Neural Comput & Applic (2015) 26:1881–1893
DOI 10.1007/s00521-015-1844-9

ORIGINAL ARTICLE

An efficient segmentation technique for Devanagari offline handwritten scripts using the Feedforward Neural Network

Vijay Pal Dhaka · Manoj Kumar Sharma

Received: 2 June 2014 / Accepted: 11 February 2015 / Published online: 26 February 2015

Abstract In this research work, we have proposed a segmentation technique for words and characters of Devanagari offline handwritten scripts. Due to complex structures and high unevenness in writing styles, recognition of words and characters from the unconstrained scripts has become a burning vicinity of interest for researchers. The proposed Pixel Plot and Trace and Re-plot and Retrace (PPTRPRT) technique extracts text region from Devanagari offline handwritten scripts and lead iterative processes for segmentation of text lines along with skew and de-skew operations. The outcomes of iterations are used in pixel-space-based word segmentation, and the segmented words are used in segmentation of characters. Moreover, PPTRPRT perform various normalization steps to allow deviation in pen breadth and slant in inscription. Investigational outcome shows that the proposed technique is competent to segment characters from Devanagari offline handwritten scripts, and accuracy of outcomes is up to 99.578 %.

Keywords PPTRPRT · Trace · Devanagari · Offline · Handwritten · Segmentation

1 Introduction

The research on auto-recognition of constrained and unconstrained Devanagari scripts started in early 1970s, and it has complex composition of its constituent symbols (13 vowels and 34 consonants along with 14 modifiers of vowels, see Fig. 1) Jayadevan et al. [1].

Devanagari script has its own specified composition rules for combining vowels, consonants and modifiers. Some of them can be combined with their type (see Fig. 2) Jayadevan et al. [1].

Another distinctive feature of the Devanagari script is the presence of a horizontal header line on the top of all characters (see Fig. 3).

The occurrence frequency of different Devanagari characters is provided in the occurrence statistics of 20 frequent characters in Devanagari script Jayadevan et al. [1] (see Table 1).

In the present scenario, due to the complex structures and high unevenness in writing styles of the scripting languages, recognition of words and characters from the Devanagari offline handwritten script has become a burning vicinity of interest for researchers. Many preprocessing steps are performed in automatic Devanagari offline handwritten recognition systems, which give a moderate deviation in the handwritten script and conserved information relevant to acknowledge.

The present research article is arranged, in the following sections: Sect. 2 gives a brief introduction to related works, Sect. 3 defines mathematical notations to be used in description algorithms, Sect. 4 discusses the methodologies of Pixel Plot and Trace and Re-plot and Re-trace (PPTRPRT) technique, Sects. 5, 6 and 7 deal with the segmentation of words and characters.

2 Related work

Numerous approaches have been proposed so far for improving segmentation of Devanagari offline handwritten

V. P. Dhaka · M. K. Sharma (✉)
Jaipur National University, Jagatpura, Jaipur 302017, Rajasthan, India
e-mail: manoj186@yahoo.co.in

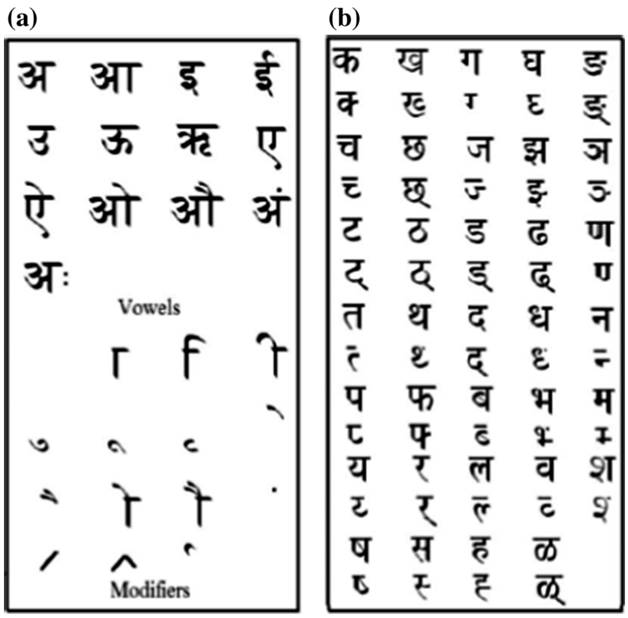

Fig. 1 **a** Vowels and modifiers of Devanagari script. **b** Consonants and their corresponding half form in Devanagari script

Fig. 2 Combination of consonants

Fig. 3 Three scripts of a word in the Devanagari script

scripts. Jayadevan et al. [1] have explored various feature extraction techniques as well as training, classification and matching techniques. Sahu et al. [2] used artificial neural network technique for preprocessing operations and designed a system for segmentation as well as recognition of Devanagari characters. Yadav et al. [3] proposed a hybrid system by combining both writer recognition and handwriting recognition for the Devanagari script. Wshah et al. [4] have proposed a script independent line-based word spotting framework based on hidden Markov models for offline handwritten documents and deal with large vocabulary without the need for word or character segmentation. Wshah et al. [5] proposed a multilingual word spotting framework based on hidden Markov models that work on corpus of multilingual handwritten documents and proposed two systems: a script identifier-based (IDB) and a script identifier-free (IDF) system. Deshpande et al. [6] proposed that the segmentation evolved regular expressions without doing preprocessing as well as training and transformed notation of regular expression directly into directed graphs and provide all the symbol strings generated by the corresponding expressions to finite-state automata. Rahul et al. [7] have proposed a Devanagari handwritten script recognition by applying a feature vector to an artificial neural network. Pal et al. [8] proposed a lexicon-driven method for multilingual (English, Hindi and Bangla) city name recognition for Indian postal automation. Doiphode et al. [9] proposed a technique which finds joint points in the word, identifies vertical and horizontal lines and finally dissects touching characters by taking into account its dimensions, namely height and width of the bounding box. Sankaran et al. [10] have proposed a formulation to minimize expectations on Unicode generation, error correction, and the harder recognition task is modeled as learning of an appropriate sequence to sequence translation scheme. Kumar et al. [11] have presented a survey on OCR of most popular Indian scripts. Wshah et al. [12] proposed a statistical script independent line-based word spotting framework for offline handwritten documents based on hidden Markov models by comparing an exhaustive study of filler models and background models for better

Table 1 Occurrence statistics of 20 frequent characters in Devanagari script

Symbol	Occurrence (%)	Symbol	Occurrence (%)
ा	10.126	ं	3.754
क	7.949	म	3.360
र	7.409	ेs	2.911
्	6.532	य	2.771
न	5.064	प	2.473
ो	4.478	ल	2.456
ह	4.396	व	2.268
स	4.368	द्	1.720
त	4.324	ज	1.462
ि	4.224	ग	1.334

representation of background or non-keyword text. Basu et al. [13] proposed a work to recognize the postal codes written in any of the four popular scripts, viz. Latin, Devanagari, Bangla and Urdu. Hassan et al. [14] proposed a framework for the application of multiple features for handwritten data-based identity recognition. They have designed a scheme for multiple feature-based identity establishment using multi-kernel learning using genetic algorithm.

3 Mathematical terms

The use of the standard symbols is highly recommended (see Table 2).

4 Methodology

This section briefly discusses descriptive algorithms of PPTRPRT technique for Devanagari offline handwritten script segmentation.

4.1 Overview of system design

Descriptive architecture of the PPTRPRT technique is given in Fig. 4.

The pattern sets used in the current study are Devanagari offline handwritten script. For desired outcomes, we are using Feedforward Neural Network.

For the evaluation of PPTRPRT technique, a gigantic database of 49,000 samples (i.e., Center for Microprocessor Application for Training Education and Research (CMATER), ICDAR-2005, Off-line Handwritten Devanagari Numeral Database, WCACOM ET0405A-U, HP Labs India Indic Handwriting Datasets) is composed for training pattern.

The PPTRPRT technique extracts text region from Devanagari offline handwritten script images and passes it to iterative processes for segmentation of text lines. The segmented text lines are used in skew and de-skew operations. These skewed and de-skewed images are provided for white pixel-based word segmentation, and these segmented words are used in an iterative process for segmentation of characters. The PPTRPRT technique embraces various dispensations to segment characters from Devanagari offline handwritten script. The normalization steps permit deviations in pen breadth and slant in inscription.

The PPTRPRT technique initiates the segmentation algorithm.

Table 2 Mathematical symbols

Symbols	Description
ε_i	Offline Handwritten Script (i.e., $i = 1,\ldots,n$)
ρ	Noise in ε_i
β_i	Extracted text segment (i.e., $i = 1,\ldots,n$)
Γ_i	Peak factor of i'th text segment
Ξ_i	Threshold value of i'th text segment
Δ_i	Black & white image of i'th text segment
Π_i	Histogram of i'th text segment
L	Image size of i'th text segment
$Max\Gamma_i$	Upper peak i'th text line
$Min\Gamma_i$	Lower peak i'th text line
P_j	Segmentation points of i'th text line
S_i	Segmented i'th text line
$\exists_k$	k'th white pixels in i'th text line (for $k = 1,\ldots,n$)
B_k	k'th black pixels in i'th text line (for $k = 1,\ldots,n$)
low_bk	Lower pixels of i'th text line
F_i	Filter Function of i'th text line
θ_i	Slope angle of i'th text line
Λ_k	Black Corner of i'th text line ($k = 1,\ldots,n$)
S_i^1	Skewed i'th text line with black corners
S_i^2	Skewed i'th text line after removing black corners
ϕ_i	Slant angle of S_i^2
ω_i	Regional boundaries of S_i^2
ξ_i	Height of S_i^2
η_i	Width of S_i^2
ζ_i	Depth of S_i^2
A_i	New text line matrix for S_i^2
R_i	Rotation of $A_i(S_i^2)$
T_i	Translation of $A_i(S_i^2)$
U_i	Scaling of $A_i(S_i^2)$
S_i^3	Final Skewed text line
Q_{ij}	Segmentation points of word from i'th text line (j = 1,…, n)
λ_{ij}	White spaces of i'th text line (j = 1,…, n)
$\aleph_i$	Cluster function for i'th text line
Ω_{ij}	Segmented word of i'th text line (j = 1,…, n)
N_k	Number of plot pixels from $B_k(\Omega_{ij})$
P_{jk}	Pixel plot function for j'th word ($k = 1,\ldots,n$)
Θ_{ij}	Strength of connection weight in Feedforward Neural Network
S_j	State for unit j in Feedforward Neural Network
Φ_i	Threshold for unit i in Feedforward Neural Network
ψ_{jk}	Outcome of Feedforward Neural Network

Algorithm 1: Segmentation ()

Step 1: Load Devanagari offline handwritten script file ε_i

Step 2: Call Preprocessing (ε_i)

The segmentation algorithm upload Devanagari offline handwritten script ε_i and outcome of an algorithm is shown in Fig. 5.

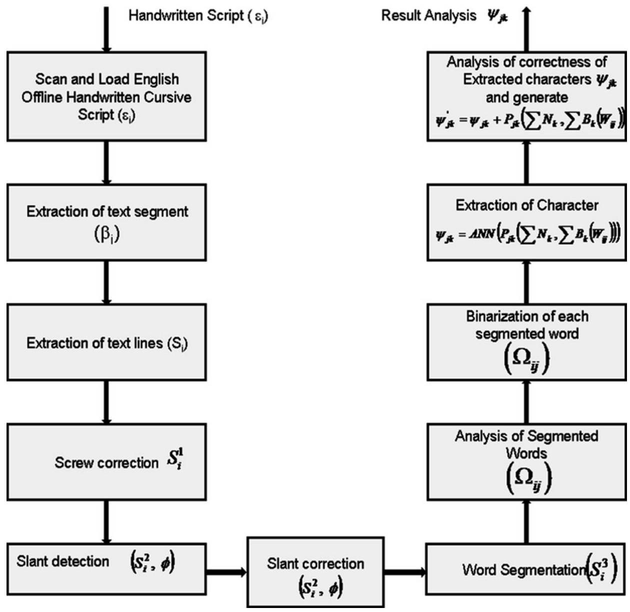

Fig. 4 Architecture of PPTRPRT technique

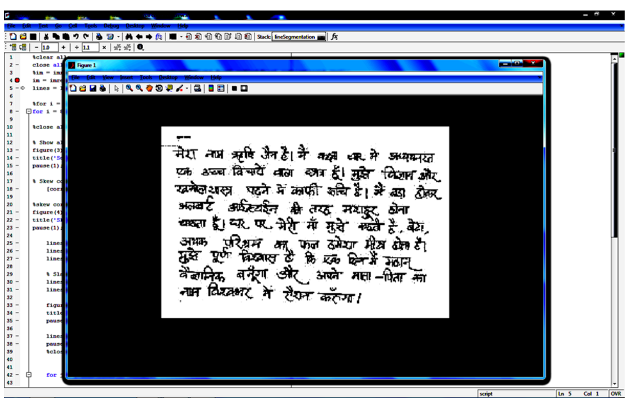

Fig. 5 Original text image

4.2 Preprocessing for characters segmentation

The outcomes of the segmentation algorithm are availed by preprocessing algorithm for forthcoming operations.

Algorithm 2: Preprocessing (ε_i)

Step 1: Extract text segment

$$\beta_i = \varepsilon_i - \rho$$

Step 2: Segment text lines from β_i

(a) Define PEAK_FACTOR (Γ_i) = 5 and THRESHOLD (Ξ_i) = 1

(b) Obtain black and white image (Δ_i)

(c) Generate Histogram (Π_i)

(d) Calculate upper peak ($Max\Gamma_i$) and lower peak ($Min\Gamma_i$)

(e) Calculate text line segmentation points (P_j)

(f) Segment text line (S_i) and drop white parts ($\exists_k$)

Step 3: Call SkewCorrection (S_i)

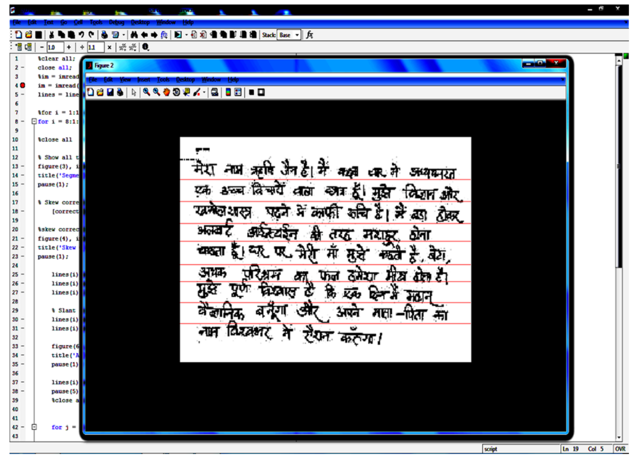

Fig. 6 Horizontal cuts of text image

The above-defined algorithm extracts the text segment β_i from ε_i by removing noises ρ. This text segment β_i is applied for text line segmentation by calculating $Max\Gamma_i$ and $Min\Gamma_i$ with PEAK_FACTOR Γ_i along with various adequate operations (see Fig. 6).

Furthermore, using $Max\Gamma_i$, $Min\Gamma_i$ and the size of an image L, we calculate text line segmentation points (P_j) and segment text line (S_i) from the text segment β_i by eliminating white parts ($\exists_k$) along with THRESHOLD value (see Fig. 7).

4.3 Skew correction

Outcomes of preprocessing algorithm are availed by skew correction algorithm for skew correction operations.

Algorithm 3: SkewCorrection (S_i)

Step 1: Improve intensity of the black pixels (Bk)

Step 2: Find (x, y) coordinates of the lower pixels (low_bk)

Step 3: Filter irrelevant pixels $F_i(low\_bk)$

Step 4: Calculate slope angle ($\theta_i(S_i)$) of the text line using linear regression

Step 5: Skew or De-skew text line(S_i) with slope angle ($\theta_i(S_i)$)

$$\left(S_i^2 = Screw\,\theta_i \ or\ De - Screw\,\theta_i\right)$$

Step 6: Remove black corners from skewed text line

$$S_i^2 = S_i^1 - \Lambda_k$$

Step 7: Call SlantDetection (S_i^2, θ_i)

An above-defined algorithm calculates (x, y) coordinate of the lower pixels and filters irrelevant pixels F_i (low_bk) from segmented text line (S_i). The slope angle ($\theta_i(S_i)$) of the segmented text line (S_i) is calculated using linear regression [$\theta_i(S_i)$] and removed black corners with skewed operation $S_i^2 = S_i^1 - \Lambda_k$. The outcomes of an algorithm are shown in Figs. 8 and 9.

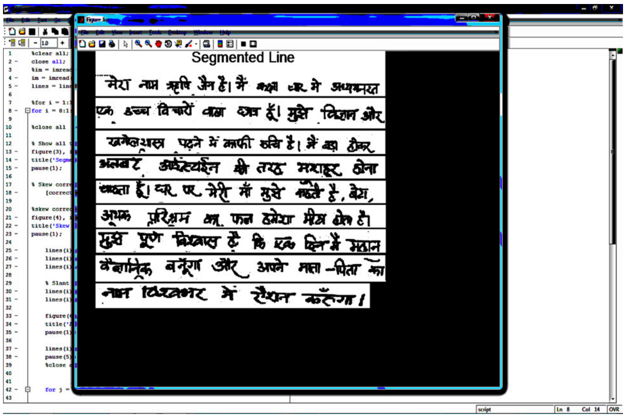

Fig. 7 Segmented text lines

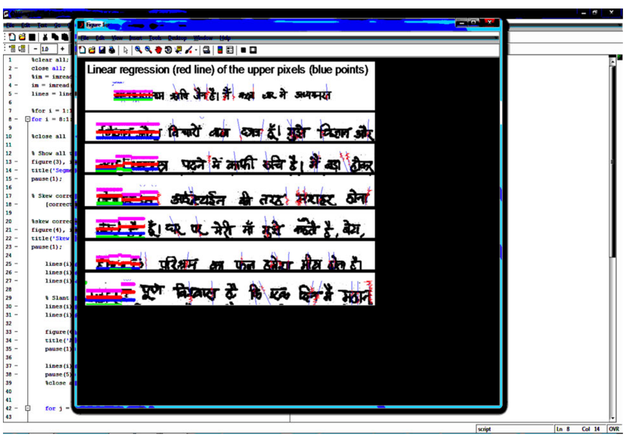

Fig. 8 Linear regression of the text lines

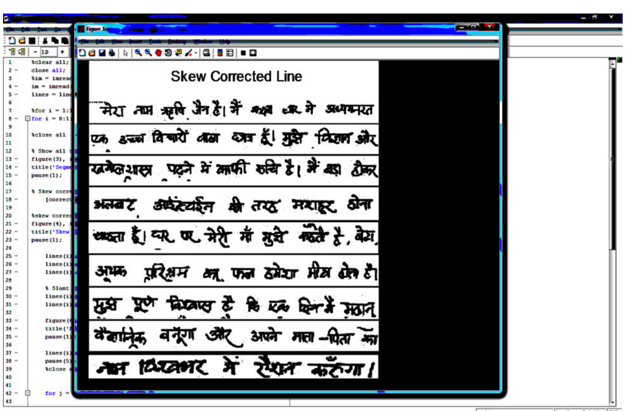

Fig. 9 Skew corrected text lines

4.4 Slant detection and correction

Outcomes of skew correction algorithm (skewed text line S_i^2 along with angle θ_i) are availed by slant detection algorithm for slant detection operation.

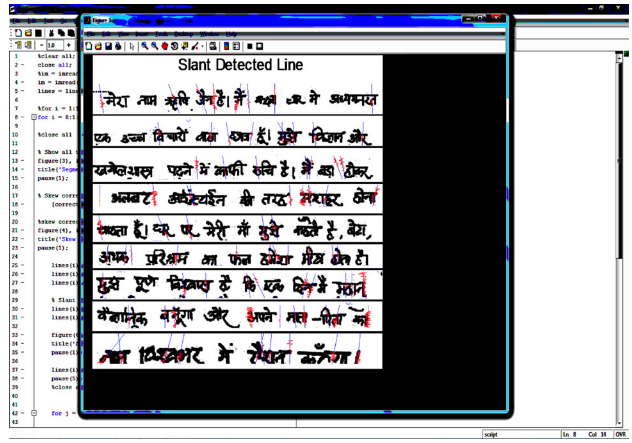

Fig. 10 Slant detection of the text lines

Algorithm 4: SlantDetection (S_i^2, θ_i)

Step 1: Trace regional boundaries of an image $\omega_i(S_i^2)$

Step 2: Define min_strock_length and step_size

Step 3: Do Histogram count operations (Π_i)

Step 4: Find indices and value of nonzero elements

Step 5: Calculate slant angle ϕ_i

Step 6: Call SlantCorrection (S_i^2, ϕ_i)

The slant detection algorithm trace regional boundaries of an image $\omega_i(S_i^2)$ by defining min_strock_length and step_size. Moreover, it traces count operations as well as the indices of the text line S_i^2 using histogram Π_i and finds nonzero elements. A slant angle ϕ_i is to be calculated using indices and value of nonzero elements. The outcomes of an algorithm are shown in Fig. 10.

Besides, slant angle ϕ_i and the text line S_i^2 are used for slant correction operations.

Algorithm 5: SlantCorrection (S_i^2, ϕ_i)

Step 1: Calculate Height ($\xi_i(S_i^2)$), Width ($\eta_i(S_i^2)$) and Depth ($\zeta_i(S_i^2)$) of the text line (S_i^2)

Step 2: Define new text line matrix $A_i(S_i^2)$

Step 3: Create spatial transformation structure of the text line

$R_i(A_i(S_i^2))$

$T_i(A_i(S_i^2))$

$U_i(A_i(S_i^2))$

Step 4: Segment final skewed text line

$S_i^3 = (R_i - T_i + U_i)$

Step 5: Call WordSegmentation (S_i^3)

The above-defined algorithm calculates height [$\xi_i(S_i^2)$], width [$\eta_i(S_i^2)$] and depth [$\zeta_i(S_i^2)$] of the text line(S_i^2). These parameters formulate a new text line matrix $A_i(S_i^2)$ and spatial transformation structures $R_i(A_i(S_i^2))$, $T_i(A_i(S_i^2))$,

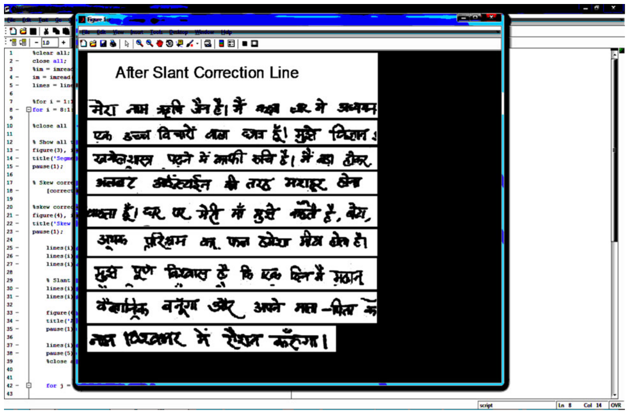

Fig. 11 Slant correction of the text line

$U_i(A_i(S_i^2))$. These spatial transformation structures are used to segment skewed text line $S_i^3 = (R_i - T_i + U_i)$ which is used in word segmentation. The outcomes of an algorithm are shown in Fig. 11.

5 Word segmentation

Moreover, the outcomes of slant correction algorithm are availed by word segmentation algorithm.

Algorithm 6: WordSegmentation (S_i^3)

Step 1: Generate Histogram Π_i

Step 2: Find Segmentation Points of Word Q_{ij}

(a) Find White Spaces $\lambda_{ij} = \sum \exists_k$

(b) Cluster Text Line to distinguish between white spaces

$\Omega_{ij} = \aleph_i(S_i^3, \lambda_{ij})$

Step 3: Call CharactersSegmentation (Ω_{ij})

Above-defined algorithm calculates histogram $\Pi_i$and white space $\lambda_{ij} = \Sigma\exists_k$ which are used to trace words segmentation pointsQ_{ij}. The outcomes of an algorithm are shown in Figs. 12 and 13.

The Character Segmentation Algorithm uses segmented words (Ω_{ij}) for the segmentation as well as extraction of characters using a Feedforward Neural Network model.

6 Feedforward Neural Network

In a multilayer Feedforward Neural Network, output feeds forward from one layer of neurons to the next layer of neurons. A multilayer Feedforward Neural Network can represent nonlinear functions and consists of one input layer, one or more hidden layers besides one output layer.

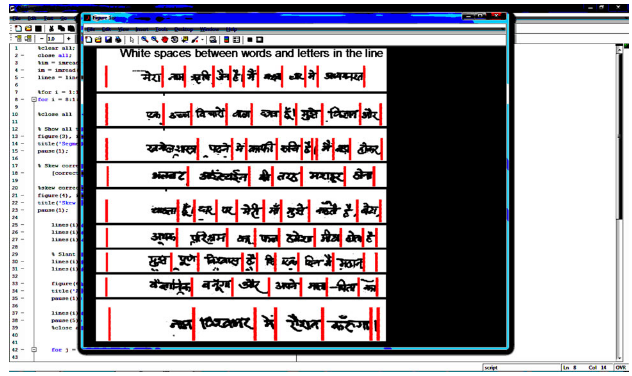

Fig. 12 White spaces between words

Fig. 13 Segmented words from the text line

Each layer has associated weights, (w0) and (z0), which feeded into the hidden layer and into the output layer respectively. Therefore, neither any backward connection nor skipping connection exists between layers. These weights will adjust while training the network using backpropagation algorithm. Typically, all input units are connected to hidden layer units, and hidden layer units are connected to the output units.

A. *Input units*

Data is feeded through Input units into the system without any processing, the value of an input unit is xj, where j goes through 1 to d input units along with a special input unit x0 contains a constant value 1 and provides bias to the hidden nodes.

B. *Hidden units*

Each hidden node calculates the weighted sum of its inputs and determines the output of the hidden node with a threshold function. The weighted sum of the inputs for hidden node z_h is calculated as:

$$\sum_{j=0}^{d} w_{hj} x_j \tag{1}$$

Table 3 Neural network specifications

Architecture	numInputs: 1
	numLayers: 4
	biasConnect: [1; 1; 1; 1]
	inputConnect: [1; 0; 0; 0]
	layerConnect: [4 × 4 boolean]
	outputConnect: [0 0 0 1]
	numOutputs: 1 (read-only)
	numInputDelays: 0 (read-only)
	numLayerDelays: 0 (read-only)
Sub-object structures	inputs: {1x1 cell} of inputs
	layers: {4x1 cell} of layers
	outputs: {1x4 cell} containing 1 output
	biases: {4x1 cell} containing 4 biases
	inputWeights: {4x1 cell} containing 1 input weight
	layerWeights: {4x4 cell} containing 3 layer weights
Functions	adaptFcn: 'trains'
	divideFcn: 'dividerand'
	gradientFcn: 'calcgrad'
	initFcn: 'initlay'
	performFcn: 'mse'
	plotFcns:
	{'plotperform','plottrainstate','plotregression'}
	trainFcn: 'traingdx'
Parameters	adaptParam:.passes
	divideParam:.trainRatio,.valRatio,.testRatio
	gradientParam: (none)
	initParam: (none)
	performParam: (none)
	trainParam:.show,.showWindow,.showCommandLine,.epochs,.time,.goal,.max_fail,.lr,.lr_inc,.lr_dec,.max_perf_inc,.mc,.min_grad
Weight and bias values	IW: {4x1 cell} containing 1 input weight matrix
	LW: {4x4 cell} containing 3 layer weight matrices
	b: {4x1 cell} containing 4 bias vectors
Other	name: ''
	userdata: (user information)

The threshold function applied at the hidden node is typically either a step function or a sigmoid function.

$$\text{sigmoid}(a) = \frac{1}{1+e^{-a}} \tag{2}$$

The sigmoid function is a squashing function, it squashes input between 0 and 1. It applies to the hidden node for the weighted sum of inputs and generates output z_h

$$z_h = \text{sigmoid}\left(\sum_{j=0}^{d} w_{hj}x_j\right) = \frac{1}{1+e^{-\sum_{j=0}^{d} w_{hj}x_j}} \tag{3}$$

for h going from 1 to H total number of hidden nodes.

C. *Output units*

Computation of the output node is either based on the type of problem (i.e., either a regression problem or a classification problem) or on number of output. The weights going into the output unit are v_{ih} and have the bias input from hidden unit z_0, where the input from z_0 is always 1. So, output unit i computes the weighted sum of its inputs as:

$$o_i = \sum_{h=0}^{H} v_{ih}\, z_h \tag{4}$$

In case of one output unit, weighted sum is

$$o = \sum_{h=0}^{H} v_h\, z_h \tag{5}$$

D. *Functions*

(1) *Regression for single and multiple outputs*

A regression function for the single output calculates the output unit value 'y' by weighted sum of its inputs:

$$y = o = \sum_{h=0}^{H} v_h\, z_h \tag{6}$$

For multiple outputs, we calculate the output value of unit y_i

$$y_i = o_i = \sum_{h=0}^{H} v_{ih}\, z_h \tag{7}$$

(2) *Classification for 2 classes*

One node can produce either 0 or 1, it can have one class correspond to 0 or 1, respectively, and generate output between 0 and 1.

$$y = \text{sigmoid}(o) = \text{sigmoid}\left(\sum_{h=0}^{H} v_h\, z_h\right) = \frac{1}{1+e^{-\sum_{h=0}^{H} v_h z_h}} \tag{8}$$

E. *Back-propagation Algorithm*

The Back-Propagation Algorithm is used to train Feedforward Neural Network using Gradient Descent method to update the weights in order to minimize the squared error between the network output values and the target output values.

(1) *Online and offline learning*

Online learning performs using Stochastic Gradient Descent and offline learning using gradient descent.

(2) *Online weight updates*

Weight updates give a single instance (x^t, r^t), where x^t is the input, r^t is the target output, and y^t is the actual output of the network along with a positive constant learning rate ϑ.

(a) Regression in a single output: the weight updates are

$$\Delta v_h = \vartheta(r^t - y^t)z_h^t \tag{9}$$

$$\Delta w_{hj} = \vartheta(r^t - y^t)v_h z_h^t\left(1 - z_h^t\right)x_j^t \tag{10}$$

(b) Regression in multiple (i.e., K) outputs: the weight updates are

$$\Delta v_{ih} = \vartheta\left(r_i^t - y_i^t\right)z_h^t \tag{11}$$

$$\Delta w_{hj} = \vartheta\left(\sum_{i=1}^{K}\left(r_i^t - y_i^t\right)v_{ih}\right)z_h^t\left(1 - z_h^t\right)x_j^t \tag{12}$$

(c) Classification for 2 classes: the weight updates are

$$\Delta v_h = \vartheta(r^t - y^t)z_h^t \tag{13}$$

$$\Delta w_{hj} = \vartheta(r^t - y^t)v_h z_h^t\left(1 - z_h^t\right)x_j^t \tag{14}$$

(d) Classification of $K > 2$ classes: the weight updates are

$$\Delta v_{ih} = \vartheta\left(r_i^t - y_i^t\right)z_h^t \tag{15}$$

$$\Delta w_{hj} = \vartheta\left(\sum_{i=1}^{K}\left(r_i^t - y_i^t\right)v_{ih}\right)z_h^t\left(1 - z_h^t\right)x_j^t \tag{16}$$

F. *Error calculation*

The sum of squared errors function is usually used to calculate errors, it can measure error with one output unit for one training example (x^t, r^t) as

$$E(W, v|x^t, r^t) = \frac{1}{2}(r^t - y^t)^2 \tag{17}$$

Error calculation with multiple output units is simply summation of errors

$$E(W, v|x^t, r^t) = \frac{1}{2}\sum_{i=1}^{K}\left(r_i^t - y_i^t\right)^2 \tag{18}$$

To calculate the error for 1 epoch with one output node (y_i)

$$E(W, v|X) = \frac{1}{2}\sum_{(x^t, r^t)\in X}\left(r_i^t - y_i^t\right)^2 \tag{19}$$

To calculate the error in the entire network for 1 epoch

$$E(W, v|X) = \frac{1}{2}\sum_{(x^t, r^t)\in X}\left(\sum_{i=1}^{K}\left(r_i^t - y_i^t\right)^2\right) \tag{20}$$

G. *Momentum*

To speed up the learning process, the momentum is used instead of the gradient descent method.

$$v_{ih} = v_{ih} + \Delta v_{ih} \tag{21}$$

$$w_{hj} = w_{hj} + \Delta w_{hj} \tag{22}$$

So, new weight update equations are

$$v_{ih}^t = v_{ih}^t + \Delta v_{ih}^t + \alpha\,\Delta v_{ih}^{t-1} \tag{23}$$

$$w_{hj}^t = w_{hj}^t + \Delta w_{hj}^t + \alpha\,\Delta w_{hj}^{t-1} \tag{24}$$

6.1 Design of training set

In the simulation and design of network learning, a multilayer Feedforward Neural Network is designed and network is trained using 'gradient descent with momentum and adaptive learning rate back-propagation,' 'gradient descent with momentum weight and bias learning function,' 'mean squared normalized error performance function' and 'log-sigmoid transfer function' and its structure is 100-5-5-5. (see Table 3).

Figure 14 shows a training state graph which contains training, validation, test and best case. The best validation performance is 9.0905e−024 at epoch 2.

Figure 15 shows the neural network regression state which contains training graph, validation graph, test graph and overall graph. It specifies the overall regression state near about linear which is 1. Table 4 shows simulation results of the training.

Since every input pattern consists of 100 distinct samples so far, Fig. 16 shows the performance graph of simulation result between neural network and input samples.

7 Characters segmentation

For the evaluation of PPTRPRT technique, a gigantic database of 49,000 samples (i.e., Center for Microprocessor Application for Training Education and Research (CMATER), ICDAR-2005, Off-line handwritten devanagari numeral

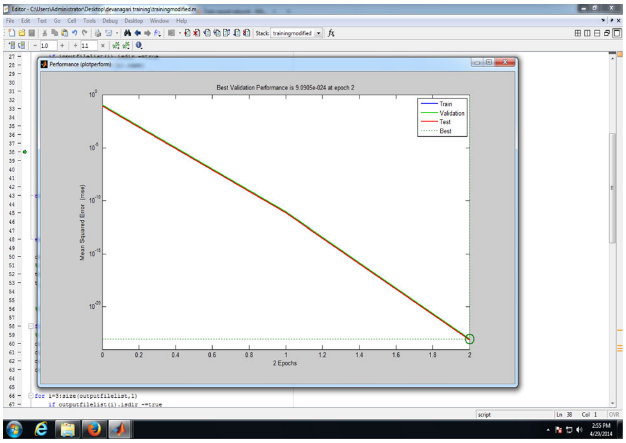

Fig. 14 Feedforward back-propagation neural network training state

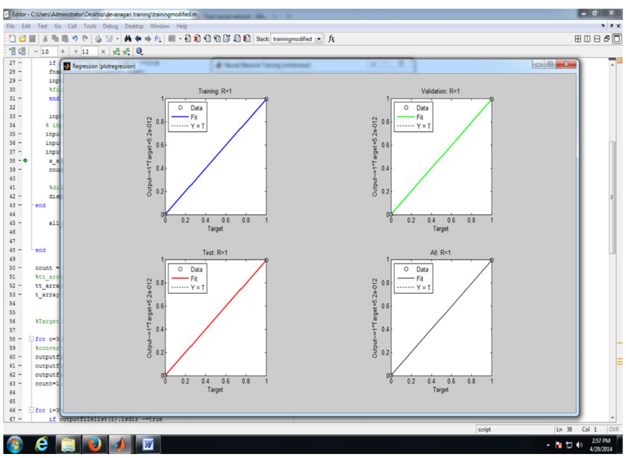

Fig. 15 Feedforward back-propagation neural network regression

database, WCACOM ET0405A-U, HP Labs India Indic Handwriting Datasets) is composed for training pattern.

Algorithm 7: CharactersSegmentation (Ω_{ij})

Step 1: Calculate all black pixels

$\sum B_k(\Omega_{ij})$

Step 2: Plot N Pixels Image

$P_{jk}(\sum N_k, \sum B_k(\Omega_{ij}))$

Step 3: Input image to Feedforward Neural Network and Recall the Input image from Input samples

$\psi_{jk} = ANN(P_{jk}(\sum N_k, \sum B_k(\Omega_{ij})))$

If ($\psi_{jk} > 95$ %)

go to step 2

else

$\psi'_{jk} = \psi_{jk} + P_{jk}(\sum N_k, \sum B_k(\Omega_{ij}))$

$\psi_{jk} = \psi'_{jk}$

Repeat step 3

Step 4: Return (ψ_{jk})

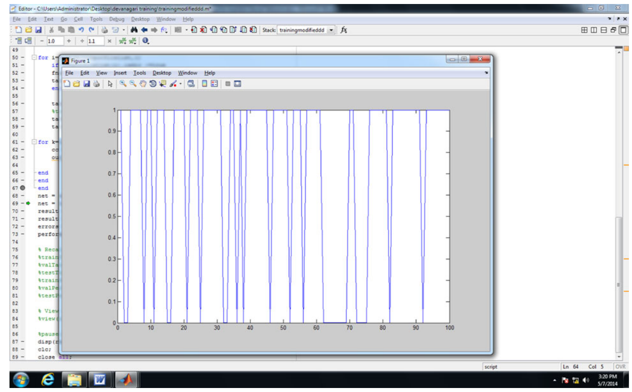

Fig. 16 Simulation results graph

Table 4 Simulation results

Columns 1 through 17																
1	0	0	1	1	1	1	0	1	1	0	1	1	1	0	0	1
Columns 18 through 34																
1	1	1	0	1	1	1	0	1	1	1	1	1	1	0	0	1
Columns 35 through 51																
1	0	1	0	1	1	1	1	1	1	1	0	1	1	1	1	1
Columns 52 through 68																
0	1	1	1	0	1	1	1	1	1	0	0	0	0	0	0	0
Columns 69 through 85																
0	1	1	0	0	0	0	1	1	1	1	1	1	0	1	1	1
Columns 86 through 100																
1	1	1	1	1	1	0	1	1	1	1	1	1	1	1		

In above algorithm, histogram Π_i is used to calculate black pixels $\Sigma B_k(\Omega_{ij})$ from the segmented words and plot N black pixels $P_{jk}(\Sigma N_k, \Sigma B_k(\Omega_{ij}))$ from the calculated black pixels $\Sigma B_k(\Omega_{ij})$. Moreover, P_{jk} pixels are used in the Feedforward Neural Network model to recall the image from samples database. The PPTRPRT technique calculates ψ_{jk} using Feedforward Neural Network

$$\psi_{jk} = ANN\left(P_{jk}\left(\sum N_k, \sum B_k(\Omega_{ij})\right)\right) \tag{25}$$

If recalled input image ψ_{jk} is matched with sample pattern up to ≥99 %, then it proceeds for the next N pixels. Otherwise, calculate ψ'_{jk}, by adding N more pixels in earlier calculated ψ_{jk} and proceed for further operations

$$\psi'_{jk} = \psi_{jk} + P_{jk}\left(\sum N_k, \sum B_k(\Omega_{ij})\right) \tag{26}$$

The outcomes of an algorithm are shown in Fig. 17.

8 Experiments and results

The PPTRPRT technique utilizes a gigantic database of offline handwritten sample scripts (each sample has 10–20

Table 5 Parameter table

No of handwritten training samples		49,000
No of scanned sample Scripts		250
Visibility ratio of samples	Min	0.1
	Max	0.9
Noise detection in unit sample	Min	5
	Max	30
Noise correction in unit sample	Min	90.078 %
	Max	99.23 %
Number of average text lines in unit sample	Min	10
	Max	20
Segmented lines in unit sample	Min	100 %
	Max	100 %
Un-segmented lines in unit sample	Min	0 %
Skew/D-skew of a line in unit sample	Max	0 %
	Min	99.97 %
	Max	100 %
Non-Skew/Non-D-skew of a line in unit sample	Min	0 %
	Max	0 %
Segmented Words of line in unit sample	Min	99.79 %
	Max	100 %
Segmented words in unit sample	Min	99.79 %
	Max	100 %
Words in a line of unit sample	Min	3
	Max	15
Words in unit sample	Min	30
	Max	300
Segmented characters in a word	Min	99.037 %
	Max	100 %
Segmented characters of a line in unit sample	Min	99.037 %
	Max	100 %
Segmented characters in unit sample	Min	97.578 %
	Max	100 %
Characters in a word	Min	1
	Max	15
Characters of a line in unit sample	Min	3
	Max	65
Characters in unit sample	Min	30
	Max	1300

lines) with distinct brightness and intensity. In this research article, descriptive algorithms are used with distinct parameter given in Table 5. The outcomes of algorithms are in the form of lines, words and characters (see Figs. 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19).

An execution of PPTRPRT technique is started with preprocessing algorithm for segmentation of text regions as well as text lines from Devanagari offline handwritten script (see Figs. 5, 6, 7). These segmented text lines were used with skew correction algorithm. Moreover, these skewed lines are available for execution of slant detection algorithm (see Figs. 8, 9, 10), and slant detected lines are used by slant correction algorithm (see Fig. 11). The slant corrected lines are used by white pixel-based word segmentation algorithm (see Figs. 12, 13). These segmented word images are used by characters segmentation algorithm (see Fig. 17).

Outcomes of the PPTRPRT technique are encouraging enough, and analyses of the result are shown in Fig. 18. Figure 18 shows the analytical performance graph in form of minimum and maximum performance ratio for the operations performed during the experiments. Graph shows that noise correction ratio is minimum 90.078 %, maximum 99.23 % and the performance ratio of segmented/unsegmented lines is minimum 0 %, maximum 100 %. The performance ratio of skew/de-skew of segmented line is minimum 99.97 %, maximum 100 % and un-skewed/deskewed lines ratio is 0 %. Besides this, the words segmentation performance ratio over segmented lines is minimum 99.79 %, minimum 100 % and segmented characters performance ratio in a word as well as in the line is minimum 99.037 % and maximum 100 %. Finally, a unit sample performance ratio is minimum 99.578 % and maximum 100 %.

9 Conclusion and further work

This research work presents a realistic characters segmentation technique for Devanagari offline handwritten scripts using a Feedforward Neural Network. The PPTRPRT technique gives a concrete basis to design an optical characters reader with finest accuracy and lowest cost. It is a new technique for reconstructing the Devanagari offline handwritten scripts.

An accuracy of segmented characters from unit sample is 99.578 % and is far enhanced from existing known techniques [15–26]. Table 6 gives comparative details of Devanagari Offline Handwritten Script Recognition systems with the performance of our proposed technique. Puri et al. [5] have proposed accuracy 80.20 % using header line as a feature vector with HMM as a classifier, the size of data set was 39,700. Pal et al. [16] and Malik et al. [17] have proposed accuracy 80.36 and 82 %, respectively, using chain code as feature vector and quadratic was well RE and MFD as classifier, the size of data set was 11,270 and 5000, respectively. Sridhar et al. [18] have proposed accuracy 84.31 % using directional chain code as feature vector and HMM as classifier, the size of data set was 39,700. Murthy et al. [19] have proposed accuracy 90.65 % using distance vector as feature vector and Fuzzy set as classifier, the size of data set was 4750. Basu et al. [20] have proposed accuracy 90.74 % using shadow and CH as feature vector and MLP and MED as classifier, the size of

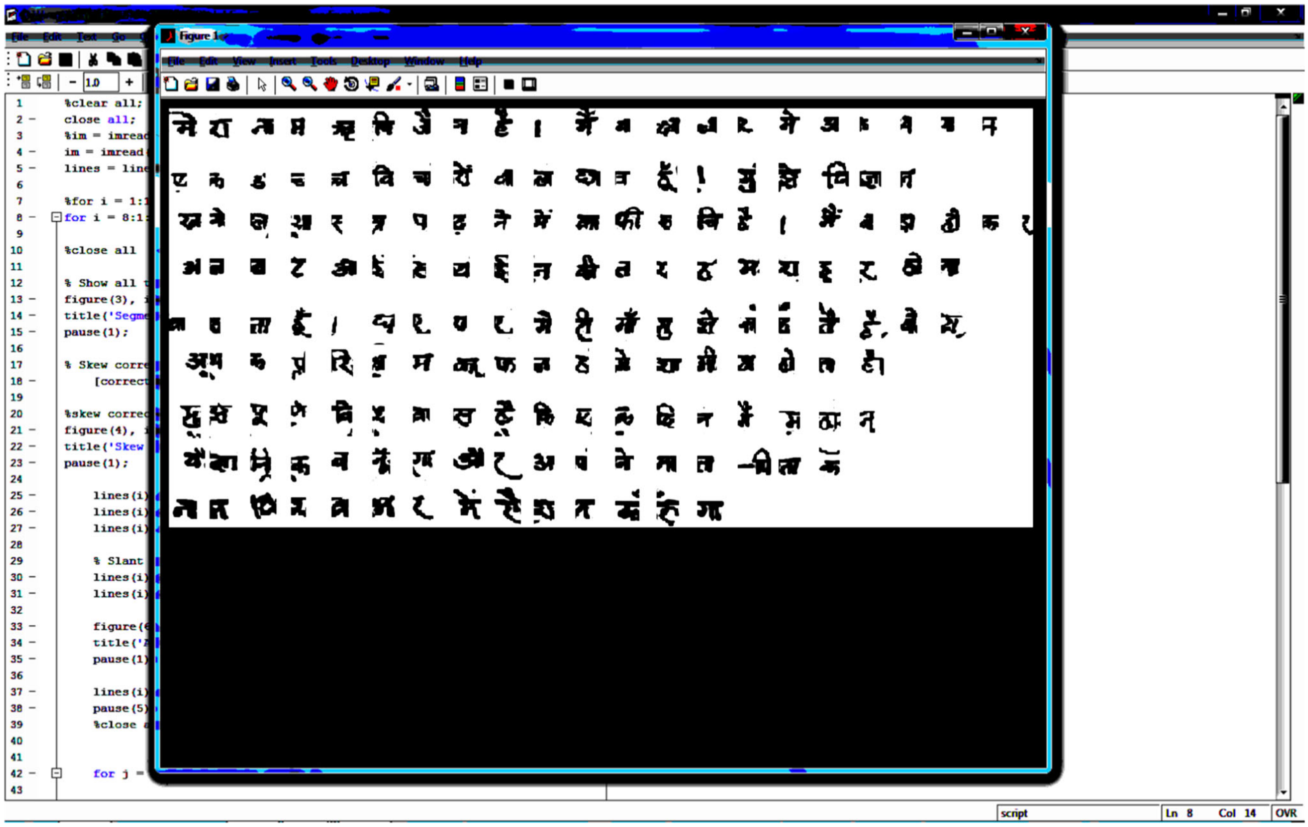

Fig. 17 Segmented characters over words

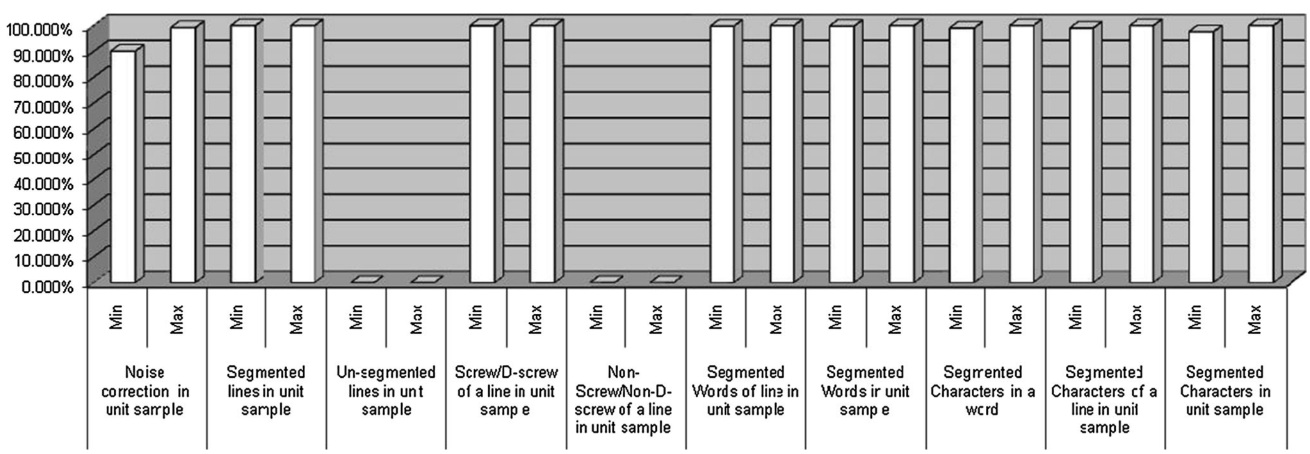

Fig. 18 Parametric analytical graph

data set was 7154. Kumar et al. [21] have proposed accuracy 94.1 % using gradient as feature vector and SVM as classifier, the size of data set was 25,000. Wakabayashi et al. [22] have proposed accuracy 94.24 % using Gaussian filter as feature vector and quadratic as classifier, the size of data set was 36,172. Rabha et al. [23] have proposed accuracy 94.91 % using eigen-deformation as feature vector and elastic matching as classifier, the size of data set was 3600. Kimura et al. [24] have proposed accuracy 95.13 % using gradient as feature vector and MQDF as classifier, the size of data set was 36,172. Bhatcharjee et al. [25] have proposed accuracy 98.12 % using structural vector as feature vector and FFNN as classifier, the size of data set was 50,000. Nasipuri et al. [26] have proposed

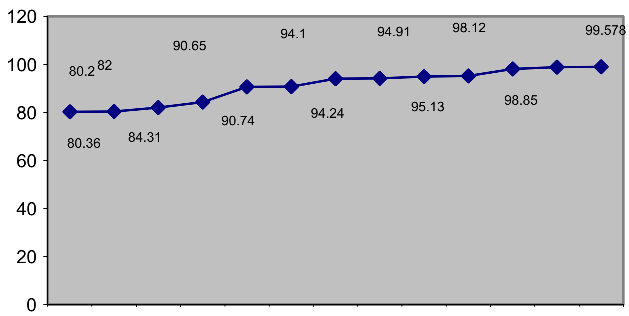

Fig. 19 Parametric analytical graph

Table 6 Comparative details of Devanagari offline handwritten script recognition systems

Methods	Feature	Classifier	Data set (size)	Accuracy %
Parui et al. [15]	Head line	HMM and string edit distance	39,700	80.20
Pal et al. [16]	Chain code	Quadratic	11,270	80.36
Malik et al. [17]	Chain code	RE&MFD	5000	82
Shridhar et al. [18]	Directional chain	HMM	39,700	84.31
Murthy et al. [19]	Distance vector	Fuzzy sets	4750	90.65
Basu et al. [20]	Shadow and CH	MLP and MED	7154	90.74
Kumar et al. [21]	Gradient	SVM	25,000	94.1
Wakabayashi et al. [22]	Gaussian filter	Quadratic	36,172	94.24
Rabha et al. [23]	Eigen-deformation	Elastic matching	3600	94.91
Kimura et al. [24]	Gradient	MQDF	36,172	95.13
Bhatcharjee et al. [25]	Structural	FFNN	50,000	98.12
Nasipuri et al. [26]	Combined	MLP	1500	98.85
Manoj et al.[a]	PPTRPRT	HFNN	49,000	99.578

[a] The notation shows the results of this research paper

accuracy 98.85 % using combined code as feature vector and MLP as classifier, the size of data set was 1500. Finally in proposed work, PPTRPRT technique gives accurate results up to 99.578 % using HFNN as classifier, the size of data set was 49,000. Furthermore, comparison of PPTRPRT technique with other techniques is given in tabular form as below (see Table 6).

Figure 19 shows analytical performance graph to compare performance of the proposed technique over the existing technique. Our future work is to extend the scope of the PPTRPRT technique to recognize English–Hindi mixed offline handwritten scripts.

In this research article, we are giving a realistic technique to reform the segmentation of words and characters from the Devanagari offline handwritten scripts over the existing techniques. It will provide a concrete basis to design an optical characters reader (OCR) with finest accuracy and lowest cost. In the knowledge of researchers, Pixel Plot and Trace and Re-plot and Re-trace (PPTRPRT) technique is new in reconstruction of the Devanagari offline handwritten scripts.

References

1. Jayadevan R, Kolhe SR, Patil PM, Pal U (2011) Offline recognition of Devanagari script: a survey. IEEE Trans Syst Man Cybern Part C Appl Rev. ISSN 1094-6977, pp. 782–976. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=5699408&queryText%3DDevanagari+handwritten+script+segmentation
2. Sahu N, Raman NK (2013) An efficient handwritten Devanagari characters recognition system using neural network. In: International multi-conference on automation, computing communication, control and compressed sensing (iMac4s), IEEE. ISBN 978-1-4673-5089-1, pp. 173–177. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6526403&queryText%3DDevanagari+handwritten+script+segmentation
3. Yadav N, Chaudhury S, Kalra P (2013) Most discriminative primitive selection for identity determination using handwritten Devanagari Script. In: 12th international conference on document analysis and recognition (ICDAR), IEEE. ISSN 1520-5363, pp. 1390–1394. Available:http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6628842&queryText%3DDevanagari+handwritten+script+segmentation
4. Wshah S, Kumar G, Govindaraju V (2012) Script independent word spotting in offline handwritten documents based on hidden Markov models. In: 2012 international conference on frontiers in handwriting recognition (ICFHR). ISBN 978-1-4673-2262-1, pp. 14–19. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6424364&queryText%3DDevanagari+handwritten+script+segmentation
5. Wshah S, Kumar G, Govindaraju V (2012) Multilingual word spotting in offline handwritten documents. In: 21st international conference on pattern recognition (ICPR), IEEE. ISBN 978-1-4673-2216-4, pp. 310–313. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6460134&queryText%3DDevanagari+handwritten+script+segmentation
6. Deshpande PS, Malik L, Arora S (2006) Characterizing hand written Devanagari characters using evolved regular expressions. In: IEEE region 10 conference TENCON 2006. ISBN 1-4244-0548-3, pp. 1–4. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=4142173&queryText%3DDevanagari+handwritten+script+segmentation
7. Rahul PV, Gaikwad AN (2012) Multistage recognition approach for handwritten Devanagari script recognition. In: 2012 World congress on information and communication technologies (WICT). ISBN 978-14673-4806-5, pp. 651–656. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6409156&queryText%3DDevanagari+handwritten+script+segmentation
8. Pal U, Roy RK, Kimura F (2012) Multi-lingual city name recognition for indian postal automation. In: 2012 International conference on frontiers in handwriting recognition (ICFHR).

ISBN 978-1-4673-2262-1, pp. 169–173. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6424387&queryText%3DDevanagari+handwritten+script+segmentation
9. Doiphode A, Ragha L (2011) Novel approach for segmentation of handwritten touching characters from Devanagari words. Das VV, Thankachand N (eds) CIIT 2011 Springer: Berlin. Vol. 250, pp. 621–623. Available: http://link.springer.com/chapter/10.1007%2F978-3-642-25734-6_106#page-2
10. Sankaran N, Neelappa A, Jawahar CV (2013) Devanagari text recognition: a transcription based formulation. In: 12th International conference on document analysis and recognition (ICDAR). ISSN 1520-5363, pp. 678–682. Available: http://ieeexplore.ieee.org/xpl/articleDetails.jsp?tp=&arnumber=6628704&queryText%3DDevanagari+script+segmentation
11. Kumar M, Jindal MK, Sharma RK (2011) Review on OCR for handwritten indian scripts character recognition, advances in digital image processing and information technology. In: First international conference on digital image processing and pattern recognition, DPPR 2011. ISBN 978-3-642-24054-6, pp. 268–276. Available: http://link.springer.com/chapter/10.1007%2F978-3-642-24055-3_28
12. Wshah S, Kumar G, Govidaraju V (2014) Statistical script independent word spotting in offline handwritten documents. Pattern Recognit 47(3):1039–1050. Available : http://www.sciencedirect.com/science/article/pii/S0031320313003919
13. Basu S, Das N, Sarkar R, Kundu M, Nasipuri M, Basu DK (2010) A novel framework for automatic sorting of postal documents with multi-script address blocks. Pattern Recognit 43(10):3507–3521. Available: http://www.sciencedirect.com/science/article/pii/S0031320310002426
14. Hassan E, Chaudhury S, Yadav N, Kalra P, Gopal M (2014) Off-line hand written input based identity determination using multi kernel feature combination. Pattern Recognit Lett 35:113–119. Available: http://www.sciencedirect.com/science/article/pii/S0167865513001955
15. Shaw B, Parui SK, Shridhar M (2008) A segmentation based approach to offline handwritten Devanagari word recognition. In: International conference on information technology, (ICIT '08). ISBN 978-1-4244-37645-0, pp. 256–257. Available: http://ieeexplore.ieee.org/Xplore/defdeny.jsp?url=http%3A%2F%2Fieeexplore.ieee.org%2Fstamp%2Fstamp.jsp%3Ftp%3D%26arnumber%3D4731338%26userType%3Dinst&denyReason=-133&arnumber=4731338&productsMatched=null&userType=inst/
16. Sharma N, Pal U, Kimura F, Pal S (2006) Recognition of offline handwritten Devnagari characters using quadratic classifier. In: Proceedings indian conference on computer vision graphics and image processing. pp. 805–816. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.333.2508
17. Deshpande PS, Malik L, Arora S (2008) Fine classification & recognition of hand written Devnagari characters with regular expressions & minimum edit distance method. J Comput 3(5): 11–17. ISBN 1796-203X. Available: http://ojs.academypublisher.com/index.php/jcp/article/view/03051117
18. Shaw B, Parui SK, Shridhar M (2008) Off-line handwritten Devanagari word recognition: a holistic approach based on directional chain code feature and HMM. In: International conference on information technology, 2008 (ICIT'08). ISBN 978-1-4244-3745-0, pp. 203–208
19. Hanmandlu M, Murthy OVR, Madasu VK (2007) Fuzzy model based recognition of handwritten hindi characters. In: 9th Biennial conference of the Australian pattern recognition society on digital image computing techniques and applications. ISBN 0-7695-3067-2, pp. 454–461. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4426832&abstractAccess=no&userType=inst
20. Arora S, Bhattacharjee D, Nasipuri M, Basu DK, Kundu M (2010) Recognition of non-compound handwritten Devanagari characters using a combination of MLP and minimum edit distance. Int J Comput Sci Secur (IJCSS) 4(1):1–14 Available: http://arxiv.org/ftp/arxiv/papers/1006/1006.5908.pdf
21. Kumar S (2009) Performance comparison of features on Devanagari hand printed dataset. Int J Recent Trends Eng 1(2):33–37. Available: http://www.academypublisher.com/ijrte/vol01/no02/ijrte0102033037.pdf
22. Pal U, Sharma N, Wakabayashi T, Kimura F (2007) Off-line handwritten character recognition of Devnagari script. In: Ninth international conference on document analysis and recognition. ISBN 978-0-7695-2822-9, vol. 1, pp. 496–500. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4378759&abstractAccess=no&userType=inst
23. Mane V, Ragha L (2009) Handwritten character recognition using elastic matching and PCA. In: International conference on advances in computing, communication and control (ICAC3'09). pp. 410–415. Available: http://robotics.csie.ncku.edu.tw/HCI_Project_2010/P78991121_%E9%84%AD%E8%A9%A0%E4%B9%8B/ICACCC2009%20Handwritten%20character%20recognition%20using%20elastic%20matching%20and%20PCA.pdf
24. Pal U, Chanda S, Wakabayashi T, Kimura F (2008) Accuracy improvement of Devnagari character recognition combining SVM and MQDF. In: 11th International conference on frontiers handwriting recognition. pp. 367–372. Available: http://www.cenparmi.concordia.ca/ICFHR2008/Proceedings/papers/cr1080.pdf
25. Arora S, Bhatcharjee D, Nasipuri M, Malik L (2007) A two stage classification approach for handwritten Devanagari characters. In: IEEE International conference on computational intelligence and multimedia applications. ISBN 0-7695-3050-8, Vol. 2, pp. 399–403. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=4426729&abstractAccess=no&userType=inst
26. Arora S, Bhattacharjee D, Nasipuri M, Basu DK, Kundu M, Malik L (2009) Study of different features on handwritten Devnagari character. In: 2nd International conference on emerging trends in engineering and technology (ICETET). ISBN 978-1-4244-5250-7, pp. 929–933. Available: http://ieeexplore.ieee.org/xpl/freeabs_all.jsp?arnumber=5395510&abstractAccess=no&userType=inst