Using Typography in Document Image Analysis

Frédéric Bapst and Rolf Ingold

IIUF, University of Fribourg
Ch. Musée 3, CH-1700 Fribourg, Switzerland

Abstract. Even if font usage plays an important role in Document Image Analysis (DIA), recognition systems generally take the concept of font management in a weaker sense than in the production cycle. With the point of view of the document recognition community, we show how typographic information (characters bitmap, metrics, etc.) can improve existing analysis methods. After a brief survey of font recognition issues, we present the advantages of a font software support in the design of recognition systems. Concrete algorithms are proposed in the subtopics of a posteriori font recognition, monofont Optical Character Recognition (OCR), and word segmentation. The reported experiments and results indicate that there are still substantial benefits to expect from the design of typography-aware analyzers.

1 Introduction

One of the goals in Document Image Analysis (DIA) is the automatic recognition of printed text. Depending on the application, the central subtask of Optical Character Recognition (OCR) needs to be completed with various levels of layout analysis, for instance block and line segmentation, read-order chaining, font or justification mode identification, and so on. The font attributes represent rich formatting parameters, because they give some hints about the document logical structure (which denotes the author's point of view).

Although document recognition has been concerned with fonts for a long time, typographers would certainly agree that only superficial aspects of font management are generally involved. We claim that a deeper study of typeface design [2, 12–14] brings an original lighting on recognition problems. The goal of this paper is to draw DIA and typography closer together. More specifically, we want to show how existing recognition techniques can be improved when the font concept is considered as seriously as in document production. As well as revealing an interesting example of know-how integration inside our *CIDRE* [1] project[2] [4], this topic leads to applicable algorithms for our community of DIA engineers.

The remainder of this paper is organized as follows: Section 2 describes to what extent font management is considered an important component in DIA research; after this brief survey, we advocate the use of a relevant font management toolkit in DIA. Section 3 presents some recognition algorithms that take advantage of typographical knowledge. Our concrete experiments and the related results are exposed in Sect. 4. Finally we draw some conclusions.

[1] For Cooperative & Interactive Document Reverse Engineering.

[2] Supported by the Swiss National Fund for Scientific Research, code 21-42'355.94.

2 From Weak To Strong Font Recognition

Many document recognition systems have to deal with fonts, especially during the analysis of logical structures. Font recognition is often taken in the weak sense, meaning that the goal is only to discover font changes in the document. In the strong sense, font recognition consists in describing the font package that was used during the formatting. This section evokes previous studies and explains that distinction.

2.1 Font Discrimination

We talk about *font discrimination* when a recognition system is able to detect the number of fonts used in a page, and to partition the entities (e.g. paragraphs or words) following this criterion. In this approach, the goal is to group together entities of the same font, not to find the exact specification of those fonts.

Sometimes the analyzer goes one step further and tries to qualify the results with informal font attributes, using tags for italic, bold, fixed-pitch or serifs. But these are only some hints and the exact typeface is still unknown. In other cases, it is during the learning phase that the user gives a symbolic name to each font.

Several approaches to font discrimination were proposed in research projects: Shi et al. [21] focus on the recognition of font families; Morris [17] performs spectral analysis; Baird et al. [3] propose an adaptive 100-font classifier; Duffy et al. [8] use prototype extraction and heuristic edition rules; Zramdini [23] uses global features and Bayesian classification (see Sect. 3.1). Some commercial OCR software [9, 18] also offer a limited form of font recognition.

For the DIA community, font discrimination is often sufficient. In dedicated applications for instance, the document class often fixes the possible fonts set; thus the meaning of fonts can be given prior to the recognition task itself.

2.2 Font Understanding

Font understanding supposes a formal definition of fonts, based on existing typography standards. The goal is to find what precise typeface was used at typesetting time. This is a strong requirement, because it brings the recognition results finally closer to the production phase. On the other hand, DIA may try to exploit the typographers' knowledge.

Some interesting ideas have been proposed in this context: Kopec [15] estimates font metrics (e.g. left and right *bearings*) from images. Hobby et al. [10] compute a high-resolution character outline from a set of low-resolution scanned samples.

Instead of actually recomputing the fonts, we claim that the recognition methods could at least make use of the same formats and tools as in typography. The effect is to give access to the huge collection of fonts that are available for the production. Once the recognition system can access such a knowledge base, it is likely that the fonts of the documents being recognized will match an existing font. Although the "classical" typefaces (like Times or Courier) exist in a great number of declinations, the precise character shapes should not change too much between designers or foundries.

2.3 Software Support

Strong font recognition forces the developers of recognition algorithms to represent the fonts in a rich data structure. The underlying software environment must offer facilities to install or load a font, ask for its character set and metrics, generate or display formatted text, and so on. Fortunately, this can be simplified through the reuse of existing toolkits.

In the *CIDRE* project, we designed a software platform that integrates recognition algorithms and typography management. The goal is to offer a relevant programming support, so that the document recognition analyzers can exploit all the information encapsulated in font definitions. Concretely, we chose to manipulate the fonts via the X11 system, with the following advantages:

- it offers a C interface (Xlib) to access all information about the fonts, e.g. the metrics (see Fig. 1);
- scalable fonts are allowed, thus avoiding multiple sizes in the font database;
- it accepts the installation of PostScript fonts, one of the most important standards;
- font aliases are supported, in order to name fonts in a consistent way;
- it is already installed in virtually all Unix systems.

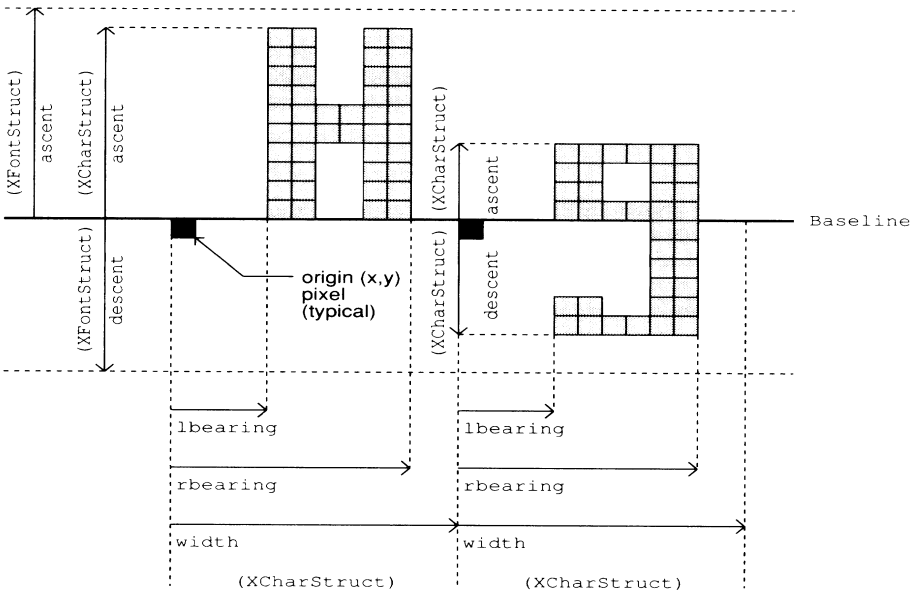


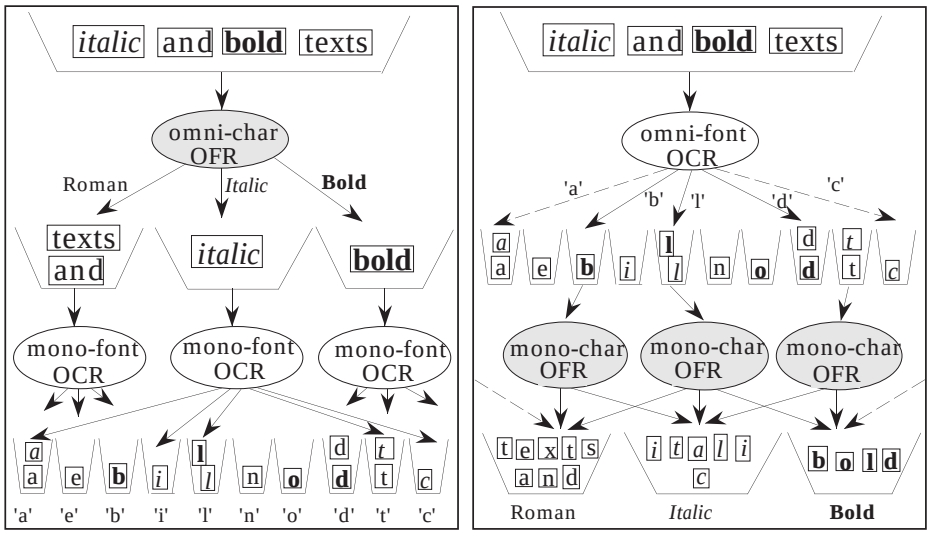
Fig. 1. Font metrics in Xlib (taken from Xlib Programming Manual).

3 Some Analysis Methods Based on Typography

This section discusses a few examples that show how document recognition systems could take advantage of typographical information. The analysis methods themselves are certainly not new, but we find interesting to present some variants that exploit a thorough font management support.

3.1 A Priori Font Recognition

Figure 2 reminds of the two major approaches to font recognition [23]. The *a priori* methods do not require OCR results and generally have a low time complexity. On the other hand, their accuracy drops for the analysis of short pieces of text, in which case an *a posteriori* algorithm is more appropriate (see Sect 3.2).



a priori font recognition **a posteriori** font recognition

Fig. 2. OCR and OFR (Optical Font Recognition) (taken from [23]).

In his PhD thesis [23], Zramdini developed the a priori font recognition system $ApOFIS$, that allows the identification of the typeface, weight, slope and size from a given image of text. It uses a font model base generated by learning from training sets. Font models are computed from features vectors, which represent global typographical properties extracted from text strings. $ApOFIS$ also supports incremental learning.

Instead of using scanned samples for the learning phase, our software platform is now able to generate synthetic images for any installed font, and to compute its corresponding font model. Thus the analyzer gains in generality, but at the price of a small decrease in accuracy when the degradation in document images is too extensive [22].

3.2 A Posteriori Font Recognition

When the a priori approach is not applicable, the alternative is to look locally at the precise character shapes. In our proposition, we make the following hypotheses: word segmentation results with their correct ASCII interpretation are provided, and each word was formatted with a single font. It would not be reasonable to require exact character bounding boxes, because it proved to be unreliable, especially for slanted typefaces.

This problem can be addressed using simple pattern matching. Recognizing the font of one word consists roughly in formatting the same text using each font in the database, and computing a dissimilarity measure from the original image. The ideal word width is computed using font metrics; the difference with the effective word box is then distributed among the character frontiers, to obtain a synthetic word of the same width. A trivial similarity score counts the black pixels after an XOR operation on both images. The algorithm can be used to estimate the closest font in a given font set, or to validate the ASCII string when the font is known.

Our brute-force method can be refined in two complementary ways. First, we try a few pixels shifts on the coordinates of each characters to optimize the possible matching. Second, we use ternary instead of binary masks [11], to reduce the effects of spatial sampling [16]; characters skeletons and envelopes are pre-computed using morphological operations (erosion and dilation). The time complexity is reduced by filtering out unrealistic comparisons, e.g. according to the word height.

3.3 Monofont OCR

Whereas the trend in OCR development (especially in commercial products) has grown toward general omnifont technology, we still feel a high interest in monofont devices. Sennhauser [20] shows how OCR can be improved using typographical constraints. Now that font recognition has reached a stable state, the expected robustness and accuracy of monofont analyzers make them a serious alternative. By the way, there are some digital typefaces explicitly designed for the OCR process (the fonts OCR-A and OCR-B, but also Bigelow's recent Lucida-Sans-OCR [5]).

The basic idea used for a posteriori font recognition can be adapted to develop a *generic monofont OCR*, i.e. which takes as parameter the name of an installed font. The OCR engine will take advantage of typographic knowledge, namely the ideal bitmap shape of all characters and the expected geometric relations between any sequence of them.

Let's sketch a naive algorithm. Suppose you know the bounding box of a word, its underlying baseline[3], and its font. Then the text can be guessed iteratively by finding the best match for the first character prefix; starting from the left, the position of the next character is estimated using font metrics, until the right bound is reached. At each position, the method tries to superimpose every printable characters and evaluates a matching score with the original image.

Some problems should not be underestimated [11]. As in Sect. 3.2, spatial sampling or local shifts may have significant effects. The matching function should allow for the

[3] The baseline can also be estimated as the bottom coordinate of the word box, maybe with an offset of the font descent.

possible overlapping of consecutive character bounding boxes. As suggested in Fig. 3, the white mask should follow the frontiers of the character shape instead of filling the whole rectangle. But the matching function must not systematically drop the question mark (?) for the dot (.), or the double quote (") for the apostrophe ('). The accuracy may also be increased using backtracking, i.e. looking for an optimum match for 2- or 3-grams instead of single characters. It is also possible to rank the characters by their a priori frequencies.

Fig. 3. Skeleton and envelope masks for monofont OCR.

In spite of these difficulties, such a monofont OCR offers four considerable advantages: first, it is robust to connected components, whereas many other OCR have difficulties with touching or split characters. Second, the recognition results will give the exact position of the individual character. Third, the method is not disturbed by *kerning* effects. Fourth and most important, a naive version can be developed in a few hours, which leaves plenty of time to perform accuracy tests on many fonts or to try some refinements. It is then a good way of designing a sound OCR package, with expected performances close to commercial softwares.

3.4 Word Segmentation

The segmentation into words is not a completely solved problem in DIA. Most techniques to detect space characters analyze the horizontal spacing inside the text lines, using e.g. run-length smoothing algorithms (RLSA) or projection profiles, but in a context-insensitive way. Figure 4 shows why these methods cannot achieve 100% accuracy: for lots of fonts, especially italic ones, horizontal spacing inside a word may be greater than between words, depending on the characters involved. We stated that even with well-known fonts and non-exotic text, neither RLSA nor projection profiles can detect the space character without mistakes.

That's why the knowledge of the ASCII interpretation is decisive for the recognition of word separators. A first family of methods for a posteriori word segmentation makes use of lexical knowledge to reject invalid frontiers or propose valid ones. This idea is interesting but not general, because it requires that the language of accepted words is known in advance.

Here we propose a second *a posteriori* approach, strictly based on font metrics. The idea is once again very simple: we suppose that the bounding boxes of each characters

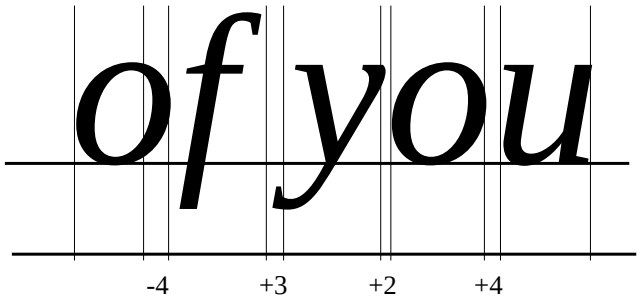


Fig. 4. Exact spacing computed from font metrics (for `tm-r-i-10`, at 300 dpi).

(or at least of word substrings) and its ASCII interpretation are provided. Then, using the distances of left and right character bearings, we can compute their ideal horizontal spacing with or without a space separator. The decision is finally based on the difference between theoretical and effective spacings. Note that the inter-word spacing is an elastic measure according to the justification mode, but the font metrics will give an approximation for the minimal gap. On the other hand, the inter-character metrics are supposed to be only determined by the font[4], possibly with the help of *kerning tables*.

3.5 Application-Specific Recognition

The previous subsections implicitly focused on a poor variety of printed documents. An immediate generalization concerns non-Latin alphabets (Arabic [1], Chinese [7], etc.), where the OCR task is quite more challenging, notably because of the influence of the context. But among the multitude of existing fonts, some were also dedicated to convey very specialized kinds of notations. Here is a non-exhaustive list:

- barcodes;
- musical scores;
- mathematical formulas;
- astrologic and astronomic symbols;
- drawings (e.g. IBM-OEM line drawing characters);
- icons.

Each of these fields comes with its own recognition applications. The use of typography will definitely not solve all of them. But we are convinced that today's font handling facilities can be a precious knowledge source, and that it is worth revising existing analysis techniques. When the objects to recognize can be matched with a typeface, the output alphabet is implicitly fixed, which precludes possible ambiguities. Even for occidental texts, most OCR encode some characters (like §, ©, fi, —, indices$_i$ and exponents^i) in a tricky way. A related advantage of using fonts in output formats is

[4] There are a few exceptions, especially in newspaper typesetting.

that a graphical user interface can present the recognition results in their most natural appearance. In general, the re-structured document automatically gets the status of a production version.

4 Experiments and Results

4.1 Testbed and Data Set

Our experiments focused on a set of 174 well-known fonts. This represents eight families (see Table 1), combined with two weights (regular/bold), two slopes (roman/italic), and six sizes (8, 9, 10, 11, 12, 14). As current DIA databases [19] are not yet indexed with detailed font attributes, we had to design our own data set. It contains one document sample for each font, representing a unique English text, formatted in two-columns. The documents were generated with LaTeX, printed on A4 paper, and scanned at 300 dpi. Finally we ground-truthed the images at the word level, in a semi-automatic process.

Table 1. Typeface set for the reported experiments.

Seriffed	Sans-serif	Typewriter	Script
lb:Lucida-Bright	ag:Avant-Garde	cr:Courier	zc:Zapf-Chancery
nc:New-Century	hv:Helvetica		
pl:Palatino			
tm:Times			

These testing conditions seem fair, in the sense that our recognition tools do not use the same font packages as for the production: we yet found a correspondence with the fonts installed in our X11 system, but they are certainly not an exact copy of those used in LaTeX.

On the other hand, our experiments are intended to give only a rough approximation of the practical reliability; the goal of this paper is to sketch some techniques based on typography, not to claim their absolute superiority.

4.2 Font Recognition

The performance of a priori font recognition using $ApOFIS$ is detailed in [22, 23]. In suitable conditions of learning, the classifier performs font recognition at the line level with 95.8% accuracy. The detection of the font weight and slope is even more accurate with rates up to 99.7%, while typeface and size are recognized with an average rate of 96%. For our testing scenario, the training images are synthetic whereas the recognition samples are scanned; moreover, the font packages are different. For 10pt fonts, this reduced the median accuracy down to about 60%, with an important offset between the best and the worst fonts.

To assess the a posteriori font recognition algorithm proposed in Sect. 3.2, we tried to identify the font from every single word of the document samples, among the whole font set. In fact, this is a very unfavourable scenario: in typical applications for instance, font homogeneity constraints could bring a decisive improvement through the processing of a sequence of consecutive words: if we err on one word with probability p (p small), then the error rate on n words tends quickly to 0 when n grows.

Table 2 presents the recognition rates for 10pt fonts. The results show that the method is as accurate as $ApOFIS$, having regards to their respective requirements. For half of the fonts, the accuracy is over 90%. As expected, the confusions only occur between very similar fonts, and half of the misrecognized words are up to three characters long. A detailed look at the worst cases revealed that there were important differences between the production and recognition typefaces: for instance, Courier-Bold characters have much smaller descenders in LaTeX than its X11 counterpart, which leads to several confusions between 9pt and 10pt when the words contain descending characters.

Table 2. Accuracy of the a posteriori font recognizer.

Font	Accuracy	Font	Accuracy	Font	Accuracy
ag-r-r-10	0.98	nc-r-r-10	0.76	lb-r-r-10	0.62
ag-r-i-10	1.00	nc-r-i-10	0.96	lb-r-i-10	0.83
ag-b-r-10	0.89	nc-b-r-10	0.89	lb-b-r-10	0.75
ag-b-i-10	0.82	nc-b-i-10	0.93	lb-b-i-10	0.85
cr-r-r-10	0.94	pl-r-r-10	0.93	zc-r-r-10	0.97
cr-r-i-10	0.89	pl-r-i-10	0.96		
cr-b-r-10	0.44	pl-b-r-10	0.49		
cr-b-i-10	0.40	pl-b-i-10	0.78		
hv-r-r-10	0.99	tm-r-r-10	0.81	*-*-*-10	0.83
hv-r-i-10	0.98	tm-r-i-10	0.73		
hv-b-r-10	0.92	tm-b-r-10	0.93		
hv-b-i-10	0.91	tm-b-i-10	0.94		

4.3 OCR

A simple version of a monofont OCR based on X11 font management tools was tested on our scanned samples. Table 3 reports the results for 8pt fonts. We must admit that the recognition rates are quite low, and we are currently refining the naive algorithm on several points.

On the same data set, ScanWorX achieves a recognition rate of over 99%, except for a few fonts: ag-r-i (98.51%), pl-r-i (95.8%), tm-r-i (93.1%), and zc-r-r (83.9%). Note that the accuracy drops when ScanWorX is parameterized for another language, and even much more when the analysis is submitted on a single word image. This means that ScanWorX makes an intensive use of lexical constraints, as well as statistical character distribution. In general, commercial products tend to privilege the

Table 3. Character accuracy of our monofont OCR.

Font	Accuracy	Font	Accuracy	Font	Accuracy
ag-r-r-8	0.952	nc-r-r-8	0.921	lb-r-r-8	0.926
ag-r-i-8	0.988	nc-r-i-8	0.928	lb-r-i-8	0.954
ag-b-r-8	0.984	nc-b-r-8	0.924	lb-b-r-8	0.924
ag-b-i-8	0.944	nc-b-i-8	0.995	lb-b-i-8	0.910
cr-r-r-8	0.691	pl-r-r-8	0.946	zc-r-r-8	0.886
cr-r-i-8	0.698	pl-r-i-8	0.924		
cr-b-r-8	0.630	pl-b-r-8	0.966		
cr-b-i-8	0.616	pl-b-i-8	0.963		
hv-r-r-8	0.972	tm-r-r-8	0.932	*-*-*-8	0.900
hv-r-i-8	0.964	tm-r-i-8	0.936		
hv-b-r-8	0.876	tm-b-r-8	0.922		
hv-b-i-8	0.950	tm-b-i-8	0.956		

most frequent usage; for special applications, the performance may not be as good as expected.

4.4 Segmentation

The superiority of our a posteriori word segmentation tool over naive thresholding is obvious, because it stems from the font design itself (see Sect. 3.4). It is more interesting to draw a comparison with a representative commercial OCR: for instance, ScanWorX avoids lots of segmentation errors using lexical information. As Table 4 shows with a focus on 8pt fonts, many mistakes are still generated with this commercial software.

On the whole set of 174 fonts, our post-segmentation method generates remarkably few errors, as stated in Table 5. A detailed look at those mistakes revealed that for Palatino-Italic, LaTeX uses a much smaller left bearing for the character *f* than the X11 font specification.

5 Conclusion

This paper discussed several possibilities of exploiting typography in the field of DIA. Our contribution ranges from theoretical to practical aspects; more specifically, we:

- discussed the state of the art in font recognition, and advocated its highest form: font understanding;
- designed a software platform that integrates a thorough font management in document recognition components;
- proposed concrete algorithms and discussed their expected advantages;
- conducted several experiments with some of these techniques to assess their potential benefits over existing analyzers.

Table 4. Number of misrecognized inter-word spaces using ScanWorX.

Font	Errors	Font	Errors	Font	Errors
ag-r-r-8	0	nc-r-r-8	0	lb-r-r-8	0
ag-r-i-8	8	nc-r-i-8	3	lb-r-i-8	9
ag-b-r-8	0	nc-b-r-8	0	lb-b-r-8	0
ag-b-i-8	8	nc-b-i-8	16	lb-b-i-8	4
cr-r-r-8	1	pl-r-r-8	0	zc-r-r-8	147
cr-r-i-8	5	pl-r-i-8	23		
cr-b-r-8	1	pl-b-r-8	0		
cr-b-i-8	1	pl-b-i-8	7		
hv-r-r-8	0	tm-r-r-8	1		
hv-r-i-8	2	tm-r-i-8	42		
hv-b-r-8	0	tm-b-r-8	0		
hv-b-i-8	3	tm-b-i-8	7		

Table 5. Misrecognized inter-word spaces using our post-segmentation tool.

Font	Errors	Font	Errors	Font	Errors
pl-r-i-8	2	pl-r-i-9	4	pl-r-i-10	1
pl-r-i-11	2			all others	0

The experiments showed that the analyzers based on typographical information can rival traditional algorithms in accuracy, but without relying on complex heuristics. The methods proposed for font recognition, OCR, or word segmentation are indeed very simple and easy to program using X11; yet they proved to solve some non-trivial recognition situations. In general, we advocate simple analyzers, because they are more easy to integrate in the design of an adaptive, interactive and cooperative system.

In the *CIDRE* project, the problematic of DIA is taken in the broader sense of document *reverse engineering*. This means that we do not want to break the inherent dependencies from the production cycle. Here we presented typography as one possible bridge between both processes, but we are also studying other implications for the modeling of physical and logical structures [6]. As a general trend, interdisciplinarity will remain a powerful inspiration engine in document recognition problems.

References

1. Adnan Amin. Arabic character recognition. In H. Bunke and P. S P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, chapter 15, pages 397–420. World Scientific, 1997.
2. J. Andre and R. D. Hersch. Teaching digital typography. *Electronic Publishing: Origination, Dissemination and Design*, 5(2):79–90, 6 1992.
3. H. S. Baird and G. Nagy. A self-correcting 100-font classifier. In *SPIE-The international Society for Optical Engeneering, Document Recognition*, pages 106–115, San Jose, California, February 1994.

4. Frédéric Bapst, Rolf Brugger, and Rolf Ingold. Towards an interactive document recognition system. Internal working paper 95-09, IIUF-Université de Fribourg, March 1995.
5. Charles Bigelow. The evolution of markings and meanings in typography, 1997. Keynote speech at ICDAR'97 (see also http://www.YandY.com).
6. Rolf Brugger, Frédéric Bapst, and Rolf Ingold. A DTD extension for document structure recognition. In *EP'98*, St-Malo, France, 1998.
7. X. Q. Ding. Machine printed chinese character recognition. In H. Bunke and P. S P. Wang, editors, *Handbook of Character Recognition and Document Image Analysis*, chapter 11, pages 305–330. World Scientific, 1997.
8. Laurence Duffy, Frank Lebourgeois, and Hubert Emptoz. Logical structure analysis by typographic characteristics extraction. In *ICIAP'97: International Conference on Image Analysis and Processing*, number 1311 in Lecture Notes in Computer Science, pages 639–646. Springer, September 1997.
9. ExperVision, Inc., 3590 North First Street, San Jose, CA 95134-9815. *TypeReader Professionnal*, February 1995. Release 1.0 for MacOS.
10. J. D. Hobby and H. S. Baird. Degraded character image restoration. In *SDAIR'96: Fifth Symposium on Document Analysis and Information Retrieval*, pages 233–246, Las Vegas, Nevada, April 1996.
11. Rolf Ingold. *Une nouvelle approche de la lecture optique intégrant la reconnaissance des structures de documents*. PhD thesis, EPFL, Lausanne, 1988. n. 777.
12. Peter Karow. *Typeface Statistics*. URW Verlag, Hambourg, 1993.
13. Peter Karow. *Digital Typefaces*. URW Verlag, Hambourg, 1994.
14. Peter Karow. *Font Technology*. URW Verlag, Hambourg, 1994.
15. G. E. Kopec. Least-square font metric estimation from images. *IEEE Transactions on Image Processing*, 2(4):510–519, October 1993.
16. D. Lopresti, J. Zhou, G. Nagy, and P. Sarkar. Spatial sampling effects in OCR. In *ICDAR'95: Third International Conference on Document Analysis and Recognition*, pages 309–314, Montreal, Canada, August 1995.
17. R. A. Morris. Classification of digital typefaces using spectral signatures. *Pattern Recognition*, 25(8):869–876, 1992.
18. Beth Paddock and Timothy J. Platt. *ScanWorX API, Programmer's Guide*. Xerox Imaging Systems, Inc., 9 Centennial Drive, Peabody, Massachusetts 01960, 1992.
19. R. P. Rogers, I. T. Phillips, and R. M. Haralick. Semiautomatic production of highly accurate word bounding box ground truth. In *Document Analysis Systems (DAS'96)*, pages 375–386, 1996.
20. R. Sennhauser. Improving the recognition accuracy of text recognition systems using typographical constraints. In *RIDT'94: Third International Conference on Raster Imaging and Digital Typography*, pages 273–282, Darmstadt, Germany, April 1994.
21. Hogwei Shi and Theo Pavlidis. Font recognition and contextual processing for more accurate text recognition. In *ICDAR'97*, pages 39–44, Ulm-Germany, August 1997.
22. A. Zramdini and R. Ingold. A Study of Document Image Degradation Effects on Font Recognition. In *ICDAR'95: Third International Conference on Document Analysis and Recognition*, pages 740–743, Montreal, Canada, August 1995.
23. Abdelwahab Zramdini. *Study of optical font recognition based on global typographical features*. PhD thesis, IIUF-Université de Fribourg, 1995. n. 1106.