OCR for World Wide Web Images *

Jiangying Zhou *Daniel Lopresti*
Matsushita Information Technology Laboratory
Panasonic Technologies, Inc.
Two Research Way, Princeton, NJ 08540-6628

Zhibin Lei
Electrical Engineering Department
Brown University, Providence, RI 02912

ABSTRACT

A significant amount of text now present in World Wide Web (WWW) documents is embedded in image data, and a large portion of it does not appear elsewhere at all. To make this information available, we need to develop techniques for recovering textual information from in-line Web images. In this paper, we describe two methods for Web image OCR.

Recognizing text extracted from in-line Web images is difficult because characters in these images are often rendered at a low spatial resolution (72 dpi). Such images are typically considered to be "low quality" by traditional OCR technologies. Our proposed methods utilize the information contained in the color bits to compensate for the loss of information due to low sampling resolution. The first method uses a polynomial surface fitting technique for object recognition. The second method is based on the traditional n-tuple technique.

We collected a small set of character samples from Web documents and tested the two algorithms. Preliminary experimental results show that our n-tuple method works quite well. However, the surface fitting method performs rather poorly due to the coarseness and small number of color shades used in the text.

Keywords: document analysis, optical character recognition, pattern recognition, World Wide Web.

1. INTRODUCTION

With the explosive growth of the World Wide Web (WWW), a great number of documents are being made accessible electronically. The issue of developing efficient methods for searching, browsing, and retrieving Web images has thus become important. Much work has already been done in the area. However, most of the research so far has mainly concentrated on processing the ASCII text part of Web documents, ignoring more complex forms of document analysis.

A significant amount of text now present in WWW pages, however, is embedded in image data (*i.e.*, bitmaps), and a large portion of it does not appear elsewhere at all. Consequently, the information contained in the image text is currently not indexed, and any search for the associated terms would fail.[7] To make this information available, we must develop techniques for assessing the contents of Web image data. In this paper, we describe methods for recognizing text embedded in in-line Web images.

There are several significant differences between the OCR problem in Web image processing and its traditional counterpart. Characters in Web images often use small fonts and are rendered at low spatial resolution (72 dpi) intended for screen display. In a recent investigation, we estimated that much of the text in Web images is roughly equivalent to a 5pt - 7pt font in a document scanned at 300 dpi.[7] Such character images are typically considered to be "difficult" by traditional OCR technologies.

On the other hand, text in Web images is rather "clean" – unlike scanned document images, many Web images are created electronically, thus, do not contain noise such as blur, speckles, *etc.* Moreover, Web images are usually

*Presented at the *IS&T/SPIE International Symposium on Electronic Imaging*, San Jose, CA, February 1997.

free of distortions caused by skew, jitter, *etc.* As a consequence, traditional OCR technologies which are designed and optimized for coping with noise typically seen in scanned document images may not be applicable to the kind of input seen in Web images.

Text in Web images is not completely distortion-free. There can be distortion, for example, from compression loss, such as in JPEG compression. Another major distortion for Web image text comes from spatial sampling variation. Spatial sampling variation is introduced when a character is rendered from an abstract, analog description. The same character in the same font can be quantized differently depending on the relative phases of the character and the sampling grid.[6]

Another difficulty we face in recognizing text from Web images is that it is often embedded in a complex color background. Traditional methods often assume that text is printed in black on a white background, while text in Web images can have arbitrary colors and is often intermingled with differently colored objects in the background. Thus, it is difficult to locate and separate text from the background by simple thresholding.

In view of the differences, we can see that there is a need to examine carefully the issues at hand and develop appropriate solutions for the Web OCR problem. In this paper, we investigate two methods for recognizing text in in-line Web images. Both methods explore the idea of using information contained in color bits to compensate for the loss of information due to low sampling resolution. The first method uses a polynomial surface fitting technique. The second method is based on the n-tuple technique.

The rest of the paper is organized as follow: in Section 2. we survey related research, Section 3 describes our approaches for recognition, Section 4 presents some experimental results. Finally, we conclude the paper in Section 5.

2. REVIEW OF RELATED WORK

The issue of recognizing low resolution, poor quality text has been addressed under a different context in the literature. For example, Lee, Pavlidis and Wasilkowski have conducted a theoretical study on the trade-off between spatial resolution and quantization resolution in signal processing.[3] The study shows that it is possible to recognize text successfully at low resolution if color information is preserved during the digitization. It is predicted that for 10 point text the sampling rate could be as low as 100 dpi (but not much lower).

Based on this result, Li and Pavlidis later proposed a character recognition technique which extracts features directly from the gray level of the input image.[9] Their method takes the digitized image as a terrain (with height denoting the "darkness" of the pixels) and analyzes the topographic structures of the terrain. The pixels are classified into different topographic feature classes such as *ridge*, *pit*, *saddle*, *ravine*, *etc.* according to the estimated first and second directional derivatives of the image intensity surface. It then extracts the basic structural information from the input image and represents the features as a topographic feature graph. The recognition process then compares the graph with predefined prototype graphs.

Li and Pavlidis' method provides a powerful framework for analyzing shapes. The surface fitting method that we describe here is, in spirit, similar to their approach. Our method assumes that the foreground color is roughly uniform for a given character in a Web image. It detects a representative foreground color for each character and computes the difference between the representative color and the color of each pixel. The result forms a 3-D surface similar to the terrain in Li and Pavlidis's method. We then derive a set of features from the surface for recognition.

Clearly, the topographic feature analysis method is a computational intensive operation. An alternative approach is to use techniques related to matched filters. A representative of this kind of method is orthogonal expansions.[5] In this method, a character in color space is expressed as a linear superimposion of different orthogonal primitive patterns. Note that a Web image usually contains a relatively small quantity of text, hence the orthogonal assumption is usually valid since the text is most likely of single font. In addition, noise commonly seen in scanned images, such as blurring, speckles, does not appear in these images so a relatively small number of primitives may be sufficient.

3. DESCRIPTION OF THE METHODS

One of the difficulties we face in Web image OCR is that the text is often blended in a complex color background. To detect and locate text from such a complex color background, we used an algorithm based on color clustering. The details of the algorithm are described elsewhere.[10] Briefly, the algorithm quantizes the color space of the input

image into a number of color classes using a parameter-free clustering procedure. Pixels then are assigned to color classes closest to their original colors. The algorithm then identifies character-like connected components in each color class based on their shapes.

The character-like connected components are subsequently segmented and extracted from the original color input image. Typically, each connected component represents a single character. Next, we convert the extracted character images into gray level by comparing the difference between the representative color of the foreground and the color of each pixel. The representative color of the connected component region is assigned gray value 255. All other colors are assigned gray value $255 - d$, where d is the distance between the color and the representative color. Once characters are extracted, we then apply the recognition process to each character image.

3.1. The Polynomial Surface Fitting Method

The first recognition method we investigated is based on polynomial surface fitting. We represent a character shape by a polynomial surface function. Here the character shape in the intensity image is treated as a 3-D surface, *i.e.*, $z = g(x, y)$ where (x, y) is the pixel location and z is the intensity value. A polynomial function of certain degree can then be used to fit to this surface.

In theory, the higher the degree of fitted polynomial is, the better it can capture the shape of the character. However, a high degree polynomial is also more sensitive to noise. Currently, we use 4th degree polynomials for the surface representation[8,4]:

$$z = f(x, y) = \sum_{0 \le i,j \le 4} a_{ij} x^i y^j$$

The fitting of the polynomial surfaces is done currently based on the least square principle. Let $Z = \{(k, l, t) \| 0 \le k < K,\ 0 < L \le J\}$ represent the image data, where (k, l) specifies the pixel coordination and t is the image intensity value. The best polynomial surface representation is a vector of coefficient a_{ij} such that the mean square error

$$\frac{1}{KL} \sum_{k,l} (\sum (a_{ij} k^i l^j - t)^2$$

is minimized.

From the polynomial representation we then extract a set of features for recognition. Notice that the monomial basis functions $x^i y^j$ described in the above equation are not orthogonal to each other, thus, the coefficients of lower order monomials will change when the degree of the polynomial surface fit increases. Hence, the coefficient vector for monomial basis functions is not a good feature vector. Instead, we use orthogonal Legendre polynomial basis functions to represent the polynomial surface:

$$P(n, x) = \frac{1}{n! 2^n} \frac{d^n (x^2 - 1)^n}{dx^n}$$

The surface will have the following representation $\sum_{0 \le i,j \le 4} b_{ij} P(i, x) P(j, y)$. Here b_{ij} is the coefficient vector for Legendre polynomial basis functions. These coefficients are used as a feature vector for the character shape. A linear classifier is used to compare each feature vector with a reference vector for each character class and assign the character to the class with the closest distance.

The polynomial surface representation only captures the global shape of a character. Characters whose shapes are similar to each other (*e.g.*, "c" and "e") may not be distinguished reliably using this method since variations in the fonts and color intensity may introduce significant "cross-over" of the class distributions.

One possible solution to this problem is to group character classes of similar shapes into "meta-classes" (*e.g.*, "c" and "e" form a meta-class). An input pattern is first classified into one of the meta-classes. A second stage classification is then used to distinguish among characters in the same meta-class by examining finer features from the image.

For example, in the second recognition stage, we may use the topographic features described by Li and Pavlidis to extract the skeleton from the image. Figure 1 shows the ridge points in images of letter "a" and "e" detected by the method. As we can see, the method recovers remarkably well the skeletons of "e" despite the severe distortion of the shape.

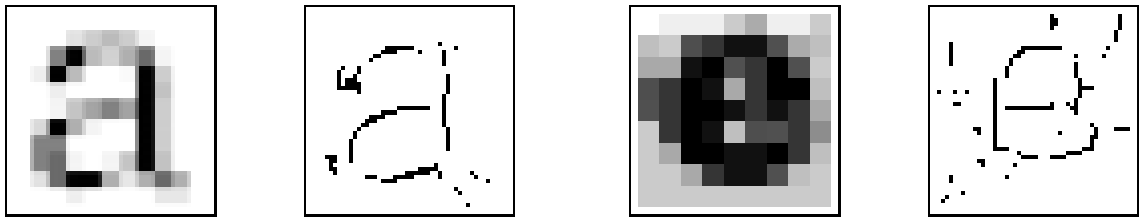

Figure 1. Ridge points detected in images of "a" and "e".

3.2. The N-tuple Method

The polynomial surface representation method is a computational intensive operation. Furthermore, the method does not work when the color intensity resolution for characters is very low to begin with (*e.g.*, a few shades of color are used to represent a character). An alternative approach is to use the n-tuple technique.

N-tuple classification is a rather old OCR technique, first proposed in 1959.[1] It has received only scant attention since the early days of OCR research. In this paper, we re-examine the applicability of the n-tuple method and show that it is useful for the Web image OCR problem.

An n-tuple is simply a set of locations in an image with specified colors. For a binary image, an n-tuple represents the presence or absence of a specific configuration of black and white pixels in a given pattern. For recognition, an n-tuple is superimposed onto an input image and the colors at the given locations are compared. An n-tuple is said to "fit" the image if the colors at all its locations match with the image color at the corresponding locations. Usually, several n-tuples are used as templates and the tuples can be shifted over the image in order to find the "fits".

Most n-tuple methods reported in the literature are defined over a binary input space. An obvious way of applying such n-tuple methods to Web image OCR is to threshold the color image into binary bitmap and treat the problem as standard n-tuple classification problem. The drawback of doing so is that the information contained in the color bits will be lost in the thresholding process. The approach we developed, on the other hand, takes into consideration the "fuzziness" of the differentiation between foreground objects (*i.e.*, text characters) and the background in the input image. Instead of an exact "fit", our method measures the degree a tuple matches the input image pattern and finds the class the tuple matches with a maximum likelihood.

To do so, we first assign to each pixel in the input image a value between 0 and 1 which represents the certainty the pixel belongs to the foreground. Let $P[i,j]$ be the input image and c_f be the representative color of the foreground determined by the text extraction algorithm. We define the normalized color difference between c_f and the pixel color at pixel (i,j) as a measure of confidence the pixel belongs to the foreground:

$$I[i,j] = 1 - \frac{d(P[i,j], c_f)}{\max_{i,j} d(P[i,j], c_f)}$$

where $d()$ is the distance between c_f and pixel color $P[i,j]$. From the definition, we have $0 \leq I[i,j] \leq 1$. $I[i,j] = 1$ indicates a definite foreground pixel, and $I[i,j] = 0$ represents a definite background pixel.

To apply the n-tuple method, let $\{t_i = (p_{i1}, p_{i2}, ...p_{in})\}, i = 1, 2, ..., m$ be a set of n-tuples defined over the input image space of dimension $W \times H$. Each p_{ij} specifies a location in the image space. Given an image pattern, we first assess the probability that the pixel in the image at location p_{ij} as specified by the i'th n-tuple is foreground and compute a joint probability distribution for all location combinations.

Formally, let $b = b_1 b_2 ... b_n$, where $b_i \in \{0, 1\}$. We say pixel location p_{ij} of tuple i is designated to be foreground if $b_j = 1$, and background if $b_j = 0$. When every b_j in b has been assigned a particular designation, we say that b represents a specific foreground location configuration. Let $P^c(i : b_1 b_2 ... b_n)$ denotes the probability that the foreground color appears in location configuration $b_1 b_2 ... b_n$ in the image pattern c. For example, for a 7-tuple, $P^c(i : 0110000)$ represents the probability of the foreground color appears at locations p_{i2} and p_{i3} simultaneously, but nowhere else. For an n-tuple, there are total 2^n different possible configurations. Let B be the set of all possible configurations of $b_1 b_2 ... b_n$. For a given image pattern, $P^c(i : b_1 b_2 ... b_n)$ is computed as:

$$P^c(i : b_1 b_2 ... b_n) = \prod_{b_j=1} I[p_{ij}] \prod_{b_j=0} (1 - I[p_{ij}])$$

One issue which is yet to be addressed is how to select a set of appropriate n-tuples for the classifier. For many non-trivial recognition problems, it is usually computationally intractable to find the optimal choice of n-tuples. In the current implementation of the n-tuple classifier, the locations p_{ij} for each tuple are chosen randomly from the input image space.

During the training process, the value $P^c[i : b_1 b_2 ... b_n]$ is evaluated for every pattern in a character class C and a summation is calculated:

$$P(i : b_1 b_2 ... b_n) = \sum_{c \in C} P^c(i : b_1 b_2 ... b_n)$$

We then estimate the conditional probability that given class C and tuple i the foreground appears in foreground configuration $b_1 b_2 ... b_n$ as:

$$P(i : b_1 b_2 ... b_n | C) = \frac{P(i : b_1 b_2 ... b_n)}{\sum_{b_1 b_2 ... b_n \in B} P(i : b_1 b_2 ... b_n)}$$

In the subsequent classification process, we compute the probability of an input pattern x given n-tuple i of class C using the following equation:

$$P(x | i : C) = \sum_{b_1 b_2 ... b_n \in B} P^x(i : b_1 b_2 ... b_n) P(i : b_1 b_2 ... b_n | C)$$

and the probability of x given all the n-tuples of class C is given as:

$$P(x|C) = \prod_{i}^{m} P(x | i : C)$$

A Bayesian maximum likelihood classifier is then used to determine the class of the input pattern. We assume here that the *a priori* probabilities are the same for all character classes and assign the pattern to the class for which $P(x|C)$ is a maximum.

4. EXPERIMENTAL RESULTS

In order to test the effectiveness of the methods, we collected 50 WWW images obtained from the Web and extracted from these images 215 lowercase characters in 8 classes. Figure 3 in the Appendix shows some sample characters from the data set. Table 1 shows certain statistics of the testing samples.

Table 1. Character sample set statistics.

character class	*number of samples*	*average width*	*average height*	*ave. number of colors*
a	32	14	14	6
c	26	8	10	5
e	37	10	12	5
i	17	5	11	3
n	27	11	13	7
o	29	9	10	4
s	26	8	10	4
t	21	7	14	3

For the surface fitting classifier, 4th degree explicit polynomial surfaces were computed to represent each character. In the test, we used half of the characters as training samples and stored the polynomial surface parameters as the feature vectors. We then ran the classifier on the whole data set. The average distance from a character image to be recognized to the stored feature vectors of an whole class was used as the distance from this character to the class. The class that a character belongs to was chosen as the class with the smallest character-to-class distance. We used the normal Euclidean distance of the polynomial coefficient vectors instead of algebraic invariants to measure

Table 2. Recognition accuracies for the surface fitting classifier.

character class	*recognition accuracy*
a	51.6%
c	88.5%
e	57.6%
i	100.0%
n	84.0%
o	44.8%
s	69.2%
t	85.7%
ave.	69.7%

Table 3. Recognition accuracies for the n-tuple classifier.

character class	*recognition accuracy*
a	90.6%
c	92.3%
e	73.0%
i	94.1%
n	100.0%
o	96.6%
s	88.5%
t	85.7%
ave.	89.3%

the distance between two characters. This is because there is not much rotation in the character dataset that we collected and we also scaled the images properly to compensate the changes in character size. Table 2 shows the recognition accuracies for the eight classes as well the overall performance.

From Table 2, we can see that the recognition results were rather poor for character classes "a", "e", "o", "s". The reason for this is that the "c", "e", "o" triplet (as well as the "a" and "s" pair) have similar shapes and it is hard to distinguish them using 4th degree polynomial surfaces since the small structural differences between these classes are usually lost in the representation. Another reason for the poor performance here is that most of the character images in our dataset are essentially binary (as we can see from Table 1, the average number of unique colors in each image is quite low). A continuous surface representation is not suitable for representing these images because of the high frequency intensity changes.

For the n-tuple classifier, we scaled the sample images to fixed-size 8×8 bitmaps. Next we set the parameters n and m (the tuple dimension and the size of the n-tuple set) and generated the n-tuples. Again, we took half of the samples from the sample set for training the classifier. We then ran the classifier over the entire data set. Table 3 shows the recognition rates for $n = 10$ and $m = 10$.

We experimented with a few combinations of n and m in the test. Figure 2 shows the correlation between the recognition rate and the tuple dimension for a fixed $m = 10$. Table 4 shows the correlation between the recognition accuracy and the tuple set size at $n = 10$.

The best recognition performance for the n-tuple classifier on this set of sample data is 90.2% at $n = 10$ and $m = 20$. Generally speaking, a large n tends to decrease the false alarm rate while increasing the mis-classification rate. Similarly, a large m also tends to increase the discrimination power of the classifier, although it also increases the computation needed. However, too many "no-so-good" tuples may also reduce the performance. Currently how to chose tuples is still an unresolved issue.

5. CONCLUSIONS

In this paper, we described two methods for recognizing text from in-line Web images. The first method is based on a polynomial surface fitting technique. The second method is based on the n-tuple technique. Both methods explore

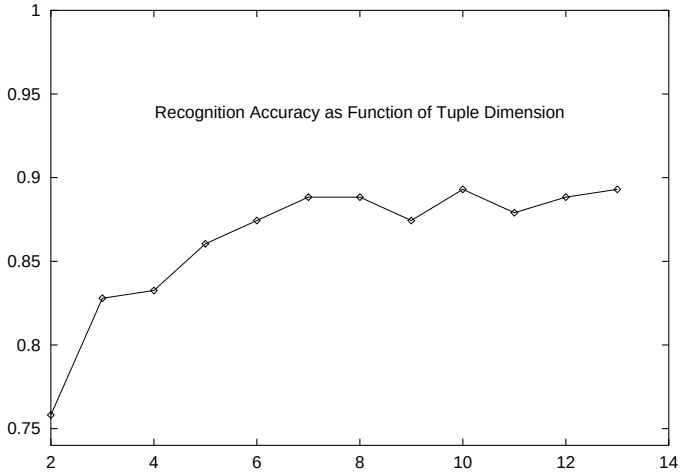

Figure 2. Correlation between recognition accuracy and tuple dimension ($m = 10$).

Table 4. Correlation between recognition accuracy and size of tuple set ($n = 10$).

number of n-tuples	*recognition accuracy*
1	76.3%
5	87.9%
10	89.3%
15	89.8%
20	90.2%

the idea of using information contained in color bits to compensate for the loss of information due to low sampling resolution.

Preliminary experimental results show that the surface fitting method performs rather poorly. We found that Web images often use only a few shades of color to represent a character. A continuous surface representation does not appear to be suitable for representing these images because of the high frequency intensity changes.

On the other hand, the results show that our n-tuple method works quite well. The major difficulty in using the n-tuple method is the selection of desirable n-tuples. It has been shown that in general generating distinguishing tuples is a computationally intractable problem. Despite this, practical searching algorithms have been developed for finding "good" tuples. For example, in a paper by Jung, Krishnamoorthy, Nagy, and Shapira, the authors proposed two algorithms for generating, from a small training set, a collection of n-tuples which are "shift" invariant.[2] Such n-tuples fit each positive pattern in at least p different shift positions and fail to fit each negative pattern by at least $n - q$ pixels in each shift position. "Shift invariance" is a desirable feature since this makes the tuples independent of sample variations.

As we mentioned earlier, text in Web images usually undergoes a relatively a few types of distortion. This makes the n-tuple method particularly applicable to the Web image OCR problem. A better understanding of the nature of distortion can lead us to the development of a practical n-tuple generator for Web OCR problem. We are currently in the process of constructing a better n-tuple classifier and conducting a larger-scale test.

APPENDIX

Figure 3. Sample characters from the testing data set.

REFERENCES

1. W.W. Bledsoe and I. Browning. Pattern recognition and reading by machine. In *Proceedings of Eastern Joint Computer Conference*, number 16, pages 225–233, December 1959.
2. D.M. Jung, M.S. Krishnamoorthy, G. Nagy, and A. Shapira. N-tuple features for OCR revisited. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(7):734–743, 1996.
3. D. Lee, T. Pavlidis, and G. W. Wasilkowski. A note on the trade-off between samping and quantization in signal processing. *Journal of Complexity*, 3:359–371, 1987.
4. Z. Lei, D. Keren, and D.B. Cooper. Computationally fast bayesian recognition of complex objects based on mutual algebraic invariants. In *Proceedings, International Conference on Image Processing*, pages 635–638, Washington, D.C., October 1995.
5. C. M. Leung. A practical basis set for chinese character recognition. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, San Francisco, California, June 1985.
6. D. Lopresti, G. Nagy, P. Sarkar, and J. Y. Zhou. Spatial sampling effects in optical character recognition. In *Proceedings of the Third International Conference on Document Analysis and Recognition*, pages 309–314, Montréal, Canada, August 1995.

7. D. Lopresti and J. Y. Zhou. Document analysis and the World Wide Web. In J. Hull and S. Taylor, editors, *Proceedings of the Workshop on Document Analysis Systems*, pages 417–424, Marven, Pennsylvania, October 1996.
8. G. Taubin. Estimation of planar curves, surfaces and nonplanar space curves defined by implicit equations, with applications to edge and range image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pages 1115–1138, November 1991.
9. L. Wang and T. Pavlidis. Detection of curved and straight segments from gray scale topography. In *Proceedings of SPIE Symposium on Character Recognition Technologies*, pages 10–20, San Jose, California, 1993.
10. J.Y. Zhou and D. Lopresti. Extracting text from WWW images. Technical report, Matsushita Information Technology Laboratory, Princeton New Jersey, April 1997. In submission.