Optical Character Recognition Without Segmentation

Mehmet Ali Özdil, Fatoþ T. Yarman-Vural, Nafiz Arica
Department of Computer Engineering,
Middle East Technical University, Turkey
E-mail : vural@ceng.metu.edu.tr

Abstract : In this study, an off-line optical character recognition scheme is presented. The proposed method performs the recognition by extracting the characters from the whole word, thus, avoiding segmentation process which is the most significant source of error in the recognition process.

A set of *control points* which is defined by the position and attribute vectors are selected as features. In the training mode, each sample character is mapped to a set of control points and is stored in an archive which belongs to an alphabet. In the recognition mode, first, the control points of the input image are extracted. Then, each control point is matched to the control points in the alphabet according to its attributes. During the matching process a *probability matrix (PM)* is constructed which holds some matching measures (probabilities) for identifying the characters. Experimental results indicate that the proposed method is very robust in extracting the characters from a cursive script and it is insensitive to noise.

1. INTRODUCTION

Off-line character recognition can be performed by either *analytic strategy* or *holistic strategy* [1]. Holistic strategy employs a top down approach for recognizing the full word, eliminating the segmentation problem. The price for this computational saving is to constrain the character recognition systems to a limited vocabulary. Also, due to the complexity introduced by the whole cursive word compared to the complexity of a single character or stroke, the recognition rate may be relatively low for a reasonable size of vocabulary (e.g.: 1000 word).

On the other hand, the analytic strategy employs bottom up approaches starting from stroke or character level and going towards producing a meaningful text [2], [3], [4], [5]. Explicit or implicit segmentation algorithms are required for this strategy, not only adding extra complexity to the problem, but, also introducing segmentation error to the system. However, once the characters in a whole word are segmented, the problem of optical character recognition is reduced to the recognition of simple isolated characters or strokes. This leads to high recognition rates for an unlimited vocabulary .

Even though some constrained algorithms are suggested, segmentation of cursive script into characters is still an unsolved problem. Especially, for cursive handwriting, the available techniques such as [6] are based on heuristics and handle segmentation problem under heavy constraints.
In this study, we present an optical character recognition scheme for cursive script which eliminates the segmentation problem, completely. Section 2, gives some

background definitions which are used in the solution stages. Section 3, explains the stages of the proposed scheme with some examples. Finally, section 4, concludes the paper by discussing the results, indicating the robustness of the proposed method in extracting the characters from a cursive script.

2. BACKGROUND

In this section, some definitions for an image matrix $I = [x_{ij}]$, where $x_{ij} \in \{0, 1\}$ is the pixel value of a scanned document, will be presented.

Definition 1: *A control Point (CP)* denotes a pixel x_{ij} with an attribute vector $\underline{a}_k$ and a position vector $\underline{p}_k$, in a given image I, $CP(k) = \{ \underline{a}_k, \underline{p}_k \}$. Attribute vector is formed by selecting features from a feature set. Position vector is defined as the (x,y) coordinates of a control point relative to the mean value of the given image (Figure 1). The mean value of an image denotes the center of mass of that image which is computed as the arithmetic mean of the (x,y) coordinates of the boundary pixels.

Definition 2: Given a character $C(m) = \{ CP(0), CP(1), ..., CP(M-1) \}$, in an alphabet $A = \{ C(0), C(1), C(2), ... , C(N-1)\}$, where M is the total number of control points in character C(m) and N is the total number of characters in the alphabet, the *character mean* $\underline{m}_m$ is defined as the center of mass of the character and computed as the arithmetic mean of the position vectors of the control points that belong to the character,

$$\underline{m}_m = 1/M \sum_{i=1}^{M} \underline{p}i \ , \qquad m=0,.....,N-1$$

The character search in a given image I is accomplished by searching the $\underline{m}_m$ values of characters. In other words, once an x_{ij} pixel is identified as character mean, then a character is recognized at that location. The reason for selecting mean value of the character as a reference point for the position vectors is because of its relative rigidity to character variations in handwriting.

Definition 3: *Probability Matrix* PM(m) of a character C(m) defined over I, is a matrix that holds the measure of existence of a character at each pixel in I.

$$PM(m) = [q_{ij}(m)]$$

where $q_{ij}(m)$ is the probabilitye of $\underline{m}_m$ being at pixel x_{ij}. The entries of PM(m) is uniquely determined for each C(m) by an algorithm that will be explained in the next section. The character recognition scheme, proposed in this study, is based on calculating the PM(m) for each character over I and identifying the maximum $q_{ij}(m)$ values for the alphabet A . Location of the maximum $q_{ij}(m)$ values indicates the center of mass of the characters that exist in the image.

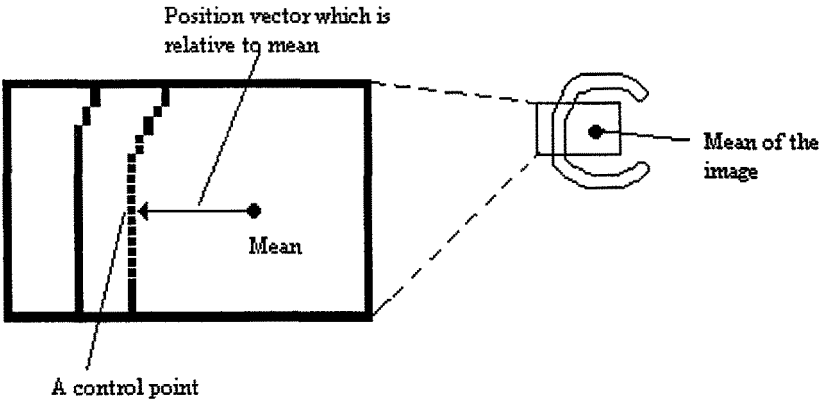


Figure 1. A sample CP representation in alphabet archive

3. SOLUTION STAGES

In our approach, there are 4 main stages which will be explained in the subsequent sections.

3.1 FEATURE EXTRACTION AND FORMING THE ALPHABET ARCHIVE

It is intuitively clear that most of the information about a character or word is held in the boundaries. Therefore, each word can be successfully, represented by its boundary. The boundary pixels of binary image are extracted and coded by Freeman' s chain code (Figure 2).

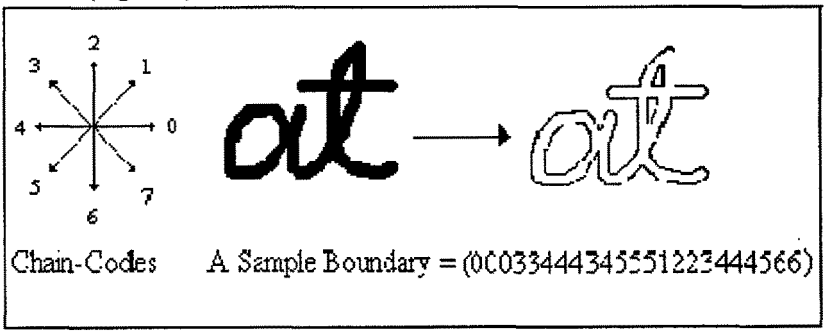


Figure 2. Boundary of the word "at" and its Freeman's chain code

In this study, the boundary points are directly taken as the CP' s. The chain-code value is considered as the attribute of a control point. Therefore, the image is converted to a CP list which is in the form of

$$I = \{ CP(0), CP(1), \ldots, CP(R-1) \}$$

where $CP(k) = (\underline{p}_k, \underline{a}_k)$, $\underline{p}_k = (x_k, y_k)$ is the position vector of CP(k) which is relative to the mean value of the image , $\underline{a}_k$ is the chain-code value and R is the total number of the control points in I .

In the training mode, each sample character is mapped into a set of control points and stored in the alphabet archive. An alphabet archive includes some additional look-up tables which supply a fast access for the control points that share the same attribute.

3.2 CONSTRUCTION OF PROBABILITY MATRICES

Probability Matrices are constructed by using the inverse position vectors with respect to the character means, in the alphabet archive (Figure 3). The following rules are employed for this purpose.

Rule 1: Given a document image $I = \{ CP(0), CP(1), \ldots , CP(R-1) \}$ to be recognized with an alphabet A , where $C(m) \in A$ and a collection of CP(k) represents character C(m) , if the attributes of $CP(k) \in C(m) \in A$ matches to the attributes of $CP(l) \in I$, then, there may be a character mean $\underline{m}_m$ around the $CP(l) \in I$.
When a control point in I is matched to the control point of the characters in A, the inverse position vector of $CP(k) \in C(m) \in A$ is carried to I to find the location $\underline{m}_m$. This may be considered as an inverse searching. That is, we don't search for the character in the document, but the character mean which belongs to the corresponding the CP's in A. Each particular match causes some increments on the probability $q_{ij}(m)$ of the character mean around $\underline{m}_m$.

Rule 2: Let $\underline{p}_l$ is the position vector of the CP(l) e I and $\underline{p}_k$ is the position vector of the $CP(k) \in C(m) \in A$ which matches with CP(I). The increment of $q_{ij}(m)$ for each additional match of the control points is calculated by ,

$$\{ q_{ij}(m) \}_{new} = \{q_{ij}(m) \}_{old} + E_m \cdot G(\underline{m}_m, S_m , W_m)$$

Since, the character in the document may be stretched or skewed due to handwritten variations, the $q_{ij}(m)$ in the surrounding points of $\underline{m}_m$ should be, somehow, increased. A Gaussian function which has nonzero values over a window W_m,

$$G(\underline{m}_m, S_m , W_m)$$

centered at $\underline{m}_m$ with character specific covariance S_m, is used for this purpose (see: Figure 5). The size of the W_m is determined during the training mode as a function S_m. The $\underline{m}_m$ is calculated by: $\underline{m}_m = \underline{p}_l - \underline{p}_k$

In the above formulation, E_m is a positive real number and depends on the size of the character. Note that, the number of CP' s in C(m) varies by size. If this number is large, E_m takes a small value. That is, the contribution of each CP to the mean value of C(m) must be relatively smaller. If the character is big, the number of the control points get larger. In order to compensate for this, E_m is adjusted to a relatively small value.

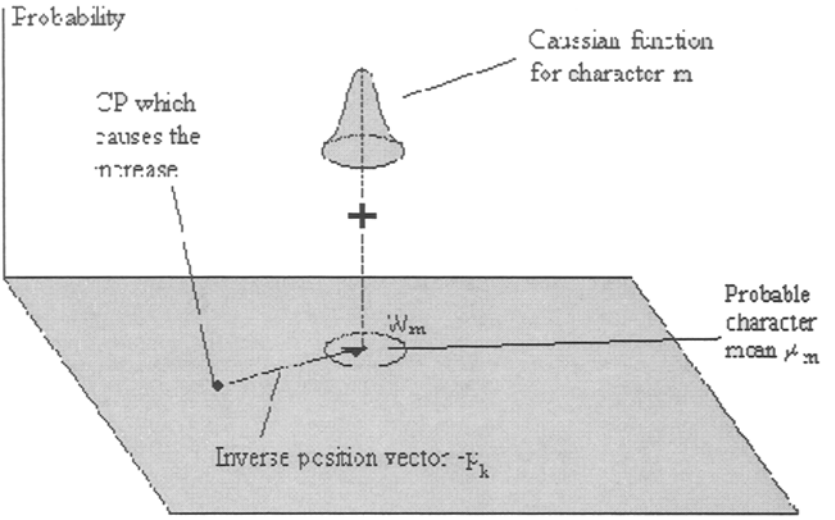


Figure 3. Computation of the Probability Matrix.

Alphabet archive is used to calculate the $q_{ij}(m)$ points in recognition mode by the following algorithm:

PM Construction Algorithm :

```
initialize PM(m) ← 0
    For each CP(l) e I
        For each C(m) e A
            For each CP(k) e C(m)
                if CP(l) matches with CP(k) then
                  Calculate the m_m.
                  Update PM(m) according to Rule 2.
                endif
            end.
        end.
    end.
```

The above algorithm computes all the PM(m)'s for the characters in the alphabet. It is intuitively clear that the mean values of the most probable character takes relatively large $q_{ij}(m)$ values with respect to the others. An experimental result indicating the Probability Matrix are shown in Figure 4.

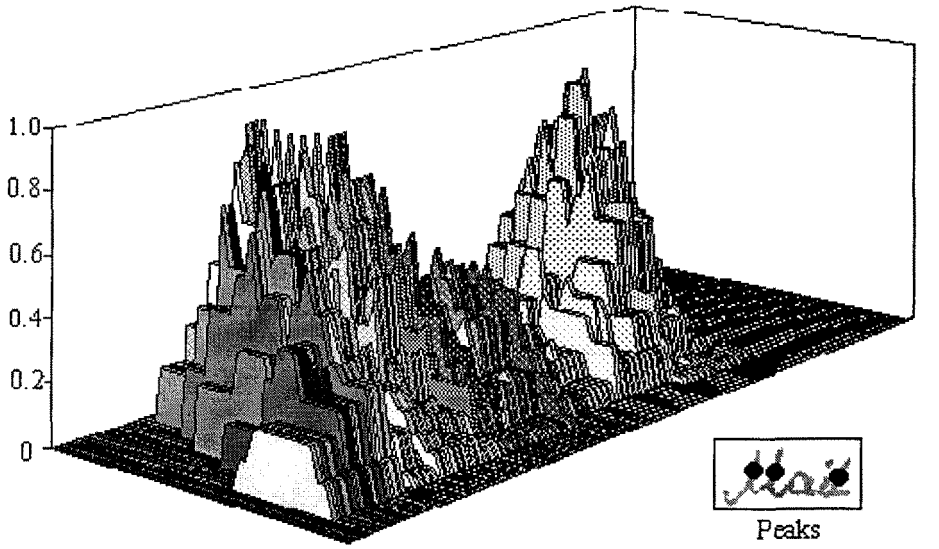


Figure 4. Probability Matrix for the character "I".

3.3 CHARACTER SEARCH ON PROBABILITY MATRICES

A character candidate point, is a point which has the local maximum q_{ij} value on the PM(m). Character search stage finds these points and constructs the Character Map List (CML).

In Figure 4, an example for the maximum values of the peaks correspond to the character candidates. We say candidates, because these peaks may be inaccurate character mappings. The final decision is made in the next stage which resolves the overlaps. The peak values are taken into account in the CML by a scanning algorithm. The heights of the peaks give the matching quality of the candidates.

CML is a list which keeps the information about the candidate characters. An item of this list contains a pointer to the character, the position of the mapping on $\underline{m}_m$, probability and an assignment list. The assignment list holds the pointers to CP(k)s which belongs to the mapping item and is constructed by means of a look-up table which is built during the execution of PM Algorithm.

3.4 FINDING AND RESOLVING THE OVERLAP-GROUPS

It is assumed that the CML contains only the candidates of true mapping. This list must be somehow resolved to make the final decision about character selection. Before doing this, a measurement for overlapping rate is defined as follows:

Definition 4: Overlapping Rate (OR) is a measurement of intersection between two CML items. Given two CML items Ui and Uj, the Overlapping Rate for Ui and Uj which is denoted as OR_{ij} is calculated by:

if $size(U_i) < size(U_j)$ **then**

$OR_{ij} = size(U_i \cap U_j) * 100 / size(U_i)$

else

$OR_{ij} = size(U_i \cap U_j) * 100 / size(U_j)$

endif

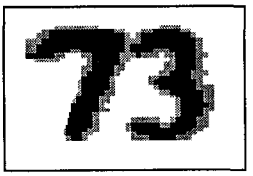
where $size(U_i)$ represents the sum of the CPs in CML item U_i.

As it can be seen from the above, overlapping is measured with respect to the smaller mapping. The overlapping rates of CML items are stored in a matrix which is named as Character Overlapping Matrix (COM). This matrix keeps both *counts* and *rates* of the overlapping as below:

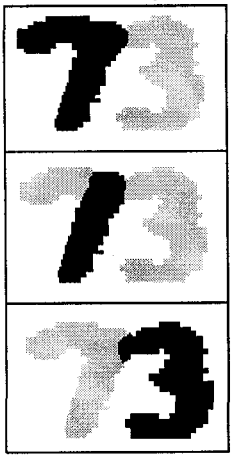
$$COM = [(count, rate)_{ij}]$$

where count is the amount of CP' s those are shared by both CML items i and j, rate is the overlapping percentage of *ith* and *jth* CML items.

Two CML items overlap with each other, if the overlapping rate of these items exceed a threshold value which is given as a parameter to the system. When the rate of two CML items exceeds the threshold value, only one of them may be a true mapping.



Input image



CML item representation for characters "7", "1" and "3"

Overlapping Matrix

(%)	7	1	3
7	100	100	5
1	100	100	5
3	5	5	100

Figure 5. An overlapping group with its CML items and overlapping matrix.

In Figure 5, an image of two touching numerals "73" is shown. At the end of PM Algorithm, three CML items for numerals "7","1" and "3" and their COM values (only rates) are obtained. For this particular example, an overlapping threshold of 20% is given. According to the COM, there is no intersection on CML(3). So, we can directly map the CP's of CML(3) to ASCII "3". However, for the rest of the image, a decision should be made, because CML(7) and CML(1) are overlapping with each other. In order to resolve the overlapping we need the definition of overlap-group,

Definition 5 : Overlap-Group (OG)

Let T_k e CML, for k=1,.....,S where S is the total number of items in CML. OG is a subset of CML if there is a sequence of overlapping $T_i \cap T_j \cap \ldots \cap T_m$ for all $T_i, T_j \ldots, T_m$

e OG. In Figure 5, there are two OG's, first is {CML(7), CML(1)} and the second is {CML(3)}.

Each OG needs to be resolved to find the ultimate solution. A subset of an OG which contains only non-overlapping CML items with the maximum matching quality and the maximum number of control points is selected as the final decision of that overlapping group. In our example, "7" and "1", represent individual solutions. Since, the number of CP' s in "7" is much more than "1", although, their match qualities (probabilities) are nearly equivalent, "7" is selected. The other OG contains only one CML item and is directly mapped to "3". Therefore, the ASCII result is found as "73" in this example.

4. CONCLUSION

In this study, a new approach is presented for handwritten cursive script which extracts the characters from the whole word. The proposed method does not need any preprocessing stage to eliminate the noise, to correct slant or to normalize the writing. It is very robust to handwriting variations and noise such as broken characters, spurious branches, ink smearage and leakage; because this type of noise does not distort the structure of the probability matrix which is the core of the proposed OCR system. The method yields 99% recognition rate for printed material, regardless of the quality of the paper and printout.

REFERENCES

[1] S. Impedevo, " Fundamentals in Handwtiting Recognition", Proc. on NATO-ASI series, Springer Verlag. 1993.

[2] Y.Bengio ,R. De Mori ,G.Flammia ,R.Kompe, "Global Optimization of a Neural Network - Hidden Markov Model Hybrid" IEEE Trans. on Neural Networks, Vol:3., No:2, pp: 252-259, 1992.

[3] O. Agazzi, S. Kuo, "Hidden Markov Model Based Optical Character Recognition in the Presence of Deterministic Transformations", Pattern Recognition, Vol. 26, No. 12, pp 1813 - 1826, 1993.

[4] L. Yang, R. Prasad, "Application of Hidden Markov Models for Signature Verification", Pattern Recognition, Vol. 28, No. 2, pp 161-170, 1995.

[5] Sid-Ahmed & F. El-Khaly, "Machine Recognition of Optically Captured Machine Printed Arabic Text" in Pattern Recognition, Vol: 23, No: 11, pp: 1207-1214, 1990.

[6] F. Yarman- Vural, A. Atici, " A Segmentation and Feature Extraction Algorithm for Arabic Script", SPIE, Visual Communication and Image processing, vol 2, pp725-738, Orlando, 1996.