COMPUTER VISION AND IMAGE UNDERSTANDING
Vol. 63, No. 1, January, pp. 66–74, 1996
ARTICLE NO. 0005

Font and Function Word Identification in Document Recognition

SIAMAK KHOUBYARI* AND JONATHAN J. HULL†

Department of Computer Science, Center of Excellence for Document Analysis and Recognition, State University of New York at Buffalo, Buffalo, New York 14260

Received November 4, 1993; accepted October 3, 1994

An algorithm is presented that identifies the predominant font in which the running text in an English language document is printed. Frequent function words (such as *the*, *of*, *and*, *a*, and *to*) are also recognized as part of the font identification. Clusters of word images are generated from an input document and matched to a database of function words derived from fonts and document images. The font or document that matches best provides the identification of the predominant font and function words. This technique takes advantage of the fact that most machine-printed documents are prepared with a single predominant font. Also, the repeated words in the document are utilized to overcome noise in the input. Advantages of this technique include its use as a preprocessing step for a document recognition algorithm. Experimental results show high accuracy is achieved on a database of original and degraded document images. © 1996 Academic Press, Inc.

1. INTRODUCTION

An important part of many algorithms that convert digital images of machine-printed text into their ASCII equivalent is information about fonts. Often this information is embedded in those algorithms by training them on most of the fonts they will encounter in practice [5] (hence the term ''omni-font'' classifier). An alternative is to allow for the interactive learning of new fonts at run-time [10]. A disadvantage of such methods is that they may need to be trained on hundreds or thousands of fonts. Also, their recognition performance may degrade as more fonts are added to their training base. The usefulness of font information in character recognition is illustrated by an approach that automatically adapts a 100-font classifier to a specific font as it is being recognized [1]. A significant improvement in recognition accuracy was observed.

An alternative approach is to employ a preprocessing step that identifies the font used in a document. Then the training data (feature descriptions, etc.) about only that font would be used during recognition. This would reduce the confusion caused by training on many fonts and would effectively reduce the recognition problem to choosing the correct class from one font rather than from many fonts.

One method for identifying the font in which a document is printed is to match the individual character images to the character images in a font database. The font of the identified characters provides the desired information. A disadvantage of such a technique is that it only uses visual information from isolated characters and is thus sensitive to noise that is commonly present in photocopies and facsimiles.

The font identification algorithm presented in this paper detects the *predominant* font in a given document, that is, the single font that is used in most documents to print more than 90% of the running text. The application of a classifier specialized for the identified font would assume that a document segmentation algorithm exists that can differentiate blocks of running text from section headings, italicized passages, tables, and so on. The specialized classifier would then be applied to the running text and an omni-font classifier to the remainder of the document.

The font identification technique presented here combines information from the image of a text page with constraints from the language in which a text is written to identify its predominant font as well as several of its function words [8]. The images of the words in an input document are compared and clusters of equivalent words are generated. Previous results have shown the utility of word image clustering [4] and that this approach can locate clusters of function words (such as *the*, *of*, *and*, *a*, and *to*) with high reliability in the presence of noise [7]. A prototype is also generated for each cluster. The prototype is an average of the individual word images in the cluster and can have less noise than the individual words. The clustering and prototype generation step thus uses the context of word images to compensate for noise that could hinder the performance of a character-based font identification algorithm. The noise reduction provided by clustering is especially useful if the input document has been degraded

* Present address: Caere Corporation, Los Gatos, CA. E-mail: siamakk@caere.com.

† Present address: Ricoh California Research Center, Menlo Park, CA. E-mail: hull@crc.ricoh.com.

1077-3142/96 $18.00

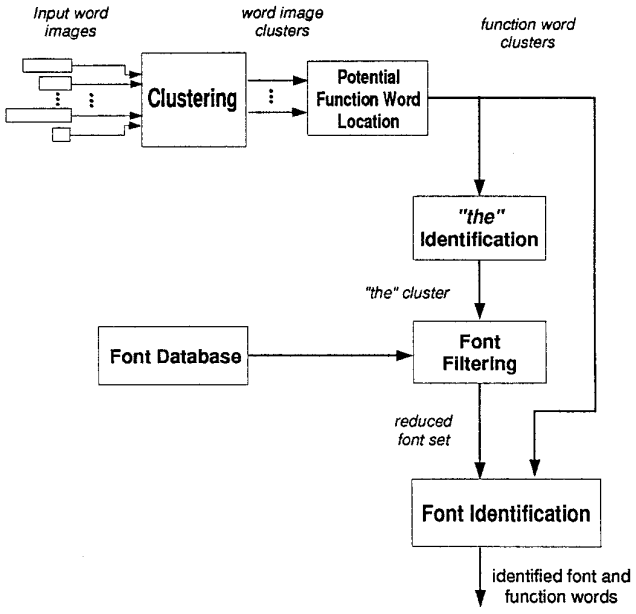

FIG. 1. The major components of the font identification algorithm.

```
WordDist (Word1, Word2)
    CurrentDist ← ∞
    RelativePositions ← { NONE, UP, DOWN, LEFT, RIGHT }
    estimate lower baselines for Word1, Word2
    center Word1, Word2
    for each shift in RelativePositions do
        TempImage ← xor (Word1, Word2, shift)
        TempDist ← sum-of-squares (DistMap (TempImage))
        CurrentDist ← min (TempDist , CurrentDist)
    end for
    return (CurrentDist)
```

FIG. 2. The word image distance measure.

with a uniform noise source. This occurs, for example, when a copier produces low-contrast images.

The cluster prototypes from an input document are matched to a database of function words in various fonts. The database is determined from a collection of fonts and document images. A font in the database is represented by the function words in that font generated at a number of point sizes. A document in the database is represented by the clusters of prototypes for the function words in that document. Thus the font identification problem is reduced to identifying the font or document in the database with function words that are visually similar to those in the input document. Images of the individual characters in the fonts and documents in the database are also stored for subsequent use by a recognition algorithm.

The identification of the specific font name (for example, Adobe Times Roman 11 point) is obtained directly if the algorithm matches the input to a font in the database. This same information can also be obtained when the input is matched to a document if the document is tagged with the name of its predominant font. However, this may not be relevant for a subsequent recognition algorithm since it only needs the individual character images associated with a matched document. Determining the predominant font by matching the input to stored documents provides the additional advantage that the other fonts (i.e., italic, bold, etc.) used in those documents would also be available for use by a document recognition system. This would be useful if the collection of fonts used in the input was the same as that used in the matched database document.

A further advantage of the approach is that the database of fonts can be easily expanded by adding new fonts or documents images. Using document images has the benefit that a recognition system could be specialized for the documents that it usually processes. This can help overcome the need for a database that contains all the fonts and sizes that it may ever encounter. Including several similar fonts should have little effect on the accuracy of a subsequent recognizer since it will still have available a more accurate representation of the font contained in an input document than an omni-font approach would.

The rest of this paper discusses the font identification algorithm. The steps of the approach are presented including the clustering algorithm and the image-matching technique. Also, experimental results are discussed that show the usefulness of the method for a database of scanned journal pages. This is followed by a presentation of conclusions and future directions.

2. ALGORITHM

The following sections describe the steps in the font identification algorithm, as illustrated in Fig. 1. The word images in an input document are first clustered and clusters that represent potential function words are chosen. Then the cluster that contains the word "the" is identified. This cluster is then matched to the "the" entries in the font

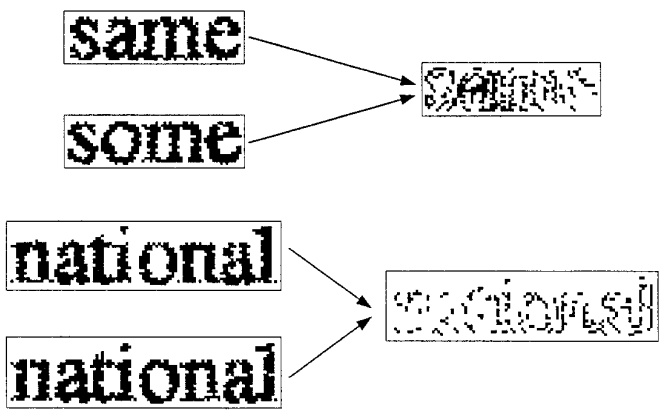

FIG. 3. Exclusive-or results for sample word pairs.

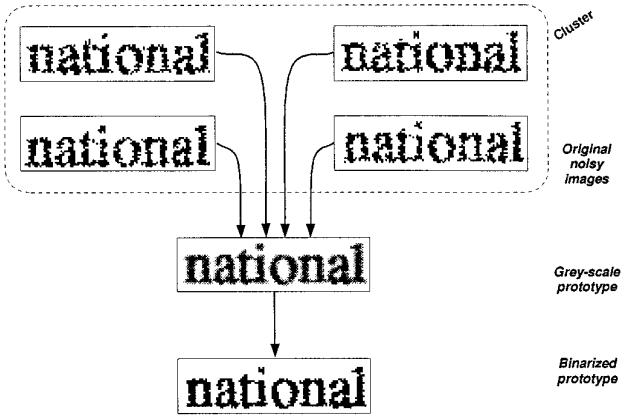

FIG. 4. An example of prototype generation.

database to choose a reduced font set. The decision about which of these is the predominant font in the document is made by the font identification step.

2.1. *Word Image Clustering*

Word images segmented from a document are sent sequentially to an agglomerative clustering algorithm [7] that places the images in clusters of equivalent words. The ideal result is a number of clusters, each containing all the occurrences of a specific word in the document. The clustering step does *not* perform any recognition, as it only locates images of *equivalent* words.

During clustering, the decision about whether to place a word in a cluster or to start a new cluster is made by a distance measure that compares an input word image to an image in the cluster. The technique used here is similar to that discussed in Ref. [2]. The three main steps in the process are outlined in Fig. 2. The lower baselines of the two input words are first calculated. These are the lines just below the lower-case letters that have neither ascenders nor descenders. The estimation of the lower baseline in a word is performed by finding a horizontal band of fixed height that covers the densest area of the image. This area represents the ''middle'' region of the word. The lower baseline is below and adjacent to this region.

The word images are then centered on their lower baselines and the *exclusive-or* (XOR) between the two images is calculated (by **xor**). The resulting image contains black regions where the two images differ. Figure 3 shows the result of calculating the exclusive-or between two pairs of images. The first involves two words with different identities; hence the process results in an image that has large black areas (referred to as ''blobs''). The second example shows two words that are equivalent, but contain a moderate amount of noise. In each case, if the distance were determined by simply counting the number of black pixels in the XOR image, the result would be very similar for both pairs of images.

The blobs in the XOR image are accounted for by calculating its *distance map* [3]. A distance-mapping algorithm (the **DistMap** function) replaces each black pixel with its distance to the nearest white pixel. This assigns higher values to the interior pixels than those close to a boundary. A scalar distance value is calculated by taking the sum of squares of the values in the distance map. The result is higher distances for XOR images that contain dense blobs, such as those at the top of Fig. 3, and lower distances for XOR images that contain random black pixels, such as those shown at the bottom of Fig. 3. The distance calculation is performed at five displacements (none, and one pixel up, down, left, and right) and the minimum distance over these displacements is chosen. The decision about whether two word images are equivalent (i.e., images of the same word) is made by comparing the value determined by **WordDist** to a threshold.

The threshold is a percentage (α) of the area of the larger of the two words, parameterized by the aspect ratio of the wider word. If the word with the largest aspect ratio has more than β times *x_height* columns (where *x_height* is the height of the middle region of the word), α is set to α times γ. The objective of this procedure is to provide a consistent method of setting the threshold that automatically adapts to changes in font and word size. The check to determine whether the larger aspect ratio exceeds a certain value is needed since a more restrictive threshold is necessary for shorter words where small differences in the distance values are indicative of differences in word identities. The settings of α, β, and γ used in the implementation are given in the Experimental Results section.

Prototype generation. One of the advantages of clustering is that the images in a cluster can be used to generate a prototype that has less noise than the individual word images. The prototype for a cluster is generated by first centering the individual words on their estimated baselines.

TABLE 1
Word Pair Statistics for the Top 20 Function Words

Word	No.	No. pair	Ratio (%)	Word	No.	No. pair	Ratio (%)
the	10570	5116	48.40	his	1439	675	46.91
of	6267	88	1.40	it	1346	238	17.68
and	4411	37	0.83	for	1272	62	4.87
to	4108	267	6.50	as	1244	64	5.14
a	3406	1590	46.68	with	1089	48	4.41
in	3356	240	7.15	I	844	128	15.17
that	1929	241	12.49	be	833	334	40.10
is	1776	497	27.98	on	825	33	4.00
he	1539	239	15.53	not	825	220	26.67
was	1449	466	30.78	by	817	45	5.51

Each pixel position in the prototype image is assigned a value equal to the proportion of times that pixel is black in the centered images. The result is a gray-scale image, where the darkest regions represent the areas of the word with the highest probability of being black. This image is binarized to generate the prototype. As mentioned before, this process is most effective at removing noise that is uniform across a page image. An example of prototype generation is shown in Fig. 4.

2.2. *Potential Function Word Location*

A set of word image clusters are identified that can potentially contain function words. This process uses the characteristic that function words are frequent and usually contains less than four characters. This is implemented by ranking the clusters in an input document by the number of images they contain and choosing the 10 clusters that contain words that are shorter than a fixed threshold.

2.3. *"The" Identification*

This step selects the cluster that contains the word "the" from among the 10 potential function word clusters. Although the identities of the words are unknown at this point, the "the" cluster is located by analyzing the number of entries in each cluster and the number of times the entries in each cluster follow entries from other potential function word clusters in the original document. Table 1 shows some statistics that illustrate the function word adjacency characteristic.[1] For each of the top 20 most fre-

[1] These statistics were obtained from a large section (150,000 words) of the Brown Corpus, a large collection of English text [9].

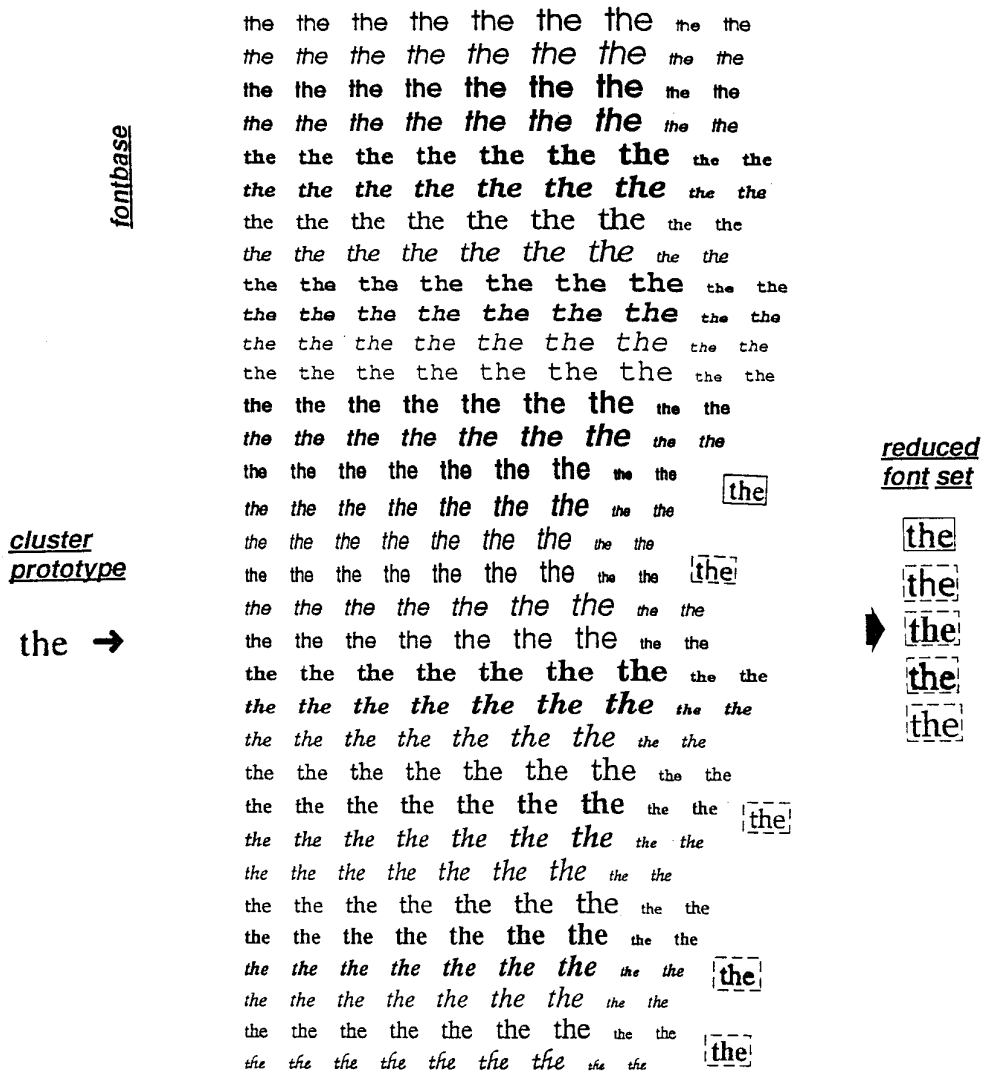

FIG. 5. An example of initial font filtering.

```
FontIdentification ( reduced_font_set , potential_function_words )
    for each font F_i in reduced_font_set do
        hits ← misses ← ratio_sum ← 0
        for each cluster C_j in potential_function_words do
            P_j ← GetPrototype ( C_j )
            decisions_{i,j} ← NULL
            for each function word FW_k in F_i do
                P_k ← GetFunctionWordPrototype (FW_k , F_i)
                distance ← WordDist ( P_j , P_k )
                threshold ← GetThreshold ( P_j , P_k )
                if distance < threshold then
                    hits ← hits + 1
                    ratio_sum ← ratio_sum + (distance/threshold)
                    decisions_{i,j} ← FW_k
                    break
                end if
            end for
            if decisions_{i,j} == NULL then misses ← misses + 1
        end for
        scores_i ← hits × (hits − ratio_sum − misses)
    end for
    SortFonts (in: scores , reduced_font_set ; out: best_matching_font , b)
    Output (best_matching_font, decisions_b )
```

FIG. 6. Font identification algorithm.

quent function words, the word's frequency is shown, followed by the number of occurrences of the word that are preceded by another function word in the set. As shown by the word-pair counts, the word "the" is preceded by the highest number of function words. The next highest function-word-pair frequency is that of the word "a". However, any confusion with "the" is avoided by comparison of the absolute frequencies of the two words as well as their physical size.

A similar analysis of the frequency of the potential function words in a document is performed to locate the "the" cluster. For each cluster C of the N potential function word clusters (say, $N = 10$), a count is kept of the number of words from the other $N - 1$ clusters which precede a word in C in the input document. The cluster which possesses the highest function-word-pair frequency is then identified as the "the" cluster. Experimental results have shown that the "the" cluster can be located with high accuracy using this technique.

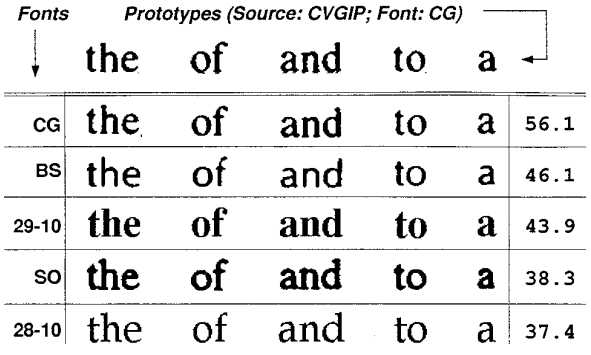

FIG. 7. An example of the font identification results.

2.4. *Font Filtering*

To increase efficiency, a filtering step is used to constrain the set of candidate fonts. This is done using the prototype from the identified "the" cluster. The "the" prototype is compared to each "the" image in the font database using the word distance measure. Those fonts that contain matching "the" images are chosen for the reduced font set. That is, if the distance between the "the" prototype from the input document and the "the" image in the font is less than the threshold for the pair, the font is placed in the reduced font set.

Figure 5 shows an example of the font filtering process. A "the" prototype generated from a document is shown on the left. The subset of the "the" images in the font base that match the input prototype is illustrated by the dashed boxes around each entry. The "the" images in the reduced font set are also shown on the right.

2.5. *Font Identification*

Figure 6 shows the algorithm used to make the final decision about the identity of the predominant font. Each font F_i in the reduced font set is compared to the clusters of potential function words found in the input document. The prototype for a potential function word (returned by **GetPrototype**) is compared to each of the prototypes for the function words in F_i (returned by **GetFunctionWord Prototype**). If the distance between the prototype for the potential function word and the prototype for the function word in the font (determined by **WordDist**) is less than a threshold for the two words (returned by **GetThreshold** as explained in Section 2.1), then a count of the number of matches (*hits*) is incremented and a measure of the quality of the match between the two prototypes (*ratio_sum*) is updated. The identity of FW_k is assigned to the cluster of words that match it. The first successful match between a potential function word and a function word in a font terminates the comparison for that potential function word. A count of the number of potential function words that did not successfully match any function word in the reduced font set is also maintained in the variable *misses*.

The variable *ratio_sum* accumulates the proportion of the distance to the threshold for the successful matches. Low values for *ratio_sum* indicate that close matches have been obtained for a given number of hits.

The overall *score* for the font F_i is calculated as the product of the number of hits and the number of hits minus the *ratio_sum* and the number of misses. This is a heuristically determined measure that allows the fonts in the reduced set to be ranked by the function **SortFonts**. This function sorts the scores and returns the identity of the font with the highest score and an index (b) that identifies it in the reduced font set. The identification of that font

TABLE 2
Font Identification Results for Original and Degraded Images

Article (font)	Identified fonts (scores)	No. hits	Corr. hits	% of text
BioScience (BI)	BI (22.3) CG (18.2) CI (17.4)	7	7	18.5
	BI (37.0) BU (14.4) CG (18.8)	8	8	17.7
BusinessSoc (BU)	BU (74.9) 28-10 (44.8) CO (24.0)	10	10	24.7
	BU (70.8) 28-10 (43.5) BI (37.8)	10	10	23.2
CACM (CO)	CO (53.0) IG (33.7) 29-09 (27.0)	9	9	21.8
	CO (37.0) IG (31.6) 29-09 (25.9)	8	8	17.8
CVGIP: GMIP (CG)	CG (82.2) 29-10 (46.2) SO (43.3)	10	10	23.3
	CG (56.1) BU (46.1) 29-10 (43.9)	9	9	17.0
CVGIP: IU (CI)	CG (56.1) CI (53.6) 32-11 (39.9)	9	9	23.0
	CG (74.4) BU (39.1) 32-11 (36.0)	10	10	23.1
Discover (DI)	DI (54.0) IG (17.1) 29-09 (14.5)	9	9	18.1
	DI (46.2) IG (12.3) 29-09 (11.1)	9	9	18.9
Geotectonics (GE)	GE (21.3) — — — —	8	8	17.8
	GE (23.5) — — — —	8	8	19.8
IEEE Computer (IC)	IC (55.0) IE1 (36.5) 32-09 (27.7)	9	9	20.6
	IC (27.3) IE1 (25.6) PR (10.1)	8	8	17.8
IEEE CGA (IG)	IG (59.1) CO (36.8) IP (31.8)	9	9	19.2
	IG (41.8) PR (18.6) CO (10.4)	9	9	16.6
IEEE Expert 1 (IE1)	IE1 (55.8) IC (52.8) PR (25.7)	9	9	20.1
	IC (39.2) IE1 (32.2) 32-09 (−6.0)	9	9	20.4
IEEE Expert 2 (IE2)	IC (53.9) IE1 (51.8) 32-09 (29.3)	9	9	17.1
	IC (29.2) IE1 (25.0) 32-09 (11.24)	8	8	16.8
IEEE PAMI (IP)	IP (21.7) IC (5.9) IE1 (5.2)	7	7	23.5
	IP (15.2) IE1 (7.6) IC (7.4)	7	7	22.8
J. Space and Rockets (JS)	JS (38.5) 29-09 (00.0) PR (00.0)	8	8	23.3
	JS (58.3) IC (14.9) IE1 (13.5)	9	9	24.1
Pattern Recognition (PR)	PR (38.1) IE1 (19.8) IC (19.0)	8	8	21.5
	PR (32.5) IE1 (17.6) IC (16.4)	8	8	22.3
The Plant Cell (PC)	PC (62.5) 13-08 (22.9) 14-08 (−2.0)	9	9	23.6
	PC (54.1) 13-08 (22.7) 14-08 (−2.5)	9	9	22.9
Proc. IEEE (PI)	PI (83.2) 28-09 (41.0) 25-09 (38.7)	10	10	24.9
	PI (73.2) IG (35.6) DI (26.7)	10	10	25.2
Scientific American (SA)	SA (33.4) TR (27.6) PI (23.8)	8	8	18.0
	SA (47.2) PI (44.1) CO (28.0)	9	9	21.8
Society (SO)	SO (58.1) CG (34.5) CI (34.1)	9	9	18.0
	CG (34.7) SO (34.4) CI (17.4)	9	9	18.4
TREE (TR)	TR (63.1) PI (34.1) SA (11.5)	9	9	25.7
	TR (76.2) PI (49.9) BU (12.6)	10	10	23.2
AVG	— (49.0) — (30.3) — (20.3)	8.8	8.8	20.9%

is returned along with the vector of *decisions* that maps identifications for function words onto clusters of word images.

An example of the results of font identification is shown in Fig. 7. The prototypes for the five most frequent function words from a journal article are shown on the top line. The matching function word images from the five fonts in the reduced font set are shown on the other lines in the figure. The table is sorted by the overall score for the fonts. That is, the top-ranked font for the document is *CG*. Inspection of the rest of the images shows that the other proposed fonts are also similar to the correct one. However, the score for the correct font is higher than that of the rest.

3. EXPERIMENTAL RESULTS

The font identification algorithm was tested by developing a font database that includes prototypes generated from fonts as well as from documents. Word images from test documents were clustered and their fonts were identified. Both clean and degraded test documents were used where the degraded documents were generated by a noise-modeling algorithm.

The font samples in the font database were generated from the 33 postscript fonts listed in the Appendix. The Unix ditroff program was used to generate postscript examples for each of the 20 function words listed in Table 1 in nine point sizes (8, 9, 10, 11, 12, 13, 14, 15, and 16). The

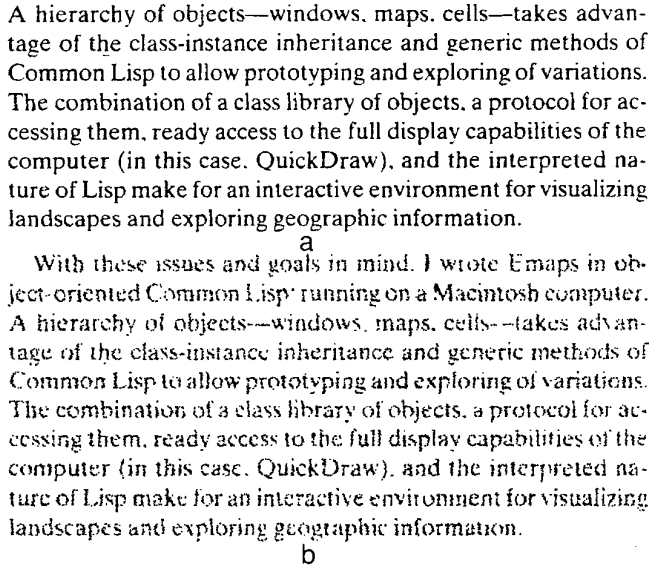

FIG. 8. (a) Original images and (b) result of degradation.

postscript for these words were converted to 300 pixel-per-inch (ppi) bitmaps using the NeWs software in the Sun operating system. This yielded 297 font entries.

An additional set of 18 documents images were placed in the font database. All the documents listed in the Appendix were used except for IE2. Each of the documents was photocopied and the copies were scanned at 300 ppi, segmented into isolated words, and each word was manually assigned its ASCII truth. The first 1000 word images in each document were automatically clustered and the clusters for the top 20 function words were used to represent the document.

The test data were generated from the next 1000 words in each document including IE2. This provided a separation of all the documents into training and testing data. The clusters for each 1000-word test document were generated and the font of each document was determined. The parameters of the threshold used during clustering were $\alpha = 0.27$, $\beta = 4.5$, and $\gamma = 0.73$.

To test the tolerance of the clustering and matching procedure to noise in the input image, the test data were corrupted by a document degradation model [6] prior to clustering. Parameters were used for this model that simulated the acquisition of a uniform low-contrast page image. Figure 8 shows the effect of the noise on a section of running text.

Table 2 shows the results from the font identification algorithm. The first column of the table shows the full name and an abbreviation of the name of the article. The second column shows the abbreviations and scores for the three font database entries that matched most closely. The font entries in the database are indicated by the code number shown in the Appendix followed by the point size. The top line in each group of two results provides the scores for the clean testing data and the second line the scores for the noisy testing data. The number of hits that occurred between a potential function word and a function word in the highest ranked font are also shown as well as the number of those hits that were correct, that is, where the identity assigned by the font identification algorithm was the same as the true identity. The percentage of running text in the test document that is covered by the identified function words is also given.

The results illustrate several interesting aspects of the algorithm. In 34 out of the 40 test cases the correct font was ranked highest. The *CVGIP: IU* article in both cases (clean and noisy) was ranked closest to the *CVGIP: GMIP* article. The *Geotectonics* article only had the other *Geotectonics* article in the reduced font set. This was because this journal used a unique Courier-like font that did not exist elsewhere in the database. The first *IEEE Expert* article had the best match to the *IEEE Expert* article in the database and the second best match to the *IEEE Computer* article. They exchanged positions when the noisy image was used. Interestingly, in both cases, the distance to the third-ranked font entry was significantly less than the second distance. The second *IEEE Expert* article exhibited similar behavior but in this case the *IEEE Computer* article was the best match in both the clean and noisy condition.

Overall, the function word identification was 100% correct for the function words it identified. On average, this covers about 21% of the word images in the test documents.

4. CONCLUSIONS AND FUTURE DIRECTIONS

An algorithm was presented that identified the predominant font in a document as well as a significant percentage of the frequent function words that it contains. The font information could be used to improve the performance of a character recognition algorithm by restricting the training data it uses. Also, knowledge of the predominant font could improve the later rendering of the ASCII recognition results. The identification of the function words can also be used to improve document recognition since function words are sometimes difficult to recognize when a document image is degraded because they contain few characters. Accurate identification of function words is useful because they supply syntactic constraints that can be used to help identify adjacent words.

Results showed that 34 out of 40 test cases were correctly identified with no ambiguity. The mismatches in all cases were fonts that were visually very similar to each other. About 21% of the function words in the test documents were recognized with no errors.

Future improvements to the algorithm could include a

modification to the word distance measure that would better account for variable intercharacter spacing. While none of the documents were observed to have this characteristic, some percentage of documents in regular use certainly do, such as those generated by some word processing software. This is also an issue if words generated from font samples are to be included in the font database. A modified version of the word distance measure would vertically segment two images into a number of strips. The strips in one word would be compared to the corresponding strips in the other word. The same distance value used here could be computed at a number of horizontal offsets. The distance for a pair of word images would be the sum of the minimum distances in each strip.

Another area of future interest is automatically building a font database. A thresholding mechanism could be developed that would separate reliable from unreliable decisions in font identification. Unreliable decisions would correspond to cases where the observed font does not exist in the database. The clusters from such documents could be used to construct new entries in the font database.

APPENDIX: FONT DATABASE

TABLE 3
Postscript Fonts in the Font Database

No.	Font name	No.	Font name	No.	Font name
1	AvantGarde-Book	2	AvantGarde-BookOblique	3	AvantGarde-Demi
4	AvantGarde-DemiOblique	5	Bookman-Demi	6	Bookman-DemiItalic
7	Bookman-Light	8	Bookman-LightItalic	9	Courier
10	Courier-Bold	11	Courier-BoldOblique	12	Courier-Oblique
13	Helvetica	14	Helvetica-Bold	15	Helvetica-BoldOblique
16	Helvetica-Narrow	17	Helvetica-Narrow-Bold	18	Helvetica-Narrow-BoldOblique
19	Helvetica-Narrow-Oblique	20	Helvetica-Oblique	21	NewCenturySchlbk-Bold
22	NewCenturySchlbk-BoldItalic	23	NewCenturySchlbk-Italic	24	NewCenturySchlbk-Roman
25	Palatino-Bold	26	Palatino-BoldItalic	27	Palatino-Italic
28	Palatino-Roman	29	Times-Bold	30	Times-BoldItalic
31	Times-Italic	32	Times-Roman	33	ZapfChancery-MediumItalic

TABLE 4
Documents in the Font Database

Code	Pages	Date	Journal
BI	738–742	1990	*BioScience*
BU	25–32	Winter 1991	*Business and Society Review*
CO	281–287	1990	*Communications of the ACM*
CG	522–528	1991	*CVGIP: GMIP*
CI	1–7	1991	*CVGIP: IU*
DI	69–76	May 1993	*Discover*
GE	411–415	1991	*Geotectonics*
IC	45–55	March 1991	*IEEE Computer*
IG	28–33	March 1993	*IEEE Computer Graphics and Applications*
IE1	13–23	April 1993	*IEEE Expert*
IE2	25–34	April 1993	*IEEE Expert*
IP	99–105	1991	*IEEE PAMI*
JS	437–443	1992	*Journal of Spacecraft and Rockets*
PR	419–429	1993	*Pattern Recognition*
PC	297–307	1993	*The Plant Cell*
PI	512–516	1990	*Proceedings of the IEEE*
SA	83–90	April 1992	*Scientific American*
SO	25–33	May/June 1992	*Society*
TR	417–419	1992	*TREE*

REFERENCES

1. H. S. Baird and G. Nagy, A self-correcting 100-font classifier, in *Proceedings, SPIE Conference on Document Recognition, San Jose, California, 1994,* pp. 106–115.
2. N. F. Brickman and W. S. Rosenbaum, Word autocorrelation redundancy match (WARM) technology, *IBM Journal of Research and Development* **26**(6), 1982, 681–686.
3. P.-E. Danielsson, Euclidian distance mapping, *Computer Graphics and Image Processing,* **14,** 1980, 227–248.
4. J. J. Hull, S. Khoubyari, and T. K. Ho, Visual global context: Word image matching in a methodology for degraded text recognition, in *Proceedings, Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, March 16–18, 1992,* pp. 26–39.
5. S. Kahan, T. Pavlidis, and H. S. Baird, On the recognition of printed characters of any font and size, *IEEE Trans. Pattern Anal. Machine Intell.* **PAMI-9**(2), 1987, 274–287.
6. T. Kanungo, R. M. Haralick, and I. Philips, Global and local document degradation models, in *Proceedings, Second International Conference on Document Analysis and Recognition, Tsukuba, Japan, October 20–22, 1993, pp. 730–734.*
7. S. Khoubyari and J. J. Hull, Keyword location in noisy document images, in *Proceedings, Second Annual Symposium on Document Analysis and Information Retrieval, Las Vegas, NV, April 26–28, 1993,* pp. 217–231.
8. S. Khoubyari and J. J. Hull, Font identification using visual global context, in *SPIE Conference on Document Recognition,* San Jose, California, 1994, pp. 116–124.
9. H. Kučera and W. N. Francis, *Computational Analysis of Present-Day American English.* Brown Univ. Press, Providence, RI, 1967.
10. H. Takahashi, A. Yamashita, N. Itoh, and T. Amano, A hybrid recognition algorithm with learning capability and its application to an OCR system, *Trans. IEICE* **E73**(4), 1990, 577–586.