Handwritten Character Recognition using Conditional Probabilities

M. PADMANABAN, AND E. A. YFANTIS
Digital Image Processing Laboratory
Department of Computer Science
University of Nevada, Las Vegas
4505 Maryland Parkway Las Vegas, NV 89154
UNITED STATES OF AMERICA
http://web.cs.unlv.edu/digitalimageprocessinglab

Abstract: - Handwritten Character Recognition is an important part of Pattern Recognition. This is also referred to as Intelligent Character Recognition (ICR). In this paper, a conditional probability based combination of multiple recognizers for character recognition will be introduced. After preprocessing the given character image, different feature recognition algorithms are employed, and their performance on a given training set is analyzed. The reliability of the recognition algorithms is measured in terms of Conditional Probabilities. A rule based on their reliability is identified to combine all these individual feature recognition algorithms by incorporating their interdependence.

Key-Words: - Character Recognition, OCR, ICR, Text-Recognition, Preprocessing, Conditional Probability.

1 Introduction

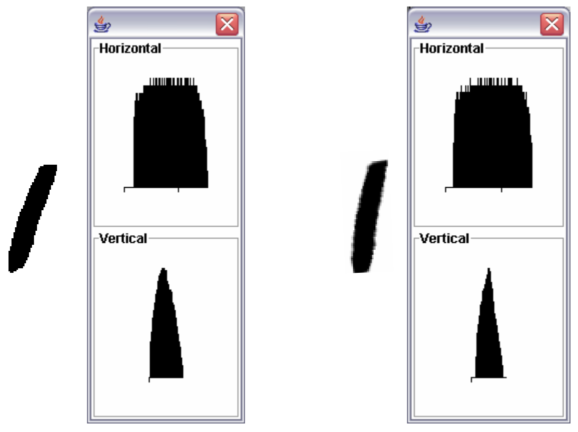
Optical Character Recognition (OCR) is the translation of optically scanned bitmaps of printed text characters into character codes, such as ASCII. Handwritten character recognition is similar in concept, except that the input is in the form of handwriting. Handwritten character recognition can be broadly divided into two categories: On-line recognition and Off-line recognition. In the on-line case, the two-dimensional coordinates of successive points of the writing as a function of time are stored in order (for example, the order of strokes made by the writer is readily available). In the off-line case, only the completed writing is available as an image. The on-line case deals with a spatio-temporal representation of the input, whereas the off-line case involves analysis of the spatio-luminance of an image.

This paper revolves around the Off-line recognition. The recognition of hand written characters is a special kind of complex mathematical problem, because the different distortions present in hand written character set makes it difficult to produce a distinct set identification. Handwritten character recognition has numerous applications such as address and zip code recognition, writer identification, Bank Check Recognition, etc [1] [2] [3].

The character recognition process begins with preprocessing where the application form is scanned and the handwritten parts are found, separated and transformed into a binary matrix with 0 representing black pixel and 1 representing white. The next phase is the character segmentation which looks for the area of each character in the matrix. In the feature extraction phase, each character is analyzed to extract different features. These features are then used for classification of characters.

2 Feature Extraction

In the preprocessing phase [4], the form is given as the input and the system extracts the content. The first job of the system is to correct the orientation of the form. Next, the noise is removed and then the characters are segmented. Preprocessing is important because it aids in improving the accuracy of the recognition.



(a) (b)

Fig.1. Skew Removal: (a) Slant Letter (b) Deskewed Letter.

2.1 Skew Removal

Skew can be removed by using the projections of the histogram. Histogram projections of the image tilted at angles between –5º to +5º are generated. The tilt angle for which the histogram gives maximum peak value is the skew angle. The image is rotated accordingly. See Fig.1.

2.2 Noise Removal

The noise is removed using smoothing algorithms or filters. The Gaussian filter is a good example of a smoothing operation. A 5 x 5 matrix mask as shown in the Fig.2 is used. The smoothing operator blurs the image and removes the spurts.

$$(1/273)* \begin{array}{|c|c|c|c|c|} \hline 1 & 4 & 7 & 4 & 1 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 7 & 26 & 41 & 26 & 7 \\ \hline 4 & 16 & 26 & 16 & 4 \\ \hline 1 & 4 & 7 & 4 & 1 \\ \hline \end{array}$$

Fig.2. Sample 5 * 5 Gaussian Mask

2.3 Thresholding

Thresholding is used to differentiate between the foreground (ink) and the background (paper). The Handwritten characters are then extracted, converted to a binary array and stored in a database.

2.4 Character Segmentation

After the preprocessing stage, the handwritten text has been transformed into matrices with binary values, where zeroes denote black pixels (i.e. the handwritten characters) and ones white pixels (i.e. the empty space around the character). The next step is to separate the characters. A simple approach is identifying the physical gaps using only the components. These methods assume that gaps between words are larger than the gaps between the characters. However, in handwriting, this is not the case. Most recognition methods call for segmentation of the word into its constituent characters. Segmentation points are determined using features like ligatures and concavities. Gaps between character segments (a character segment can be a character or a part of character) and heights of character segments are used in the algorithm.

2.5 Extraction of Features

Feature Extraction is the process of extracting a set of parameters that define the shape of the underlying character as precisely and uniquely as possible. Generally, the character images are not of the same size; they are of arbitrary size. In machine-print, a height to width ratio (aspect ratio) of 4:3 is common, but there is no fixed aspect ratio in handwritten characters. Sizes of the image may vary from approximately 8 x 8 to 200 x 200 for document images digitized at 300 ppi. So, larger variations in size are noticeable among handwritten rather than machine-printed characters. Any feature extractor for such images should be applicable for all input image sizes. There are some cases in which the characters from different classes look similar, like the characters ‘I’ and ‘J’. So, localized analysis of contours is essential.

3 Recognition Algorithms

Each image of size D1 x D2 is divided into N1 x N2 parts. Each partition has dimension (D1/N1) x (D2/N2) approximately. The following are some of the algorithms used for Feature Recognition applied to each partition. A feature vector is obtained for each algorithm.

3.1 Local Center of Gravity

The “Center of Gravity” [5] for each partition is computed. The local center of gravity (X_{cg}, Y_{cg}) is found by determining the center of gravity for each partition. The distance between the local center of gravity and the whole center of gravity of the number is found, and the distance is normalized by dividing the distance by the width of the number.

$$CenterofGravity\ X_{cg} = \frac{\sum m_i x_i}{m_i} \qquad (1)$$

$$CenterofGravity\ Y_{cg} = \frac{\sum m_i y_i}{m_i} \qquad (2)$$

3.2 Connectivity

The connectivity between segments adjacent to each other in horizontal and vertical directions is determined. Connectivity between segments is represented by a ‘1’ and no connectivity is represented by a ‘0’. Image is thinned and partitioned as shown in Fig.3.

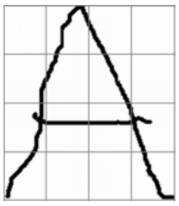


Fig.3. Connectivity

3.3 Extreme Points

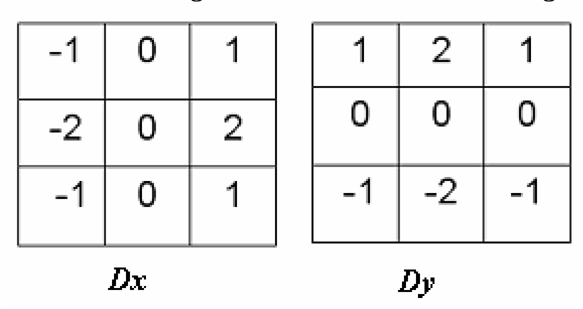
The horizontal mid-point of the character image along with the extreme left and right points in each partition is calculated. If the extreme point [6] is on the left side, then a negative sign is introduced. Then the distance between extreme points and the midpoint is calculated and normalized by dividing it with the width.

3.4 Number of Tips

Characters are thinned. For every black pixel, it is checked whether it has more than one black neighboring black pixel. If it has only one black neighboring pixel, it is noted as tip. The number of tips [6] in each partition is stored.

3.5 Gradients

The gradient map of each partition is determined to find the local contour variations. Then the gradient angles are quantized. The gradient angle is determined using the Sobel masks shown in fig.4.



-1	0	1
-2	0	2
-1	0	1

Dx

1	2	1
0	0	0
-1	-2	-1

Dy

Fig.4. Sobel Masks

The Sobel Operator for horizontal component is

$$Dx\,(i,j) = I(i-1,j+1) + 2I(i,j+1) + I(i+1,j+1) - I(i-1,j-1) - 2I(i,j-1) - I(i+1,j-1) \tag{3}$$

and the Sobel Operator for vertical component is

$$Dy\,(i,j) = I(i-1,j-1) + 2I(i-1,j) + I(i-1,j-1) - I(i+1,j-1) - 2I(i+1,j) - I(i+1,j+1) \tag{4}$$

The angle of the edge pixels is then calculated using the formula

$$Theta\,(i,j) = Arc\,Tan\,(\,Dy(i,j)\,/Dx(i,j)\,) \tag{5}$$

Quantizing gradient directions into a small number (K) of ranges aids in the generation of fixed number of features. For example, we can set 12 ranges (bins) : 0^o-30^o, 30^o-60^o, 60^o-90^o…….. 330^o-360^o.

3.6 Wavelets

The image is first normalized to size N*N pixels, where N = 2^K. The Haar or Daubechies's kernels are applied recursively (2–levels). At each level, 4 quadrants are available. The top-left is decomposed recursively as shown in the figure below.

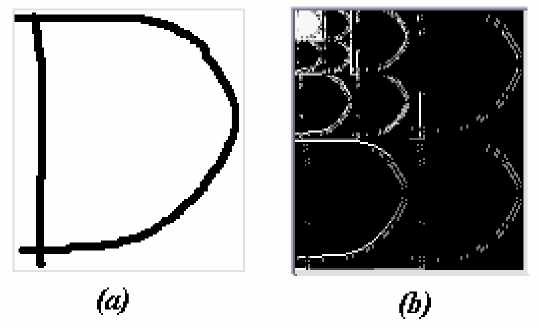


Fig.5. (a) Original Image, (b) Wavelets

Each sub-image is partitioned into m*m blocks. The 1st and 2nd Moments [7] are computed for each block as follows:

1st Moment:

$$\mu = \frac{1}{n^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} b(x,y) \tag{6}$$

2nd Moment:

$$\sigma = \sqrt{\frac{1}{n^2}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1}\left[\mu - b(x,y)\right]^2} \tag{7}$$

These moments are stored in the feature vector.

3.7 String Distance Measurement

First, the image is thinned. Row by row, two successive pixel rows are considered. The distance 'S' between the current black pixel (B_i) and the next row black pixel (B_{i+1}) is calculated. All the distances for a total of k rows are summed up to compute the string distance measurement (SD) [8].

$$SD = \sum_{i=1}^{k} S_i \qquad (8)$$

"String Line Measurement" (SLIM) is also added to differentiate the horizontal and vertical lines. If the line is horizontal or vertical, then SLIM is set to one. The final equation is:

$$SDM_i \quad = \quad \alpha factor_i + \sum_{j=1}^{n} \beta factor_j \qquad (9)$$

$$\begin{aligned} &where \\ &\alpha factor_i = SD_i \\ &\beta factor_j = SD_i * (SLIM_j * BlackPixels\ within \\ &\qquad V/H\ segment_j / Black\,Count_i) \end{aligned} \qquad (10)$$

SDM for each partition is stored in to the feature vector.

3.8 Angular Moments

The image is partitioned with respect to center of gravity. Then the three Moments [9] (mass, distance and standard deviation) are calculated for each partition.

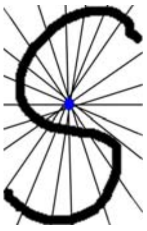


Fig.6. Angular Partitions

Mass:

$$m(p) \quad = \quad \frac{1}{M}\sum_{r \in p} r_i^0 \qquad (11)$$

Distance:

$$r(p) \quad = \quad \frac{1}{R}\sum_{r \in p} r_i^1 \qquad (12)$$

Standard Deviation:

$$\sigma^2(p) \quad = \quad \frac{1}{S}\sum_{r \in p} (r_i - \bar{r}_p)^2 \qquad (13)$$

Summation goes over the black pixels of the partition. The normalizing factor M, R and S are the summation (Moments) over the whole image.

4 Classifications

A classifier that is trained on a labeled data set can be used for future prediction of class labels for unknown instances.

The classifier predicts a class label, w_u, for an unknown feature vector 'y' from a discrete set of previously learned labels $\{w_1, w_2 \ldots w_n\}$. It can be shown that to minimize classification errors, one should assign the example to the class with the largest posterior probability $P(w_i \mid y)$. This is known as the maximum aposteriori rule [10], and is the best any classifier can do. So, the job of the classifier is to estimate $P(w_i \mid y)$ from the unknown data set.

Training a classifier can be time consuming and require significant amounts of memory, especially for large data sets.

4.1 K – Nearest Neighbor Classifiers

The K Nearest Neighbor (kNN) algorithm [11] [12] [13] predicts the outcome y for an unknown w_u by finding the k labeled training data set nearest to w_u within a pre-classified dataset D. It classifies w_u using the maximum vote of the nearest 'k' neighbors. The kNN method is a powerful technique that can be used to generate highly nonlinear classifications with limited data.

Normally, the "closeness" is measured by Euclidean distance. For two tuples, $X = \langle x1, x2, x3 \ldots xn\text{-}1\rangle$ and $Y = y1, y2, y3\ldots yn\text{-}1\rangle$, the Euclidian distance is

$$D(X,Y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} \qquad (14)$$

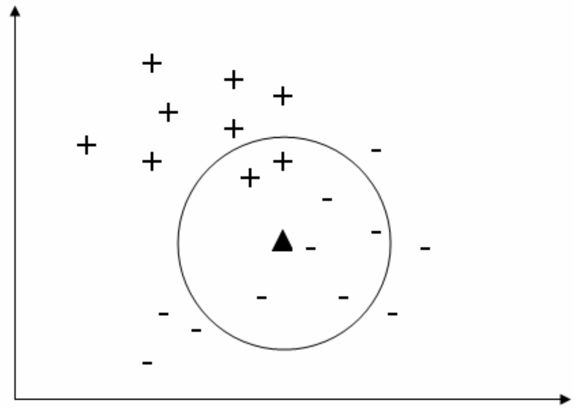


Fig.7. K Nearest Neighbor

The main limitations of kNN are storage requirements (since the entire dataset needs to be available for matching) and the computational cost (since for each unknown data, the distance to all training samples needs to be computed). These limitations, however, can be overcome by editing the training set [14], and generating a subset of prototypes. The kNN algorithm is extremely sensitive to the dimension of the features. So, special attention must be paid to the scaling of each feature dimension.

5 Combining Recognizers

The "confusion matrix", C, of each recognizer on a training set of data is used as indicators of the recognizers performance. C is an M × (M^K) matrix, for M classes and K recognizers. From the matrix C, the row sum C_{ij} is calculated, which gives the total number of samples belonging to class 'i'. Column sum C_{ij} gives the total number of samples that are assigned a class pattern 'j'.

The conditional probability [15] that a pattern, x, actually belongs to class 'i', given that the recognizers classify it as pattern 'j', can be estimated as

$$P(x \in C_i \mid e(x) = j) = \frac{C_{ij}^{(k)}}{\sum_{i=1}^{M} C_{ij}^{(k)}} \qquad (15)$$

The class 'i' with the maximum probability value is the correct output. A two class problem with three recognizers produces a 2 x 8 matrix as follows:

	0,0,0	0,0,1	…	1,1,0	1,1,1
0	A0	B0	…	G0	H0
1	A1	B1	…	G1	H1
	Col SumA	Col SumB	…	Col SumG	Col SumH

Fig.8. Confusion Matrix

The row values are the input, and the column values are the outputs of the 3 recognizers i, j, k respectively.

So,

$$TotalSum = \sum ColSum \qquad (16)$$

$$P(X = 0 \mid Xi = 0, Xj = 0, Xk = 0)$$

$$= \frac{P(X = 0, Xi = 0, Xj = 0, Xk = 0)}{P(Xi = 0, Xj = 0, Xk = 0)}$$

$$= \frac{(A0 / TotalSum)}{(ColSumA / TotalSum)}$$

Therefore,

$$P(X = 0 \mid Xi = 0, Xj = 0, Xk = 0) = \frac{A0}{ColSumA} \qquad (17)$$

By using these probability values, the correct output class 'i' (having the maximum conditional probability value) can be determined, given that the recognizers classify it as pattern 'j'.

6 Conclusions

Recognition of complex handwritten characters remains a great research area. By using the conditional probabilities to combine the multiple recognizers, high accuracy of recognition is achieved.

References:

[1] Sung-Hyuk Cha, Sargur N. Srihari, Writer Identification: Statistical Analysis and Dichotomizer, *Lecture Notes in Computer Science*, Volume 1876, Jan 2000, Page 123

[2] G. Leedham and S. Chachra, Writer identification using innovative binarised features of handwritten numerals. *In Seventh International Conference on Document Analysis and Recognition*, pages 413–417, 2003.

[3] G. Kim and V.GovindaRaju, A Lexicondriven approach to Handwritten Word Recognition for Real Time Applications, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Volume 11, no.4, pp.366-379, Apr.1997.

[4] P. Slavik, V. Govindaraju, Equivalence of different methods for slant and skew corrections in word recognition applications. *IEEE Trans. Pattern Analysis and Machine Intelligencee*, Volume 23, Issue 3, March 2001 Page(s):323 - 326

[5] S. Mori, H. Nishida, and H. Yamada. *Optical Character Recognition*. John Wiley and Sons, Inc., 1999.

[6] Heutte, L., Paquet, T., Moreau, J. V., Lecourtier, Y., and Olivier, C. 1998. A structural/statistical feature based vector for handwritten character recognition. *Pattern Recogn. Lett.* 19, 7 (May. 1998), 629-641.

[7] Jong-Hyun Park, Il-Seok Oh, Wavelet-Based Feature Extraction from Character Images, *Lecture Notes in Computer Science*, Volume 2690, Aug 2003, Pages 1092 - 1096

[8] S. Singh. Shape Detection Using Gradient Features for Handwritten Character Recognition, *icpr*, vol. 03, no. 3, p. 145, 13th 1996.

[9] Tsang, I.J., Tsang, I.R., Dyck, D.V., 1998. Handwritten character recognition based on moment features derived from image partition.

In: International Conference on Image Processing, vol. 2, pp. 939-942.

[10] R. O. Duda, P. E. Hart, and D. G. Stork, *Pattern Classification*, 2nd ed. New York: Wiley, 2000.

[11] Y. Hammamoto et. al, A Bootstrap Technique for Nearest Neighbor Classifier Design, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 1, Jan. 1997.

[12] T. Hastie and R. Tibshirani, Discriminant Adaptive Nearest Neighbor Classifier, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 18, no. 6, June 1996.

[13] S.A. Nene and S.K. Nayar, A Simple Algorithm for Nearest Neighbor Search in High Dimensions, *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 19, no. 9, pp. 989-1003, Sept.1997.

[14] B. V. Dasarathy, Nearest Neighbor (NN) Norms: NN Pattern Classification Techniques. Los Alamitos, CA: *IEEE Comp. Soc.Press*, 1991.

[15] John Haigh. *Probabilty Models*. Springer, 2002.