DEVELOPMENT OF A HANDWRITTEN TURKISH CHARACTER RECOGNITION SYSTEM

Sema KOÇ, Ayşegül ÇUHADAR
University of Gaziantep
Electrical and Electronics Engineering Department
27310 Gaziantep / TURKEY
e-mail : skoc@gantep.edu.tr

ABSTRACT
Character recognition systems have many applications such as document processing, language translation, electronic publication, office documentation, etc. In this study, an optical character recognition system for Turkish handwritten characters is developed.
The recognition method consists of following stages: Preprocessing, feature extraction, dictionary building and pattern matching.
In the preprocessing stage normalization and thinning algorithms are used. For the feature extraction vertical, horizontal and two diagonal (VH2D) method is used. In the VH2D method, projections on character images are made from four directions; vertical, horizontal and two diagonals ($45^0$and 135^0). Then a dictionary of feature vectors will be built up to be used as a base for pattern matching. Finally, when an unknown character is input, its feature vector is obtained and this feature vector is used to match the feature vectors of the characters in the dictionary. The unknown character will be recognized as the one who has the minimum difference between the feature vectors of the unknown character and all the candidates in the dictionary, provided that the minimum difference is within a reasonable threshold.

1. INTRODUCTION
The technology of converting printed or handwritten materials into corresponding symbolic forms, accessible to computer manipulation, is known as "Optical Character Recognition (OCR)". OCR has been an active research area for more than three decades and has still been receiving intensive attention.
In this paper, an algorithm for the handwritten Turkish character recognition is presented. The algorithm consists of 4 stages, namely; preprocessing, feature extraction, dictionary building and pattern matching.
In the preprocessing stage, normalization and thinning algorithms are used. Good features extraction is critical for the success of character recognition. In this study, the VH2D method is used for the feature extraction. And then, feature vectors of the data set will be used to build a data dictionary, which is used to be a base for character recognition.
The detail of the recognition algorithm is given in the rest of the paper.

2. PREPROCESSING
The goal of the preprocessing is to increase the quality of the handprinted data. And includes all the steps that are necessary to bring the input data into a form acceptable to the feature extraction portion of the system.
Our preprocessing stage consists of normalization and thinning.

2.1 Normalization
Normalization of the character dimensions is one of the most efficient preprocessing techniques.
The height of the characters can differ considerably and this invites normalization of character dimensions [1].

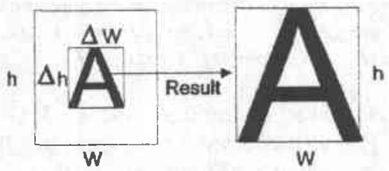

Figure 1. Normalization of the character dimensions

For the normalization algorithm character widths and heights are measured by putting a rectangular sub-frame around the character with respect to the rows and the columns of the grid. The actual character width, and height Δw and Δh serve as basic parameters for the normalization algorithm shown in Figure 1. Assuming a character width Δw is 5 points and maximum grid width is 9 points a binary vector of 5x9 elements is composed first. Starting from the left groups of 5 components are picked from this vector and summed.

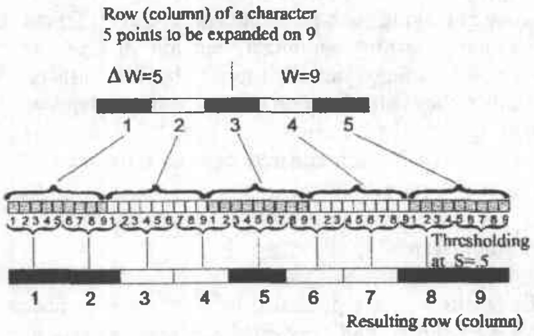

Figure 2. Normalization Algorithm

A thresholding procedure is next. If the sum exceeds 5xs, where s is a threshold parameter, the result is 1 otherwise the result is 0. Consequently a vector of 9 components is obtained being the expansion of the row originally containing 5 grid points. This operation is sequentially carried out for all rows of the sub grid and for the columns next. At the end of the procedure a character is obtained touching all limiting lines of the original grid as demonstrated schematically in Figure 2.

2.2 Thinning

Thinning is the most common name for the process of reducing the width of a line-liked object from many pixels wide to just a single pixel. And a large reduction in volume of data resulting from this representation produces a significant improvement in the efficiency of OCR system.

The thinning algorithm, which is used in this study, requires two successive iterative passes as described in [2], [3]. At step 1, a logical rule P_1 is applied locally in a 3x3 neighbourhood to flag border pixels that can be deleted. These pixels are only flagged until the entire image is scanned. Deletion of all flagged pixels is performed afterwards. At step 2, another logical rule P_2 is applied locally in a 3x3 window to flag border pixels for deletion. When the entire image has been scanned, the flagged pixels are deleted. This procedure is applied iteratively, until no more image thinning can be performed. Let:

$$N(p_0)= \sum_{i=1}^{8} p_i \,, \tag{1}$$

denotes the number of objects pixels (p_i=0,1, i=1,...8) in the 3x3 window (excluding the central pixel) and

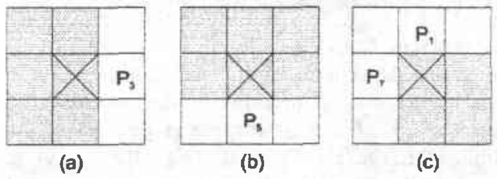

Figure 3. Central window pixels belonging to:
(a) an East boundary; (b) a South boundary;
(c) a North-West corner point

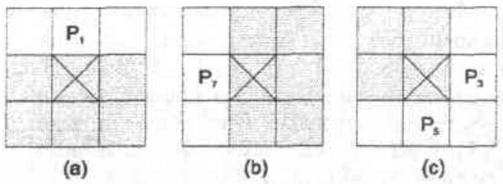

Figure 4. Central window pixels belonging to:
(a) a North boundary; (b) a West boundary;
(c) a South-East corner

$T(p_0)$ denote the number of $0 \rightarrow 1$ transitions in the pixel sequence $p_1p_2p_3 \ldots p_8p_1$. The logical rules P_1, P_2 used in two steps of the algorithm have the following form:

$$P_1:(2<N(p_0)\leq 6)\&\&(T(p_0)=1)\&\&(p_1.p_3.p_5=0)\&\&(p_3.p_5.p_7=0) \tag{2}$$

$$P_2:(2<N(p_0)\leq 6)\&\&(T(p_0)=1)\&\&(p_1.p_3.p_7=0)\&\&(p_1.p_5.p_7=0) \tag{3}$$

The products of the form $p_i.p_j.p_k$ denote logical conjunctions of the corresponding pixels. The first condition of both logical predicates states that the central pixel can be deleted if it possesses at least one and at most six 8-connected neighbours in a 3x3 window. If the central pixel has only one neighbour, it cannot be deleted, because the skeleton limb will be shortened. If the central pixel has more than six neighbours, central pixel deletion is not permitted because it will cause object erosion. The second predicate $T(p_0)$=1 checks if the pixels of the perimeter of the 3x3 neighbourhood form only one connected component. The third and fourth predicates ($p_1.p_3.p_5$=0) and ($p_3.p_5.p_7$=0) in (2) are satisfied if p_3=0 or p_5=0, or if (p_1=0 and p_7=0). Examples of these three cases are shown in Figure 3. The central pixel belongs to an East boundary (p_3=0), to a South boundary (p_5=0), or to a North-West object corner (p_1=0, p_7=0). In the first pass, the algorithm removes pixels belonging to one of these three cases. In the second pass, the thinning algorithm removes pixels having p_1=0 or p_7=0 or (p_3=0 and p_5=0) (3). In the second pass, pixels lying at North boundaries, or West boundaries, or South-East corner points are removed, as can be seen in Figure 4.

3. FEATURE EXTRACTION

In the data emerging from the preprocessing stage, the number of dimensions is high and very little information is concentrated in each. Feature extraction is applied in order to concentrate the information in the image into a few highly selected features.

In this work, we used the VH2D method [4] for the feature extraction. The VH2D approach firstly will extract feature vectors from the projections of each character's binary image in four directions, which are vertical, horizontal and two diagonal (45^0 and 135^0) directions, respectively. Secondly, since a randomly scanned image does not always have a character located in the center of its image, in order to ensure that every imaged character is located in the center of its image, the feature adjustment is essential to reduce the possibility of misclassifiying patterns due to the difference between images without centralization of patterns. Four subfeature vectors from four directions will be adjusted according to character's central point and then combined to form a complete feature vector for each individual character. Thirdly, when all the feature vectors of the character set are obtained, a dictionary of feature vectors is built up to be used as a base for pattern matching. Finally, when an unknown character is input, its feature vector is obtained by using the same way as in the second task. Then the unknown character's feature vector is used to match the feature vectors of the characters in the dictionary. The unknown character will be recognized as the one who has the minimum difference between the feature vectors of the unknown character and all the candidates in the dictionary, provided that the minimum difference is within a reasonable threshold. Otherwise, the unknown characters will be rejected which means this character is not in the dictionary. The details of the feature extraction stage is given below.

A binary image of a character is denoted as follows:

$$I_{ij} = \begin{cases} 1 & \text{If pixel } I_{ij} \text{ is black} \\ 0 & \text{otherwise} \end{cases}$$

where i,j=1,2,…,N (N is the size of the image).

If the image size of a character is NxN, then the central point of the image I_{ij} is $I_0([N/2],[N/2])$. In the rest of the paper, we assume N is even.

For a character C which has a binary image I_{ij}, its central point is denoted as $I_c(x,y)$, where;

$$x = \left[\frac{\sum_{k=1}^{N} \sum_{j=1}^{N} I_{kj} * j}{\sum_{k=1}^{N} \sum_{j=1}^{N} I_{kj}} \right] \quad (4)$$

$$y = \left[\frac{\sum_{k=1}^{N} \sum_{i=1}^{N} I_{ik} * i}{\sum_{k=1}^{N} \sum_{j=1}^{N} I_{kj}} \right] \quad (5)$$

i,j,k=1,2,…,N(N is the size of the image).

To move a character C into the center of its image, it should be moved to the position of the image in which the I_c overlaps the I_0.

For an image of character, $I_0=(N/2,N/2)$, $I_c=(x,y)$, let $\Delta_x=x-N/2$, $\Delta_y=y-N/2$, to have the character located in the center of its image it is necessary to transform the image according to following relations:

$$\text{If } \Delta_x \begin{array}{ll} >0 & \text{the image shifts left} \\ <0 & \text{the image shifts right} \\ =0 & \text{the image does not shift} \end{array} \quad (6)$$

$$\text{If } \Delta_y \begin{array}{ll} >0 & \text{the image shifts left} \\ <0 & \text{the image shifts right} \\ =0 & \text{the image does not shift} \end{array} \quad (7)$$

Image projections are made by adding the pixel values of the image I_{ij}, along certain directions such as vertical, horizontal or diagonal directions.

For a binary image I_{ij} of a character C (i,j=1,..,N), the image projection in vertical direction (v) is denoted as $F_v=[F_{v1},\dots F_{vj},\dots F_{vN}]$, where:

$$F_{vj} = \sum_{i=1}^{N} I_{ij} \quad (8)$$

For a binary image I_{ij} of a character C (i,j=1,..,N), the image projection in horizontal direction (h) is denoted as $F_h=[F_{h1},\dots F_{hi},\dots F_{hN}]$, where:

$$F_{hi} = \sum_{j=1}^{N} I_{ij} \quad (9)$$

For a binary image I_{ij} of a character C (i,j=1,..,N), the image projection in 45^0 diagonal direction (d1) is denoted as $F_{d1}=[Fd_{11},\dots Fd_{1i},\dots F_{d12N-1}]$, where:

$$F_{d1i} = \begin{cases} \sum_{l=N-i+1}^{N} \sum_{k=1}^{i} I_{lk} & \text{if } 1 \le i \le N \text{ and } k = l - N + i \\ \sum_{l=1}^{2N-i} \sum_{k=i-N+1}^{N} I_{lk} & \text{if } N+1 \le i \le 2N-1 \text{ and } k = l + i - N \end{cases} \quad (10)$$

For a binary image I_{ij} of a character C (i,j=1,..,N), the image projection in 135^0 diagonal direction (d2) is denoted as $F_{d2}=[Fd_{21},\dots Fd_{2i},\dots Fd_{22N-1}]$, where

$$F_{d2i} = \begin{cases} \sum_{l=1}^{i} \sum_{k=1}^{i} I_{lk} & \text{if } 1 \le i \le N \text{ and } k = i - l + 1 \\ \sum_{l=i-N+1}^{N} \sum_{k=i-N+1}^{N} I_{lk} & \text{if } N+1 \le i \le 2N-1 \text{ and } k = i - l + 1 \end{cases} \quad (11)$$

Image projections in four directions (v,h,d1 and d2) can be used as four subfeature vectors of a character. For an image of a character I_{ij}, its feature vector is defined as:

$F=(F_x,F_y,F_{45^0},F_{135^0})$.

3.1 Feature Adjustment

The VH2D projections of a centralized image I'_{ij} can be obtained by adjusting the VH2D projections of its uncentralized image I_{ij} according to the central point of the character $I_c(x,y)$.

Assume that the central point of the character C is Ic (F_v,F_h,F_{d1},F_{d2}), four projections of the uncentralized image I_{ij} $(F'_v,F'_h,F'_{d1},F'_{d2})$ are four projections of the centralized image I'_{ij}. From the way the $I_c(x,y)$ is calculated (4), (5) and the way the image is centralized (6), (7) the numbers of "1"s in I_{ij} and I'_{ij} should not be changed. So, the values of the elements in F_v should be the same as those in F'_v, as in F_h and F'_h,F_{d1} and F'_{d1}, F_{d2} and F'_{d2}. The only difference is that the positions of the corresponding elements in (F_v,F_h,F_{d1},F_{d2}) and $(F'_v,F'_h,F'_{d1},F'_{d2})$ different. Therefore, we can adjust the positions of elements in (F_v,F_h,F_{d1},F_{d2}) to make them equal to those in $(F'_v,F'_h,F'_{d1},F'_{d2})$. Obviously, the adjustment of projections in vertical and horizontal directions is dependent on Δ_x and Δ_y respectively. As a result, we have

(1) For vertical direction:

$$F_{vj} = \begin{cases} F_{vj} + \Delta_x & \text{if } 1 \le j + \Delta_x \le N \\ 0 & \text{otherwise} \end{cases} \quad (12)$$

(2) For horizontal direction, for the same reason as in (1):

$$F_{hi} = \begin{cases} F_{hi} + \Delta_x & \text{if } 1 \le i + \Delta_y \le N \\ 0 & \text{otherwise} \end{cases} \quad (13)$$

(3) For 45^0 diagonal direction:

$$F_{d1i} = \begin{cases} F_{d1i} + \Delta_{45} & \text{if } 1 \le i + \Delta_{45} \le 2N-1 \\ 0 & \text{otherwise} \end{cases} \qquad (14)$$

here, Δ_{45}=N-(x+y).

(4) For 135^0 diagonal direction:

$$F_{d2i} = \begin{cases} F_{d2i} + \Delta_{135} & \text{if } 1 \le i + \Delta_{135} \le 2N-1 \\ 0 & \text{otherwise} \end{cases} \qquad (15)$$

here, Δ_{135}=x-y.

Then, combine the four adjusted subfeature vectors F'_v, F'_h, F'_{d1}, and F'_{d2} into complete feature vector F.

4. BUILDING DICTIONARY

The dictionary is composed of all the feature vectors of the character set. If the size of the character set is M, then the dictionary is simply a linear array of D[M]. Each element of the array D[i] (i=1,...,M) is a basically a tuple, which represents a character,

D[i]=(D[i].I,D[i].C,D[i].F)

Here, D[i].I is the index of D[i], D[i].C is the code of D[i], D[i].F is the feature vector of D[i]. D[i].I is obtained by summations of all the vector elements of four subfeature vectors of D[i]. It is represented in the following way:

$$D[i].I = \sum_{j=1}^{N} D[i].F'_{hj} + \sum_{j=1}^{N} D[i].F'_{vj} + \sum_{j=1}^{2*N-1} D[i].F'_{d1j} + \sum_{j=1}^{2*N-1} D[i].F'_{d2j} \qquad (16)$$

where D[i].F'_{vj}, D[i].F'_{hj}, D[i].F'_{d1j}, D[i].F'_{d2j} are the jth vector elements of D[i].F'_v, D[i].F'_h, D[i].F'_{d1}, D[i].F'_{d2}, which are the four subfeature vectors of D[i].N is the image size.

The dictionary is organized in a way that it is sorted according to the ascending order of the index value. If there are two similar indices, they are located consecutively.

5. PATTERN MATCHING

When the dictionary of feature vectors has been built up, we are able to perform pattern matching, to classify the input unknown character.

For a test sample, its feature vector is obtained as explained before and its index is calculated using (16). Then we find the position of the element whose index is nearest to the index of the test sample in the dictionary array, which is sorted according to the indices of characters in the dictionary set. We then match the feature vector of the test sample with those of the elements within a certain boundary around the position.

The pattern matching process can be summarized as follows:

MDV=min(S.F-D[k].F),

Where;

$$S.F\text{-}D[k].F = \sum_{j=1}^{6*N-2} |S.F_j - D[k].F_j| \qquad (17)$$

Here, S is the test sample.

6. CONCLUSION

In this study we have developed an algorithm for the recognition of Turkish characters. In the recognition algorithm, before feature extracting stage we have used preprocessing techniques; normalization and thinning to increase the recognition rate of the system. In the feature extraction stage the VH2D method was used. This method has several advantages:

Good features of characters have been extracted, which are very important in resulting in a high classification rate.

The problem of image centralization has been considered without an actual process of image transformation, thus, a large amount of computation time is avoided.

REFERENCES

[1] A. Güdesen, 'Quantitative analysis of preprocessing techniques for the recognition of handprinted characters, Pattern Recognition, vol.8, pp 219-227, 1976.

[2] R.C.Gonzolez, P.Wintz, Digital Image Processing, Addison Wesley, 1987.

[3] T.Y.Zhang, C.Y.Suen, 'A fast parallel algorithm for thinning digital patterns', Communications of the ACM, vol.27, No.3, pp. 236-239, 1984.

[4] H. D. Cheng , D. C. XIA, 'A Novel Parallel Approach to Character Recognition and its VLSI implementation, Pattern Recognition, vol.29, pp 97-119, 1996.