International Journal of Computer Theory and Engineering, Vol. 2, No. 1 February, 2010
1793-8201

# Neighborhood Pixels Weights-A New Feature Extractor

Satish Kumar

***Abstract*—In OCR applications, the feature extraction methods used to recognize document images play an important role. The feature extraction methods may be statistical, structural or transforms and series expansion. The structural features are very difficult to extract particularly in handwritten applications. The structural behavior of the strokes existing in the handwritten expressions can be estimated through statistical methods too. In this paper, a feature extraction method is proposed that measures the distribution of black and white pixels representing various strokes in a character image by computing the weights on all the four corners on a pixel due to its neighboring black pixels. The feature is named as Neighborhood Pixels Weights (NPW). Its recognition performance is compared with some feature extraction methods, which have been generally used as secondary feature extraction methods for the recognition of many scripts in literature, on noisy and non-noisy handwritten character images. The experiments have been conducted using 17000 Devanagari handwritten character images. The experiments have been made using two classifiers i.e. Probabilistic Neural Network and *k*-Nearest Neighbor Classifier. NPW feature is better as compared to other features, studied here, in noisy and noise-less situation.**

***Index Terms*—Devanagari script, Hand-printed recognition, NPW (Neighborhood pixels weights), Random noise, Statistical features, Weighted map.**

## I. Introduction

An OCR or ICR works in various phases such as: scanning, pre-processing, feature extraction, classification and post-processing. The feature extraction phase is quite important since a set of useful properties of a graphical symbol/expression/character available as an image is defined and extracted during this phase. On the basis of these properties, the given graphically expressed character/symbol is categorized/ assigned a label. These properties are quite valuable for taking classification decision and are known as features in pattern recognition terminology. The feature extraction methods used must be robust for expressing the properties of a character/script under consideration. If there is a slight variation in a character image either due to printing/writing or due to instrument used, the features used must be able to absorb the same. There are many feature extraction methods available in literature. Govindan et al [1] classified the various feature extraction methods into three categories i.e. statistical, structural, and global transforms and series expansion. A character image is composed of black and white pixels having two levels either 0 or 255 in binary format or gray colors having a range of 256 colors between 0 and 255 in gray format. The images may be colored but in case of document images only binary or gray formats are mostly considered.

The structural features are based on the geometrical and topological properties of a character under consideration and these properties may be local or global [1]. A character is composed of number of components in the form of strokes. These strokes may be lines, arcs, curves, etc and may or may not be connected to each other depending upon the structure of a character. These components are also called as stroke primitives and can be extracted from either skeleton or contour of a character image. In structural based recognition process, the various stroke primitives of a character are extracted and approximated. The relationships between various stroke components are established. It is somewhat difficult to extract and approximate the various stroke primitives existing in a character image as in some cases the strokes may not touch where touching is required and strokes may unnecessarily touch where touching is not required in the basic structure of a character while printing or writing. This approach also requires matching an approximated stroke primitive with stored prototypes which is not only complex to model but also requires multi-level heuristics. Moreover, these features are extracted from binary images only.

The problems faced with structural features can be easily overcome with statistical features which are based on statistical distribution of black and white pixels in a character image. These features may be extracted from binary or gray scale images and are invariant to character distortions and writing styles to some extent. The features are easy to extract and can be computed with high speed as at a given pixel only some arithmetic or logic operations are required to perform which take less computational time and are not difficult to compute. The various authors have used different statistical methods to estimate the presence or absence of a stroke primitive in the given region of a character image. Some most commonly used statistical features for character recognition are: zoning, moments, projection histograms, crossings, character loci and *n*-tuple. The topological or geometrical features are also extracted from a skelotonized character image. A thinning process is not only computationally expensive but also losses some important parts of some character images. The character images, particularly the hand-printed ones, have

Dr. Satish Kumar is with Panjab University and is serving at its Regional Centre, Muktsar, Punjab, India ; email: satishnotra@yahoo.co.in;

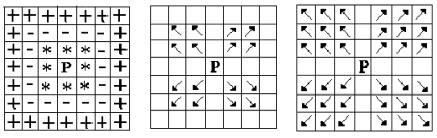
certain degree of blur that leads to the loss of important information on their blur parts. In some cases, some parts of a character image is merged together which also leads to loss of important information. Some authors have also used binary feature vector obtained by using topological or geometrical features [10] , [11]. In this case, a character image is divided into number of regions and the presence or absence of a particular feature in a region is detected and represented using binary features. This representation contributes to large variation in feature position and creates discontinuity in feature space [13].

In this paper, some of these feature extraction methods are being studied (theoretically) briefly along with their recognition performance on noisy and non-noisy situation (practically). The paper only gives the performance of proposed feature against some features in literature using two classifiers *i.e. k*-NN (*k*-Nearest Neighbor) and PNN (Probabilistic Neural Network). The features covered are zoning, profiles, histograms, crossings and Kirsch directional edges. A new proposed statistical feature is also covered in detail and its performance is compared with the above said features which are mostly used as secondary features to enhance the recognition performance of various performing features. Our newly proposed feature may work on binary as well as on gray images. The details of newly proposed feature are given in section *II*. Section *III* covers some features from literature for making comparison with proposed feature. A little description of the classifiers used for conducting study is made in Section *IV*. Experimental results are given in Section *V* in detail. The analysis of results is given in Section *VI*. The study work is concluded in Section *VII*.

## II. Proposed Feature In Detail

### A. Neighborhood Pixels Weight

In this feature, the weights of neighboring pixels of a pixel are considered for feature computation. The neighboring pixels which are presented on any one of the four corners to a pixel have been considered in study. A pixel has 8, 16 and 24 neighboring pixels if we consider first level, second level and third level neighboring layers respectively. These neighborhood levels considered are shown in Fig. 1(a). The pixels marked as '*', '- and '+' belong to first, second and third level neighborhood layers respectively. If we consider all the three levels then on each corner 5 pixels from third level layer, 3 pixels from second level layer and one pixel from first level layer are considered for contributing their weights to a pixel P.



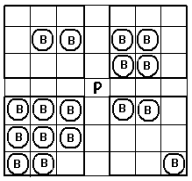

(a) (b) (c)

Fig. 1 (a) Neighborhood pixels of all the three level neighborhood layers, (b) Neighboring pixels of a pixel *P*, on four corners, due to first and second level neighborhood layers, and (c) Neighboring pixels of a pixel *P*, on four corners, due to first, second and third level neighborhood layers.

We have performed our experiments by considering the pixels of two level as well as all the three level neighborhood layers to a pixel P. If we consider two levels then on each corner 3 pixels from second level layer and one pixel from first level layer are considered for contributing their weights to a pixel *P*. Consider the case of entire three level neighborhood layer based feature extraction method. In Fig. 1(c), the weights of the pixels marked with an arrow are only considered. In our experiments, the weights on a pixel due to black pixels on all the four corners are considered. At a given pixel the feature is encoded using 4-cell array. A cell contains the weight on a pixel corresponding to its neighborhood pixels on any one of the four corners. Suppose any wants to encode the feature of a pixel P given in Fig. 2(a). The weights at this pixel corresponding to all the four corners are given in Fig. 2(b). Here encircled 'B' represents black pixel. Initially, all the four cells corresponding to a pixel P are zero. Consider the case of top-left corner pixels where only two pixels are black. The weight of this cell is obtained by summing weights of (weight of each black pixel is 255 and of white pixel is 0) all the nine pixels *i.e.* 2×255=510 divided by maximum possible weight due to all the nine pixels *i.e.* 9×255=2295 which is 0.22. Similarly the weights of bottom-left, bottom-right and top-right cells are 0.89, 0.33 and 0.44 respectively.

NPW feature can be extracted from gray images too. Suppose the gray color pixels surrounding a pixel P as given in Fig. 3 (a). The weight on a pixel P due the right-top corner pixels is obtained by summing the gray level of all the nine pixels *i.e.* 815 divided by maximum possible weight due to all the nine pixels *i.e.* 9×255=2295 which is 0.36. Similarly the weights of bottom-left, bottom-right and top-right cells are 0.79, 0.29 and 0.21 respectively.

In Fig. 1, only the pixels at corner are considered and the pixels along row/column of a pixel P are not considered as it has been observed from the number of experiments that the inclusion of these pixels does not affect the accuracy much. So it is better not to consider these pixels for feature computation. It also reduces the number of neighborhood pixels for feature computation. Here, the experiments have been conducted on binary images only. A weight map (WM) corresponding to all the pixels for a given image is prepared. The weight map consists of four planes, each having size same as the size of normalized character image.

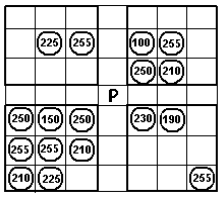



(a)

| Corner | Top Right | Top Left | Bottom Left | Bottom Right |
|---|---|---|---|---|
| Weights | 0.44 | 0.22 | 0.89 | 0.33 |

(b)

Fig. 2 (a) Neighboring black pixels of a pixel *P* due to first, second and third level neighboring layers, and (b) Weights due to neighboring pixels on four corners.




(a)

| Corner | Top Right | Top Left | Bottom Left | Bottom Right |
|---|---|---|---|---|
| Weights | 0.36 | 0.21 | 0.79 | 0.29 |

(*b*)

Fig. 3 (*a*) Neighboring gray pixels of a pixel *P* due to first, second and third level neighboring layers, and (b) Weights due to neighboring pixels on four corners.

Each plane is due to neighborhood pixels weights along a particular corner for all the pixels in an image. The four WM planes for the binary image given in Fig. 4(*a*), obtained by considering all the three levels are given Fig. 4(*b-e*) and the four WM planes for the binary image given in Fig. 4(*a*), obtained by considering only two level neighborhood pixels are given Fig. 4 (*f-i*) To extract feature vector, each plane is divided into X ×Y regions and average weight in each region is computed. For our experiments, each WM plane is divided into 5 ×5 regions. The four features of top-left corner regions (enclosed inside boxes) due to all the four WM planes given in Fig. 4(*b-e*) are given in Table 1. Fig. 4 is labeled as (*a-i*) from top to bottom and left to right.

As already mentioned that the NPW feature can be computed or used by taking two level neighborhood layer pixels as well as three level neighborhood layer pixels. Both ways have been used to conduct experiments to know if there is any enhancement in discrimination ability of NPW feature by including all the three levels over two levels. The results are given Tables 2 and 3, rows (1-2) in each. For two level the feature name is NPW2 (Neighborhood Pixels Weights-2) and three level the feature name is NPW3 (Neighborhood Pixels Weights-3).

The weight maps shown in Fig. 4 contain only integer values for weights. In actual, these values will be real. In Fig. (*b-e*), the total black pixels out of 9 pixels and in Fig. (*f-i*), the total black pixels out of 4 pixels have been taken.

In WM feature, the relationship of a pixel with its neighboring pixels is used to compute the hidden properties pertaining to a character image which are very useful for classification purpose.

TABLE 1

AVERAGE WEIGHT FOR TOP-LEFT CORNER REGION ON ALL THE FOUR WM PLANES WITH THREE LEYER NEIGHBORHOOD

| Features | 0.67 | 0.0 | 0.0 | 1.08 |
|---|---|---|---|---|
| Corner | Top Right | Top Left | Bottom Left | Bottom Right |

## III. Features From Literature

### *A. Profiles*

The profiles based features are motivated for recognition of hand-printed numerals by Shridhar et al [12], where authors used left and right profiles only. The use of profiles as complementary feature for recognition of hand-printed pattern is carried by many authors and some authors are: Heutte et al [13], Liu et al [14] and Koerich [15]. Profiles extract the structural information of outer contour and do not provide any information about the interior structure of a character image such as loops, number of strokes, etc. Many features can be computed from profiles. In order to find profiles, an image is projected from outside and the distance of its outer contour from the boundary of the bounding box of a character image, which is either square or rectangle, is obtained. The left and right profiles of character image, given in Fig. 4(a), at row no. 7 are 14 and 5 respectively and the top and bottom profiles at column no. 7 are 2 and 17 respectively. Taking profiles according to all rows and columns for a given image, for classification purpose, unnecessarily contributes to the size of feature vector. So the profiles may be taken at some selective rows and columns.

### *B. Histograms*

Glauberman [16] used projection histograms first time in a hardware based OCR. This technique is generally used to detect orientation in a document page or segmenting a page into lines, words and characters. In order to find the projection histograms, an image is tracked along a path from a side and the number of black pixels in that path is counted. A histogram gives the width of character strokes along a particular path (either row or column). The vertical histograms are slant invariant whereas horizontal histograms are not [17]. The histograms only give the stroke information along the given path and do not cover any other properties such as number of strokes along a path, width of each stroke, location of each stroke, etc. The horizontal histograms of character image, given in Fig. 4(a) , at row no. 4 and 9 are 10 and 9 respectively and vertical histograms at column no. 4 and 9 are 2 and 5 respectively.

### *C. Crossings*

Crossings based methods have been used in [6], [18] for hand-printed character recognition and are generally used to detect the number of strokes presented in a character along a particular path. If this path is along the rows, then it is called horizontal crossings and if this path is along the columns, then it is called vertical crossings. The crossings can be considered as the number of transitions either from black to white or from white to black pixels along a particular path. Crossings can be extracted from an original character as well as from its

skeleton. Kim et al [6] used crossings in raw form but Arica et al [18] used median of the black pixel runs in each scan line. The region in which the median of a black pixel run falls is assign its code value otherwise the code value for the region is 0. The sum of codes due to all regions in a scan line is taken as feature.

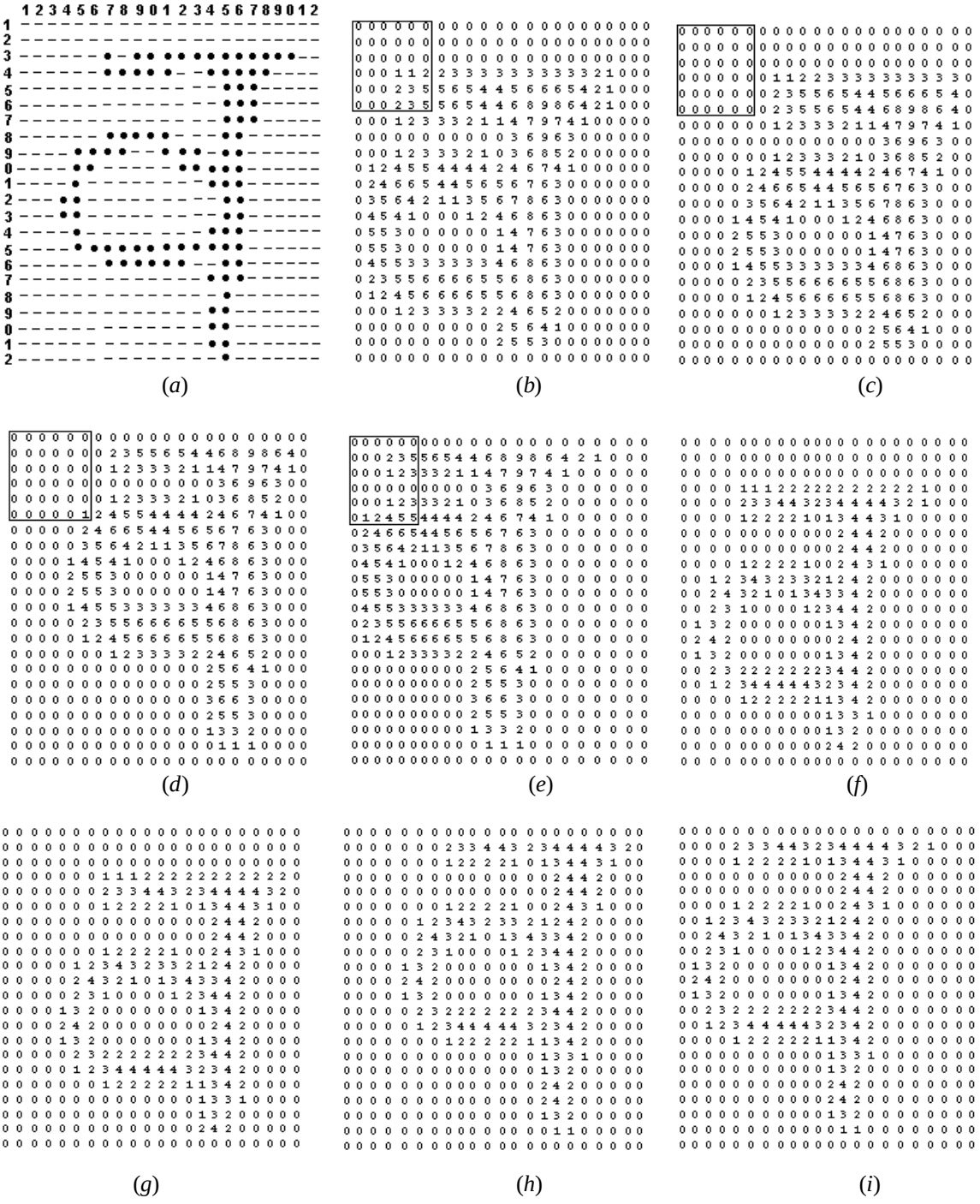


Fig. 4 (a) Binary image 22 ×22 pixels size, (*b-e*) Four WM planes for image *4(a)* computed by considering the pixels of all the three neighborhood layers, (*b-e*) Four WM planes for image *4(a)* computed by considering the pixels of two neighborhood layers only.

The crossings of character image, given in Fig. 4(a), at row no. 4 and 9 are 2 and 3 respectively whereas column no. 4 and 9 are 1 and 3 respectively.

*D. Zoning*

Zoning feature extraction method was implemented in Calera OCR system used to recognize machine-printed non-decorative fonts. Bosker [3] gives some details about zoning method used in Calera. Cao et al [4] used zoning on numeral contours where the images are divided into 4×4 zones. Zoning can be implemented on various forms of a character image such as original (solid character), character contour and character skeleton. In addition, this method can be used to recognize both gray as well as binary images. It is used to capture local properties of a character. The various authors have implemented zoning. In this feature extraction method, a character image (character bitmap / character bounding box) is mostly divided into zones. Mostly, non-overlapping zones are taken but Cao et al [4] divided the character bitmap into

overlapping zones. In this method a character image is divided into $X \times Y$ regions, each having $a \times b$ pixels size. In its simplest implementation the percentage density of black pixels in each zone is computed. Zoning is not slant invariant.

*E. Kirsch Directional edges*

Kirsch defines a non-linear edge enhancement algorithm as follows [8],[24]:

$$G(x, y) = \max\{1, \max_{i=0}^{7}[|5S_i - 3T_i|]\} \quad (1)$$

where $S_i$ and $T_i$ are defined as:

$$S_i = A_i + A_{i+1} + A_{i+2} \quad (2)$$

$$T_i = A_{i+3} + A_{i+4} + A_{i+5} + A_{i+6} + A_{i+7} \quad (3)$$

The subscript $i$ of pixels $A_i$ is taken as modulo 8 and $A_i$ ($i$ = 0, 1, 2,…,7) is eight neighbors of pixel ($x$, $y$) starting from top-left pixel and moving clock-wise. The edge strength at a pixel ($x$, $y$) along horizontal ($H$), vertical ($V$), left-diagonal ($L$) and right-diagonal($R$) is calculated from an image as follows:

$$G_H(x,y) = max(|5S_0 - 3T_0|, |5S_4 - 3T_4|) \quad (4)$$

$$G_V(x,y) = max(|5S_2 - 3T_2|, |5S_6 - 3T_6|) \quad (5)$$

$$G_R(x,y) = max(|5S_1 - 3T_1|, |5S_5 - 3T_5|) \quad (6)$$

$$G_L(x,y) = max(|5S_3 - 3T_3|, |5S_7 - 3T_7|) \quad (7)$$

The directional edges for an image at each pixel are computed using (4-7) and these give four sub-images according to four edge levels. Four directional sub-images for a Devanagari character, given in Fig. 4(a) normalized to 70×70 pixels size, due to four directional edges (last four) are given in Fig. 5.

Some authors who used/studied this feature for handwritten recognition are: Wen[5], Kim[6] , Knerr et al[7] and Chao[9].

## IV. Classification Methods

The various experiments have been conducted in this paper on handwritten characters using two classifiers *i.e.* Probabilistic Neural Networks (PNN) and *k* nearest neighbor (*k*-NN). The characters belong to Devanagari script. For classification purpose the standard codes available in MATLAB [22] have been used.

Fig. 5 Four sub-images (right side) obtained using four Kirsch operators (4-7) and fifth sub-image (left side) obtained due to original image, Fig. 4(a), after normalizing its size to 70×70 pixels.

*A. k-Nearest Neighbors Classifier*

*k* nearest neighbor classifier is one among the instance-based methods and it is also called as lazy algorithm. In *k*-nearest neighbors (*k*-NN), the posteriori probability of occurrence of unknown pattern is predicted on the basis of frequency of its nearest *k*-neighbors in a given training sample set. Computation time to test a pattern in *k*-NN depends upon the number of training samples *m* and the size of feature vector *n* which is O(*n*×*m*). The performance of a classifier further depends upon the value of *k*, the size of training data set, the metric distance used to measure the distance between a test sample and the training samples and the mode of decision (majority rule, weighted decision etc.). Various efforts have been made to improve the efficiency of *k*-NN classifier in respect of computational speed and classification accuracy. Liu et al [19] gives some references about such efforts. Despite its lazy behavior, it has been successfully used for character recognition problems.

*1) k-Nearest Neighbors Rule*

To predict the class of a new unlabeled sample *u*, *k* samples are searched from the training sample set, which are closest to *u* and assign *u* the label of samples that appears most frequently out of *k* samples. In another way, assign *u* the label of samples that appears in majority out of *k* nearest samples. This is also called as majority rule.

There may be some cases where the value of *k* is even or the value of *k*>2 and all the *k* samples belong to different classes leading to ambiguity. In such cases it is essential to break a tie in the number of neighbors. A random and nearest tiebreaker is taken which uses the nearest neighbor among tied groups to break the tie in conflicting situation. There are many methods; those can be used as a metric for computing the distance between training and a test sample. Here Euclidean distance has been used.

*B. Probabilistic Neural Network*

Bayes classifier assigns a pattern *u* to a class whose decision function $\Psi_j(u)$ , which is given below, yields maximum value [20] (*i.e.* maximum posterior probability)

$$\Psi_j(u) = p(u/\omega_j)P(\omega_j) \quad j = 1,2,3,\ldots,q; \quad (8)$$

Probabilistic neural network (PNN) has been designed for pattern recognition problems and is based on the Bayesian classification theory [21]. The PNN architecture is 4-layered which consists of two hidden layers in addition to one input

and output layer each. The input layer is used to upload the input pattern and its size is same as the dimension of the input pattern. First hidden layer is fully connected to the input layer. This layer is called as pattern layer and its size (number of neurons or nodes) is equal to the number of input patterns *i.e.* there is one neuron for each pattern. This is a layer of radial basis neurons in which each neuron acts as a detector for a different input pattern. Second hidden layer consists of one summation node for each class to realize summation. Final layer is decisive. It performs the comparison between the outputs appearing on the summation nodes. Since the outputs of the summation nodes are the estimated probability density for various classes under consideration, the output of the final layer is corresponding to the class having largest probability. In this way the final output of the PNN is corresponding to the Bayes rule (8).

One advantage of PNN is its very small training time as compared to other neural network based classifiers. In classification phase it has to combine all the training patterns to yield require estimate, so it is slow in classification phase. It may be difficult to implement in those applications where there is a need of large size of training data samples since the size of the pattern layer is same as the size of the training data set. It is important in those pattern recognition applications where learning is more important than generalization.

The standard implementation of this classifier has been used for conducting experiments [22] and the readers are suggested to consult the help of same. The performance of the classifier depends upon the number of samples and the spread factor mentioned as *Sp* here. If *Sp* is near zero the network will act as a nearest neighbor classifier. As *Sp* becomes larger the designed network will take into account several nearby design vectors. In this paper, the experiments have been conducted for different values of *Sp* by taking fixed size of training set.

## V. Experimental Results

A number of experiments have been conducted to know the performance of newly developed feature. Experiments are also conducted to know its performance in comparison with some features available in literature. Devanagari handwritten character database have been used for conducting experiments. Though the size of our database is large but we have used 400 characters per class for 43 Devanagari alphabet letters (classes) only. Out of these first 100 characters from each class have been used for testing purpose and remaining 300 characters from each class have been used for training purpose. Both noise-free and pseudo-noisy images are taken for the purpose. The classifiers used are *k*-NN and PNN. The feature extraction methods have been computed using java. Images have been size normalized to 45×45 pixels. Before performing recognition we suggest headline removal from each Devanagari handwritten character as with its removal the recognition rate is enhanced a lot [23]. The algorithm used for this purpose is not described here and is beyond the scope of this paper. To know the effect on features in noisy situation, a random white noise which is about 15% to the normalized image size is added to the images. The features have been extracted using all methods from the images once created by adding random white noise. Some hand-printed normalized Devanagari character images before and after noise are given in Fig. 6.

### *A. Feature Deduction Strategy*

In Section *III*, the various features have been discussed to be considered for examining against proposed feature. In this Subsection, the strategy used to extraction features to make comparison is discussed. Since the size of normalized image used is 45×45 pixels.

Fig. 6 Normalized Devanagari hand-printed characters before and after random noise introduction

The profiles are computed at 30 locations on each side *i.e.* left, right, top and bottom and the size of feature vector used is 30×4=120. Each feature component is divided either by width or height of character bitmap depending upon the side of image projected. The feature is named as Prof-120. We have considered the histograms according to all rows and columns. The size of feature is 45×2=90 and is named as His-90. The feature vector is normalized by finding the maximum value of feature component out of 90 feature components so computed. The same is situation with crossings and its name is Cros-90.

In case of Zoning, the given normalized character image is divided into 10×10 regions and percentage density of black pixels in each region is computed. The feature is named as Zon-100. In case of Kirsch directional Edges, each sub-image (Horizontal, vertical, left-diagonal and right diagonal) is divided into 5×5 regions and frequency of occurrence of pixels corresponding to each edge level in each region is computed and feature vector is normalized by the maximum value of frequency obtained out of all regions and the feature is named as Kir-100. In case of NPW, each weighted map (WM) is divided into 5×5 regions and average neighborhood pixels weight is computed in each region. The feature vector is normalized by dividing the each feature component with the maximum value of average neighborhood pixels weight out of all the regions. The term NPW2 is used where two level neighborhoods are used and NPW3 is used where three level neighborhoods are used.

### *B. Results with k-NN AND PNN classifiers*

The experimental results on noise-free and noisy Devanagari handwritten character images using *k*-NN as classifiers are given in Table 2. The experiments have been conducted on different values of *k*. Only odd values have been

considered for conducting experiments in order to avoid conflict arisen due to majority rule. The experimental results with *k*-NN at different values of *k* are reported in Table 2. The Euclidean distance is used as metric. At $k$ =1, the results are low as compared to the results listed in Table 2. As we have already mentioned that the output of PNN fully depends on the Spread factor *i.e. Sp*. The recognition rate with PNN classifier on various features under study at various values of *Sp* is given in Table 3.

## VI. Result Analysis And Discussion

The role of both classifiers on the recognition performance of various features is observed quite different. The effect is not uniform on all the features. On some features the recognition rate is high with PNN classifier whereas in case of others it is high with *k*-NN. The analysis of recognition rate of various features with PNN and *k*-NN classifiers in noisy and noise-less situation is given in Fig. 7.

The crossings feature is least performing in respect of recognition performance with both classifiers in noisy as well as noise-less situation. NPW3 (Neighborhood pixels weights by considering the pixels of all the three neighborhood layers) feature is performing far better as compared to other features studied here. However, the difference in recognition performance of two variations of NPW feature *i.e.* NPW2 and NPW3 is not much large and both are performing better as compared to others. The difference in these two variations is only in computational time as in case of NPW2 only 16 pixels needs to be check at each pixel whereas in case of NPW3 36 pixels needs to be check at each pixel for weight calculation. The difference in error rate in noisy and noise-less situation with *k*-NN and PNN has been analyzed in Fig. 8. The difference in error is very small in case of NPW3 and PNN and quite large in case of Histograms and PNN. It means, the performance of NPW3 is not affected much if the images are noisy and the performance of Histograms affected a lot if images are noisy with PNN classifier. Some effect of noise is also observed in case of NPW2 feature. From the above discussion we can say that newly developed feature NPW3 is robust in noisy as well as noise-less situation as compared to other features studied here.

TABLE 2

RECOGNITITON RESULTS WITH VARIOUS FEATURES USING *K*-NN CLASSIFIER AT DIFFERENT VALUES OF *K* ON DEVANAGARI HAND-PRINTED CHARACTER IMAGES

| | Devanagari Handwritten Character Samples | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Noisy-less | | | | | Noisy | | | | |
| No of Nearest Neighbor (*k*) | 3 | 5 | 7 | 9 | 11 | 3 | 5 | 7 | 9 | 11 |
| Type | | | | | | | | | | |
| NPW2-100 | 84.2 | 85.0 | **85.1** | 85.1 | 84.9 | 83.4 | 84.2 | **84.7** | 84.1 | 83.7 |
| NPW3-100 | 86.2 | 86.3 | 86.6 | **86.8** | 86.8 | 85.7 | 86.4 | **86.7** | 86.4 | 86.2 |
| His-90 | 57.4 | 59.1 | 61.0 | 61.2 | **68.3** | 56.7 | 58.3 | 59.1 | 59.8 | **59.9** |
| Prof-120 | 67.6 | 69.1 | 69.2 | 69.0 | **69.3** | 65.9 | 67.3 | 68.2 | 68.2 | **68.3** |
| Cros-90 | 54.4 | 56.9 | 58.0 | 58.4 | **59.0** | 42.5 | 47.9 | 50.3 | 51.9 | **53.0** |
| Kir-100 | 72.0 | 74.0 | 74.6 | 75.3 | **75.3** | 61.3 | 64.0 | 65.5 | **66.0** | 65.9 |
| Zon-100 | 79.7 | 80.3 | **80.3** | 79.1 | 79.7 | 78.7 | **79.7** | 79.2 | 79.5 | 79.4 |

TABLE 3

RECOGNTITON RESULTS WITH VARIOUS FEATURES USING PNN CLASSIFIER AT DIFFERENT VALUES OF SPREAD (*Sp*) ON DEVANAGARI HAND-PRINTED CHARACTER IMAGES.

| Feature Type | Devanagari Handwritten Character Samples | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Noisy-less | | | | | | Noisy | | | | | |
| | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 | 0.7 |
| NPW2-100 | 83.0 | 83.6 | 84.3 | 85.4 | **85.6** | 84.9 | 82.1 | 82.6 | 84.4 | 84.4 | **84.5** | 84.0 |
| NPW3-100 | 84.7 | 85.2 | 86.1 | 87.0 | **87.2** | 86.2 | 84.7 | 85.0 | 86.0 | 86.8 | **86.9** | 85.7 |
| His-90 | 56.4 | **60.7** | 59.6 | 55.5 | 51.5 | 47.7 | 40.4 | 43.6 | 53.2 | **55.0** | 55.0 | 51.2 |
| Prof-120 | 64.9 | 65.7 | 67.4 | 69.6 | **70.1** | 69.4 | 63.6 | 64.4 | 65.9 | 67.9 | **68.6** | 67.2 |
| Cros-90 | 53.1 | 54.6 | 58.1 | **59.2** | 57.3 | 54.1 | 40.4 | 43.6 | 49.8 | **55.0** | 55.0 | 51.2 |
| Kir-100 | 70.7 | 73.5 | **76.0** | 75.5 | 73.1 | 70.7 | 58.9 | 62.9 | **69.1** | 68.1 | 66.5 | 66.3 |
| Zon-100 | 78.2 | 78.3 | 78.8 | 79.7 | 80.2 | **80.6** | 76.2 | 76.8 | 77.9 | 78.9 | **79.8** | 78.7 |

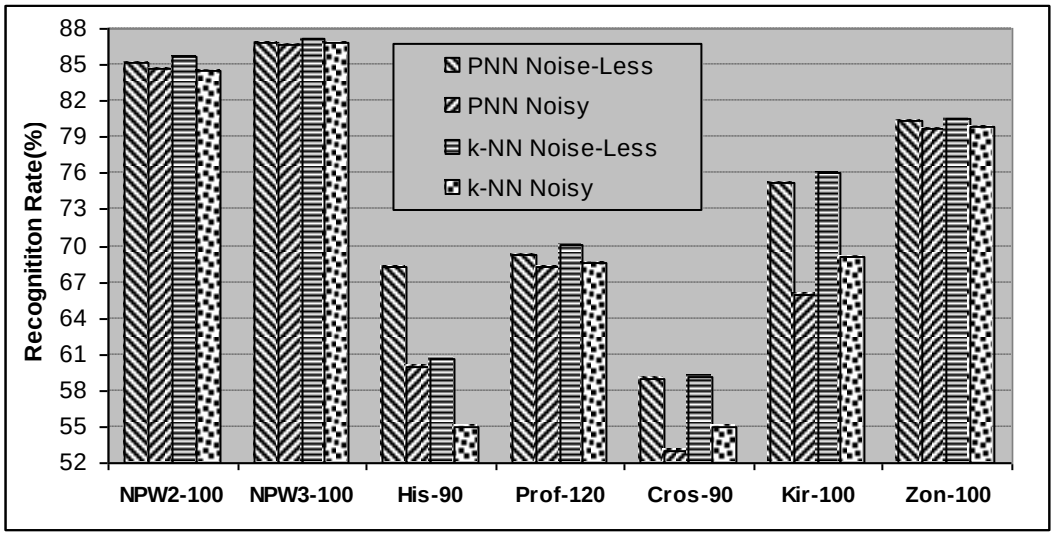



Fig. 7. The analysis of recognition rate of various features with PNN and *k*-NN classifiers in noisy and noise-less situation.

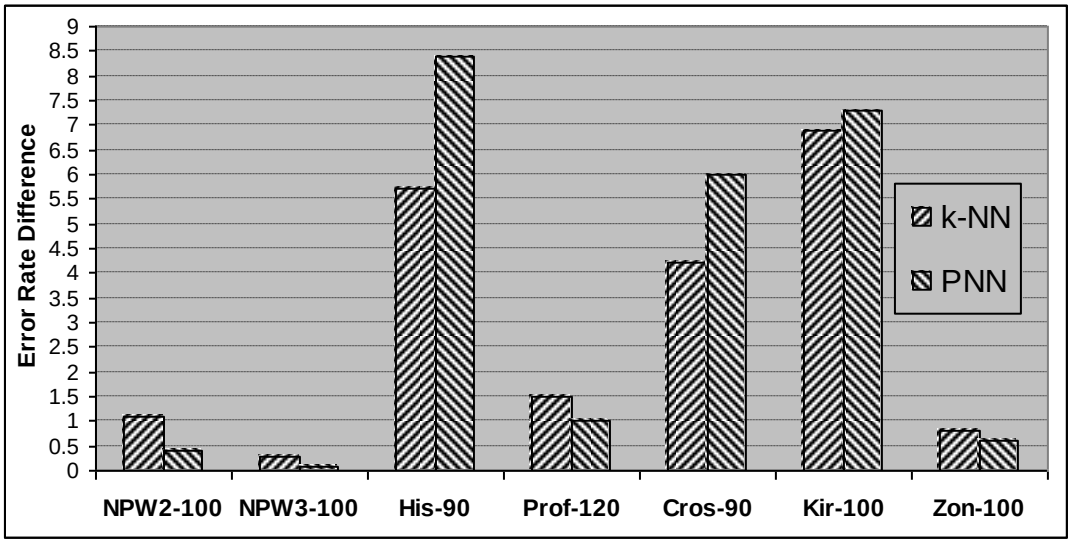



Fig. 8: Error Rate difference in noisy and noise-less situation with *k*-NN and PNN

As the various features studied here except NPW have been mostly used as complementary features for recognition of various scripts but we suggest to use NPW feature either as primary feature or secondary feature (supporting) for document level recognition.

## VII. Conclusion

In this paper a feature extraction method is proposed that measures the distribution of black and white pixels representing the stroke(s) in a character image by computing the weights on all the four corners on a pixel due to its surrounding black pixels. The feature is named as NPW (Neighborhood Pixels weights) as it is based on the weights on a pixel due to neighboring pixels. It can be computed from binary as well as gray images. We have tested its performance against some features, which have been used as supporting features in many OCR/ICR applications, on noisy as well as noise-less situation. NPW feature is better than other features studied here. This feature is also robust in noisy situation as its recognition rate is not dropped much as compared to other features. This feature may be used as primary or secondary feature for pattern recognition applications.

## REFERENCES

[1] V. K. Govindan and A. P. Shivaprasad, "Character Recognition – a Review", Pattern Recognition, 1990, vol. 23, no. 7, pp. 671-683.

[2] N. Arica and T. Yarman-Vural, "An Overview of Character Recognition Focused on Off-line Handwriting", IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews, 2002, vol. 31, no. 2, pp. 216-233.

[3] M. Bosker, "Omnidocument Technologies", Proceedings of the IEEE, 1992, vol. 80, no. 7.

[4] J. Cao, M. Ahmadi and M. Shridhar, Recognition of Handwritten Numerals with Multiple Feature and Multistage Classifier, Pattern Recognition, 1995, vol. 28, no. 2, pp. 153-160.

[5] Y. Wen, Y. Lu and P. Shi, "Handwritten Bangla Numeral Recognition System and its Application to Postal Automation", Pattern Recognition, 2007, vol. 40, pp. 99-107.

[6] K. M. Kim, J.J. Park, Y.G. Song, I. C. Kim and C. Y. Suen, "Recognition of Handwritten Numerals Using a Combined Classifier with Hybrid Features", SSPR & SPR, LNCS 3138,2004, pp. 992-1000.

[7] S. Knerr, L. Personnaz and G. Dreyfus, "Handwritten Digit Recognition by Neural Networks with Single-Layer Training", IEEE Transactions on Neural Networks, 1992, vol. 3, no. 6, pp. 962-968.

[8] W. K. Pratt, "Digital Image Processing", Third Edition, Wiley, New York , 2001.

[9] S.-B. Cho, "Neural-Network Classifiers for Recognizing Totally Unconstrained Handwritten Numerals", IEEE Transactions on Neural Networks, 1997, vol. 4, no. 1, pp. 43-53.

[10] H. S. Baird, "Feature Identification for Hybrid Structural/Statistical Pattern Classification", Computer Vision, Graphics and Image Processing, 1988, vol. 42, pp. 318-333.

[11] K. Anisimovich, V. Rybkin, A. Shamis and V. Tereshchenko , "Using Combination of Structural, Feature and Raster Classifiers for Recognition of Hand-printed Characters", Proceedings of Fourth International Conference of Document Analysis and Recognition, Ulm, Germany, 1997, pp. 881-885.

[12] M. Shridhar and A. Badreldin, "Recognition of Isolated and Connected Handwritten Numerals", Proceedings of the IEEE International Conference on Systems, Man and Cybernetics,1984, pp. 142-146.

[13] L. Heutte, T. Paquet, J. Moreau, Y. Lecourtier and C. Olivier, "A Sturctural / Statistical Feature Based Vector for Handwritten Character Recognition", Pattern Recognition Letter, 1998, vol. 19, pp. 629-641.

[14] C.- L. Liu, K. Nakashima, H. Sako and H. Fujisawa, "Handwritten Digit Recognition: Benchmarking of State-of-the-Art", Pattern Recognition, 2003, no. 36, pp. 2271-2285.

[15] L. Koerich, "Large Vocabulary Off-line Handwritten Word Recognition", Ph. D. Thesis, École de Technologie Supérieure, Montreal-Canada,2004.

[16] M. H. Glauberman, "Character Recognition for Business Machines", Electronics, 1996, pp. 132-136.

[17] O. D. Trier, A. K. Jain and T. Taxt, "Feature Extraction Method for Character Recognition – a Survey", Pattern Recognition, 1996, vol. 29, no. 4, pp. 641-662.

[18] N. Arica and F. T. Yarman-Vural, "Optical Character Recognition for Cursive Handwriting", IEEE Transactions on Pattern Analysis and Machine Intelligence,2002, vol. 24, no. 6.

[19] C.- L. Liu and M. Nakagawa, "Evaluation of Prototype Learning Algorithms for Nearest- Neighbor Classifier in Application to Handwritten Character Recognition", Pattern Recognition, 2001, vol. 34, pp. 601-615.

[20] R. C. Gonzalez and R. E. Woods, "Digital Image Processing", 3rd Ed., Pearson Education.

[21] P. D. Wasserman, "Advanced Methods in Neural Computing", New York: Van Nostrand Reinhold, 1993, pp. 35-55.

[22] Neural Network Toolbox, version 5.0.2 and Bioinformatics Toolbox, version 2.6, The Math Works, Inc.

[23] Satish Kumar, "The Headline Removal Algorithm and its Effect on Recognition of Devanagari Handwritten Characters", International Journal of Systemic, Cybernetics and Informatics, April 2009.

[24] R. Kirsch, "Computer Determination of the Constituent Structure of Biomedical Images," *Computers and Biomedical Research*, 1971, 3-4, pp. 315–328.

**Dr. Satish Kumar** is a faculty member of Panjab University, Chandigarh (India) , currently posted at Panjab University Regional Centre, Muktsar , Punjab (India) and have about ten years experience of teaching Post-graduate classes. His areas of interest are Image Processing, Pattern Recognition and Artificial Intelligence involved with these fields.