A Methodology for Special Symbol Recognitions

Jisheng Liang[1] Ihsin T. Phillips[2] Vikram Chalana[1] Robert Haralick[3]

[1] MathSoft, Inc. 1700 Westlake Ave N, Suite 500, Seattle, WA 98109, U.S.A.

[2] Department of Computer Science/Software Engineering, Seattle University Seattle, WA 98122 U.S.A.

[3] Department of Electrical Engineering, University of Washington Seattle, WA 98195 U.S.A.

{jliang, yun, haralick@george.ee.washington.edu}

Abstract

This paper presents a special symbol recognition system that incorporates the result of an OCR to recognize the special symbols those not handled by the current commercial OCR systems. Given a document image and the OCR output, we first refine the character coordinates produced by the OCR. Then, the special symbols are distinguished from the normal characters. Finally, we compute the features from the special symbol sub-images and a supervised classifier is used to assign the sub-images to one of the predefined special symbol categories. The system was tested on 5516 images from the National Library of Medicine. The evaluation results are reported in the paper.

1. Introduction

Optical character recognition (OCR) is a success story among the applications of the field of computer vision and pattern recognition [4]. For example, most of the commercial OCR systems on the market today can produce nearly perfect results on quality printed documents and can yield over 90% accuracy rate on moderately degraded documents. However, when in the domain of special symbols, these same systems can fail in miserably. The reason is simple – these systems were not trained to recognize the special symbols. As a result, when encountering a special symbol, such as a Greek letter or a mathematical symbol, most OCR systems do not know it is a special symbol. It would either recognize it as one or more of its regular symbols with a low confident level or assign it as a "non-recognizable" symbol. The task is left for the operator who is assigned to cleaning-up the OCR errors, manually.

The reason for not developing an OCR system that handles the special symbol is obvious. First of all, there is no commercial market there to support the additional development of an OCR system that handles the special symbols. Second, by including the special symbols into the recognition engine, the speed of the system will be slower. This is not very desirable in the commercial world. As a result, without a special symbol recognition system, the conversion of documents from paper format to electronic format remain costly for those documents that contain a substantial amount of special symbols. This paper presents a special symbol recognition system that incorporates the result of an OCR to recognize the special symbols of those not handled by the current commercial OCR systems.

Our recognition system consists of three major modules: character segmentation, special symbol detection, and special symbol classification modules. The system architecture is shown in Figure 1. The description of these three modules are given in Section 2, 3, and 4 respectively. Our experimental results are given in Section 5.

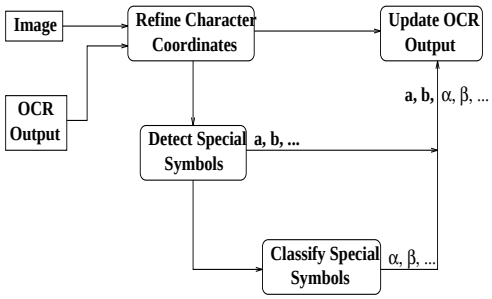


Figure 1. System Architecture.

2. Character Segmentation Refinement

Inputs to the character segmentation module are – a binary document image and the output of a commercial OCR system. The output of the OCR system includes character strings, word boxes, text-line boxes and text-block boxes. First, we compute the set of bounding boxes of the connected components from the input image. Then, we find the correspondences between the computed connected component bounding boxes and the word boxes (of the OCR), and the correspondences between the characters (of the OCR) and the connected components. A character may correspond to one connected component (one-to-one match), or two or more components (one-to-many match). Or, two or more characters may correspond to one connected component (many-to-one match). We use the "relative position"

Proceedings of the International Conference on Pattern Recognition (ICPR'00)
1051-4651/00 $10.00 @ 2000 IEEE

of the characters as a clue for finding the correspondences between the characters and the connected components. The following is the formal definition for the correspondence matching problem.

Problem Statement: Let $A = (a_1, \cdots, a_N)$ be a sequence of characters and let $B = (b_1, \cdots, b_M)$ be a sequence of connected components, the problem is to decompose (split or merge if necessary) the elements of B into a sequence $G = (g_1, \cdots, g_N)$ of glyphs, such that each element of G is associated with a character in A,

$$[(g_1, a_1), (g_2, a_2), \cdots, (g_N, a_N)],$$

that minimizes the criterion function D(F(A), F(G)), where $F(X)$ is a sequence of features of X, and D is the distance measurement between F(A) and F(G). We define $F(X)$ as a transformation that converts each character in X to a character position class. The character position class is defined as the classes follows:

- C 0: within baseline and x-height, e.g. a, c, e, o.
- C 1: that extends above x-height (ascender), e.g. b, A, C, P.
- C 2: character that extends below baseline (descender), e.g., p, g, q
- C 3: character that extends above x-height and below baseline, e.g. {, }.
- C 4: superscript or higher punctuation mark.
- C 5: subscript or lower punctuation mark.

The computation of the character position sequence for a character string is straight forward. However, the computation of the position feature sequence for a sequence of connected components within a word is not trivial. Let a connected component be represented as (x_1, y_1, x_2, y_2), the coordinates of the upper-left and the bottom-right corners of it's bounding box. Taking the bottom-right corners of all the connected component boxes within a text-line, we use a robust line-fitting algorithm [2] to estimate the text line's baseline coordinates. Then, the x-height is estimated from the distance of all the components' upper-left corners to the baseline. The character position class is assigned to each component based on the position of its bounding box with respect to the detected baseline and the x-height of the text-line. Instead of using a global threshold, we build an adaptive classifier for each text-block to determine the character position feature vector for the connected components. A binary tree classifier is adaptively trained given the computed position features and their known position classes. At each node of the classification tree, we search for the best threshold values of the decision rule by minimizing the number of misclassification errors. These feature strings are then matched with each other to decide which connected components correspond to which characters.

3. Special Symbol Detection

A special symbol is usually recognized by an OCR system as a short string of one or more regular characters, where characters within the string, in general, are given low recognition confidence levels by the OCR system. We collect a set of potential special symbol strings among the short strings produced by the OCR. (We consider a string having less than 4 characters long with low character confidence levels as a potential special symbol string.) For each potential special symbol string, we compute the posterior probability of this string being a special symbol, base on the confidence levels of the characters within the string. Using the computed probability, the character on the left and the character on the right of the string, we compute a list of possible special symbol candidates for the string. The actual assignment is done by the classification module (given in Section 4.) Our special symbol detection method is given as follows.

Let the observed character string be $X = x_1 x_2 \cdots x_m$ $m \leq 3$. Each x_i is associate with a pair (a, c), where a is a character and c is the OCR confidence level for the character. The probability that a special symbol $s \in S$ (a known special symbol set) has caused the OCR to produce the string X can be expressed by the use of Bayes' rule as,

$$P(s|X) = \frac{P(X|s)P(s)}{P(X)}. \tag{1}$$

$P(X|s)$ is the probability of observing X under the condition that s is a certain symbol. $P(s)$ is the a priori probability of s, and $P(X)$ is the probability of the character string X.

In the training step, for each given sequence of characters and the confidence levels of the characters, the probability that this sequence of characters is indeed a certain special symbol is calculated. A probability look-up table is constructed for all special symbols $s \in S$.

The context information is also very useful in determining whether a sequence of characters is actually a special symbol. For example, the special symbol "γ" is often recognized as "y" with relatively high confidence. It causes posterior probability $P(s = \gamma | a = y, c)$ to be very low and the symbol "γ" is missed. If we can observe from the data that the probability of "γ" followed by "-" is greater than the probability of the character "y" followed by "-", this context information can be used to update the posterior probability and to detect the symbol "γ".

Let the observed character string before X be $X^- = x_1^- x_2^- \cdots x_n^-$. Let the observed character string after X be $X^+ = x_1^+ x_2^+ \cdots x_p^+$. We use $n = p = 1$. The probability that a special symbol $s \in S$ has caused the OCR to produce X can again be expressed by the use of Bayes' rule as,

$$\begin{aligned} &P(X^-, s, X^+ | X^-, X, X^+) \\ &\quad = \frac{P(X^-, X, X^+ | X^-, s, X^+) P(X^-, s, X^+)}{P(X^-, X, X^+)}. \end{aligned} \tag{2}$$

Based on the assumption of conditional independence among X^-, X, and X^+, then

$$\begin{aligned} &P(X^-, X, X^+ | X^-, s, X^+) \\ &\quad = P(X^- | X^-) P(X|s) P(X^+ | X^+) = C \times P(X|s), \end{aligned} \tag{3}$$

where C is a constant. Therefore, the probability (2) can be approximated as

$$\begin{aligned} & P(X^-, s, X^+ | X^-, X, X^+) \\ & \propto \frac{P(X|s)P(X^-, X^+|s)P(s)}{P(X^-, X^+|X)P(X)} \\ & = P(s|X)\frac{P(X^-, X^+|s)}{P(X^-, X^+|X)} \end{aligned} \tag{4}$$

where $P(X^-, X^+|s)$ is the probability when s is a know special symbol, then its left neighbor is X^- and its right neighbor is X^+. And $P(X^-, X^+|X)$ is the probability that, given an observed character sequence X, its left neighbor is X^- and its right neighbor is X^+.

4. Special Symbol Classification

At the end of the special symbol detection module, all potential special symbol strings are given a list of special symbol candidates and the candidates' probabilities. Next step is to determine which symbol to assign to each of these strings. The classification method is as follows.

A sub-image is computed and normalized for each potential special symbol string from the input image. Then, we compute the distance from this sub-image to each of the trained probability maps. (The probability map is computed, off-line, for each special symbol $s \in S$ from the training samples.) Using the probabilities associated with the special symbol candidates as a prior, we use the Bayesian framework to update the probability of each candidate.

To achieve scale and translation uniformity, the regular moments (i.e., m_{pq}) of each image are utilized. An image function $f(x, y)$ can be normalized with respect to scale and translation by transforming it into $g(x, y)$, where

$$g(x, y) = f(\frac{x}{a} + \bar{x}, \frac{y}{a} + \bar{y}), \tag{5}$$

where $\bar{x} = \frac{m_{10}}{m_{00}}$, $\bar{y} = \frac{m_{01}}{m_{00}}$, and $a = \sqrt{\beta / m_{00}}$ [6]. The normalized image is sampled to an $n \times m$ grid.

Given a set of normalized training samples, we compute the probabilities that a symbol produces foreground value at each pixel, and generate a probability map for each symbol. A probability map is the histogram of a special symbol's normalized images within the training set. Given the image I of a special symbol S_k, its probability map T_k is computed as

$$T_k(i, j) = T_k(i, j) + I(i, j) \tag{6}$$

The values of the probability map is normalized in the range from 0 to 255. Figure 2 shows the computed probability maps for a set of 10 special symbols.

Given a normalized binary (0, 255) sub-image of a given string, we first sample the sub-image into a $n \times m$ grid. Then, we compute the "distance" between the sampled sub-image and each of the trained probability maps. We assign the map with the smallest distance to the sub-image. The

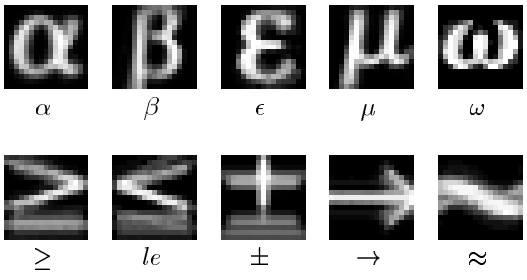


Figure 2. Illustrates the probability maps of some special symbols.

distance d between an image I and a probability map T is defined as the sum of absolute difference:

$$d(I, T) = \sum_{i}^{N} \sum_{j}^{M} |I(i, j) - T(i, j)|. \tag{7}$$

Let $D = \{d_1, d_2, \cdots, d_i, \cdots, d_N\}$ denotes the distance between an input image I and the trained probability maps $\{T_1, T_2, \cdots, T_N\}$. The likelihood that an input glyph I is indeed a special symbol s_i can be computed as

$$P(I|s_i) = \frac{1/d_i}{\sum_{k=1}^{N}(1/d_k)} \tag{8}$$

Using the probabilities of the special symbol candidates as the a prior probability, we update the probability of each candidate by observing the image features. Give a sequence of OCR produced character-confidence pairs $X = (X_1, X_2, \cdots, X_N)$, where $X_i = (a_i, c_i)$, we search for the special symbols using the following algorithm.

Algorithm 4.1 *Special symbol recognition*

For $i = 1$ to N, Do

1. For $j = 0$ to 2, Do
   (a) If $i + j > N$, Stop.
   (b) Let $\hat{X} = \bigcup_{k=i}^{i+j} X_k$.
   (c) Determine the left neighbor $X^- = X_{i-1}$ and the right neighbor $X^+ = X_{i+j+1}$.
   (d) Compute the probability that $\hat{X}$ is a special symbol $s \in S$, as the multiplication of the probabilities in Equation 4 and 8,

$$\begin{aligned} & P(X^-, s, X^+ | X^-, \hat{X}, X^+) \\ & \propto P(I|s)P(s|\hat{X})\frac{P(X^-, X^+|s)}{P(X^-, X^+|\hat{X})}, \end{aligned} \tag{9}$$

   where I is the image associated with $\hat{X}$.

   End

2. Select $\hat{X}$ that produces the maximum probability. If the probability is larger than a predetermined threshold, replace $\hat{X}$ by the special symbol s.

End

Table 1. Performance of the character segmentation algorithm.

	Total	Correct	Splitting	Merging	Mis-False	Spurious
Ground Truth	36394	36250 (99.61%)	43 (0.12%)	88 (0.24%)	1 (0.00%)	12 (0.03%)
Detected	36396	36250 (99.60%)	90 (0.25%)	42 (0.11%)	1 (0.00%)	13 (0.04%)

Table 2. Performance of the special symbol classification using different features.

Features used	Number of correct detection
Moment invariants	3122 (82.3%)
Zernike moments	3297 (86.9%)
Probability maps	3630 (95.7%)

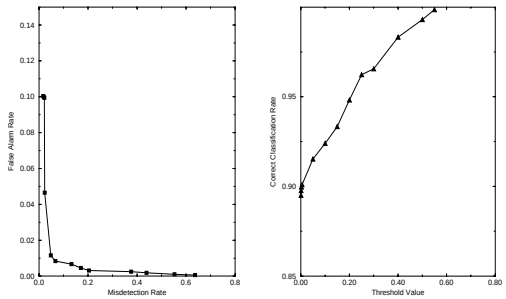
5. Experimental Results

The data set used in our experiment consists of 5516 pages from the National Library of Medicine.

To evaluate our character segmentation module, we select 32 pages among the 5516 pages and manually groundtruthed the character boxes for the 32 images – a total of 36394 character boxes. The segmentation module was tested on these 32 character-box-groundtruthed images. The evaluation results are shown in Table 1. The evaluation results show that 99.6% (36250) of ground truth character boxes (36394) have been correctly segmented, while 43 boxes are split into total of 90 boxes and 88 boxes are merged into total of 42 detected boxes.

To evaluate our symbol classification module, we selected from the data set 13 special symbols with relatively large number of samples among the special symbols in the data set for training. The selected symbols are: α, β, $\circ$ (Degree), δ, ϵ, γ, $\geq$, κ, $\leq$, μ, $\pm$, $\rightarrow$, $\approx$ (or $\sim$, $\simeq$, $\cong$). The total number of samples is 3794. Three set of features were used in the evaluation: the moment invariants[5], the Zernike moments[6], and the probability map. A decision tree classifier from S-PLUS [3] was used to compute the two moment features and the nearest neighbor classifier was used to compute the probability maps. A 5-fold cross validation method was used to estimate the accuracy of the classification module. The experimental results for the classification modules on the three sets of features are shown in Table 2.

Finally, we evaluated the performance of the combination of the detection and the classification modules. First, we find matches between the special symbols boxes detected by the detection module and the special symbol boxes in the ground truth. The matching results are the numbers of the correct detections, the miss-detections, the false alarms, the splitting and the merging errors. Next, for the special symbols which have been correctly detected (one-to-one match), we determine the rate in which these special symbols are correctly classified. The correct classification rate is used as the performance measure. The miss-detection and the false alarm rates based on different threshold values are plotted in Figure 3(a). The correct classification rate versus the threshold values are shown in Figure 3(b).



(a) (b)

Figure 3. Plots (a) false alarm rate vs. Mis-detection rate; (b) correct classification rate; using different threshold values.

Acknowledge

The authors would like to thank the National Library of Medicine for the support of this work.

References

[1] T.H. Cormen, C.E. Leiserson, and R.L. Rivest. *Introduction to Algorithms*, MIT Press, 1990.

[2] J. Liang *Document Structure Analysis and Performance Evaluation*, Ph.D. thesis, University of Washington, 1999.

[3] MathSoft. *S-PLUS Guide to Statistics*, 1997.

[4] O.D. Trier, A.K. Jain and T. Taxt. Feature extraction methods for character recognition - a survey. *Pattern Recognition*, pp 641-662, Vol. 29, No. 4, 1996.

[5] T. H. Reiss. *Recognizing planar objects using invariant image features*, Lecture notes in computer science 676, Springer-Verlag, 1991.

[6] A. Khotanzad and Y.H. Hong. Invariant image recognition by Zernike moments. *IEEE Transactions on Pattern Recognition and Machines Intelligence*, Vol. 12, No. 5, May 1990.