EFFICIENT AND FAST DISTORTED CHARACTER RECOGNITION ALGORITHM

Wejdan A. Amer
Department of Computer Science, College of Science, University of Baghdad. Baghdad- Iraq

Abstract

An algorithm to recognize distorted Machine and handwritten characters is proposed. It uses a feature point extraction-based recognition approach. A new verification scheme, which deals with this problem, is presented. A new feature extractor set is introduced. This research explores best sets of feature extraction techniques and studies the accuracy and speed of the suggested procedures. Finally, experimental results on one database are presented, their results showing the robustness of the algorithm using small database.

خوارزمية كفوءة وسريعة لتمييز الحروف

وجدان عبد الامير حسن
قسم علوم الحاسبات، كلية العلوم، جامعة بغداد. بغداد- العراق

الخلاصة

تم اقتراح خوارزمية لتمييز الأحرف المطبوع المشوهة والمطبوعة يدوياً. حيث تستخدم طريقة تعتمد استخراج خواص النقطة. أسلوب تحقق جديد ليتعامل مع هذه المشكلة تم تحضيره. مجموعة خواص جديدة تم تقديمها في البحث. هذا البحث يستغل افضل مجموعات استخراج الخواص ويدرس الدقة والسرعة في الإجراءات المقترحة. وأخيرا، نتائج عملية على قاعدة بيانات تم اعدادها، والتي تشير الى قوة الخوارزمية المقترحة باستخدام قاعدة بيانات صغيرة.

Introduction

Optical character recognition (OCR) has been a topic of interest since possibly the late 1940's when Jacob Rabinow started his work in the field [1]. The earliest OCR machines were primitive mechanical devices with fairly high failure rates. Due to increasing in amount of new written material increased, the need to process it all in a fast and reliable manner. They quickly gave way to computer-based OCR devices that could outperform them both in terms of speed and reliability.

Today there are many OCR devices in use based on a plethora of different algorithms. All of the popular algorithm sport high accuracy and most high speed, but still many suffer from a fairly simple flaw: when they do make mistakes (and they all do), the mistakes are often very unnatural to the human point of view. That is, mistaking a "5" for an "S" is not too surprising because most people are willing to agree that these two characters are similar, but mistaking a "5" for an "M" is counter-intuitive and unexpected. Algorithms make such mistakes because they generally operate on a different set

of features than humans for computational reasons. Humans observe strokes and the relations between them, while algorithms measure anything from Transformation Ring Projections of a character to the Fourier Transform of the Horizontal-Vertical Projections of a character [2]. These methods do work and are often computationally efficient, but they make the computer see letters through a decidedly non-human set of eyes [3].

The importance of making the same sorts of mistakes as a human may not be immediately obvious, but it is important to realize that the main purpose of OCR is to facilitate communication between humans. Mistakes typical of humans can be more readily corrected by humans (i.e. "5ave" is easier to connect with "Save" than "Mave").

This paper describes an algorithm that attempts to work with a subset of the features in a character that a human would typically see for the identification of machine-printed or handwritten characters. Its recognition rate is currently not as high as the recognition rates of the other algorithms, more developed character recognition algorithms, but it is expected that if it were expanded to work with a larger set of features this problem would be removed. If it were expanded to use more features, it would be made correspondingly slower; with the advent of faster microprocessors this fact is not viewed as a crippling problem.

Extracting feature points thus reduced to calculating a number between zero and one to describe a pixel's neighborhood and then comparing that number against a database of known feature points. Missing feature points is certainly not a limiting factor in the algorithm's accuracy. It also does not suffer from labeling too many uninteresting points as being feature points. The feature point extractor is thus fast and reliable [4].

OCR system consists of four major stages:

1) Pre-processing, 2) Binarization,
3) Feature Extraction, 4) Recognition.

In the Pre-processing, the raw data is subjected to a number of preliminary processing steps to make it usable in the descriptive stages of character analysis. Pre-processing aims to produce data that are easy for the OCR systems to operate accurately. The main objectives of pre-processing are [5]:

- Binarization
- Noise reduction
- Stroke width normalization
- Skew correction
- Slant removal

The presently proposed scheme in Pre-processing step focus on binarization then feature extraction and recognition (verification) implemented by assuming that the input ten by ten data is not particularly aberrant -- lines in an ten by ten grid should not normally be thicker than one pixels. With this assumption, it then proceeds to look for feature points.

Method description

The ability to identify handwritten or machine printed characters in an automated or a semi-automated manner has obvious applications in numerous fields. Since creating an algorithm with a one hundred percent correct recognition rate is quite probably impossible in our world of noise and different font styles, it is important to design character recognition algorithms with these failures in mind so that when mistakes are inevitably made, they will at least be understandable and predictable to the person working with the program [4]. The current work explores one such algorithm and tests it on same font. The results are discussed and several improvements are suggested.

The procedure for extracting feature points utilized by current algorithm is fairly straightforward. Fig.3 shows character in its array 0 and 1 for each pixel. Since an eight by eight character consists of only sixty-four pixels, it is viable to simply loop through the entire character and examine each pixel in turn.

Characters cannot be identified by the extraction of feature points alone. Without a database of characters and their associated feature points, the ultimate feature point extractor would be useless. Only with such a database can the feature point extraction results from an unknown character be compared against what is expected for real-world characters and a judgment of the unknown's identity made. Thus a "gold-standard" dictionary of characters and their associated features must be defined. Ideally this dictionary should contain details for the average appearance of every character manifestation (many English characters have multiple different accepted manifestations -- note "Z" versus "Ƶ"). If poor representative's appearances for characters are chosen valid characters at the extremes will not be identified as readily. If some manifestations of characters are missed, the program will certainly not be able to identify characters belonging to these groups at all.

With both methods for extracting feature points and a dictionary of characters and associated feature point data for reference, identifying characters becomes a problem of measuring the degree of similarity between two sets of features. The method employed by this algorithm is just a slight matching of set of eight strings each time. All the string sets each of the feature points in the unknown character and their closest corresponding feature points in the reference character are summed and missing or extra feature points are penalized. Identification is then a matter of finding the character in the dictionary that is within a certain threshold matching of the unknown character. In practice, the algorithm currently checks every character in the reference set to first locate the maximum matching, and then verify that the maximum matching is less than the threshold. Additionally, the algorithm assumption no noises present.

Suggested Algorithm

The algorithm of the method demonstrated in details in the following flow chart diagram (for more details see a pseudo code listed in appendix A.

Fig. 1 show general block diagram recognition, while Fig 2 show learning block diagram.

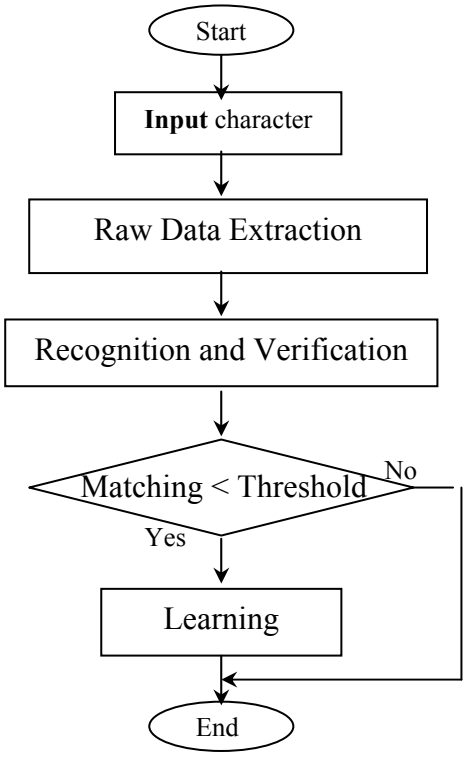

Figure 1: general character recognition block diagram

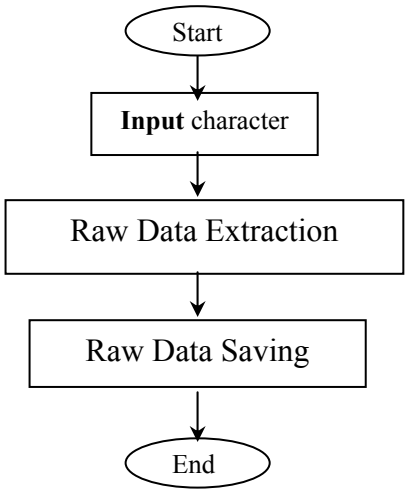

Figure 2: Learning block diagram

The following lines demonstrate ambiguous variables of the pseudo code showed in the next pages:

Wid is image width, **Hgt** is image height,
Xb is X coordinates beginning of the character
Xe is X coordinates beginning of the character
Yb is Y coordinates beginning of the character
Yb is Y coordinates beginning of the character
strText is character/symbol type, **Bool1stScan** is character image scanning flag, **strData** is character string (i.e., 0 or 1 for each pixel represent character).

The variables in Fig 2 showed in the following:
StrRecognized is recognized character, **intMatch** is matching percentage, **intMaxMatch** is threshold/maximum allowed matching parentage, **C** is counter of saved character string within data base file, **arrRawData** is charcter string saved in data base file, **arrTagData** is character type/definition, d is counter dedicated to elements for each character string to do matching.

Algorithm Begin:

Step 1:

(Pre-Processing and character Feature extraction process)

The following list code shows our method of extracting data (Character features extraction):-

Step 1-1:

Read Image pixels and put it in an array (Zero value for black pixel and one value for white pixel),
let be $A_{x,y}$ denote original character array:

where Wid is character image width, and
Hgt is character image Height

Step 1-2:

Variable: scan = True
Let Yb=0: Ye=0: Xb=0: Xe=0 ,
Where: Yb, Xb denote to **Y** and **X** beginning of character coordinate, and
Xb, Xe denotes **X** and **Y** ending of character coordinate.

```
For X=0 to Wid-1 Do begin
For Y=0 to Hgt-1 Do begin
      If A X, Y = 0 then begin
         If scan = False then
            If Y < Yb Then Yb = Y
            If Y > Ye Then Ye = Y
            If X < Xb Then Xb = X
            If X > Xe Then Xe = X
         Else
           Set scan = False
            Yb = Y
            Ye = Y
            Xb = X
            Xe = X
         End If
      End If
End y, x
```

Step 1-3:

```
If Xe - Xb <> 0 And Ye - Yb <> 0 Then
  For X = Xb To Xe
   For Y = Yb To Ye
     If A X, Y = 0 then
          DataString = DataString & 0
     Else
          DataString = DataString & 1
     End If
End Y, X
```

Step 2:

(Character Recognition (Verification))

The following list code show our suggested method of data matching (Character Identification)

Step 2-1:

Open Data_Base File and get Data from first Raw
Set variable j =0 : MaxMatch =0
Do while Not End Of Data_Base File
Get first raw data and put it in a string of array denoted by: $CharArr_j$
$CharIndex_j = Mid(CharArr_{j,1,1})$
(Retrieve First character (Character Symbol) in $CharArr_j$ string)
Example: $CharArr_{j,1,1}$ =
A,11111111111111111111111111111001111100011110001101100111110111100011011111111000111111111111001111111111
Then $CharIndex_j$ =A
Where: **Mid** is a function in VB6 compiler extract each element exist in string j=index, 1=start position, 1=char length
$CharArr_j := CharArr_{j,3}$
(reminder characters (Character String) in $CharArr_j$ string)

Step 2-2:-

Do while Not End Of Data_Base File
Get first raw data and put it in a string of array denoted by : $CharArr_j$
$CharIndex_j = Mid(CharArr_{j,1,1})$
(Retrieve First character (Character Symbol) in $CharArr_j$ string)
$CharArr_j := CharArr_{j,3}$
(Reminder characters (Character String) in $CharArr_j$ string)

Step 2-3:-

```
For i = 1 To 10 Do begin
 For j = 1 To 10 Do begin
  If Mid(CharArr (c),d+1,1)= 0 Then begin
   If Mid(DataString,d+1,1)= 0 Then begin
        Match = Match + 1
   Else
         Match = Match - 1
   End If
  Else

 If Mid(DataString,d+1,1)<>0 Then begin
         Match = Match + 1
 Else
         Match = Match - 1
 End If
End If
d = d + 1
End j, i
```

Step 2-4:-

```
If MaxMatch < Match Then begin
      MaxMatch = Match
      RecogSt = CharIndex(c)
     (RecogSt == recognized string)
      If  MaxMatch > 60 Then
         GoTo  Step 2-6
      End If
End If
c = c + 1
```

Step 2-5:-
While Not End Of DataBase File Goto Step **2-2.**

Step 2-6:-
Data Base File Close
Print "The highest possible of drawn character is recognized as ' " & strRecognised & " ' = " & intMaxMatch & "%"

Algorithm End.

Fig. 3 show array of a symbol "B" character. Fig. 4 show results for three type distorted character recognition, where 100% mean the system was learning on this character.

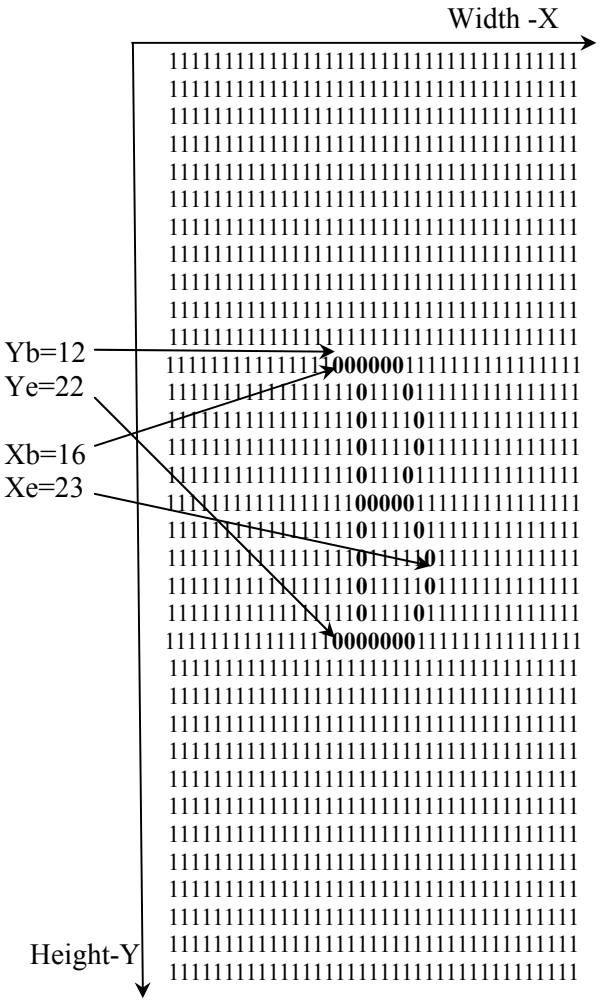

Figure 3: B character image string (i.e., each pixel is 0 or 1.

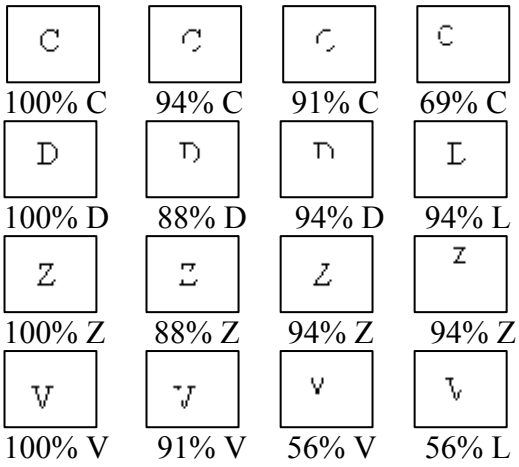

Figure 4: Three type distorted character recognition results.

Conclusions

The results of matching percents were showed in Fig. 5 & 6, but in general they did make numerous mistakes. More than half the mistakes were understandable considering the difference in font styles or general character appearance (distinguishing between a "l" and an "I" or a "?" and a "Q" sometimes has to be done by context even for humans). The remainders were understandable when the nature of the features was considered. These mistakes provide the information needed to improve the algorithm.

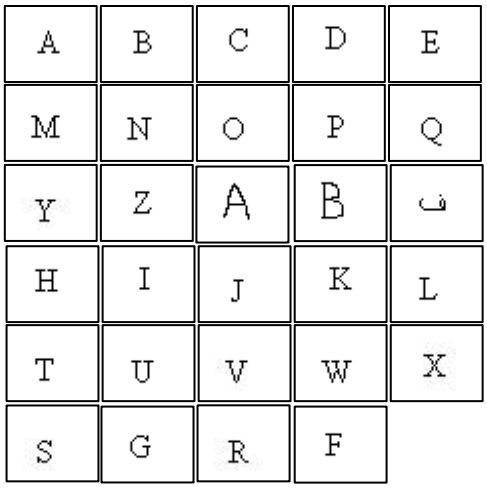

Figure 5: Original Characters/images (matching is %100)

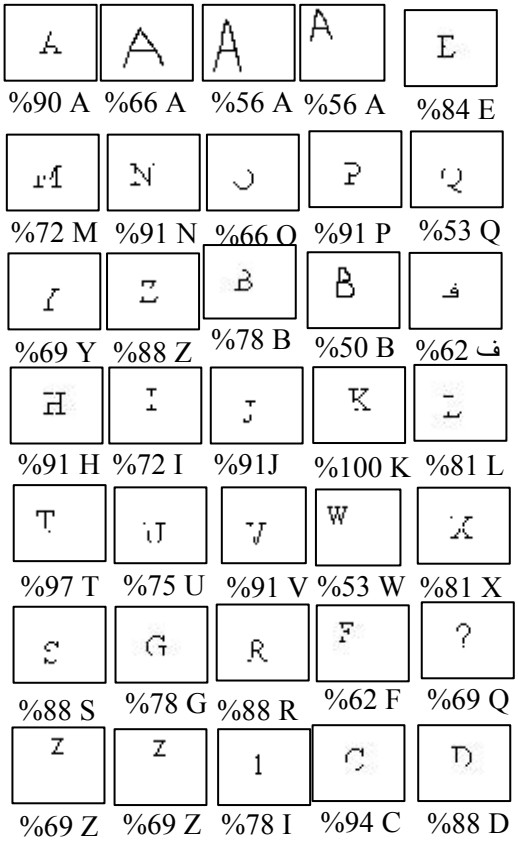

Figure 6: Distorted Characters Recognition Results with their matching percent ratio (average matching ratio for all above characters is %61).

Several improvements are possible. In the current algorithm, the different types of feature points ("T" intersections, "Y" intersections, ends, etc.) are all treated as generic feature points. This approach was taken because not every type of feature point can be unambiguously defined for a given eight pixel.

Overall the results of this experiment were mixed. On the one hand, the initial results certainly are not of commercial quality. When only a couple of pixels differed between the unknown character and the reference, the results were fairly good, but larger differences often made the algorithm unable to correctly identify the unknown character. On the other hand, the low success rate is not indicative of the general algorithm, just the current implementation.

Suggestion and feature works

Suggested future work includes both the testing of the algorithm with more character data. Of particular interest would be character data that is deliberately noisy and character data that has been reduced to eight by eight resolutions from some greater resolution. Both of these cases reflect real-world problems.

There are many possible changes that could vastly improve the algorithm's recognition abilities. With a few of these changes implemented, the mistakes the algorithm would make would indeed be very similar to the types of mistakes humans would make. Thus general algorithm holds promise as a character recognizer that identifies characters in a manner similar to the way that humans identify characters.

Another future work could be summarized as follow:

- Creating new hierarchical classification schemes based on rules after examining the corresponding confusion features.
- Exploiting new features to improve the current performance.

References

1. Vamvakas,G.; Stamatopoulos ,N.; Gatos, B.; Pratikakis, I. and Perantonis,S.J. Standard Database and Methods for Handwritten Greek Character Recognition. **2007**. The 11th Panhellenic Conference on Informatics. pp.64-71.
2. Hiroshi Shimodaira, Takashi Sudo and Mitsuru Nakai. Online Overlaid-Handwriting Recognition Based on Substroke HMMs. **2003**. Seventh International Conference on Document Analysis and Recognition Volume II. P.102-108.
3. Bourbakis, N. G. and Gumahad II, A. T. Knowledge-Based Recognition of Typed Text Characters. **1991**. Character & Handwriting Recognition: Expanding Frontiers, World Scientific Publishing Co. pp.12-17.
4. Toscano, K., Sanchez, G., Nakano, M., Perez, H., Yasuhara, M. Novel Cursive Character Recognition System. **2006**. Artificial Intelligence. Fifth Mexican International Conference on Volume, pp. 101–110.
5. Vamvakas,G.; Gatos,B.; Pratikakis,I.; Stamatopoulos,N.; Roniotis, A. and Perantonis,S.J. Hybrid Off-Line OCR for Isolated Handwritten Greek Characters. **2007**. The Fourth IASTED International Conference on Signal Processing, Pattern Recognition, and Applications. pp. 197-202.