Font Identification Using Visual Global Context

Siamak Khoubyari and Jonathan J. Hull

Center of Excellence for Document Analysis and Recognition
Department of Computer Science
State University of New York at Buffalo
Buffalo, New York
khoub-s@cs.buffalo.edu hull@cs.buffalo.edu

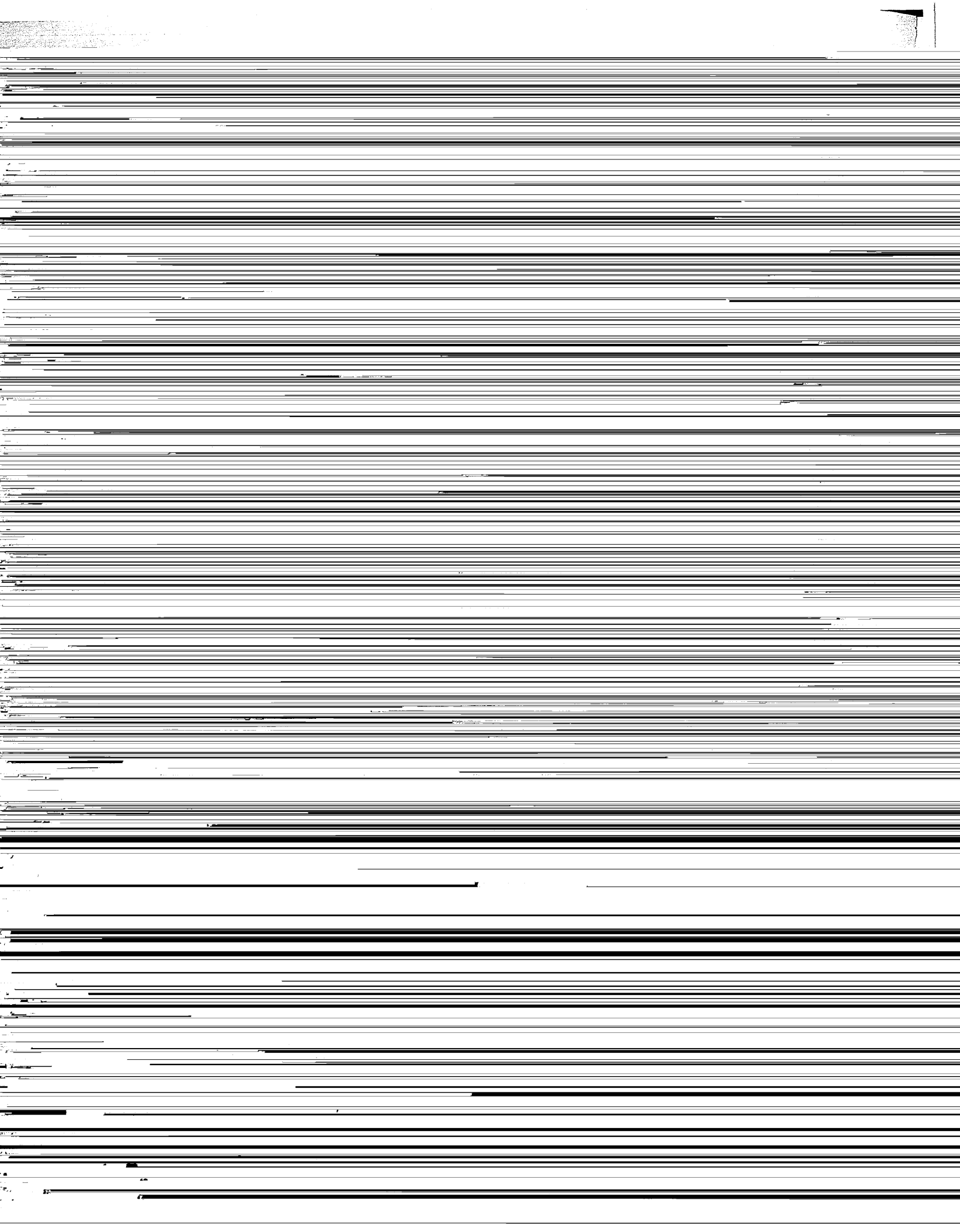
ABSTRACT

One method for detecting the font in which a document is written is to match the individual character images to a large 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.

This paper proposes an algorithm for font detection that uses contextual information from approximately 30 percent of the word images in a document to identify the font. This method utilizes the redundancy within and between words to compensate for noise that would be detrimental to the performance of techniques that use isolated characters.

The algorithm proposed in this paper is an extension of an earlier technique for the recognition of function words [4]. This method locates clusters of equivalent word images within a document and uses cluster characteristics to locate and recognize the function words (i.e., words such as *the, of, and, a,* and *to*). Also, an improved prototype (cluster center) that represents each cluster is calculated.

The method proposed here matches the cluster prototypes to a database of function words in various fonts. The database is determined from a collection of document images that have been segmented into isolated words. Each document contributes a unique set of function word



prototypes [4] is used in this work.

```
GetTop10 (in: all_clusters ; out: top_clus_list)
Get'the'Clus (in: top_clus_list ; out: 'the'_clus)
GetFontList (in: font_lib , 'the'_clus ; out: font_set)
for each font F_i in font_set do
    hits ← misses ← ratio_sum ← 0
    for each cluster C_j in top_clus_list do
        GetPrototype (in: C_j ; out: prototype_image)
        Autocorrelation (in: prototype_image , F_i_images ; out: distance , threshold)
        if distance < threshold then
            hits ← hits + 1
            ratio_sum ← ratio_sum + (distance/threshold)
        else
            misses ← misses + 1
        end if
    end for
    scores_i ← hits × (hits − ratio_sum − misses)
end for
SortFonts (in: scores , font_set ; out: best_matching_font , b)
if scores_b ≥ T then
```

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

Table 1: Word pair statistics for the top 20 function words

is less than the normalized threshold for the pair, the font is placed in the candidate set. The idea is that if a satisfactory word-level match is achieved for a word which has already been identified, the given font is a good candidate for more detailed analysis.

Ranking the Fonts

In the last step in the process, prototypes from all of the frequent function word clusters are used to rank the fonts in the font subset chosen by the above process. The grading scheme for the fonts in the reduced font set is as follows. Each match (a "hit") between a function word prototype from the document and a function word in one of the fonts produces a "score". This score is given by the ratio of the actual autocorrelation distance between the two word images to the normalized threshold for the pair. Therefore, a low ratio signifies a close match. By adding the individual scores, a measure of how closely the function words matched the font is obtained. A count of the number of non-hits ("misses") is also kept and used in the calculation of the overall score (goodness) for a given font. A non-hit occurs when a function word prototype

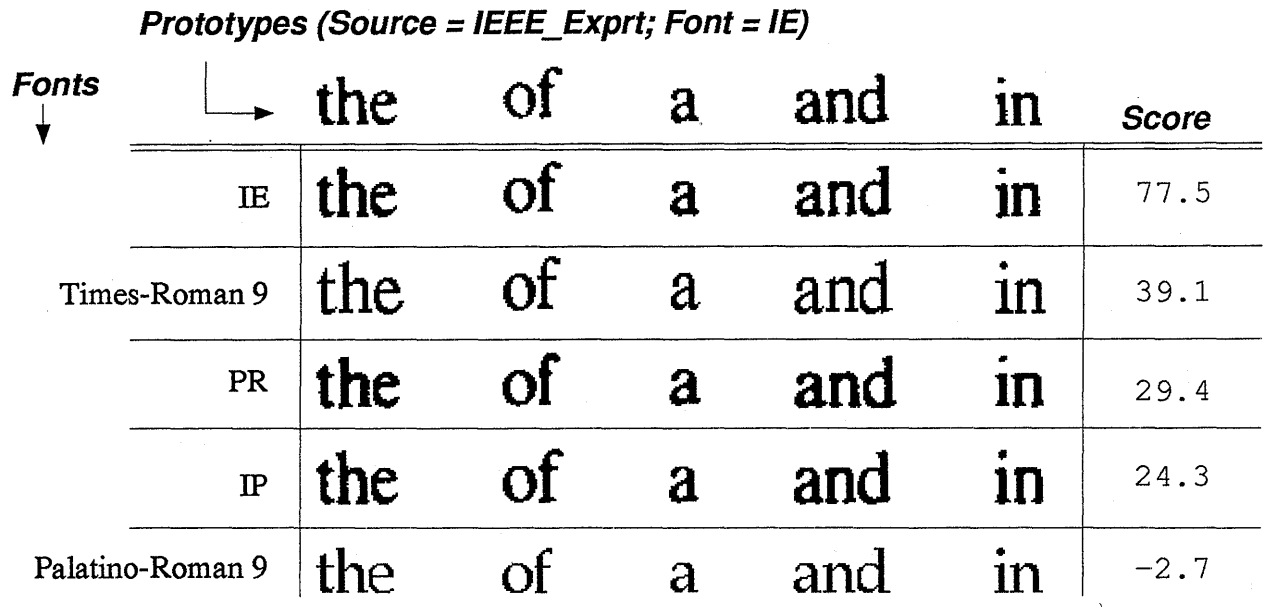


Figure 3: An example of the font identification results

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 of the fonts. That is, the top-ranked font for the document is *IE*. Inspection of the rest of the images shows that the other proposed fonts are also very similar to the correct one. However, the score for the correct font is much higher that that of the rest. This illustrates the ability of the proposed algorithm to detect such subtle differences among fonts. This also demonstrate the power of using visual context versus using individual characters.

3 Results

3.1 Experimental Environment

In order to fully test the font identification algorithm, a database of 30 journal articles from 28 unique sources was created. Each article contains several pages, each of which was photocopied, scanned and binarized. The text blocks were located, and word images extracted.

3.2 Font Identification Performance

The proposed algorithm was tested on ten of the articles in the database. The word clusters for the first few pages of each article were used as training data, while the rest of the article was

used as test data. In one case, no training was necessary, since a different article from the same source (IEEE Expert magazine) was already in the database. The font library also contains entries generated using 30 PostScript fonts in nine point-sizes (8 pt. to 16 pt.).

Table 2 shows the results from the font identification algorithm. Several components of the results are listed, all relating to the testing portion of each of the ten articles. The first column of the table displays the article name (an abbreviation of its source) and the font name associated with that article. The second column shows the performance of the function word location step (i.e., the number of frequent function words in the top 10 clusters of short words). The third column contains the number of fonts in the reduced candidate set. The average score of these fonts is given in the fourth column, while the fifth column shows the name and the score for the top-ranked font.

For example, in the article identified by "CACM", nine of the ten potential function word clusters were actually function words; seven fonts were proposed by the initial font filtering step; the average font match score of these fonts was 7.8, and the top-ranked font among them, which is the correct font (CM), had a score of 52.7. The results illustrate the accuracy of the font identification algorithm.

Of particular interest are the results for the article named "IEEE Expert". This article

4 Conclusions and Future Directions

An algorithm for font detection in document images was presented that effectively utilizes word and language-level contextual information to recognize the font in which a document is printed. Such a capability is useful for improving the performance of text recognition algorithms that