

The Snippet Statistics of Font Recognition

Jakub Lidke Christian Thureau
Christian Bauckhage

Fraunhofer IAIS, Schloss Birlinghoven, 53754 Sankt Augustin, Germany
<http://www.iais.fraunhofer.de/>

Abstract

This paper considers the topic of automatic font recognition. The task is to recognize a specific font from a text snippet. Unlike previous contributions, we evaluate, how the frequencies of certain letters or words influence automatic recognition systems. The evaluation provides estimates on the general feasibility of font recognition under various changing conditions. Results on a data-set containing 747 different fonts shows that precision can vary between 16% and 94%, dependent on (i) which letters are provided, (ii) how many letters are provided, and (iii) which language is used – as these factors considerably influence the text snippet statistics. As a second contribution, we introduce a novel bag-of-features based approach to font recognition.

1. Introduction

The task of Optical Font Recognition (OFR) is to recognize a specific font based on a supplied text-snippet or scan of a document. The advent of the Internet gave rise to a variety of important applications for OFR. Font owners have to detect copyright violations from visual inspection (usually the font names are not included in the documents), and end-users try to find fonts solely based on visual similarity, e.g. to replace costly protected fonts by free ones. Despite its practical relevance, only a few approaches were introduced so far.

Font recognition faces two main challenges: First, the number of different classes (fonts) tends to be very large. Second, a supplied text sample used for querying a font is usually unconstrained, i.e. the number of letters/words is unknown as is the content or the language.

In this contribution, we explore the statistics of these unconstrained text-snippets. We assume that certain letters are more discriminable than others (see also Fig. 1). If this assumption holds, OFR results must vary for (i)

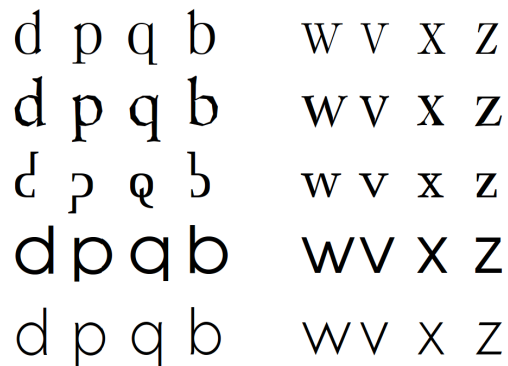


Figure 1. 8 letters from 5 different fonts.

different combinations of letters (which letters are provided?), (ii) different numbers of letters (how many letters are provided?), and (iii) different languages (how do letter/word priors influence the recognition?). Our main goal is to give estimates on the general feasibility of font recognition under various conditions. Further, we provide statistically plausible recommendations – detailed evaluation on which letters, languages, and words are most appropriate for OFR – that should underpin all future developments in font recognition systems.

Furthermore, we present an efficient approach to font recognition. The approach meets certain crucial requirements that we found essential for the task at hand. It builds on the popular idea of histograms of visual words [4]. In fact, we express each font as a histogram over a number of binary image patches. From a new text-snippet, we now extract another histogram of binary patches. Recognition can be achieved by computing the minimum distance to the database histograms. Using normalized patch histograms, we can easily compare different lengths of text snippets without violating the approach (see [7], where this was applied for recognizing human activity sequences of varying length).

1.1. Related Work

Feature-wise, OFR has been applied using local or global feature sets [1, 2, 3, 5, 6, 9]. Local features are usually more flexible but they often require well working segmentation. Global approaches, on the other hand, are more forgiving with respect to noise and wrong segmentations but they usually require a large training base. Feature classification, in a subsequent step, is usually achieved using common classifiers, e.g. Bayes-classifiers or Support Vector Machines.

In [9] Gabor filters are applied to text blocks for recognition of Chinese and English typefaces. Another global approach is presented by [1], where high order statistical moments (3rd and 4th order) and principal-component analysis are used for font recognition. In [3] font styles (bold, italic, normal) are recognized using textons[8], basic texture elements that are linearly combined. [2] present another basis reconstruction approach by employing non negative matrix factorization (NMF) applied to font shapes. In [6] stroke templates are automatically generated from individual characters belonging to a specific font, and subsequently stored in a database. Given novel input stroke data, a simple Bayes classifier decides about the most likely font. In [10] vertical and horizontal projection profiles from text lines are used for typeface and style classification with a Bayes classifier. In [5], the content of the text is used as an additional cue. While the approach itself is rather simple, employing an eigenvector decomposition of shapes, recognition results on 2763 different fonts are sufficient. In contrast to some of the aforementioned methods, we aim at unconstrained font-recognition, i.e. we focus solely on visual recognition and do not use additional cues. Moreover, we use a relatively large database of 747 different fonts, whereas many approaches use less than 20 or even 10 different fonts.

2. Patch-based Font Recognition

In this paper, we propose an approach that is based on comparison of histograms of local shape descriptors. Thus, we mainly follow the popular histogram of visual words approach [4]. To best of our knowledge, this is the first time that such an approach is applied to font recognition.

Given a shape, in our case a character or word, we first sample a number of locations where image patches are extracted. This can be achieved by e.g. applying k-means clustering to all black pixels in a binary image, yielding a set of cluster centers $C = \{c_1, \dots, c_n\}, c_i \in \mathbb{R}^2$. A dense distribution of cluster centers showed to be important.

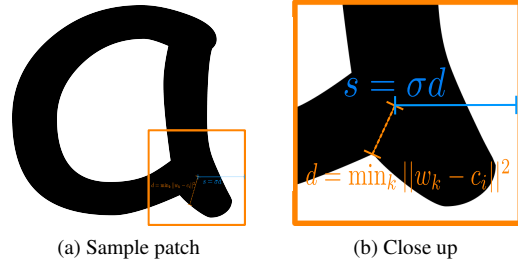


Figure 2. Centered on black pixels, binary image patches are extracted. Patch sizes are determined automatically and lead to variations in the amount of context a patch encodes.

Centered around each cluster center c_i we extract a quadratic image patch p_i . The sizes $S = \{s_1, \dots, s_n\}, s_i \in \mathbb{R}$ of the extracted binary patches are automatically determined. We measure the s_i using the distance between c_i and the nearest white pixel and set $s_i = \sigma \min_k \|w_k - c_i\|^2$, $w_k, c_i \in \mathbb{R}^2$, where the w_k are the white pixels of the shape image and σ denotes a constant scaling factor, see Figure 2 for an illustration. All extracted binary patches are finally scaled to a uniform size, e.g. of 5×5 or 10×10 pixels, using a simple image resizing procedure.

The proposed binary image patch representation is useful for representing shapes for three different reasons. First, it is invariant to scale since the scale is determined strictly based on local conditions and the patches are afterwards scaled to uniform size. Second, patches extracted from the inner part of a shape and from the border region of a shape contain different amounts of local context, Third, there is no need for specific x-height or capital-height segmentation (even character segmentation is not crucial, as the patches seldom extend beyond the boundaries of a specific character).

In order to get a single descriptor for a complete font or a single text-snippet, we compute histograms of certain prototypical shape patches. The shape prototypes (or codebook vectors) are computed during training by applying k-means clustering to all resulting patches (the later presented experiments used a total of 5000 prototypes). Given a number of patches, a histogram $H(\mathbf{p})$ for all codebook vectors $[c_1, \dots, c_n]$ that were clustered during the training phase is defined as a mapping

$$H(\mathbf{p}): \mathbf{p} \rightarrow \mathbb{N}^+ \cup 0, H(\mathbf{p})_{c_i} := \text{occ}(c_i, \mathbf{p}), \quad (1)$$

where $i = 1, \dots, n$, and n denotes the total number of codebook vectors, and $\text{occ}(c_i, \mathbf{p})$ denotes the number

of occurrences of \mathbf{c}_i in \mathbf{p} . Finally, we normalize the histograms s.t. each font is now represented by a distribution over a set of codebook shapes. Thus, we can now compare histograms independent of the number of letters or words or patches (we convert them to a distribution anyway). A normalized patch histogram $\phi(\mathbf{p})$ is then defined as $\phi(\mathbf{p}) = H(\mathbf{p}) / |H(\mathbf{p})|$.

Given the normalized histogram $\phi(\mathbf{p})$ for a number of patches \mathbf{p} , the classification is performed according to the minimum distance D to a set of normalized training histograms $\phi(\mathbf{f}_i)$ of font \mathbf{f}_i

$$j = \underset{i}{\operatorname{argmin}} D(\phi(\mathbf{p}), \phi(\mathbf{f}_i)), \quad (2)$$

For distance computation, we tried the following distance measures: L1, L2, Kullback-Leibler Divergence and Bhattacharyya Distance. As also other results indicate [7], KL-divergence and Bhattacharyya usually gave the best results for comparing sequences of different length (or different number of patches in our case). The later presented experiments mainly use the Bhattacharyya distance ($D = -\ln \sum_k \sqrt{\mathbf{p}_k \mathbf{f}_k}$, where k marks the k -th histogram bin)

3. Evaluation

For evaluation, we conducted a set of 747 different true-type fonts. The data-set contains all standard fonts (Arial, Times New Roman) but also some more exotic fonts. We considered a classification of a new text snippet to be correct, if the *true* font is within the first five closest histograms. However, it should be noted that the first returned fonts are usually rather similar, therefore restricting font classification to a best in five selection seems acceptable.

The font histograms $\mathbf{f}_i, i = 1, \dots, n$ are constructed from all lower case letters a-z. Thus, we skipped e.g. german Umlaute or other language specific characters, but also capital letters. The reason for this is simple; most letters encountered in the wild are lower case, and not all true-type fonts provide uppercase letters, numbers, or special character. Therefore, adding other characters would unbalance the training data.

We evaluated various experimental setups. For each experiment, we randomly generated large amounts of synthetic test data (in total more than 500000 images), randomly varying the combination of letters, and the total number of letters. It should be noted that the results also (besides letter number and word length) depend on well-working segmentation. To avoid random artifacts, we averaged over several experimental trials.

The first experiment evaluated OFR for only a single used character (can we tell the font based on a single

| | | | | |
|-------------|-------|-------|-------|-------|
| Word length | 2 | 3 | 4 | 5 |
| Recognition | 60.8% | 68.9% | 73.3% | 77.2% |
| Word length | 6 | 7 | 8 | 9 |
| Recognition | 78.7% | 81.8% | 82.0% | 82.4% |

(a) Unsegmented random characters

| | | | | |
|-------------|-------|-------|-------|-------|
| Word length | 2 | 3 | 4 | 5 |
| Recognition | 76.0% | 86.1% | 90.6% | 91.9% |
| Word length | 6 | 7 | 8 | 9 |
| Recognition | 92.7% | 93.1% | 93.4% | 93.7% |

(b) Segmented random characters

| | | | | |
|-------------|-------|-------|-------|-------|
| Word length | 2 | 3 | 4 | 5 |
| Recognition | 89.6% | 91.0% | 92.6% | 93.4% |
| Tag length | 6 | 7 | 8 | 9 |
| Recognition | 93.7% | 93.7% | 93.9% | 94.3% |

(c) Optimal case character selection

| | | | | |
|-------------|-------|-------|-------|-------|
| Word length | 2 | 3 | 4 | 5 |
| Recognition | 33.2% | 58.3% | 75.1% | 84.0% |
| Tag length | 6 | 7 | 8 | 9 |
| Recognition | 91.1% | 92.0% | 92.4% | 92.4% |

(d) Worst case character selection

Table 1. Average recognition results

supplied character?). Table 2 summarizes the results. Interestingly, we found a huge difference for individual characters. The letters 'd,p,q' (> 70%) seem to be best suited for font recognition, whereas 's,v,x,z' (< 30%) gave the worst results.

Further experiments considered different word lengths. Table 1 summarizes the results. Table 1a shows the results for using unsegmented randomly generating words of different length. It can be seen that segmentation is in fact an important aspect of font recognition, although it does not lower the results dramatically. While an increased number of letters leads to better recognition rates, we usually did not exceed 85% on average. Table 1b repeated the first experiment for segmented characters. The precision is overall significantly higher, however, a further increase in word length only slightly improved the results. Table 1c and Table 1d show results for the best possible and the worst possible combination of characters respectively. For more than 6 characters we ended up with a precision above 90% in both cases.

Finally, we evaluated the influence of different languages to font recognition by generating random words (one per trial) in varying fonts, picked according to the

| | | | | | | | |
|---|-----|---|-----|---|-----|---|-----|
| d | 74% | a | 56% | k | 48% | w | 30% |
| p | 72% | f | 55% | c | 46% | s | 26% |
| q | 70% | n | 54% | e | 42% | v | 25% |
| b | 69% | m | 52% | y | 41% | x | 20% |
| r | 60% | h | 50% | j | 39% | z | 16% |
| g | 59% | u | 49% | l | 35% | | |
| o | 57% | t | 48% | i | 33% | | |

Table 2. Single letter recognition results.

relative word frequency of the 100 most popular words in that language (here only for English and German). Based on these generated words, we estimated 87.99% as an average recognition rate for German words, English came up to 86.50%. However, it should be noted that the 100 most frequently used English words are on average shorter than the Germans. Additionally, we give an estimate for OFR dependent on the derived letter statistics combined with the known letter frequency in a given language. Results for the five most frequent letters are summarized in Table 3. Here, English gave the best results, followed by Spanish, Italian, German, and French. Although with respect to letter frequency western European languages are rather similar, we could still observe differences in recognition.

To summarize the most important results; in the optimal case, using the most discriminable characters (d,p,q), three characters can already lead to a precision of 91% on 747 different fonts. However, in the worst case (the letters v,x,z) the precision reduces to only 58%. On average (random selection), three distinct characters lead to a precision of 87%. Often only 3-5 characters are sufficient, using more than 5 characters often leads to rates above 90%. Thus, assuming a sufficiently large text snippet, font recognition can be done reliably with a very high best in five accuracy, i.e. the correct font is within the list of the five best matching fonts. It should be noted that the presented results depend to a certain extent on the introduced approach and used features. However, given that the results are based on rigorous statistical evaluation, we believe that this also holds for different methods of unconstrained font recognition.

4. Conclusions

This paper presented an evaluation of the statistics of text snippets used in automatic font recognition. Besides, we introduced a novel approach to font recognition that fulfilled certain requirements of the evaluation and also shows a high precision for automatic font recognition.

| English | German | French | Italian | Spanish |
|-------------|---------|---------|---------|---------|
| 5.3 → e | 7.2 → e | 6.4 → e | 6.5 → a | 7.0 → a |
| 4.5 → a | 5.3 → n | 4.4 → a | 5.6 → o | 5.7 → e |
| 4.3 → t | 4.2 → r | 4.0 → r | 4.9 → e | 4.9 → o |
| 4.3 → o | 3.9 → a | 4.0 → n | 3.8 → r | 4.4 → d |
| 3.7 → n | 3.8 → d | 3.6 → t | 3.8 → i | 4.1 → r |
| : | : | : | : | : |
| : | : | : | : | : |
| 1.54 | 1.42 | 1.42 | 1.42 | 1.46 |

Table 3. Language specific font recognition estimates.

Extensive experiments revealed novel insights into the feasibility of font recognition in general. In fact, under certain conditions font recognition seems to be an ill posed problem as there basically is a lack of information when only a very few letters are supplied. Also, when simply the *wrong* letters are supplied, we can not expect convenient recognition. As we could show, with an increasing number of different letters, the task becomes more and more manageable, ending up in recognition precision higher than 94% for 747 different fonts.

References

- [1] C. Avilés-Cruz, R. Rangel-Kuoppa, M. Reyes-Ayala, A. Andrade-Gonzalez, and R. Escarela-Perez. High-order statistical texture analysis: font recognition applied. *Pattern Recogn. Lett.*, 26(2), 2005.
- [2] C.-W. Lee and K. Jung. Nmf-based approach to font classification of printed english alphabets for document image understanding. In *Proc. MDAI*, 2005.
- [3] A. Schreyer, P. Suda, and G. Maderlechner. A formal approach to textons and its application to font style detection. In *Proc. Workshop on Document Analysis Systems*, 1999.
- [4] J. Sivic, B. C. Russell, A. A. Efros, A. Zisserman, and W. T. Freeman. Discovering object categories in image collections. In *Proc. ICCV*, 2005.
- [5] M. Solli and R. Lenz. Fyfont: Find-your-font in large font databases. In *Proc. SCIA*, 2007.
- [6] H.-M. Sun. Multi-linguistic optical font recognition using stroke templates. In *Proc. ICPR*, 2006.
- [7] C. Thureau and V. Hlaváč. Pose primitive based human action recognition in videos or still images. In *Proc. CVPR*, 2008.
- [8] S.-C. Zhu, C.-E. Guo, Y. Wang, and Z. Xu. What are textons? *Int. J. Comput. Vision*, 62(1-2), 2005.
- [9] Y. Zhu, T. Tan, and Y. Wang. Font recognition based on global texture analysis. *IEEE Pattern Analysis and Machine Intelligence*, 23(10), 2001.
- [10] A. Zramdini and R. Ingold. Optical font recognition using typographical features. *IEEE Pattern Analysis and Machine Intelligence*, 20(8), Aug 1998.