Shape Representation and Recognition of Bangla Characters using Fourier Descriptor

M. A. Naser
Dept. of EEE
Islamic University of Technology (IUT)
Gazipur Bangladesh

Abdur Raquib Ridwan
Dept. of EEE
Islamic University of Technology (IUT)
Gazipur Bangladesh

Md.Murad Hossain
Dept. of EEE
Islamic University of Technology (IUT)
Gazipur Bangladesh

ABSTRACT
Fourier descriptors are used for representing the shapes as well as extracting the features of the characters. It has the advantage of being invariant to the affine transforms. It possesses the ability to reconstruct the original shape. Moreover, all the descriptors are not required to describe the character. Hence huge reduction in feature size is possible. K-nearest neighbor classifier is used to recognize the characters and the accuracy is 98.4% for training set while it is 89.3% for a test set. A case wise in-depth analysis is provided for each misclassified character at the end of the result.

General Terms
Feature extraction, Fourier descriptor, Bangla character recognition, k-nearest neighbor classifier.

1. INTRODUCTION
Feature extraction is one of the crucial steps for any character recognition system. To extract the features from the image of the character, the character itself has to be represented in a way suitable for the operation. In image processing, this representation is done either by the pixels comprising the object or by the contour. First one is region based representation while the second one is contour or shape based. Examples of region based representation are area, Euler number, texture and statistical moments. Contour based representations are Fourier descriptors, chain codes etc. Among these techniques, Fourier descriptor is the most useful one in image processing. This is because of its distinctive properties. These features are rotation, translation, scaling and mirror-reflection invariant. Above all, original shapes can be fully reconstructed from these features [1]. Reconstructions of the original characters are shown in Fig. 1 for different numbers of descriptors.

Many literatures are found to work on region as well as contour based techniques. Pal and Tripathy [2] proposed contour-distance based features where the distance of the contour points from the center of gravity is taken as features. These distances are then arranged in a specific order to make the feature size and rotation invariant. N. Sharma and U. Pal [3] proposed a directional chain code features based quadratic classifier. But all of these works are based on Dev Nagari characters, which only have a few characters similar to Bangla. Fourier descriptor is used in many pattern recognition problems [4,5,6,7,8] including character recognition [9,10]. Datta and Chaudhury [11] experimented with Fourier descriptors and multi-layer perception (MLP) neural network on Bengali handwritten and printed numerals and they reported a maximum accuracy of 90%. But their work is limited to the numerals only. Hasnat et. al used Discrete Cosine Transform (DCT) which is a variation of Discrete Fourier Transform (DFT). DCT is applied for each dimension individually.

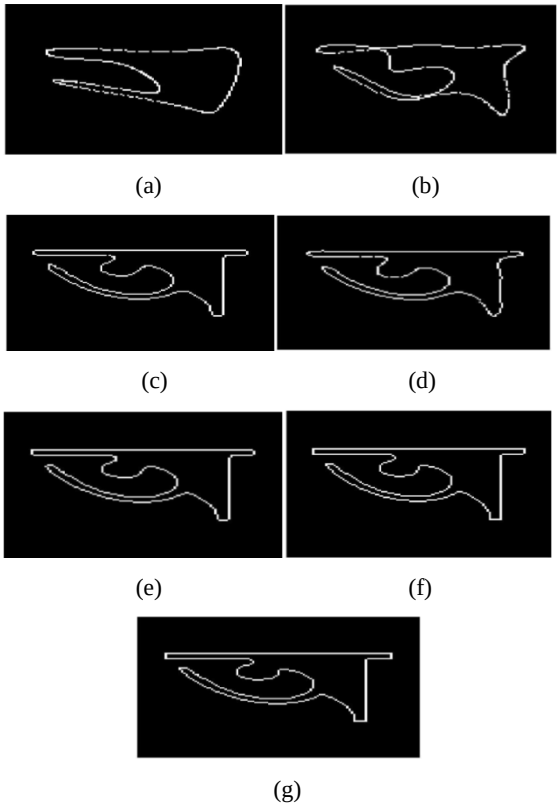


Fig 1: Reconstruction of the original shape using 10, 20, 40, 60, 80, 100, 200 numbers of fourier descriptors is shown in the images (a), (b), (c), (d), (e), (f) and (g) respectively.

In this paper, Fourier descriptors are used as features and the accuracy is tested for Bangla characters. For recognition purpose, k-nearest neighbor classifier is used. The rest of the paper is arranged as following. In section 2, theory of Fourier descriptor is discussed. Section 3 deals with the concept of extracting character features and the results are illustrated in section 4. Finally, section 5 concludes with an overview and possible advancement possible for improving the accuracy.

2. FOURIER DESCRIPTORS

To describe the shape of the character through Fourier descriptors, the contour [Fig. 2] has to be represented first. The contour can be represented by the *N* point digital boundary in the *xy* plane. These points can be represented by coordinate pairs for example $(x_0, y_0), (x_1, y_1), (x_2, y_2), \ldots\ldots\ldots (x_{N-1}, y_{N-1})$ which are travelled through the boundary starting from an arbitrary point. So the boundary can be represented as a sequence of coordinates $s(n) = [x(n), y(n)]$ for $n = 0, 1, 2, \ldots\ldots\ldots N-1$ and each coordinate pair may be expressed as a complex variable, $s(n) = x(n) + jy(n)$ for $n = 0, 1, 2, \ldots\ldots\ldots N-1$.

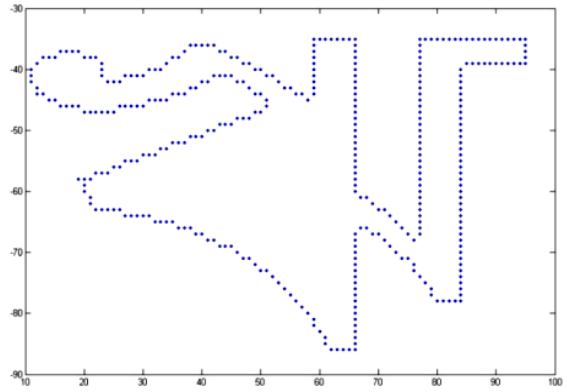


Fig 2: Contour of the Bangla character ঋ

The Fourier descriptors of the boundary can be found by Discrete Fourier Transform [1] written by,

$$C_u = \frac{1}{N}\sum_{n=0}^{N-1} s(n) e^{-j2\pi un/N} \quad \text{for } u = 0, 1, 2, \ldots\ldots\ldots N-1 \tag{1}$$

The inverse Fourier Transform of C_u can be written by,

$$s(n) = \sum_{u=0}^{N-1} C_u e^{j2\pi un/N} \quad \text{for } n = 0, 1, 2, \ldots\ldots\ldots N-1 \tag{2}$$

The lower order terms of this series describe the macroscopic behavior of the boundary curve while the higher order terms describe the detailed microscopic behavior of the boundary. So instead of taking all the coefficients, less number of coefficients can be used to reconstruct *s(n)*. Hence *s(n)* can be approximated to $\hat{s}(n)$ where $\hat{s}(n)$ contains adequate lower order terms as well as higher order terms. Finally only the first *P* coefficients of $\hat{s}(n)$ is used for reconstruction,

$$\hat{s}(n) = \sum_{u=0}^{P-1} C_u e^{j2\pi un/N} \tag{3}$$

Fourier descriptors calculated in the above equations are invariant to translation, rotation and scaling. Translation can be described as addition (introduction) of any complex number *s*, to every element in the array *s(n)* which contains boundary coordinates. The Discrete Fourier transform of such array is

$$\frac{1}{N}\sum_{n=0}^{N-1}(s(n)-s)e^{-j2\pi un/N} = \frac{1}{N}\sum_{n=0}^{N-1} s(n) e^{-j2\pi un/N} - s\frac{1}{N}\sum_{n=0}^{N-1} e^{-j2\pi un/N} \tag{4}$$

Rotation of angle ϕ can be described as multiplication of every element in the array with $e^{-\phi i}$. The Discrete Fourier transform of such array is,

(5)

$$\frac{1}{N}\sum_{n=0}^{N-1} e^{-\phi i} s(n) e^{-j2\pi un/N} = \frac{1}{N} e^{-\phi i}\sum_{n=0}^{N-1} s(n) e^{-j2\pi un/N}$$

As $\left|e^{-\phi i}\right| = 1$, it is enough to take the absolute value of each element for achieving the rotational invariance. Scaling can be described as multiplication of every element in the array with a real constant *c*. The discrete Fourier transformation of such an array is,

$$\frac{1}{N}\sum_{n=0}^{N-1} cs(n) e^{-j2\pi un/N} = \frac{1}{N} c\sum_{n=0}^{N-1} s(n) e^{-j2\pi un/N} \tag{6}$$

3. FEATURE EXTRACTION

First, the shape or contour of the character is obtained as shown in Fig. 2. Then each point describing the boundary is converted to complex number as mentioned in previous section. This array of numbers functions as a signal which is then finally fourier transformed. The fourier transform coefficients are the extracted features. Finally taking only 57 coefficients is sufficient to rebuild the characters to its original form. As Fig. 3 indicates, the most significant values are at the middle, so the coefficients are taken from the middle portion of the transformed signal. If the characters do not have enough boundary points to generate 57 coefficients, zero padding is done in that case.

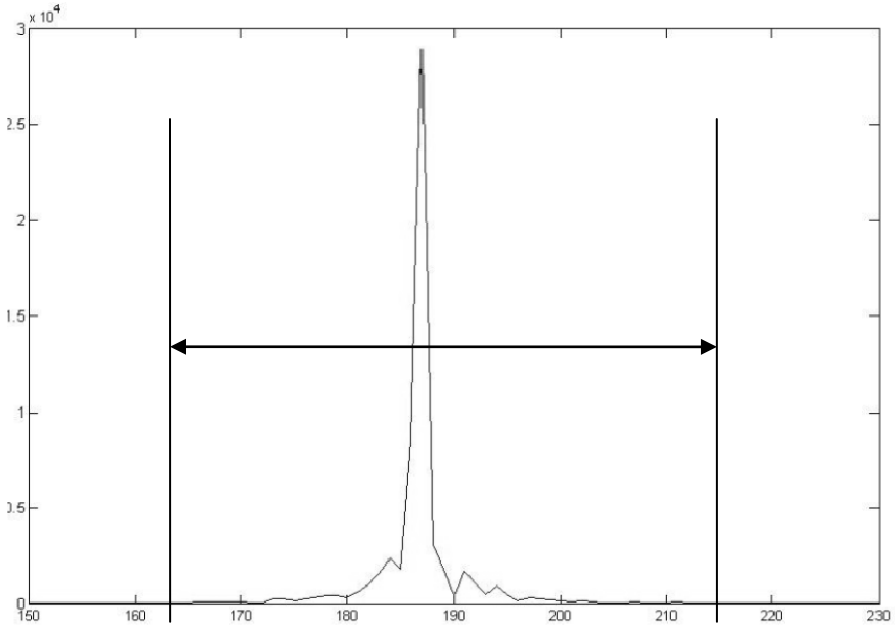


Fig 3: The range of the 57 coefficients having the highest significant peak at the middle.

4. RESULT

Once the features are generated and rearranged, they are ready for recognition. For the present work, K-nearest neighbor (KNN) classifier is used. The total set contains 250 characters of 5 different fonts. The whole set is then divided into train and test sets containing 150 and 100 characters respectively. The accuracy is 89.33 %. Table I shows some of the characters recognized incorrectly.

The misclassified characters can be categorized into three types: similar boundary errors, disconnected region errors and mirror-image errors. Here is the case wise analysis of the result. In Bangla alphabet, many characters have similar boundary. This poses problem when features are calculated based on their boundaries. Like, ঈ is recognized as উ due to the presence of the similar upper portion. উ and জ have similar lower-round shape. ঔ and ঐ have similar shape on their right side. In case of ঘ and ষ, both have almost similar boundary. Because fourier descriptor is utilized only on the boundary of the characters, so no difference can be acknowledged by the classifier. Fig. 4 shows the boundaries of the characters.

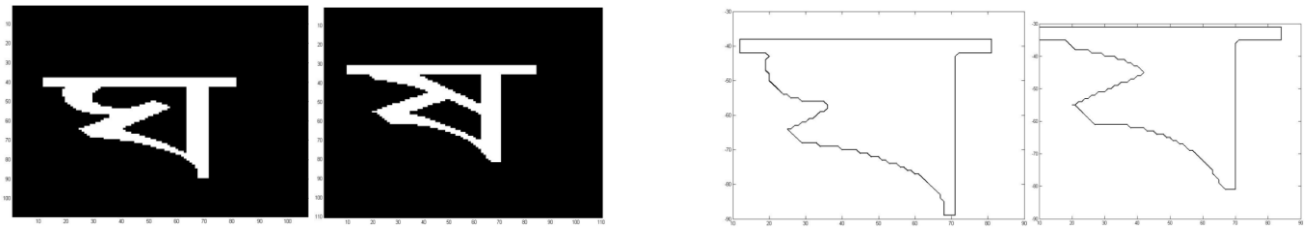
When the fourier descriptor is taken, it is taken on the boundary of a region. That region is a whole entity and not disconnected. If there is another disconnected portion, it is regarded as another region. Hence, the dot under the character ন় is discarded while taking the boundary of the character. But this alters the character to ব. Hence the character ন় is misclassified as ব [fig. 5]. Similar case is observed for ং and ঁ.

TABLE I: List of the incorrect recognized characters.

Original	Recognized as	Original	Recognized as
ঈ	উ	ণ	ন
উ	জ	ব	চ
ঔ	ঐ	ন়	চ
ঘ	ষ	ব	ন়
ন়	ঋ	ং	ঃ
ড	ঢ়	ঁ	ং

Next case is the mirror-image error. This is due to the property of the Fourier descriptor itself. Fourier descriptor can identify both an object and its mirror-image ব and চ are two different characters. But they look like each other's mirror image. Fig. 6 shows the characters and their mirror images. It becomes obvious that the descriptor fails to extract information that makes them separate. The confusion matrix is provided in the Appendix.

Then, the rotation invariance of the descriptor is examined. Here the classifier is trained and tested by the same 250 characters. Rotation is taken at angles of 15, 30, 45, 90 and 180 degrees. Table II contains the results. Angles near to the axis show better results than the angles far from the axis. This is because the more the angle moves away from the axis, the more distorted the character becomes. In this case, the characters undergo maximum distortion at 45 degree. Fig. 7 shows characters rotated at different degrees.



(a) (b) (c) (d)

Fig 4: The typical boundaries of the character **ঘ** and **ষ** in the image (a), (c) and (b), (d) respectively.

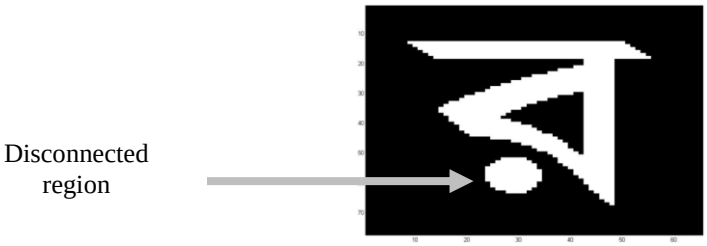
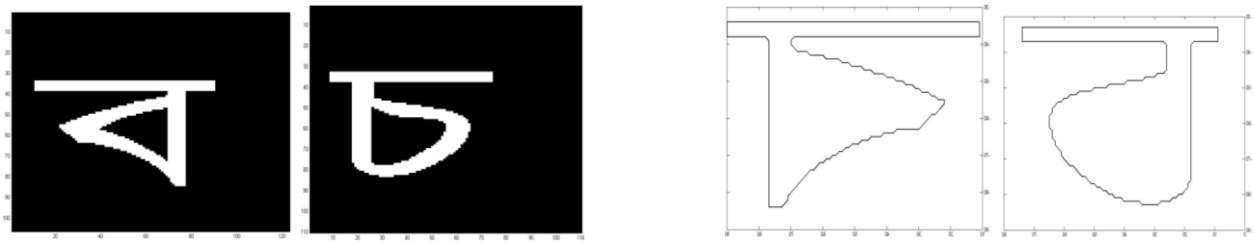


Fig 5: The disconnected region of the character **র়**



(a) (b) (c) (d)

Fig 6: (a) & (b) showing the two different character **ব** & **চ** and Mirror image showing the similarity between of **ব** & **চ** .

(a) (b) (c) (d) (e)

Fig 7: Rotation of the character **অ** in the images (a), (b), (c), (d) and (e) at angles 15, 30, 45, 90 and 180 degree respectively

Table II: Percentages of accuracy at different angles of rotation.

Angle (Degree)	Accuracy (%)
0	98.4
15	87.6
30	73.6
45	42
90	87.6
180	96.4

5. CONCLUSION

In this paper, the features of Bangla characters have been extracted using Fourier descriptors. The accuracy found is 98.4% for training set and 89.3% for an unknown test set. The cause of all the discrepancies found in recognition has been explored. Some misclassifications occurred due to the inherent nature of bangla characters. Again, some occurred due to the built in properties of the Fourier descriptor algorithm itself. The future goal will be the modification of fourier descriptor that enhances the local properties of a contour to avoid misclassification of similar contours.

6. Appendix

=== Confusion Matrix ===

```
 a  b  e  d  e  f  g  h  i  j  k  l  m  n  o  p  q  r  s  t  u  v  w  x  y  z  A  B  C  D  E  F  G  H  I  J  K  L  M  N  O  P  Q  R  S  T  U  V  W  X   <-- Recognized As
 2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0 | a = AW
 0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | b = E
 0  0  2  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | c = EE
 0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | d = U
 0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | e = UU
 0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | f = RI
 0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | g = AE
 0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | h = OI
 0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | i = O
 0  0  0  0  0  0  0  1  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | j = OUI
 0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | k = KA
 0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | l = KHA
 0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | m = GA
 0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | n = GHA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | o = UO
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | p = CA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | q = CHA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | r = JA
 0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | s = JHA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | t = EO
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | u = TA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | v = THA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | w = DA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0 | x = DHA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | y = MURDHANNO
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | z = TW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | A = THW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | B = DW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | C = DHW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | D = DONTONNO
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | E = PA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | F = PHA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | G = BA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | H = BHA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | I = MA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0  0  0 | J = ONTOSTHO-JAW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  0  0  0  0  0  0  0  0  0  0  0  0  0 | K = RA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0  0  0  0  0  0 | L = LA
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  1  0  0  0  0  0  0  0  0 | M = BW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0  0  0 | N = TALLOBOSHW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  2  0  0  0  0  0  0 | O = MURDHONOSSAW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0  0 | P = DONTOSSAW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0  0 | Q = HAW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0  0 | R = DRW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0  0 | S = DHRW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  0  0  0  0  0  0  0  0  0  2  0 | T = ONTOSTHIO
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3 | U = KHONDO-TAW
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  2  1 | V = ONUSSWAR
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  3  0 | W = BISORGO
 0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  0  1  2 | X = Chandrabindu
```

7. REFERENCES

[1] Gonzalez, R, C & Woods R, E. 'Digital Image Processing', Second Edition, Prentice-Hall of India Private Limited, 2006.

[2] Pal, U & Tripathy, N. 'A contour distance-based approach for multi-oriented and multi-sized character recognition'. Journal of Sadhana. Vol 34, pp. 755-765, October, 2009.

[3] Sharma, N, Pal, U, Kimura, F & Pal, S. 'Recognition of Off-Line Handwritten Devnagari Characters Using Quadratic Classifier', Computer Vision, Graphics and Image Processing. ICVGIP , LNCS 4338, pp. 805 – 816, 2006.

[4] Persoon, E, & Fu, K, S. 'Shape discrimination using Fourier descriptors'. In the Proceedings of the IEEE Trans., SMC-7, No.3. pp. 170-179, March 1977.

[5] Richard, C, W & Hemami, H. 'Identification of three-dimensional objects using Fourier descriptors of the boundary curve' In the Proceedings of the IEEE Trans., SMC4. pp. 371-378, July 1974.

[7] Wallace, T, P & Wintz, P, A. 'An efficient three-dimensional aircraft recognition algorithm using normalized Fourier descriptors', Computer Graphics Image Processing. Vol 13. pp. 99-126, 1980.

[8] Zahn, C, T & Roskies, R, Z. 'Fourier descriptors for plane closed curves', In the Proceedings of the IEEE Trans. Computer. Vol.21. pp. 269-281, 1972.

[9] Knyrak, A & Buaeshi, H, E. 'Classification of digitized curves represented by signatures and Fourier Descriptors'. In Computer Vision and Shape Recognition, Krzyzak and Kasvand, Eds, World Scientific. pp. 241-259, 1989.

[10] Granlund, G,H. 'Fourier preprocessing for hand print character recognition', In the Proceedings of the IEEE Trans. Computers, Vol.21, pp. 195-201, 1972.

[11] Shridhar, M & Badreldin A. 'High accuracy character recognition algorithm using Fourier and topological descriptors", Pattern Recognition, Vo1.17, NOS, pp. 515-524, 1984.

[12] Datta, S, Chaudhury, S & Parthasarathy G. 'On recognition of Bengali numerals with backpropagation learning', In Proceedings of the IEEE International Conference on Systems, Man and Cybernetics, Vol.1, pp. 94 - 99, 1992, USA.