Pergamon

Pattern Recognition, Vol. 30, No. 8, pp. 1293–1300, 1997
© 1997 Pattern Recognition Society. Published by Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0031-3203/97 $17.00+.00

PII: S0031-3203(96)00167-7

IMAGE-BASED KEYWORD RECOGNITION IN ORIENTAL LANGUAGE DOCUMENT IMAGES

JASON ZHU,[†] TAO HONG[‡] and JONATHAN J. HULL[§,*]
[†]Microsoft Corporation Seattle, WA, U.S.A.
[‡]Center of Excellence for Document Analysis and Recognition, State University of New York at Buffalo, Buffalo, NY 14260, U.S.A.
[§]Ricoh California Research Center, 2882 Sand Hill Road, Suite 115, Menlo Park, CA 94025, U.S.A.

(*Received* 16 *July* 1996; *in revised form* 30 *July* 1996; *received for publication* 21 *October* 1996)

Abstract—An algorithm is presented for keyword recognition in Oriental language document images. The objective is to recognize keywords composed of more than one consecutive character in document images where there are no explicit visually defined word boundaries. The technique exploits the redundancy expressed by the difference between the number of possible character strings of a fixed length and the number of legal words of that length. Sequences of character images are matched simultaneously to a dictionary of keywords and illegal strings that are visually similar to the keywords. A keyword is located if its image is more likely to occur than any of the illegal strings that are visually similar to it. No intermediate character recognition step is used. The application of contextual information directly to the interpretation of features extracted from the image overcomes noise that could make isolated character recognition impossible and the location of words with conventional post-processing algorithms difficult. Experimental results demonstrate the ability of the proposed algorithm to correctly recognize words in the presence of noise that could not be overcome by conventional character recognition or post-processing algorithms. © 1997 Pattern Recognition Society. Published by Elsevier Science Ltd.

Word recognition　Chinese　Japanese　Oriental languages　Text recognition
Contextual post-processing　Word spotting

1. INTRODUCTION

The recognition of keywords in Oriental languages such as Chinese and Japanese is an important part of extracting information about the content of a passage of text. Keywords help indicate the topic of a document and are useful for later retrieval operations. Keyword recognition is also an important part of automatic translation systems. Better recognition of keywords in degraded images such as facsimile transmissions or poor photocopies will also improve the overall quality of OCR output.

The recognition of words in Oriental language document images is a difficult problem since words are composed of one or more characters and a running text contains no visual indication of word boundaries. The *keywords* recognized by the technique proposed in this paper are words that are composed of two or more characters and that occur in a fixed list or *dictionary*. Any arbitrary word composed of two or more characters can be included in this dictionary.

Word recognition in Oriental languages has traditionally been solved by post-processing the results of isolated character recognition.[1,2] The design of a typical word recognition post-processing algorithm is shown in Fig. 1. A passage of text is processed character-by character and feature vectors (fv_i) are extracted from each character. These feature vectors are independently classified and the n "best" decisions ($d_{ij}, j = 1, \ldots, n$) for each character are output. These are the n characters that are most visually similar to the corresponding characters in the input document. A post-processing algorithm compares these decisions to the entries in a dictionary. Each dictionary entry contains sequences of characters that correspond to words. The post-processing algorithm outputs a sequence of character decisions that correspond to an entry in the dictionary if it is confident those characters actually occurred in the input text image.

Word recognition techniques take advantage of the redundancy present in Oriental languages. For example, in Chinese there are approximately 3500 commonly used characters. Out of the $3500^2 \approx 12$ million possible strings of two-character words, only about 25,000 are legal words. Since legal words make up such a small percentage of all possible character strings, their occurrence in the output of a recognizer is a strong indication that they are correct.

Word recognition techniques that depend on such inter-character redundancy can only be applied to words that contain two or more characters. Thus, such methods will only be able to improve OCR performance on about 50% of the characters in a typical Chinese text passage. However, it is important to recognize these character strings correctly since they are useful for subsequent indexing and retrieval of the document.

* Author to whom correspondence should be addressed. E-mail: hull@crc.ricoh.com.

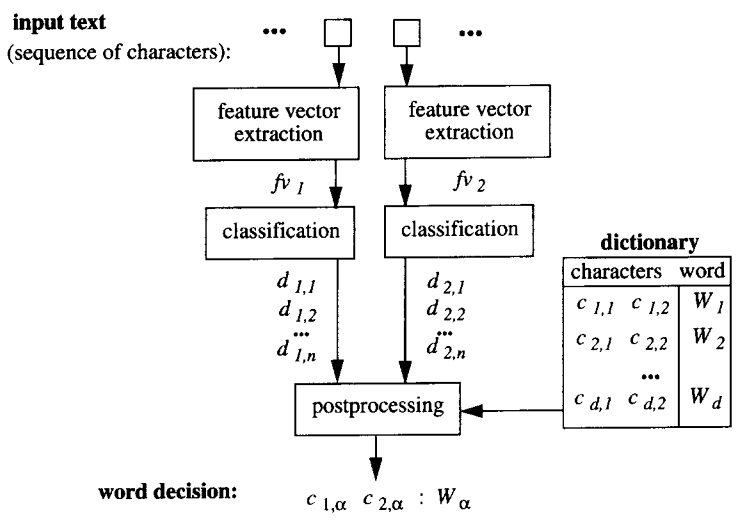
input text
(sequence of characters):
feature vector extraction
feature vector extraction
fv_1
fv_2
classification
classification
$d_{1,1}$ $d_{1,2}$ ... $d_{1,n}$
$d_{2,1}$ $d_{2,2}$ ... $d_{2,n}$
dictionary
characters word
$c_{1,1}$ $c_{1,2}$ W_1
$c_{2,1}$ $c_{2,2}$ W_2
$c_{d,1}$ $c_{d,2}$ W_d
postprocessing
word decision:
$c_{1,\alpha}$ $c_{2,\alpha}$: W_α

Fig. 1. Design of a post-processing algorithm.

A problem with post-processing methods is that they can require many of the characters in a word to be recognized correctly in the top n decisions output by the character recognizer. This assumption is difficult to guarantee when the input image is subject to degradations that occur in facsimile transmissions or photocopies. The visual context between characters is not available to the post-processing algorithm. It only receives the symbolic decisions output by the classifier.

Visual context has been utilized in English word recognition by methods that use segmentation-based recognition.[3] These techniques use the visual context between characters during post-processing. Figure 2 shows how a segmentation-based recognition strategy is applied to English text. A segmentation algorithm is applied to each word image. Feature vectors (fv_i) are then extracted from the isolated character images and compared to dictionary entries. Each dictionary entry is composed of a sequence of feature vectors [v_{ij}, $i = 1, \ldots,$ no. of dictionary words (d), $j = 1, \ldots,$ word length] that correspond to a word [W_i, $i = 1, \ldots,$ no. of dictionary words (d)]. The algorithm outputs the word with the feature vectors that best match the feature vectors extracted from the input image. This strategy works well for English language documents since word boundaries are well defined.

An algorithm is proposed in this paper that applies a similar approach to Oriental language documents.[4] The feature vectors extracted from sequences of characters in an input image are matched to sequences of feature vectors for words. The absence of explicit word segmentation marks is compensated for by a two-stage recogni-

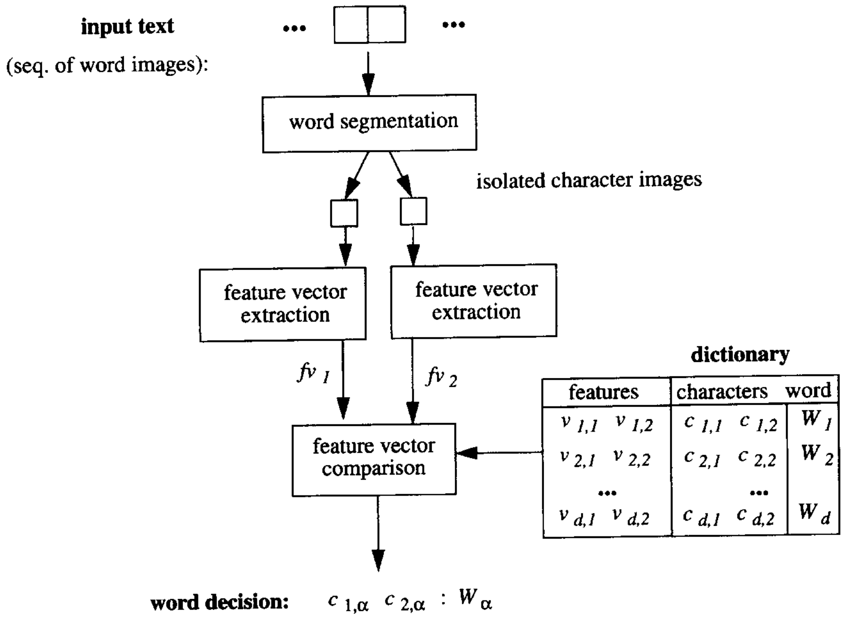
input text
(seq. of word images):
word segmentation
isolated character images
feature vector extraction
feature vector extraction
fv_1
fv_2
feature vector comparison
dictionary
features characters word
$v_{1,1}$ $v_{1,2}$ $c_{1,1}$ $c_{1,2}$ W_1
$v_{2,1}$ $v_{2,2}$ $c_{2,1}$ $c_{2,2}$ W_2
$v_{d,1}$ $v_{d,2}$ $c_{d,1}$ $c_{d,2}$ W_d
word decision:
$c_{1,\alpha}$ $c_{2,\alpha}$: W_α

Fig. 2. Design of a segmentation-based word recognition algorithm for English.

tion algorithm. In a hypothesis generation stage a dictionary word is proposed that could match a given sequence of character images. In a hypothesis testing stage those character images are compared to a list of character sequences that are visually similar to the proposed dictionary word. That list of character sequences is known as the confusion set for that word. The proposed dictionary word is output by the hypothesis testing stage only if it is a "better" decision than any of the other members of its confusion set. The confusion represents the contextual knowledge of the algorithm. That is, the character strings that are likely to be misrecognized as the dictionary word are contained in its confusion set.

The advantages of this technique include its application of word-level contextual information directly to the interpretation of the feature vectors. The direct use of image data allows for the recognition of words in context even though their individual characters may be unrecognizable in isolation. This overcomes a problem in traditional post-processing techniques that require reliable character recognition results.

An additional advantage of the proposed algorithm is that it is intended to recognize entries from a fixed dictionary of keywords (a similar task for English language documents has been called word spotting). The proposed algorithm is not designed to recognize all the words that occur in a passage of text. This is a more difficult problem that would require the recognition of single-character words that possess little image context. Also, a parsing algorithm may be needed to distinguish inflections in Japanese and Korean documents, among other characteristics.

The rest of this paper presents the algorithm in detail. The model of legal and illegal words is explained and an example is given of how it is used in practice. Experimental results are presented on a database of Chinese text that illustrates the ability of the algorithm to overcome noise that would be difficult to compensate for in a post-processing approach.

2. ALGORITHM DESCRIPTION

An example of the operation of the proposed algorithm is given in Fig. 3. A Chinese language document is input and the characters in the document are processed sequentially. Beginning at each character, the algorithm attempts to match words from its dictionary in decreasing order by length. If the algorithm makes a successful match, it outputs the characters in the decision word and starts matching again after the last character in the word. If the algorithm does not make a successful match, it goes to the next character and begins the process again.

The example shows the algorithm processing the next four characters in the input text. The hypothesis genera-

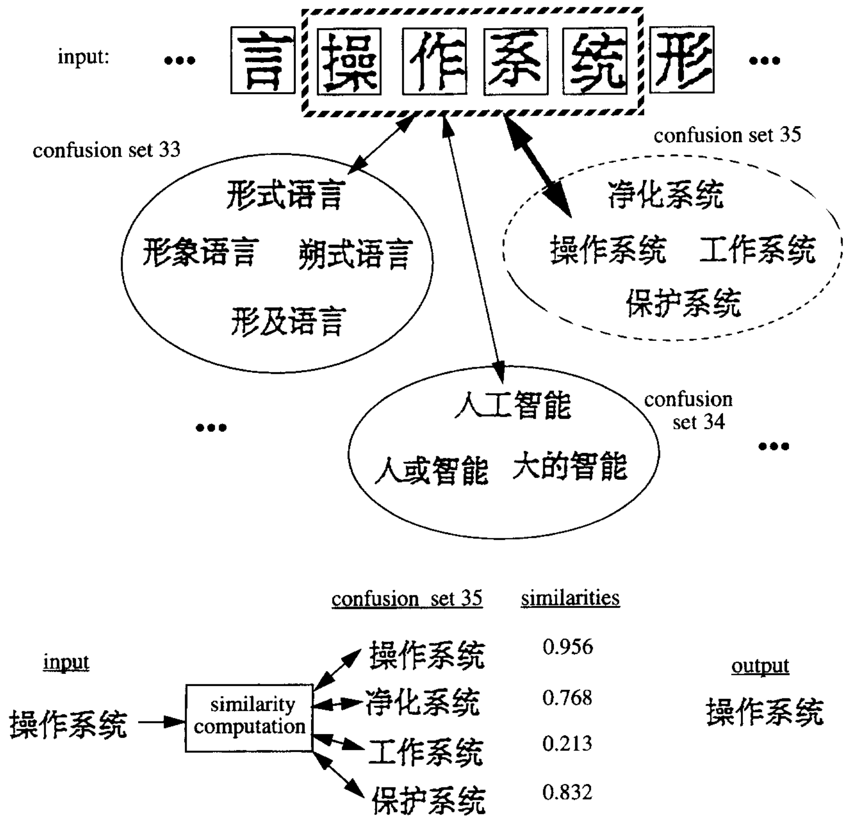

Fig. 3. Example of algorithm application. Hypothesis generation locates dictionary words that could match the input. Hypothesis testing determines which entry in a confusion set is the best match (i.e. maximizes similarity between images).

```
1.   for each length $l$ = 2,3,4,5,6,7,8
      - extract all the character strings of length $l$ and
          their frequencies from a training corpus;

2.   for each dictionary word $dw$

      - determine the character strings $cw$ found in step 1 that
          are the same length as $dw$ and have at least one
          character in common;

      - calculate the Similarity between each such $cw$ and $dw$;

      - if ( Similarity ( $cw$, $dw$ )  > $\delta_0$ or
               frequency ( $cw$ ) > $t_{conf}$ and
               Similarity ( $cw$, $dw$ ) > $\delta_1$( $cw$ ) )
            then  add $cw$ to the confusion list for $dw$
```

Fig. 4. Training phase.

tion phase compares the next four characters to the legal word in each confusion set. Three of the confusion sets in a hypothetical database are shown in Fig. 3. These confusion sets are numbered 33, 34, and 35. The comparison is performed on feature vectors extracted from the character images. In this example, the word in confusion set 35 (the sequence of four characters in the case of the arrow) produced the maximum similarity value (i.e. the best match). The similarity between two strings of character images ranges from 0 to 1 where values close to 1 indicate character strings that are similar to each other.

The hypothesis testing phase is illustrated at the bottom of Fig. 3. The next four characters from the input are compared to each member of confusion set 35. The similarities between the input characters and each member of the confusion set are shown. In this example the dictionary word in confusion set 35 was output because it maximized the similarity value.

The confusion sets are important determiners of performance algorithm. The confusion sets represent the contextual knowledge of the method. The more accurately the confusion sets represent the character strings that are likely to be confused with a given word, the higher the recognition accuracy should be.

The confusion sets are compiled from a large training corpus by comparing consecutive strings of characters to every dictionary word. If a string of characters contains at least one character in common with a dictionary word and those characters occur consecutively more than a fixed number of times in the corpus, they are added to the confusion set for that dictionary word. The reasoning behind using the confusion set is that false positives are most likely to happen when the input is visually similar to a dictionary word. The confusion sets model the most likely false positives and provide a method to prevent their output.

A precise statement of the algorithm is given in Figs 4 and 5. A training phase shown in Fig. 4 is used in which the confusion sets for each dictionary word are calculated. In step 1 all the character strings of each length from 2 to 8 are extracted from a training corpus along with their frequencies. In step 2 the character strings are determined that are the same length as each dictionary word and have one character in common with it. These strings are added to the confusion list for the dictionary word if the similarity between the character string and the dictionary word is greater than δ_0 or the similarity is greater than δ_1 and the character string occurs more than t_{conf} times in the corpus.

In the testing phase shown in Fig. 5, each character position p is inspected. Each dictionary word dw of a given length l is compared sequentially to the input stream starting at position p in a text that contains characters numbered $1, \ldots, N$. Longer words are compared before shorter words. If the similarity between the next l characters from the input stream starting at position p and dw is greater than $t_{\text{recog},l}$, each word cw in the confusion set of dw is compared to the input. If none of the members of the confusion set match the input better than dw (i.e. the similarity between the input characters and every word cw is less than the similarity between the input and dw), then dw is added to the output set. If after all the dictionary words have been considered, the output set contains a single word, that word is written out and the next l characters are skipped in the input stream. If the output set contains more than one word, this indicates that the algorithm found more than one word that was a plausible output at that location. In this case, none of the words are output.

Referring to the example in Fig. 3, p is the position of the character on the left-hand side of the cross-hatched box. Three of the dictionary words (dw) are at the ends of the arrows. The other character strings in the confusion sets are the words (cw) that are compared to the cross-hatched character sequence when its similarity to dw exceeds $t_{\text{recog},l}$. The selection of the output word by the inner loop of the algorithm in Fig. 5 is shown at the bottom of Fig. 3.

The similarity function is applied to two strings of character images. It is calculated as the average of the similarity between their feature vectors. The similarity between two feature vectors X and Y of length n

```
for each character position $p = 1 \ldots N$ in the input {

 for $l$ = MAX_l downto 2 {

   output_set = NULL;

   for each dictionary word $dw$ of length $l$ { /* hyp. gen */

     if (Similarity($l$ input chars, $dw$) > $t_{recog,l}$) then {

        output_word = $dw$;

        for each $cw$ in confusion set of $dw$ /* hyp. test */
           if (Similarity($l$ input chars, $cw$) >
                Similarity($l$ input chars, $dw$))
           then
             output_word = $cw$;

        if (output_word == $dw$) output_set += $dw$;

     }

   } /* end for each $dw$ of length $l$ */

     if (|output_set| == 1) {
      write(output_set);
      $p = p + l$;
      break;
   }

 }  /* end for l = MAX_l downto 2 */

} /* end */
```

Fig. 5. Testing phase.

is computed:

$$\frac{2 * \sum_{i=1}^{n}(x_i * y_i)}{\sum_{i=1}^{n} x_i^2 + \sum_{i=1}^{n} y_i^2},$$

where the feature vectors X and Y are computed by a method called local stroke direction (LSD).[5–7] The LSD technique divides each character image into a 8×8 grid and assigns each pixel the direction (vertical, horizontal, diagonal right, and diagonal left) of the vector that covers the maximum number of consecutive black pixels. The number of pixels in each grid cell that are assigned each direction are output. This provides a feature vector of 256 elements for each isolated character.

When given a character image, its LSD feature vector is computed as follows (see Fig. 6 for an example in detail): first, for each black pixel, compute its directional run-length for each of four directions and normalize it as a ratio to the total run-length in all directions; second, partition the image into an 8×8 grid and compute the directional run-length of each area as an average of the pixels in the area. In Fig. 6(c) the values of the LSD feature vector are scaled to integers in the range 0–255.

The similarity calculation assumes that there exists an image of each character in the training corpus and the dictionary. This image data is extracted from digitized fonts that contain labeled images of isolated characters.

The actual run time of the algorithm is proportional to the product of the length of the input and the number of entries in the dictionary. An obvious improvement could be obtained by adopting a dynamic programming approach that matches strings of character images to dictionary entries.[8]

The performance of the algorithm is dependent on several thresholds. The training procedure in which the confusion lists are constructed uses three thresholds. The δ_0 threshold is a general-purpose value that specifies the level of similarity needed for a character string to be inserted in the confusion list for a dictionary word. Any character string that does not pass this threshold can still be included in the confusion list for that dictionary word. However, this only happens if the frequency of the character string exceeds the general-purpose threshold t_{conf} and the similarity exceeds δ_1. This second level of filtering provides a way to include frequent strings in the confusion set but only if their similarity to the dictionary word is greater than a minimum value. The hypothesis generation step of the algorithm depends on $t_{\text{recog},l}$. This threshold is dependent on the length of the word so that it can compensate for the averaging of the similarity values.

3. EXPERIMENTAL RESULTS

Experiments were conducted to investigate several aspects of the proposed word recognition algorithm. An implementation of the algorithm was constructed

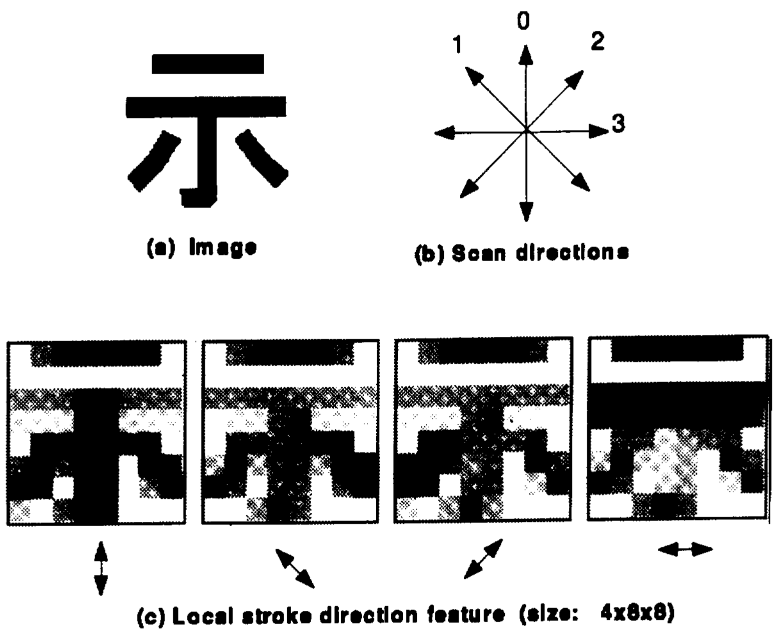

Fig. 6. Chinese character image and its local stroke direction (LSD) feature vector.

that located keywords in a Chinese language document. This image is the introduction from a book on Chinese OCR techniques that was scanned at 300 ppi in binary mode.[9] The document image was degraded to simulate the noise present in poor-quality facsimiles or photocopies. The recognition performance of the proposed algorithm was compared to that of a character recognition algorithm and a post-processing technique.

The training data for building the confusion tables was composed of two million characters of running text from the Pin–Yin Hanzi (PH) corpus[10] plus three million characters of running text from the China News Digest (distributed by electronic mail).

The dictionary was composed of the 20 keywords shown in Table 1.

Fourteen of those words occur 48 times in the test image and the other six words do not occur. The length of each keyword as well as its English translation are given. The confusion lists on average contained 16.9 character strings for each word.

The similarities between the characters in the test image and the characters in the dictionary words (and confusion sets) were calculated as described above. A Kunlun 96 $\times$ 96 pixel Sung font was used as the font training data. Noise was introduced into the test image by downsampling it to 200 ppi and 100 ppi and corrupting the 300 ppi and 200 ppi versions with a Gaussian bit-flip model. The value of a pixel was changed (from black to white or white to black) if the absolute value of a random number drawn from a $N(0,1)$ distribution was greater than 1. Three levels of noise were applied by repeating this procedure three times.

Figure 7 shows examples of the test data. Non-degraded versions of the sample at the three resolutions (300 ppi, 200 ppi, and 100 ppi) are shown in the top row of Fig. 6. The second, third, and fourth rows show how the images look after one, two, and three iterations of noise. It can be seen that most characters are severely broken after two or three iterations of noise. At 100 ppi, many of the characters are unrecognizable in isolation.

The experimental results are reported in Table 2. The proposed algorithm for word recognition is compared to a character recognition technique and a post-processing algorithm. The character recognition algorithm com-

Table 1. Keywords in the dictionary

Length of Chinese word	English translation	Length of Chinese word	English translation
2	Recognition	4	Formal language
2	Chinese character	3	Automaton
3	Computer	2	System
2	Information	5	Electron microscope
2	Pattern	4	Operating system
2	Technique	4	Programming design
2	Chinese language	3	Percentage
2	Mathematics	3	Workstation
4	Artificial intelligence	2	Chinese language
5	Language and character study	2	Language character

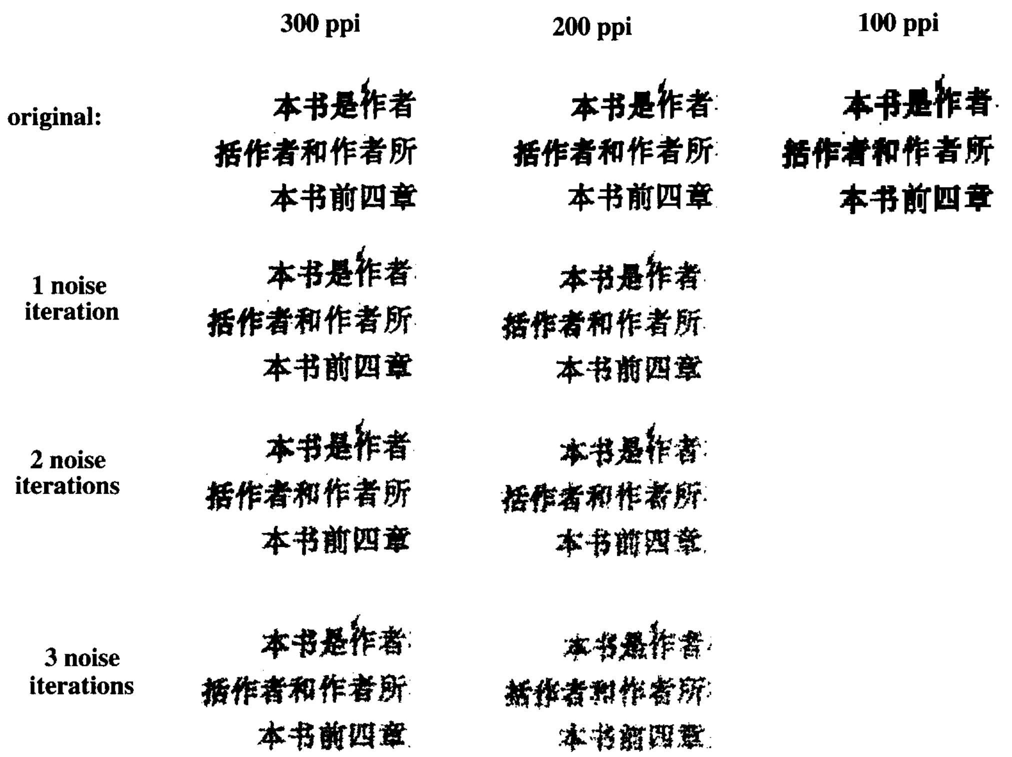

Fig. 7. Examples of test data.

Table 2. Recognition performance comparison

	ppi / Noise	300 ppi				200 ppi				100 ppi
		Clear	1	2	3	Clear	1	2	3	Clear
Character recognition performance	Top 1	96.7	94.2	91.6	86.8	94.8	85.0	77.1	64.3	67.1
	Top 5	99.5	99.0	97.7	96.5	99.2	96.5	93.6	83.9	88.9
	Top 10	99.7	99.3	98.7	97.5	99.7	98.3	95.4	89.3	93.4
Post-processing performance	Top 1	89.5	84.9	87.2	75.6	91.9	66.3	60.5	38.4	45.4
	Top 5	100.0	98.8	93.0	93.0	100.0	91.9	87.2	74.4	83.7
	Top 10	100.0	98.8	95.3	93.0	100.	94.2	94.2	82.6	88.4
Word recognition performance	Correct	100.0	100.0	100.0	100.0	100.0	98.8	96.5	94.2	96.5
	Error	0.0	0.0	0.0	0.0	0.0	0.0	2.3	2.3	0.0
	Missing	0.0	0.0	0.0	0.0	0.0	1.2	3.5	5.2	3.5

pared the LSD feature vector extracted from characters in the test image to characters in the font training data. The percentage of characters from the test image that are correctly located as the first choice of the character recognition technique are reported. The percentage correct in the five and 10 best choices found by the LSD recognizer are also given.

The post-processing algorithm was given the *m* best decisions from the character recognition algorithm. It was assumed that a keyword could be located if all its characters (excluding at most one) were found in the *m* best decisions. The percentage of keywords correctly located by this criterion are reported. This is an estimate of the expected performance of a post-processing technique.

The performance of the keyword recognition algorithm is reported as the percentage of keywords in the test image that are correctly located as well as the number of errors that occurred, that is, the number of keywords that were incorrectly recognized. The percentage of keywords in the test image that were not located by this technique are also reported.

The experimental results show that the proposed algorithm for keyword recognition outperforms the post-processing algorithm in all cases. Correct rates above 94% are achieved in the 300 and 200 ppi images. When the 100 ppi image was processed, the post-processing algorithm needed 10 choices to achieve an 88% correct rate, while the keyword recognition method found 96.5% of the target keywords correctly with a 0% error rate.

4. DISCUSSION AND CONCLUSIONS

An algorithm for keyword recognition in Oriental language document images was presented. Feature vectors extracted from sequences of characters were compared directly to feature vectors for words. A simultaneous comparison to character strings that are likely to be confused with each dictionary word was used to suppress false positives. This provides a method that effectively uses word level contextual information directly at the image level. The bypassing of an intermediate character recognition step allows for the recognition of words in the presence of noise that would make successful post-processing of those decisions versus a dictionary difficult.

Future work on this method should include development of a pre-processing step predicting the vocabulary contained in an input document. Such a technique has been successfully applied to English language documents(11) and could significantly reduce the run time of the matching process. A dynamic programming method should also be applied to speed up the matching of character images in the input document to dictionary entries.(6) An HMM formulation is also an obvious next step.

Acknowledgements—This work was partially supported by a grant from the Ricoh California Research Center while the authors were at the Center of Excellence for Document Analysis and Recognition, State University of New York at Buffalo. The authors also thank the anonymous reviewers who contributed several useful comments.

REFERENCES

1. S. Mori, C. Y. Suen and K. Yamamoto, Historical review of OCR research and development, *Proc. IEEE* **80**(7), 1029–1058 (1992).
2. K. Seino, Y. Tanabe and K. Sakai, A linguistic post-processing based on word occurrence probability, in *From Pixels to Features III: Frontiers in Handwriting Recognition*, S. Impedovo and J. C. Simon, eds, pp. 191–199. Elsevier, Amsterdam (1992).
3. T. K. Ho, J. J. Hull and S. N. Srihari, A computational model for recognition of multifont word images, *Mach. Vision Appl.*, special issue on Document Image Analysis, Summer, **5**(3), 157–168 (1992).
4. J. Zhu and J. J. Hull, Image-based word recognition in Oriental language document images, *Int. Conf. on Pattern Recognition*, B, Jerusalem, Israel, 9–13 October, pp. 300–303 (1994).
5. T. Akiyama and N. Hagita, Automated entry system for printed documents, *Pattern Recognition* **23**(11), 1141–1154 (1990).
6. T. K. Ho, J. J. Hull and S. N. Srihari, A word shape analysis approach to lexicon based word recognition, *Pattern Recognition Lett.* **13**, 821–826 (1992).
7. S. Naito, K. Komori and S. Mori, Stroke density feature for handprinted Chinese character recognition, *J. IECE Japan* **PRL**, 81–32 (1981).
8. J. J. Hull, S. N. Srihari and R. Choudhari, An integrated algorithm for text recognition: comparison with a cascaded algorithm, *IEEE Trans. Pattern Analysis Mach. Intell.* **PAMI-5**(4), 384–395 (July 1983).
9. X. Z. Zhang, *Chinese Recognition Techniques*. QingHua University, Beijing, China (1992) (in Chinese).
10. J. Guo and H. C. Liu, PH—A Chinese corpus for Pinyin–Hanzi transcription, ISS Technical Report, National University of Singapore, Singapore (1992).
11. J. J. Hull and Y. Li, Interpreting word recognition decisions with a document database graph, *Proc. Second Int. Conf. on Document Analysis and Recognition*, Tsukuba Science City, Japan, 20–22 October, pp. 488–492 (1993).

About the Author—JASON ZHU received the B.S. and M.S. degrees in Computer Science from Beijing University in 1986 and 1989. He also received an M.S. degree in Computer Science from the State University of New York at Buffalo in 1993. He is now a software design engineer with Microsoft Corporation. Before that, he was a software engineer at the Progress Software Corporation. His areas of interest include Chinese information processing, pattern recognition, database development and applications, and software internationalization and localization.

About the Author—TAO HONG received a B.S. degree in Computer Science from Beijing University in 1986, an M.S. in Psychology from Beijing University in 1989, and a Ph.D in Computer Science from the State University of New York at Buffalo in 1995. He is presently a research scientist in CEDAR at the State University of New York at Buffalo. His areas of interest include pattern recognition, visual document analysis, and natural language processing.

About the Author—JONATHAN J. HULL received the B.A. degree in Computer Science and Statistics and the M.S. and Ph.D. degrees in Computer Science from the State University of New York at Buffalo (SUNYAB) in 1980, 1983 and 1987, respectively. He is currently a Senior Computer Scientist at the Ricoh California Research Center in Menlo Park, California, where he also heads the Document Analysis Research Group. From 1987 to 1994 he was a member of the research faculty of the Department of Computer Science at SUNYAB and was also Associate Director of the Center of Excellence for Document Analysis and Recognition (CEDAR). His research interests include document analysis, image processing, computer vision, pattern recognition, and information retrieval. Dr Hull is a member of the ACM, the IEEE Computer Society, the Pattern Recognition Society, and is an Associate Editor of IEEE PAMI and Pattern Recognition. He served as Program Chair for Document Analysis of the Third Annual Symposium on Document Analysis and Information Retrieval at the University of Nevada at Las Vegas in April, 1994. He also served as co-chair of the Second IAPR Symposium on Document Analysis Systems, Malvern, Pennsylvania in October 1996.