6×6 zones, resulting in the average skeleton points with 8, 10, and 12 pixels per zone, respectively. On computing the expected risk in each case, the value of $R(f)$ is the resulting minimum for the 5×5 zone. For a random case, for the optimal quadratic curve $y = -17.556x^2 + 31.930x + 0.003$, the corresponding minimal empirical loss is $Remp = 6.91$ and expected error $R(f) = 0.78$. Thus, in the proposed technique, the character image of 50×50 pixels is partitioned into 5×5 horizontal and vertical zones.

The reason to implement the hybrid feature extraction approach has been proved by measuring the performance of statistical, structural, and proposed techniques

Table III. Justification of Hybrid Feature Extraction Approach

Method of Feature Extraction	Accuracy(in %)
Statistical	.74
Structural	.47
Proposed approach	.98

Table IV. Performance Evaluation of Proposed Feature Set Extraction and Classification Approach

Database	Number of Images	TP	TN	FP	FN	Accuracy (in %)	Error Rate (in %)
Hindi character [Dongre 2012]	4,000	744	2778	367	111	91.4	8.6
Hindi numeral [Bhattacharya and Chaudhuri 2009]	2,000	431	1393	135	41	95.5	4.5

individually in Table III. The statistical feature extraction is based on discriminating the data using quantitative features of data, which hinders the detection of morphological patterns in the character. Furthermore, extracting features based on the statistical distribution of pixels in a character is inadequate to diagnose distortions and style variations.

4. NEURAL NETWORK CLASSIFIER

The classification stage uses the feature vector generated in the feature extraction stage to identify the text image. Classification is accomplished by a two-layer feed-forward neural network with back-propagation learning. The topology of the classifier consists of input and output layers with neurons determined by the length of feature vector and number of classes, respectively. The number of neurons in the first and second hidden layer are 70 and 40, respectively. The network employs the gradient descendent rule in an attempt to minimize the squared error between the network output values and the target values for these outputs. The trained network is used to test an input character image entered by the user and the classifier maps any input pattern to a number of classifications. The probability density function (pdf) of each of the classes is generated, and then an unknown image, X, belongs to class i if $f_i(X) > f_j(X)$, $\forall j \neq i$, where f_k is the pdf for class k.

5. EXPERIMENTAL RESULTS

The proposed feature set generation technique has been trained and tested on 22,556 standard dataset samples. Tenfold cross-validation is used for testing results using 75% of the data as training and 25% for validation. The samples consist of handwritten Hindi numerals, the character open-source database *"Devanagari numeral and character database for offline handwritten character recognition"* [Dongre 2012], and a database obtained by the CVPR Unit, ISI Kolkata [Bhattacharya and Chaudhuri 2009]. The results for the various testing parameters have been summarized in Table IV. Equations (10) and (11) present the accuracy and error measure of the proposed method, which achieves an average accuracy of 93.4% and a false ratio of 6.6%:

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{10}$$

$$Error\ ratio = \frac{FP + FN}{TP + TN + FP + FN}, \tag{11}$$

Table V. Comparative Performance Analysis with Existing Classifiers for Devanagari Numerals

Author	Features	Classifier	Accuracy
Bhattacharya and Bhattacharya [2006]	Shape features	HMM+ANN	95.64%
Bajaj et al. [2002]	Density, moment, and descriptive	Multiclassifier connectionist	89.68%
Hanmandlu [2007]	Vector distance	Fuzzy model classifier	90%
Bhattacharya and Chaudhuri [2009]	Wavelet based	MLP	99.06%
Proposed approach	Hybrid features	MLP with 70 and 40 neurons	95.5%

Table VI. Comparative Performance Analysis with Existing Classifiers for Devanagari Characters

Author	Features	Classifier	Validation (in %)
Umapada Pal et al. [2008]	Gradient features using Gaussian filter	Quadratic classifier	94%
Hanmandlu [2007]	Vector distance	Fuzzy model	90%
Sharma [2006]	Chain code	Quadratic classifier	80
Proposed approach	Local and global features	MLP	91.4%

where TP = true positive, TN = true negative, FP = false positive, and FN = false negative.

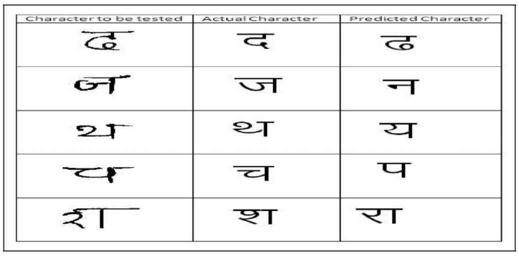
6. PERFORMANCE ANALYSIS

The proposed feature extraction technique has been compared with existing techniques with their proposed feature vector and parameter settings on the same datasets. The database used for comparison is obtained by ISI Kolkata [Bhattacharya and Chaudhuri 2009], of Devanagari numerals. The comparative performance details are presented in Table V. It is observed that the proposed approach by using a simple *MLP* classifier gives approximately the same results as obtained by using the complex classifier combination of *HMM* and *ANN*.

For comparative analysis of Devanagari characters, most of the approaches [Umapada Pal et al. 2008; Hanmandlu 2007; Sharma 2006] have used their own datasets, which are not publicly available for comparison. For comparative analysis though, we illustrate their claimed and published results with our proposed configuration in Table VI.

7. CONCLUSIONS

This article proposes a model of an efficient feature set extraction technique using statistical and structural features of the text image for script-independent character recognition (it has been tested on English and Urdu also). Results generated by the experiments on the testbed images conform to the greater level of efficiency with respect to the size of the feature set, time utilized to train the network, and its complexity. The approach of Quadratic Polynomial Curve Fitting used for generating a feature set has been experimentally derived by testing various degrees of polynomials. For n data points, the complexity of the algorithm is $O(kn)$, where k is the number of iterations. As the image is divided into 25 zones, the complexity achieves $O(25*kn) \approx O(kn)$. As a measure of statistical comparison, the proposed approach gives accuracy of 93% in an established testbed environment These results uphold the application of the approach to generate feature vectors. Experimental results prove that the algorithm shows incorrect results for characters with similar structural appearance or character images resembling other characters due to writer variations. Some sample characters



Character to be tested	Actual Character	Predicted Character
द	द	ढ
ज	ज	न
थ	थ	य
च	च	प
श	श	रा

Fig. 6. Sample character images providing false results.

providing false results are shown in Figure 6; however, most of these characters are difficult to be recognized even by humans.

REFERENCES

R. Bajaj, L. Dey, and S. Chaudhury. 2002. Devnagari numeral recognition by combining decision of multiple connectionist classifiers. *Sadhana* 27, 1 (2002), 59–72.

U. Bhattacharya and B. B. Chaudhuri. 2009. Handwritten numeral databases of Indian scripts and multi-stage recognition of mixed numerals. *IEEE Trans. Pattern Anal. Mach. Intell.* 31, 3 (2009), 444–457.

M. Blumenstein, B. Verma, and H. Basli. 2003. A novel feature extraction technique for the recognition of segmented handwritten characters. In *Proceedings of the 7th International Conference on Document Analysis and Recognition (ICDAR'03)*. 137–141.

K. Haghighat, A. T. Mowlaei, and A. Faez. 2002. Feature extraction with wavelet transform for recognition of isolated handwritten Farsi/Arabic characters and numerals. In *Proceedings of the 14th International Conference on Digital Signal Processing*, Vol. 2. 923–926.

D. Hearn and P. Baker. 1997. *Computer Graphics: C Version*. Prentice Hall International.

D. Hush and C. Scovel. 2001. On the VC dimension of bounded margin classifiers. *Mach. Learn.* 45, 1 (Oct. 2001), 33–44.

F. Kimura, N. Sharma, and U. Pal. 2006. Recognition of off-line handwritten Devnagari characters using quadratic classifier. In *Comp. Vision, Graphics & Image Proc.* Springer, Berlin. 805–816.

V. H. Mankar and V. J. Dongre. 2012. Development of comprehensive Devanagari numeral and character database for offline handwritten character recognition. *J. Appl. Comput. Intell. Soft Comput. (ACISC)* 2012. Article ID 871834. 1–5.

N. Nain and D. Khanduja. 2013. Segmentation and recognition techniques for handwritten Devanagari script. In *Proceedings of the MUSCLE International Workshop on Computational Intelligence for Multimedia Understanding*. 105–111.

F. Perez-Cruz, A. Navia-Vazquez, A. R. Figueiras-Vidal, and A. Artes-Rodriguez. 2003. Empirical risk minimization for support vector classifiers. *Trans. Neur. Netw.* 14, 2 (March 2003), 296–303.

E. Himavathi Pradeep and J. Srinivasan. 2011. Neural network based handwritten character recognition system without feature extraction. In *Proceedings of the International Conference on Computer, Communication and Electrical Technology (ICCCET'11)*. 40–44.

S. V. Rajashekararadhya and P. V. Ranjan. 2009. Zone based feature extraction algorithm for handwritten numeral recognition of Kannada script. In *Proceedings of the IEEE International Advance Computing Conference (IACC'09)*. 525–528.

O. V. Ramana, M. V. K. Hanmandlu, and M. Murthy. 2007. Fuzzy model based recognition of handwritten hindi characters. In *Proceedings of the 9th Biennial Conference of the Australian Pattern Recognition Society on Digital Image Computing Techniques and Applications*. 454–461.

B. Shaw U. Bhattacharya, S. K. Parui, and K. Bhattacharya. 2006. Neural combination of ANN and HMM for handwritten Devanagari numeral recognition. In *Proceedings of the 10th IWFHR*. 613–618.

T. W. Umapada Pal, S. Ch, and F. Kimura. 2008. Accuracy improvement of Devnagari character recognition combining SVM and MQDF. In *Proceedings of the 11th International Conference Frontiers Handwritten Recognition SVM and MQDF*. 367–372.

Received March 2014; revised December 2014; accepted December 2014