ABSTRACT

Clustering and identification of fonts in document images impacts on the performance of optical character recognition (OCR). Therefore font features and their clustering tendency are investigated. Font clustering is implemented both from shape similarity and from OCR performance points of view. A font recognition algorithm is developed to identify the font group with which a given text was created.

1 INTRODUCTION

The increasing diversity of font styles is a factor limiting the performance of general-purpose OCR systems. In fact new computer-generated fonts are being added to the existing fonts in the printing industry, their total running into hundreds. The detection of font style of a text is an obvious way to improve the performance of optical recognition algorithms.

We investigate the clustering behavior of fonts and propose a method for font/font class identification. One application of font clustering is the design of a more robust OCR system. The underlying scheme here is that the character recognition system will consist of a font cluster identifier unit followed by an OCR unit specific to each font group. A second application is the identification of individual fonts when reproduction of a document in its original appearance is desired. In addition font recognition aids in the recognition of logical structures of documents. Finally it is also conjectured that for many of these tasks it may suffice to find out the font cluster to which a specific text belongs, so that the cluster centroid can be used as the representative font. The goals of the study can be itemized as follows:

- To find the minimal set of clusters that provides adequate OCR performance across all fonts.
- To determine the representative font for each font cluster
- To develop an algorithm to detect the font type or cluster in a document.

There have been few studies on font recognition. Notably R. Morris [10] has considered classification of digital typefaces using spectral signatures. Khoubiyari and Hall in [1] have considered a scheme based on the identification function words e.g., “the”, “and”. Zramdini and Ingold [2, 9] consider a font identification approach based on the projection profiles. Our contribution is in the investigation of the clustering behavior of fonts and it’s impact on the OCR performance.

2 PRELIMINARIES: FONTS AND TYPEFACES

A font can be modeled by the following attributes[2, 9]:

- The **font family** like “Times”, “Helvetica” and “Courier”.
- The **size** expressed in typographic points.
- The **weight** of the font, having one of the following values: *light, normal or bold*.
- The **slope** indicating the orientation of the letters main strokes. A font could be *roman* or *italic*.
- The **spacing mode** specifying the pitch of the characters.

The fonts we have considered belong to 28 different font families and together with weight and slope varieties their total number amounts to 65 as listed in the Appendix. The fonts, are referred to in the paper via their associated ID number.

We assume that the font attributes of spacing mode and size have been normalized as follows: The spacing mode uniformized via the bounding boxes of characters; the size is normalized via bilinear interpolation to a standard size of \(32 \times 32\). Thus the font variables that remain are font family, slope and weight. For each font we considered the 68 different characters, consisting of the lowercase letters, uppercase letters and numerals. A total of \(68 \times 65 = 4420\) character images of different fonts were prepared using a computer graphics program.

There are several possible feature sets to be extracted from characters for recognition and clustering purposes [5]. In this paper we tested the following features: bitmaps, DCT coefficients, eigencharacters and Fourier descriptors.

3 FEATURES of CHARACTERS AND FONTS

3.1 Character Bitmaps

The 1024-dimensional binary pattern of a character is indicated by the lexicographic ordering of its \(32 \times 32\) bitmap. Any jitter, due to the finite resolution grid or interpolation, can be mitigated by considering four other templates in one pixel: up, down, right, left shift positions. The maximum of the match scores is considered for classification.

3.2 Fourier Descriptors [5]

The cosinusoidal and sinusoidal components of the \(x\) and \(y\) – projections of character chain codes have been used. For \(n\)
number of harmonics the total feature set contains $4 \times n$ features for each character.

3.3 Eigencharacters

The subspace features often prove to be more efficient in the classification task and for dimension $m = 2$ the visualization of the multivariate data scatter becomes useful. The subspace features for a character are obtained by projecting the original features on the subspace defined by the $m$ principal eigenvectors [7]. The large dimensional (e.g., 1024) eigenvalue problem is solved by a method described in [4].

3.4 DCT Features

Features can be extracted by considering the low order DCT transform coefficients of the $32 \times 32$ character block. Alternatively one can consider the juxtaposition of the selected DCT coefficients of the sub-blocks of the character images. For example for sub-block size 8, if one selects the first 4 coefficients of each block DCT, one obtains a $4 \times (32/8)^2 = 64$ element feature vector for each character.

3.5 Font Features

Font feature vectors are constructed by serial juxtaposition of the character feature vectors in some order. Thus if the bold characters denote symbolically the respective feature vectors, the font feature vector appears as $\Phi = [\mathbf{a} \ldots \mathbf{z} \ \mathbf{A} \ldots \mathbf{Z} \ 0 \ldots \ 9]^T$. The dimensionality of the font feature vector adds up then to $d = K \times m$, where $K$ is the number of characters and $m$ is the number of features per character.

4 CLUSTERING OF FONTS

We investigate in this section the distance functions of character feature vectors and comparative evaluation of the clustering methods.

4.1 Distance Functions

Let $c_{pk}(i,j)$ represent the bitmap at coordinates $(i, j)$ of the $k$th character in the $p$th font style. The weighted Hamming distance between any two character images $k$ created in fonts $p$ and $q$ respectively is calculated as follows:

$$H(c_{pk}, c_{qk}) = \sum_{i=1}^{32} W(i,j) |c_{pk}(i,j) \oplus c_{qk}(i,j)|$$

where the weights $W(i,j)$ from 1 to 4 are assigned proportional to the number of neighbors (weight 1 also for isolated pixels). An $F \times F$ proximity matrix can be constituted for each character, averaged, $F$ being the number of fonts considered.

Averaging font-to-font proximity matrices over each of the $K$ characters, we obtain the general $F \times F$ "font proximity" matrix, where the $(p, q)$ cell has:

$$D(font_p, font_q) = \sum_{k=1}^{K} P(char_k)H(char_k, font_p, char_k, font_q)$$

$$= \sum_{k=1}^{K} P(char_k)H(c_{pk}, c_{qk})$$

$P$ being the character occurrence probability. Finally the distance between clusters can be defined in terms of interfont distances. We have considered the mean distance, the centroid distance, the nearest-neighbor distance and finally the mean Hamming distance. For example the mean distance was defined as:

$$D_{\text{mean}}(cluster_p, cluster_q) = \frac{1}{N_p N_q} \sum_{i=1}^{N_p} \sum_{j=1}^{N_q} D(font_{pi}, font_{jq})$$

For features other than bitmaps, (Fourier descriptors, DCT coefficients and eigenfeatures), the above font and font cluster distance definitions remain valid if Euclidean instead of Hamming distance is used, i.e.:

$$H(c_{pk}, c_{qk}) = \sum_{i=1}^{m} (c_{pk}(i) - c_{qk}(i))^2$$

if $c_{pk}$ denotes the $k$th character feature vector in font $p$.

4.2 Cluster Centroids

Using interfont distances the centroidal font can be defined as the font having the smallest sum of feature distances to all other fonts. We compared three centroid definitions, that is font feature centroids, juxtaposition of individual character centroids, and averaging-thresholding of character bitmaps. The first two methods proved to be equally good choices.

4.3 Clustering Algorithms

We have experimented with three clustering algorithms, namely, the linkage algorithm, the mean-square distance based algorithm and finally with the Ward’s algorithm.

In the linkage algorithm the font pairs having minimum average distance in the font proximity matrix are merged and new clusters are formed iteratively. However as clusters become sizeable, the single linkage method suffers from false merges and the disparateness in the groups increases.

In the second method the mean square Hamming distance between clusters defined as:

$$D(cluster_p, cluster_q) = \frac{N_p N_q}{N_p + N_q} \left( \frac{1}{K} \sum_{k=1}^{K} \left[ H(c_{pk}, c_{qk}) \right] \right)^{1/2}$$

where $N_p$ and $N_q$ denote the respective font cluster populations. An illustrative output of this method is shown in Fig. 1. Thirdly a sample result of the Ward’s algorithm is shown in Fig. 2. Recall that Ward’s algorithm aims to minimize the within-cluster variation and this action is equivalent to maximizing the between-cluster variation [3]. The organization and clear separation of the 10 font clusters using Ward’s algorithm are shown in Fig. 3.

![Figure 1 Clustering with average mean square Hamming distance using bitmap features; the cluster centroids are shown on the right.](image-url)
Figure 2: Clustering with Ward’s algorithm using 20 principal components of the bitmap features

Figure 3: Clustering of the 65 fonts using Ward’s algorithm projected on the plane of the first two principal components.

4.4 Clustering Performance

An extensive set of experiments have been carried out using different clustering techniques and different feature sets. [6]. As an indicator of cluster performance the ratio of the mean of within-cluster variances to the between-cluster variances has been used [8]. The goal of these experiments was to establish on the one hand the more favorable features and clustering algorithms, on the other hand to cross-check whether fonts clustered with respect to one specific feature resulted in good scores with respect to all other features as well.

It was concluded that as a method the Ward’s clustering algorithm was the most satisfactory and among features the eigenfeatures (KL features) result in the most compact representation.

5 OCR PERFORMANCE AND FONT CLASSIFICATION

5.1 Inter– and intra-cluster OCR Performance

We investigated the extent of the OCR performance variation across different fonts. The performance differentials were measured using recognition scores with template matching. The OCR performance when text was created in one font, but recognized with templates from another font was calculated by simulation. The recognition percentages are given in Table 1. One can observe that in the off – diagonal terms there are unacceptably high performance drops due to font mismatches. This OCR cross-performance table proves the importance of font recognition.

<table>
<thead>
<tr>
<th>Template font #1</th>
<th>Template font #2</th>
<th>Template font #3</th>
<th>Template font #4</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text font #1</td>
<td>89.6</td>
<td>55.6</td>
<td>71.8</td>
</tr>
<tr>
<td>Text font #2</td>
<td>64.4</td>
<td>98.6</td>
<td>20.6</td>
</tr>
<tr>
<td>Text font #3</td>
<td>69.0</td>
<td>39.2</td>
<td>100.0</td>
</tr>
<tr>
<td>Text font #4</td>
<td>74.4</td>
<td>76.2</td>
<td>83.4</td>
</tr>
<tr>
<td>Text font #5</td>
<td>93.2</td>
<td>52.0</td>
<td>94.8</td>
</tr>
<tr>
<td>Text font #6</td>
<td>64.6</td>
<td>88.0</td>
<td>74.8</td>
</tr>
<tr>
<td>Text font #7</td>
<td>84.6</td>
<td>64.2</td>
<td>64.4</td>
</tr>
<tr>
<td>Text font #8</td>
<td>64.4</td>
<td>71.6</td>
<td>37.2</td>
</tr>
<tr>
<td>Text font #9</td>
<td>59.4</td>
<td>34.6</td>
<td>58.8</td>
</tr>
<tr>
<td>Text font #10</td>
<td>40.6</td>
<td>70.4</td>
<td>41.0</td>
</tr>
</tbody>
</table>

Similarly we considered the intra-cluster variation of the OCR performance. In other words the correct recognition scores when different fonts in the same class were used for both the text and template.. As expected the performance differential across fonts in the same cluster is much less. When we tested whether the centroid yields also the highest OCR score, we discovered that the centroid template and the font that would result in the best average character were not always the same.

Table 2: Intracluster OCR recognition rates for a cluster consisting of fonts numbered 6, 51, 10, 21, 30, 27, 47, 57, 63 (the last 4 not shown). The centroid font is 6, which is also used as a template.

<table>
<thead>
<tr>
<th></th>
<th>6</th>
<th>51</th>
<th>10</th>
<th>21</th>
<th>30</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>97.0</td>
<td>97.0</td>
<td>78.2</td>
<td>86.6</td>
<td>86.6</td>
</tr>
<tr>
<td>51</td>
<td>96.2</td>
<td>97.0</td>
<td>86.0</td>
<td>83.4</td>
<td>83.4</td>
</tr>
<tr>
<td>10</td>
<td>86.0</td>
<td>78.6</td>
<td>97.0</td>
<td>97.0</td>
<td>97.0</td>
</tr>
<tr>
<td>21</td>
<td>82.0</td>
<td>74.4</td>
<td>91.6</td>
<td>92.4</td>
<td>92.4</td>
</tr>
<tr>
<td>30</td>
<td>86.0</td>
<td>78.8</td>
<td>97.0</td>
<td>93.4</td>
<td>97.0</td>
</tr>
<tr>
<td>27</td>
<td>96.6</td>
<td>96.8</td>
<td>86.2</td>
<td>77.0</td>
<td>84.0</td>
</tr>
<tr>
<td>47</td>
<td>87.0</td>
<td>88.2</td>
<td>69.4</td>
<td>80.0</td>
<td>69.6</td>
</tr>
<tr>
<td>55</td>
<td>95.6</td>
<td>94.8</td>
<td>85.4</td>
<td>87.8</td>
<td>86.4</td>
</tr>
<tr>
<td>63</td>
<td>95.6</td>
<td>95.6</td>
<td>85.6</td>
<td>78.0</td>
<td>86.4</td>
</tr>
<tr>
<td>Avg</td>
<td>91.3</td>
<td>89.0</td>
<td>86.2</td>
<td>86.2</td>
<td>87.0</td>
</tr>
</tbody>
</table>

5.2 OCR Based Clustering of Fonts

Motivated by the observation that the cluster centroid did not always yield the template resulting in the best recognition score, OCR performance-based clustering was carried out. To this purpose we estimated a 65 x 65 correct recognition score matrix using a corpus of texts. The resulting 65 x 65 matrix is not symmetric, therefore the mutual OCR rate between fonts $p$ and $q$ (that is text with font $p$ and template with $q$ and viceversa) is selected as the minimum of the two scores when font $p$ is matched with font $q$ and viceversa. In this clustering scheme one selects a seed, preferably the highest scoring font and joins to its group all such fonts whose performance differential
remains below a threshold \( \varepsilon \). Obviously the more one lowers the threshold the more clusters one obtains but with less populations and eventually all fonts become a separate clusters.

6 DETECTING THE FONT CLASS

To determine to which font class a text region belongs, we ran the template matching procedure. We selected characters randomly from text and we incremented the score of the winning font by one, each time that font resulted in the highest match value. When the score of the highest scoring font exceeded that of its nearest competitor by a certain amount, we declared that font as the document font. In our work the score gap between the winner and its nearest competitor was chosen as 10 and the correct font was usually identified within 10–15 steps.

7 CONCLUSION

Clustering tendency of fonts has been investigated. Among the tested features, (DCT features, Fourier descriptors, bitmaps and eigenfeatures) eigenfeatures result in the most parsimonious and compact representation. Clustering of fonts have been effected with two different goals:

i. Clustering based on “shape similarity” using font features is needed when the problem domain is the recognition of logical document structures, document reproduction, document indexing and information retrieval.

ii. Clustering based on “differential error performance” using character templates when the problem domain is merged character segmentation or enhancement of the recognition rate of the OCR system.

It has been shown that for all the fonts six to eight clusters prove adequate; on the other hand for the more limited set of commonly used in daily newspapers three to four clusters were sufficient. The font recognition algorithm can be used both to recognize font groups or individual fonts. In the latter case it is preferable to have a hierarchical approach, that is, to first recognize the font cluster and subsequently the specific font within that cluster.

Appendix


References


