Neural Comput & Applic (2016) 27:1369–1379
DOI 10.1007/s00521-015-1940-x

ORIGINAL ARTICLE

Segmentation of english Offline handwritten cursive scripts using a feedforward neural network

Manoj Kumar Sharma[1] · **Vijay Pal Dhaka**[1]

Received: 16 March 2015 / Accepted: 3 June 2015 / Published online: 16 July 2015
© The Natural Computing Applications Forum 2015

Abstract In the present paper, we used the Pixel Plot and Trace and Re-plot and Re-trace (PPTRPRT) technique for English offline handwritten curve scripts and leads. Unlike other approaches, the PPTRPRT technique prioritizes segmentation of words and characters. The PPTRPRT technique extracts text regions from English offline handwritten cursive scripts and leads an iterative procedure for segmentation of text lines along with skew and de-skew operations. Iteration outcomes provide for pixel space-based word segmentation which enables segmentation of characters. The PPTRPRT technique embraces various dispensations in segmentation of characters from English offline handwritten cursive scripts. Moreover, various normalization steps allow for deviations in pen breadth and inscription slant. Investigational outcomes show that the proposed technique is competent at extracting characters from English offline handwritten cursive scripts.

Keywords PPTRPRT · Trace · Plot · Offline · Handwritten · Segmentation

1 Introduction

The recognition of English offline handwritten cursive script is of burning interest for researchers due to the high unevenness of scripting styles. The high gratitude tariff has been published to recognize remote digits, characters or words [11, 12, 15–24]. The gratitude performance achieved for recognition systems of unconstrained English offline handwritten cursive scripts is drastically poorer. It has transpired since research focused on relevant domains wherever solitary words are implicated (e.g., handwritten cursive addresses, names on bank check lists and drafts or signature reading). Many pre-processing steps are performed by automatic English offline handwritten cursive script recognition systems, which allow for moderate deviation in the handwritten script and conserved information relevant to acknowledgement of the signature.

In this research article, we present a realistic technique for improving word and character segmentation [25] from offline scripts over the existing techniques. This technique will provide a concrete basis for design of optical character readers with the fine accuracy and low cost. In the realm of research, the "Pixel Plot and Trace and Re-plot and Re-trace" (PPTRPRT) technique has been applied by the authors with Devanagari script segmentation [15] and has been newly applied in the reconstruction of offline handwritten English cursive scripts.

The present research article is arranged as follows: Sect. 2 gives a brief introduction of related works; Sect. 3 defines mathematical notations used in the description algorithms; Sect. 4 discusses the methodologies of the PPTRPRT technique; Sects. 5, 6 and 7 deal with word and character segmentation.

2 Related work

The researchers have published numerous approaches for improving segmentation of English offline handwritten cursive script. Dhaval et al. [1] tested segmentation by evaluating the local stroke geometry (imposed the width, height and aspect-ratio constraints in resultant characters),

✉ Manoj Kumar Sharma
manoj186@yahoo.co.in

[1] Jaipur National University, Jagatpura Jaipur Rajasthan, Jaipur 302017, India

without making limiting assumptions on the characters size and number of characters in a word. This approach found the segmented text with the most average liveliness of the resulting characters. A possible location of the segmenting neighboring was described by a graph model. Ram et al. [2, 14, 15] used text line contour estimation to analyze and place neighboring text line segments in their real boundaries. Impedovo et al. [3] has proposed a database for handwritten cursive basic words recognition. The database includes various instances of basic words for bank checks. Lee et al. [4] discussed over-segmentation of the words in a text line to reduce the chances of under-segmentation. Over-segmented words have been used in neural network binary validation systems for producing results on behalf of fitness function. Florence et al. [5] proposed a technique for selecting a certain place in a text line and checked that the selected places belong to a letter of a word, or a space between two words. Njah et al. [6] obtained segmentation of bank cheques using a new database. Shivram et al. [7] used English character features with geometrical points for the segmentation of postal addresses.

3 Mathematical terms

The use of the following standard symbols is strongly recommended (Table 1).

4 Methodology

In this Section, algorithms of the PPTRPRT technique are discussed in detail.

4.1 Overview of system design

The architecture of the PPTRPRT technique is given in Fig. 1.

The pattern sets used in the current study are English offline handwritten cursive script and a feedforward neural network is used to draw desired outcomes with improved accuracy.

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

The PPTRPRT technique extracts a text region from English offline handwritten cursive script images and passes it through an iterative process for segmentation of text lines. These extracted text lines were uses in skew and de-skew operations and skewed/de-skewed images were provided for white pixel-based word segmentation. The segmented words were used in an iterative process for segmentation of characters. The PPTRPRT technique embraces various dispensations to extract cursive

Table 1 Mathematical notations in the segmentation

Symbols	Description
ε_i	English offline handwritten cursive 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 and 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	Extracted 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 ($j = 1,\ldots, n$)
λ_{ij}	White spaces of i'th text line ($j = 1,\ldots, n$)
$\aleph_i$	Cluster function for i'th text line
Ω_{ij}	Segmented word of i'th text line ($j = 1,\ldots, n$)
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 the FFNN

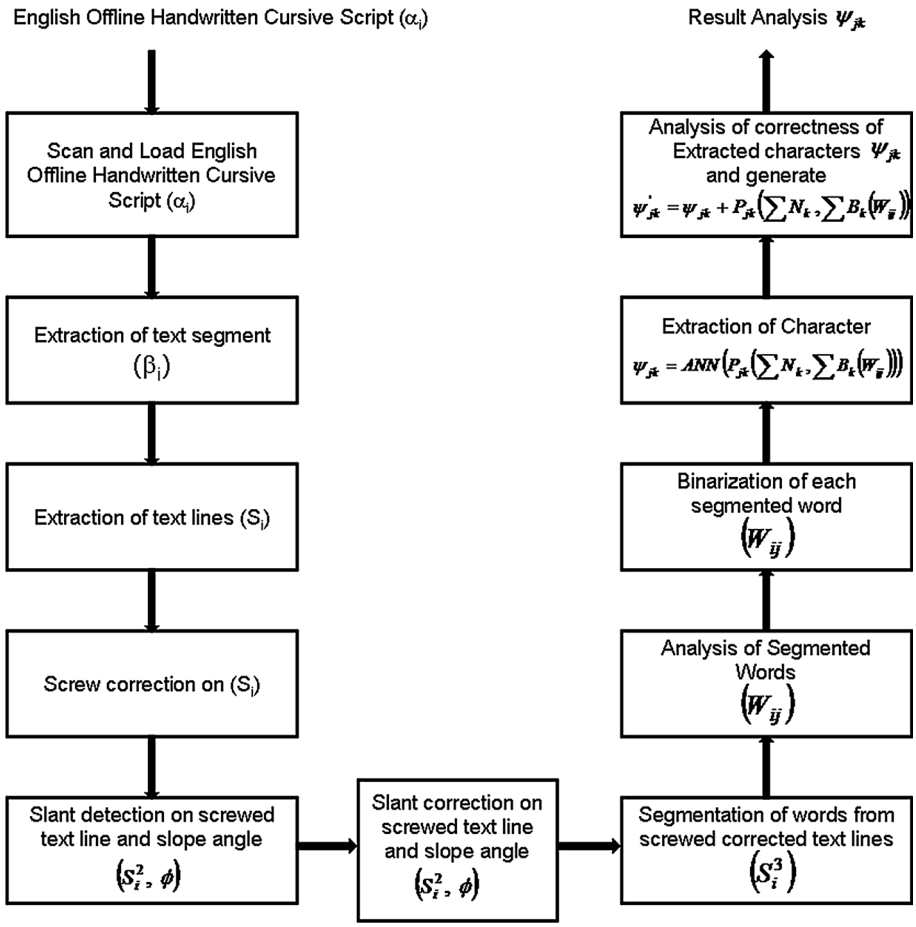

Fig. 1 Architecture of the PPTRPRT technique

characters from English offline handwritten cursive script. Normalization steps allow for deviations in pen breadth and inscription slant.

The PPTRPRT technique initiates with a segmentation algorithm.

Algorithm 1: Segmentation ()

Step 1 Load offline cursive English script file ε_i
Step 2 Call preprocessing (ε_i)

The segmentation algorithm starts with uploading English offline handwritten cursive script ε_i. The outcome of an algorithm is shown in Fig. 2.

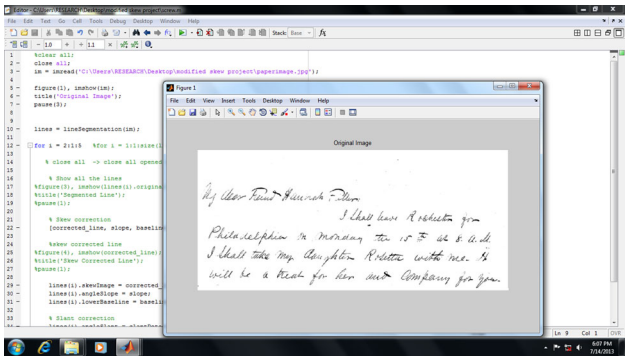

Fig. 2 Original text image

4.2 Pre-processing for character segmentation

The outcome of the segmentation algorithm is availed by the preprocessing algorithm for forthcoming operations.

Algorithm 2: Preprocessing (ε_i)

Step 1 Extract text segment
$\beta_i = \varepsilon_i - \rho$

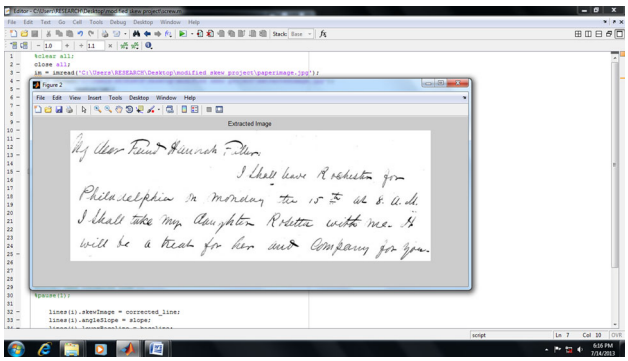

Fig. 3 Extraction of text image

Step 2 Extract text lines from β_i
(a) Define "PEAK_FACTOR" $(\Gamma_i) = 5$ and "THRESHOLD" $(\Xi_i) = 1$
(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) Extract text line (S_i) and drop white parts $(\exists_k)$
Step 3 Skew correction (S_i)

In the above defined algorithm, operations start with extraction of the text segment β_i from α_i by removing noise ρ. This text segment β_i is applied for text line segmentation by calculating Max Γ_i and Min Γ_i using PEAK_FACTOR Γ_i with various adequate operations. Furthermore, using Max Γ_i, and Min Γ_i along with size of an image L one can calculate the text line segmentation points (P_j). Moreover, it extracts the text line (S_i) from the text segment β_i by eliminating the white parts $(\exists_k)$ and using the THRESHOLD value. The outcome of an algorithm is shown in Fig. 3.

4.3 Skew correction

Moreover, the outcomes of the preprocessing algorithm were availed by the skew correction algorithm for further operations.

Algorithm 3: Skew correction (S_i)

Step 1 Improve intensity of the black pixels (Bk)
Step 2 Find the (x, y) coordinates of the lower pixels (low_bk)
Step 3 Filter the irrelevant pixels $F_i(low\_bk)$
Step 4 Calculate the slope angle $(\theta_i(S_i))$ of the text line using linear regression
Step 5 Skew or de-skew the text line (S_i) with slope angle $(\theta_i(S_i))(S_i^2 = \text{Screw}\,\theta_i\ or\ De - \text{Screw}\,\theta_i)$

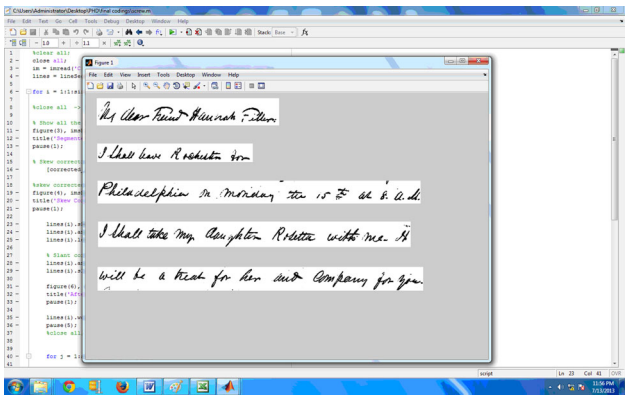

Fig. 4 Slant detection of the text lines

Step 6 Remove the black corners from the skewed text line
$S_i^2 = S_i^1 - \Lambda_k$
Step 7 Slant detection $(S_i^2,\ \theta_i)$

The above defined algorithm starts by calculating the (x, y) coordinate of the lower pixels and filtering the irrelevant pixels $F_i(\text{low\_}bk)$ from extracted text line (S_i). The slope angle $(\theta_i(S_i))$ of the text line (S_i) is calculated using linear regression $(\theta_i(S_i))$ and removes the black corners with the skewed operation $S_i^2 = S_i^1 - \Lambda_k$. The outcomes of an algorithm are shown in Fig. 4.

4.4 Slant detection and correction

Moreover, outcomes of the skew correction algorithm (skewed text line S_i^2 along with the angle θ_i) availed by the slant detection algorithm for further operations.

Algorithm 4: Slant Detection $(S_i^2,\ \theta_i)$

Step 1 Trace the regional boundaries of the image $\omega_i(S_i^2)$
Step 2 Define the "min_strock_length" and "step_size"
Step 3 Do histogram count operations (Π_i)
Step 4 Find the indices and values of non-zero elements
Step 5 Calculate the slant angle ϕ_i
Step 6 Slant correction $(S_i^2,\ \phi_i)$

Operation of the slant detection algorithm starts by tracing the regional boundaries of an image $\omega_i(S_i^2)$ and by defining the min_strock_length and step_size. Moreover, using histogram count operations Π_i, the indices of the text line S_i^2 are traced and the non-zero elements are counted. A slant angle ϕ_i is calculated using indices and values of the non-zero elements. The outcomes of an algorithm are shown in Fig. 5.

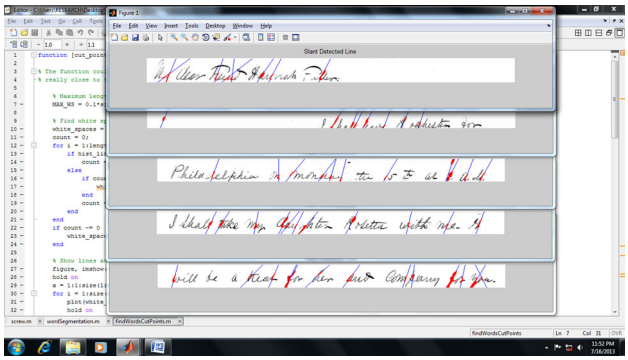

Fig. 5 Slant detection of text lines

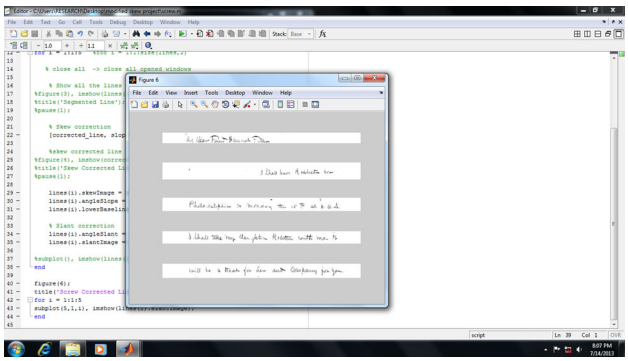

Fig. 6 Slant correction of the text line

Besides, slant angle ϕ_i along with the text line S_i^2 is used for slant correction operations.

Algorithm 5: Slant Correction $\left(S_i^2,\ \phi_i\right)$

Step 1 Calculate the height $\left(\xi_i\left(S_i^2\right)\right)$, width $\left(\eta_i\left(S_i^2\right)\right)$ and depth $\left(\zeta_i\left(S_i^2\right)\right)$ of the text line $\left(S_i^2\right)$

Step 2 Define the new text line matrix $A_i\left(S_i^2\right)$

Step 3 Create a spatial transformation structure of the text line

$R_i\left(A_i\left(S_i^2\right)\right)$

$T_i\left(A_i\left(S_i^2\right)\right)$

$U_i\left(A_i\left(S_i^2\right)\right)$

Step 4 Extract the final skewed text line

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

Step 5 Word segmentation $\left(S_i^3\right)$

In the above defined algorithm, the operation starts with calculation of the height $\left(\xi_i\left(S_i^2\right)\right)$, width $\left(\eta_i\left(S_i^2\right)\right)$ and depth $\left(\zeta_i\left(S_i^2\right)\right)$ of the text line $\left(S_i^2\right)$. These parameters are used in formulating a new text line matrix $A_i\left(S_i^2\right)$ and spatial transformation structures $R_i\left(A_i\left(S_i^2\right)\right)$, $T_i\left(A_i\left(S_i^2\right)\right)$, $U_i\left(A_i\left(S_i^2\right)\right)$. These spatial transformation structures have been used for segmentation of skewed text line $S_i^3 = (R_i - T_i + U_i)$. The skewed text line $S_i^3 = (R_i - T_i + U_i)$ is to be used in word segmentation. The outcomes of an algorithm are shown in Fig. 6.

5 Word segmentation

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

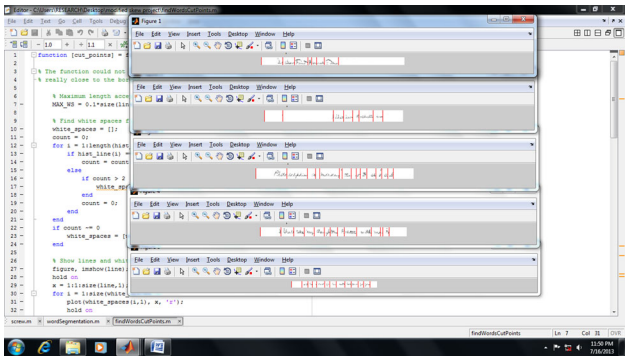

Fig. 7 White spaces between words

Algorithm 6: Word Segmentation $\left(S_i^3\right)$

Step 1 Generate a histogram Π_i

Step 2 Find the segmentation points of word Q_{ij}

(a) Find white spaces $\lambda_{ij} = \sum \exists_k$

(b) Cluster the text line to distinguish between white spaces

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

Step 3 Character segmentation $\left(\Omega_{ij}\right)$

The above algorithm starts by calculating a histogram Π_i and the white space $\lambda_{ij} = \Sigma\exists_k$. These white spaces are uses to calculate word segmentation points Q_{ij}. The outcomes of an algorithm are shown in Figs. 7 and 8.

The segmented words $\left(\Omega_{ij}\right)$ are to be used by character segmentation algorithm for segmentation of characters using a feedforward neural network model.

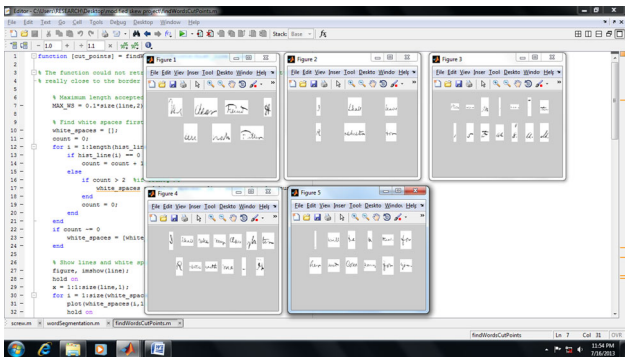

Fig. 8 Segmented words from the text line

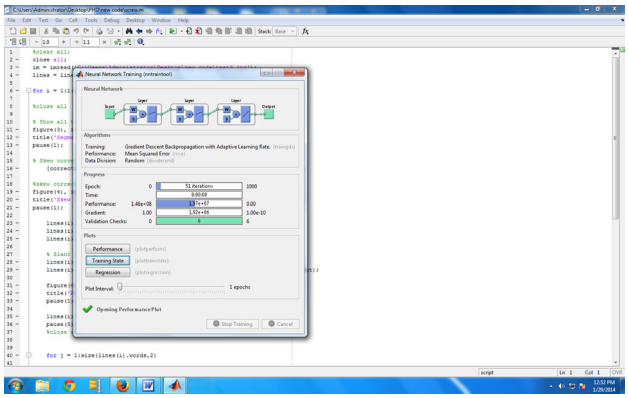

Fig. 9 Feedforward backpropagation neural network training (20–5–5–5)

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; i.e., one or more hidden layers besides one output layer. Each layer has an associated weight [i.e., weight (w_0) feeds into the hidden layer and weight (z_0) feeds into the output layer]; there are neither backwards connections nor skipping connections between layers. These weights will adjust while training the network using a backpropagation algorithm. Typically, all input units are connected to hidden layer unit and hidden layer units are connected to the output units.

6.1 Input units

Input units feed data into the system without any processing; the value of an input unit is x_j, for j going for 1 to d input units along with a special input unit x_0 that contains a constant value 1 and provides bias to the hidden nodes.

6.2 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:

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

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

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

6.3 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 the number of the output. The weights going into the output unit is 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, the weighted sum is

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

6.4 Functions

6.4.1 Regression for single and multiple outputs

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

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

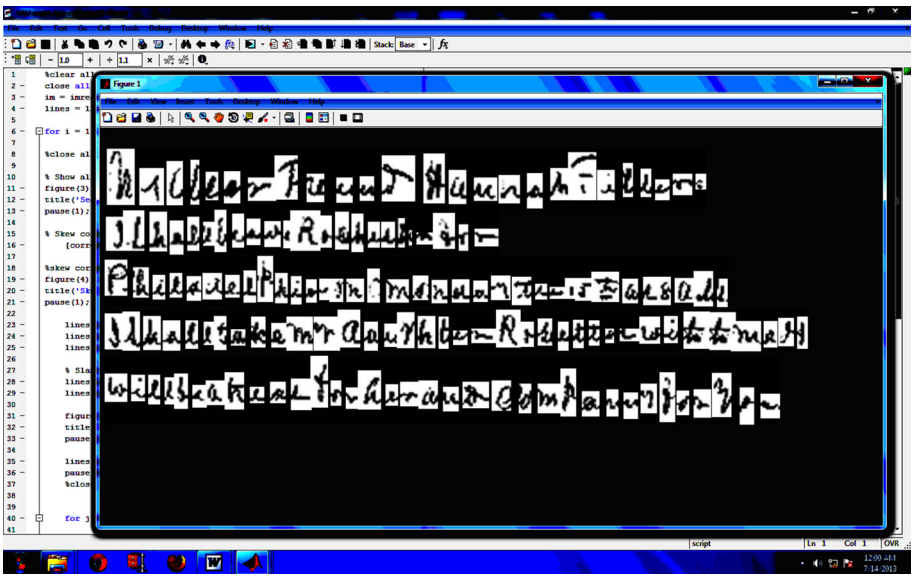

Fig. 10 Segmentation of characters over the words

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

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

6.4.2 *Classification for two 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.

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

7 Character segmentation

7.1 Design of the training set

In simulation design of network learning, a multilayer feedforward neural network is utilized. The neural 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'. Since every input pattern consist of 20 distinct features. A neural network has 20 processing units in the input layer. The architecture of the neural network consists of the input layer as well as two hidden layers with five units and one output layer with five units, i.e., a 20–5–5–5 architecture (see Fig. 9).

Table 2 Distinct parameters

No. of scanned sample	250
Visibility ratio of samples	
Min.	0.1
Max.	0.9
Noise detection in unit sample	
Min.	5
Max.	30
Number of average text lines in unit sample	
Min.	10
Max.	20
Words in a Line of unit sample	
Min.	3
Max.	15
Words in unit sample	
Min.	30
Max.	300
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

For evaluation of the PPTRPRT technique, a gigantic database was composed for training patterns. A

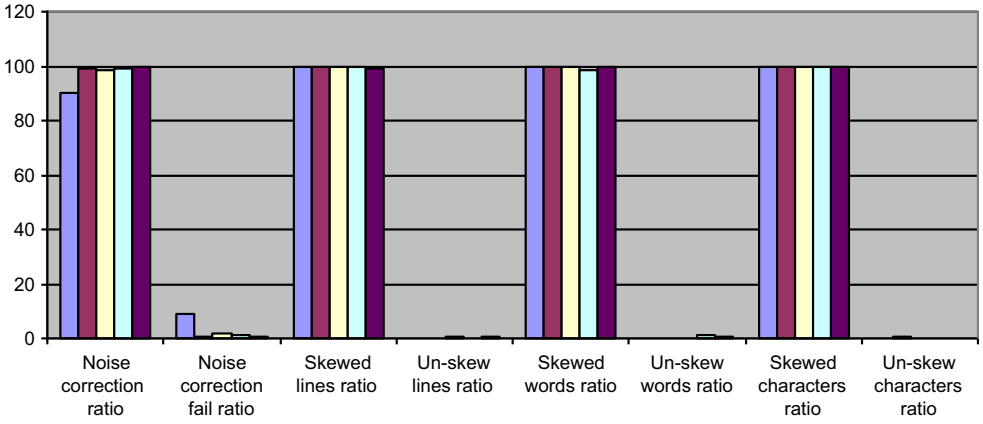

Fig. 11 Pre-processing analytical results

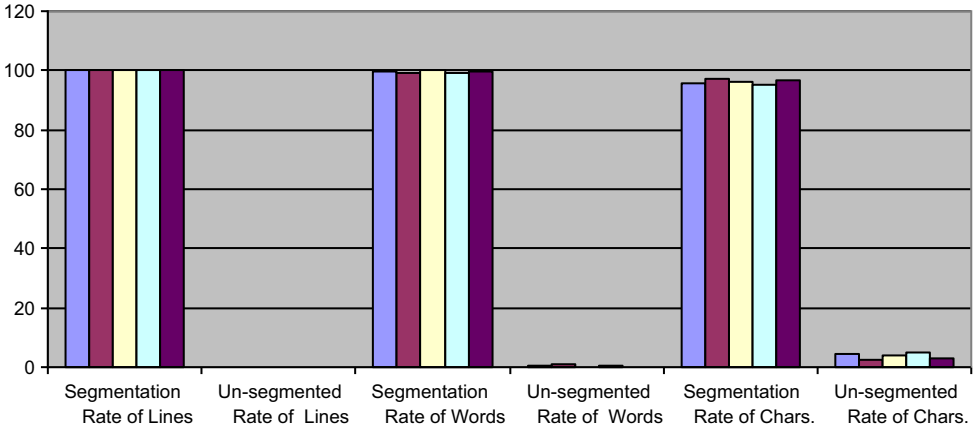

Fig. 12 Analytical graph of the final segmentation results of the proposed technique

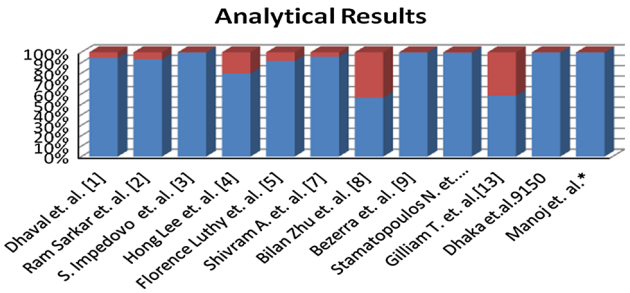

Fig. 13 Comparative analysis of the proposed technique with existing segmentation techniques

histogram of offline images is in 20 columns to deliberate the compactness of pixels. Moreover, we obtained 20 input pattern vectors (arranged in 20×1) of the training set for each English offline handwritten cursive script image. Output pattern vector corresponded to the input pattern vector which consists of 5×1 binary values. Sample test patterns were used to verify the performance of the qualified neural network. So, segmentation of characters over the segmented words is applied using a feedforward neural network.Algorithm 7: Character Segmentation (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 Give image to the feedforward neural network and recall the given image from the given samples

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

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})

A histogram Π_i is used in the above algorithm and calculate the black pixels $\Sigma B_k(\Omega_{ij})$ from the segmented words, and a plot of the N black pixels $P_{jk}(\Sigma N_k, \Sigma B_k(\Omega_{ij}))$ is used calculate the black pixels $\Sigma B_k(\Omega_{ij})$. Moreover, P_{jk} pixels are used by the feedforward

Table 3 Final pre-processing results of the proposed technique

Data set	Noise correction ratio	Noise correction fail ratio	Skewed lines ratio	Un-skewed lines ratio	Skewed words ratio	Un-skewed words ratio	Skewed characters ratio	Un-skewed characters ratio
CMATER	90.078	9.022	99.97	0.03	100.00	0.0	99.99	0.01
ICDAR-2005	99.23	0.77	99.67	0.23	99.90	0.10	99.66	0.34
WCACOM ET0405A-U	98.45	1.55	99.43	0.57	99.95	0.05	100.0	0.0
HP Labs India Indic handwriting datasets	99.00	1.00	99.89	0.11	98.78	1.22	99.99	0.01
Off-line handwritten numeral database	99.43	0.57	99.32	0.68	99.65	0.35	99.83	0.17

Table 4 Final segmentation results of the proposed technique using different datasets

Data set	Scanned text images	Words	Characters	Segmentation rate of lines	Un-segmented rate of lines	Segmentation rate of words	Un-segmented rate of words	Segmentation rate of chars.	Un-segmented rate of chars.
CMATER	250	5000	8000	100.00	0.0	99.45	0.55	95.57	4.43
ICDAR-2005	300	8000	11,000	100.00	0.0	99.00	1.00	97.32	2.68
WCACOM ET0405A-U	120	6400	19,000	100.00	0.0	100.00	0.0	96.00	4.00
HP Labs India Indic handwriting datasets	200	4000	9000	100.00	0.0	99.32	0.68	95.00	5.00
Off-line handwritten numeral database	180	3000	2000	100.00	0.0	99.76	0.24	96.78	3.22

Table 5 Comparative analysis of the proposed technique with existing segmentation techniques

Methods	Feature	Classifier	Data set (size)	Accuracy (%)
Dhaval et al. [1]	Graph model	Average longest path Algo	1300	73.45
Sarkar et al. [2]	Fragments	Line contour estimation	1240	88.44
Impedovo et al. [3]	SIDB	HMM	95,760	89.35
Lee et al. [4]	TCM	OSNVA	311	79
Luthy et al. [5]	TWN	HMM	1025	92.36
Shivram et al. [7]	CRF	HMM	2127	92
Zhu et al. [8]	P2DBMN	MRF	100	76.89
Bezerra et al. [9]	LSTM	Hybrid RNN	57,293	84.37
Stamatopoulos et al. [10]	NCSR	SVM	23,525	94.17
Gilliam et al. [13]	De facto standards	Grapheme codebook	136	97
Dhaka et al. [15]	PPTRPRT	FFNN	49,000	99.578
Manoj et al. [this study]	PPTRPRT	FFNN	49,000	97.32

neural network model to recall an image from the sample database.

The PPTRPRT technique (which is based on the under-segmentation to over-segmentation approach) calculates ψ_{jk} using a feedforward neural network

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

where ψ_{jk} denoted the percentage of extracted features using feedforward neural network (FFNN).

If the recalled image ψ_{jk} matched with the sample pattern up to $\geq$ 95 %, then proceed to the next set of N pixels. Otherwise, calculate ψ'_{jk} by adding N more pixels in the earlier calculation ψ_{jk} and proceed with further operations

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

The outcomes of an algorithm are shown in Fig. 10.

8 Experiments and results

The PPTRPRT technique utilizes a gigantic database of offline handwritten samples (each sample has 10 to 20 lines) with distinct brightness and intensity. In this research article, descriptive algorithms are used with distinct parameters given in Table 2. The outcomes of the 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).

The PPTRPRT technique starts its execution with the preprocessing algorithm by extracting text regions and segmenting text lines from English offline handwritten cursive script images (see Fig. 3). These segmented text lines (see Fig. 4) passed through the skew correction algorithm, and skewed lines were used by the slant detection algorithm (see Fig. 5). The detected line slants-were then used by the slant correction algorithm (see Fig. 6). The pre-processing results from noise detection/correction to skewed/un-skewed character ratio is given in Table 3.

Figure 11 shows the analytical performance graph of the pre-processing operations. The graph is showing achieved and un-achieved performance ratios for distinct datasets used in the experiments.

Therefore, the slant-corrected lines were used by the white pixel-based word segmentation algorithm (see Figs. 7, 8) and the outcomes of word segmentation were used by the characters segmentation algorithm. A multilayer FFNN is designed for network learning, and the network is trained with 'gradient descent with momentum and adaptive learning rate back-propagation', a 'gradient descent with momentum weight and bias learning function', a 'mean squared normalized error performance function' and a 'log-sigmoid transfer function', and the architecture of the network is 20–5–5–5 (see Fig. 9). Outcomes of the PPTRPRT technique are encouraging enough and analyses of the results are presented in Figs. 11 and 13.

Figure 12 shows the analytical performance graph between the final segmentation results of the proposed technique for distinct datasets used in the experiments (Table 4).

Finally, in the proposed work, the PPTRPRT technique provides the best segmentation results, up to 97.32 %, using a FFNN as a classifier; the size of data set was 49,000. Therefore, a comparative analysis of the proposed technique with existing techniques is given in Table 5.

Figure 13 shows a results analysis of the proposed technique as compared to the performance of the existing techniques. Our future work intends to extend the scope of the PPTRPRT technique to segmentation of multilingual offline handwritten scripts.

9 Conclusion and further work

This research work presents a realistic technique for character segmentation of English offline handwritten cursive scripts using a FFNN. The PPTRPRT technique is a new technique for reconstructing English offline handwritten cursive and is driving the results by keeping an approach between under-segmentation and over-segmentation. The technique will provide a concrete basis by which design of an optical character reader with fine accuracy and low cost will be achieved.

References

1. Salvi D, Zhou J, Waggoner J, Wang S (2013) Handwritten text segmentation using average longest path algorithm. In: Workshop on applications of computer vision (WACV), IEEE, ISBN 978-1-4673-5053-2, pp 505–512
2. Sarkar R, Halder S, Malakar S, Das N, Basu S, Nasipuri M (2012) Text line extraction from handwritten document pages based on line contour estimation. In: 3rd international conference on computing communication and networking technologies (ICCCNT), IEEE, INSPEC 13252116, pp 1–8
3. Impedovo S, Facchini G, Mangini FM (2012) A new cursive basic word database for bank-check processing systems. In: 10th IAPR international workshop on document analysis systems (DAS), IEEE, ISBN 978-1-4673-0868-7, pp 450–454
4. Lee H, Verma B (2010) Over-segmentation and neural binary validation for cursive handwriting recognition. In: International joint conference on neural networks (IJCNN), IEEE, ISBN 978-1-4244-6916-1, pp 1–5
5. Luthy F, Varga T, Bunke H (2007) Using hidden markov models as a tool for handwritten text line segmentation. In: Ninth international conference on document analysis and recognition (ICDAR 2007), IEEE, ISBN 978-0-7695-2822-9, p 8
6. Njah S, Nouma BB, Bezine H, Alimi AM (2012) MAYASTROUN: Multilanguage handwriting database. In: International conference on frontiers in handwriting recognition (ICFHR), IEEE, ISBN 978-1-4673-2262-1, pp 308–312
7. Shivram A, Zhu B, Setlur S, Nkagawa M (2013) Segmentation-based online word recognition: a conditional random field driven beam search strategy. In: 12th International conference on document analysis and recognition (ICDAR), IEEE, ISSN 1520-5363, pp 852–856

8. Zhu B, Shivram A, Setlur S, Govindaraju V, Nakagawa M (2013) Online handwritten cursive word recognition using segmentation-free MRF in combination with P2DBMN-MQDF. In: 12th international conference on document analysis and recognition (ICDAR), IEEE, ISSN 1520-5363, pp 349–353
9. Bezerra BLD, Zanchettin C, Bragad de Andrade V (2012) A hybrid RNN Model for cursive offline handwriting recognition.In: Brazilian symposium on neural networks (SBRN), IEEE, ISBN 978-1-4673-2641-4, pp 113–118
10. Stamatopoulos N, Gatos B, Louloudis G, Pal U (2013) Handwriting segmentation contest. In: 12th international conference on document analysis and recognition (ICDAR), IEEE, ISSN 1520-5363, pp 1402–1406
11. Pradeep J, Srinivasan E, Himavathi S (2012) Performance analysis of hybrid feature extraction technique for recognizing English handwritten characters. In: World Congress on Information and Communication Technologies (WICT), IEEE, ISBN 978-1-4673-4806-5, pp 373–377
12. Yuan A, Bai G, Yang P, Guo Y (2012) Handwritten english word recognition based on convolution neural networks. In: International conference on frontiers in handwriting recognition (ICFHR), IEEE, ISBN 978-1-4673-2262-1, pp 207–212
13. Gilliam T, Wilson RC, Clark John A (2011) Segmentation and normalisation in grapheme codebooks. In: International conference on document analysis and recognition (ICDAR), IEEE, ISBN 978-1-4577-1350-7, pp 613–61
14. Ryu J, Koo HI, Cho NI (2015) Word segmentation method for handwritten documents based on structured learning. IEEE Trans Signal Process Lett 22(8)
15. Sharma MK, Dhaka VP (2015) An efficient segmentation technique for Devanagari offline handwritten scripts using the Feed-forward Neural Network. Neural Comput Appl. doi:10.1007/s00521-015-1844-9
16. Dhaka VP, Sharma MK (2015) Classification of image using a genetic general neural decision treeInt. J Appl Pattern Recogn 21:76–95
17. Obaidullah SM, Mondal A, Roy K (2014) Structural feature-based approach for script identification from printed Indian document. In: IEEE international conference on signal processing and integrated networks (SPIN), pp 120–124
18. Zhu G (2009) Language and Media Process. Laboratory, University of Maryland, College Park, MD, USA. In: Zheng Y, Doermann D, Jaeger S (eds) Signature detection and matching for document image retrieval, IEEE transaction on pattern analysis and machine intelligence, 31(11): 2015–2031
19. You J (2003) Department of Computer, Hong Kong Polytechnic University, Kowloon. In: Zhang D, Cao J, Guo M (eds) Parallel biometrics computing using mobile agents, IEEE international conference on parallel processing, pp 305–312
20. Pervouchine V (2005) School of Computer Engineering, Nanyang Technology University, Singapore. In: Leedham G (ed) Document examiner feature extraction: thinned vs. skeletonised handwriting images, TENCON 2005 IEEE Region 10, pp 1–6
21. Tomai CI (2004) Department of computer science and engineering, center of excellence for document analysis and recognition, Amherst, MA, USA. In: Zhang B, Srihari SN (eds) Discriminatory power of handwritten words for writer recognition", Proceedings of the 17th International Conference on Pattern Recognition, ICPR 2004, vol 2, pp 638–641
22. Jiang Y, Ding X, Fu Q, Ren Z (2006) Context driven chinese string segmentation and recognition. In: Proceedings structural, syntactic, and statistical pattern recognition: joint IAPR Int'l Workshops, pp 127–135
23. Gatos B, Louloudis G, Pratikakis I, Halatsis C (2009) Text line and word segmentation ofhandwritten documents. Pattern Recogn 42(12):3169–3183
24. Stafylakis T, Papavassiliou V, Katsouros V, Carayann G (2010) Handwritten document image segmentation into text lines and words. Pattern Recogn 43(1):369–377
25. Sharma MK, Dhaka VP (2015) Pixel plot and trace based segmentation method for bilingual handwritten scripts using feed-forward neural network. Neural Comput Appl. doi:10.1007/s00521-015-1972-2