CATEGORIZING DOCUMENT IMAGES INTO SCRIPT AND LANGUAGE CLASSES

C. Y. Suen, S. Bergler, N. Nobile, B. Waked, C. P. Nadal, and A. Bloch

CENPARMI, Concordia University, 1455 de Maisonneuve Blvd West, Montréal, Québec H3G 1M8, Canada
Email contact: bergler@cs.concordia.ca

ABSTRACT

In order to properly archive and index large numbers of international documents, several challenging processing steps must be completed even before optical character recognition (OCR) can be applied. We present a system that preclassifies documents for further processing and OCR. The system operates in four phases: preprocessing (including skew detection, segmentation, and noise removal), script (Latin, Arabic, Ideographic, or Cyrillic) classification, shape coding, and language classification for seven European languages.

We present a set of statistical techniques, based fundamentally on connected component analysis and horizontal projections, for the first two phases. Even with little training, the system predicts the correct script category in 91% of the cases, when tested on real-life documents of varying kinds, diverse formats and qualities from many sources. The third and fourth phase are based on expert systems approaches. Language identification combines several heuristics based on a statistical analysis of our training corpus. It currently has a 95% success rate on real-life documents of moderate quality. We will discuss the techniques and their combination and the process of improving performance.

1 Introduction

The need to archive paper documents electronically and to process information contained in them has led to the development of sophisticated commercial optical character recognition (OCR) software for different scripts, which is usually also optimized for the language used. In a multilingual environment, neither script nor language can be assumed prior to processing the documents. Our efforts are intended to provide a preprocessing stage that transforms the image of the scanned page into several low-level representations which can be exploited to determine the script (Arabic, Ideographic, or Latin), and in the case of Latin script also whether the document is written in one of seven languages: English, French, German, Spanish, Portuguese, Italian, or Russian[1]. This categorization can then be used to select the proper Latin or Arabic OCR modules for any given document.

This paper describes a system that performs both tasks, but focuses mainly on the language classification task. The script classification will be sketched only briefly; a more extensive account can be found in [1].

Before the script classification task can actually begin, several preprocessing steps must be performed[2]. Thus the scanned image is segmented into text regions and other

[1]Cyrillic script has characteristics very similar to the Latin script, and for ease of processing and describing the systems we pretend that it is a Latin script.

[2]see Hochberg *et al.* [2], Ratner [3], and Lee *et al.* [4] for approaches that do not require preprocessing.

regions, then the text is deskewed and segmented into paragraphs, lines, words, and characters. During this process, noise is filtered out and some distracting text segments (such as very oversized headlines) are removed as well.

For robustness in the face of poor quality data from a variety of sources, we use very low-level statistics throughout the process. One core characteristic is the structure of *bounding boxes*, that is the smallest rectangle that encloses a set of touching pixels (*connected component*). Bounding boxes are used for all preprocessing steps, including deskewing.

The script classification module SCRIPTID relies also on the notion of *horizontal projections*, that is a histogram plotting the number of black pixels for each scan line (see Wood *et al.* [5] for a different use of projection profiles). It turns out that the three script types we aim to distinguish have quite different characteristics in the height distribution of the bounding boxes, in density[3], and in the horizontal projections of a text line. Based mainly on these three aspects we obtain a 91% recognition rate.

Subsequent language identification, in the case of Latin script, is based on the notion of character shapes. Because the final system works from scanned documents without the benefit of OCR, we cannot rely on the shape of the characters, but at best on shape characteristics of the characters, which can be inferred from our low-level data. Based on Sibun and Spitz [7] we developed a new set of character shape codes that map characters with similar characteristics into the same class. Shape codes are determined based on the bounding boxes compiled in the preprocessing phase, taking ascenders, descenders, special diacritics (umlaut, dots, accents), and the size of the character into account.

The language identification module IDDOC computes selected statistics for a given test document, and compares them to the averages extracted in the training phase for all languages. A ranking of each language hypothesis for the given document is produced first, based on the statistics, then an expert system prioritizes certain heuristics or combinations of heuristics with respect to the others. A language hypothesis results from this process, unless a pre-set certainty threshold cannot be met.

The heuristics for IDDOC are based on the occurrence of unique word shapes, character n-grams, and word n-grams. Together these characteristics perform very well. Character n-grams are well-known tools (see Sibun and Spitz [7] and Lee *et al.* [4] for the use of word unigrams on document images; Cavnar and Trenkle [8] for the use of n-grams in determining newsgroup membership; and Gokcay and Gokcay [9] for use of n-gram statistics on text); we have introduced word n-grams and our expert system approach for improved performance. An earlier version of IDDOC has been presented in [10]. The accuracy of IDDOC in combination with SCRIPTID on scanned real-world documents of moderately good quality is 95%.

2 Preprocessing

Figure 1 presents a typical scanned input image. Preprocessing consists of a connected component analysis, filtering of large areas and noise, deskewing, and segmentation.

Creating Bounding Boxes

The bounding box is used in every phase of our SCRIPTID system, starting from deskewing and segmentation to script identification. The bounding box is simply a

[3] We use density information only as an auxiliary function and in particular do not compute reading order distribution of optical density as reported in Spitz [6].

Missa do
5.º Aniversário

Por ocasião do 5.º aniversário de sua morte, informamos os amigos que no próximo dia 27-09-97, celebrar-se-á missa por sus intenção a Celina de Lurdes. Pelas 11h30, a vós que dignarem a comparecer a tal acto, os nossos agradecimentos, marido, filhas, genros e netos.

Descanse em Paz querida.

Jovens fugitivas viviam num pinhal

Três raparigas adolescentes que fugiram de casa há um mês, em Coimbra, foram encontradas ontem de madrugada, no pinhal onde viveram desde a fuga, por uma patrulha da BT da GNR. As fugitivas, de 13, 14 e 17 anos, residem no bairro da Rosa, na zona norte da cidade de Coimbra, e despareceram de casa no dia 12 de Agosto.

O pinhal onde foram encontradas, às 5 horas e 30 da madrugada, por uma patrulha da BT da GNR de Coimbra, situa-se numa zona isolada da povoação de Ademia, a cerca de quatro quilómetros das residências. A primeira adolescente, de 13 anos, foi descoberta na berma da estrada da Ademia, o que suscitou algumas interrogações, dada a sua pouca idade e a hora, pois era de madrugada.

A rapariga pôs-se em fuga, juntando-se às outras duas, vindo a ser interceptadas pelos militares da BT. As adolescentes confessaram que haviam fugido de casa no dia 12 de Agosto e desde então dormiam no pinhal e alimentavam-se de frutos que colhiam nas propriedades agrícolas das redondezas.

As três raparigas encontravam-se bem de saúde, apenas a de 14 anos parecia "um pouco debilitada". Transportadas para o quartel da corporação até à chegada dos pais, as adolescentes "devoraram todo o pequeno-almoço". A fuga, participada à PSP de Coimbra no dia 12, pelos pais, foi justificada com "problemas familiares" que se escusaram a especificar.

Durante a sua estada no pinhal, deitavam-se no chão sem qualquer aconchego, pois apenas dispunham da roupa que vestiam quando saíram de casa. As duas raparigas mais novas são irmãs e a de 17 anos é amiga de ambas.

Figure 1: Image of a scanned Portuguese document

rectangular box which circumscribes the connected components of black pixels; each box can be represented by two coordinate points, the upper-left and the lower-right corners.

Large Area Filtering

Bounding boxes with more than five times the average area for bounding boxes for a document are removed, as well as bounding boxes with a disproportionate height to width ratio (vertical or horizontal lines). These components are unlikely to contain text, and can be ignored for both script and language identification.

Noise Filtering Bounding boxes with dimensions smaller than the text font stroke width (most frequently occurring black run) are considered noise, and are removed.

Deskewing the Document

Using a Hough transform (see Duda and Hart [11], Le *et al.* [12]) we deskew to an angle of accuracy of 0.5 degrees, limited to a range of ± 25 degrees.

Segmentation

The page is segmented into paragraphs, lines, and words. Following Ha *et al.* [13], we rely mainly on white space deduced from horizontal and vertical projections.

3 Script Classification

SCRIPTID classifies the text into 3 types of script: Arabic, Ideographic, and Latin. The identification process is based on the bounding box size distribution and horizontal projections.

First, we extract information from the height distribution of the bounding boxes before the preprocessing phase. Figure 2 shows three typical height distributions for an Arabic, an Ideographic, and a Latin document. The position of the two largest peaks in relation to each other and to the minor peaks determines the relative position of the most frequent bounding box size. Since Latin script has predominantly characters that span from the baseline to the x-height (the size of the letter x), its major peaks lie above the average box size, the same as for Ideographic scripts, whereas Arabic scripts usually have at least one major peak below the average bounding box size.

A second distinguishing feature is character density. Ideographic scripts include very

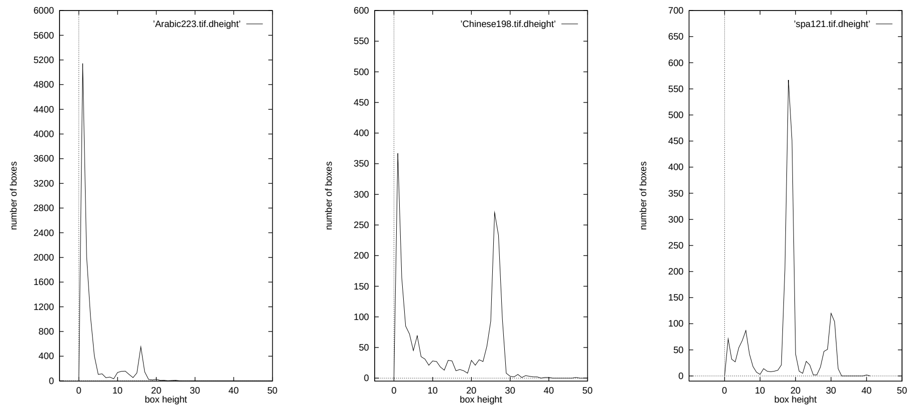

Figure 2: Bounding box height distribution for Arabic, Ideographic, and Latin scripts.

dense characters, whereas Arabic scripts are characterized by low density.

The third and most prominent feature we use to distinguish the three script classes is the horizontal projection of a line. As Figure 3 illustrates, Arabic script is characterized by one prominent peak, Latin scripts by two prominent peaks (at the base line and the x-height), and ideographic scripts have differing patterns, each characterized by more than two peaks.

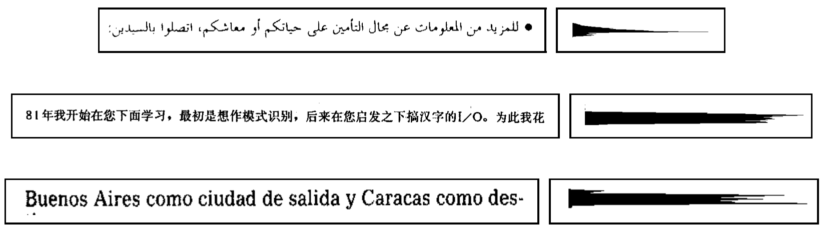

Figure 3: Horizontal projections for Arabic, Ideographic, and Latin scripts.

Script classification is thus a procedure that uses very basic document features, which are largely insensitive to bad data quality (a large percentage of touching lines from smearing or extremely noisy data poses a problem, as does the presence of a large percentage of horizontally broken characters). In fact, testing on 220 documents supplied by a client, ranging in quality from good to extremely poor (photocopies of faxes scanned at 200 DPI) results in 91% accuracy, 7% error, and 2% rejection. A more detailed discussion of the procedures can be found in [1].

Lee *et al.* [4] report an interesting approach based on position of concavities, bounding box height and position, and density. They distinguish Ideographic (Chinese and Japanese) from Latin (English, French, German, Italian, and Spanish) scripts with 98% accuracy on degraded and unoriented pages. While the emphasis of our approach is very different, we feel that our results are comparable given that we include more Asian languages in our testing for Ideographic script and our input includes four scripts, Arabic and Cyrillic in addition to Ideographic and Latin.

4 Shape Coding

Shape coding is a process which converts the bounding boxes contained in each extracted word into one of 13 character shape codes. The following steps describe the conversion process:

1. Determine baseline and x-height line.
2. For each word, correct the position of the baseline and the x-height as required by residual skew or bad alignment of characters to the baseline.
3. Convert the bounding boxes to character shape codes, using the heuristics outlined in Figure 4.

	Simplified calculation	**Samples**
x	A single small box.	a o x
A	A single large box on the baseline.	A T k l
i	A box with one very small box above it.	i é Ñ
ü	A box with two very small boxes above it.	ü ä ë Ö
g	A single large box that descends the baseline.	g p q
j	A single large box which descends the baseline with a very small box above	j ¡ ¿
.	A single small box near the baseline.	. _
!	A larger box above a very small box with a height ratio of at least 3:1.	! ?
:	Two small boxes above each other; the bottom box is near the baseline.	: ;
-	A long box located between the x-height line and the base line.	= ×
,	A small box that crosses the baseline.	,
’	A small box above the x-height line.	” ‘ 1
?	Any unrecognized character.	

Figure 4: Computation of character shape codes

This procedure is very sensitive to skew angle, noise, and document quality. The number of broken or touching characters in a document affects the shape and position of bounding boxes. But character shape coding is an accurate indicator for the underlying language only when bounding boxes correspond to individual characters. Figure 5 shows the character shape coding for the last two paragraphs of the document shown in Figure 1; this segment has 16 shape coding errors and 2 codes “unknown” (“?”). The system is trained on real world data, which compensates somewhat for these errors (assuming that the training set has an average number of broken or touching characters).

To evaluate the performance of this function, we tested it on different test sets. The first consisted of documents from the WorldWideWeb which were electronically converted from ASCII to image, thus avoiding any skew, noise, or quality problems. The second test set consisted of 298 scanned images. Because proofreading the shape coding output must be done manually, we have evaluated the shape coding procedure indirectly by comparing the language classification results. The test sets performed at 96% and 95%, which is strong indication that our preprocessing phase is very accurate, and that training on scanned documents adequately enhances the system.

5 Language Classification

The language classification module is called IDDOC. It exploits statistics gathered from a training set to match the statistical profile of a test document. It discriminates

```
Ax AxAx xxgxxigxx xxxxxAxxxxx-xx Axx Ax xxiAx, xgxxxx x Ax
xx xxxx gxxxix xx gxxxx AxAiAiAxAx’. AxxxxgxAxAxx gxxx x gxxx-
AxA Ax xxgxxxgix xAi i xAxgxAx Axx gxix, xx xAxAxxxxxAxx Axxx-
xxxxx AxAx x gxgxxxx-xAxxgx’. A Axgx, gxAixigxAx A AAA Ax
AxixAxx xx Aix AA, gxAxx gxix. Axi jxxAiAixxAx xxx gxxAAxxxx Axxi-
Aixxxx’ gxx xx xxxxxxxxx x xxgxxiAixxx.
AxxxxAx x xxx xxAxAx xx gixAxA, Axi?xxxx-xx xx xAix xxx gxxA-
gxxx xxxxxAxgx, gxix xgxxx?x Aixgxxxixx Ax xxxgx gxx xxxAixx
gxxxAx xxxxxx Ax xxxx. Ax Axxx xxgxxigxx xxix xxxxx xix ixxix x
x Ax AA xxx i xxigx Ax xxAxx.
```

Figure 5: Shape codes for a fragment of Figure 1.

between seven languages, English, French, German, Spanish, Portuguese, Italian, and Russian. The challenge in this set of languages was to distinguish between Spanish and Portuguese based on the statistics and in the absence of full character recognition.[4]

The languages have been chosen because commercial OCR software has specialized language modules for these languages plus a set of Scandinavian languages. We experimented with several of the Scandinavian languages at the beginning of our project, but found it very hard to compile enough documents for training and testing. Indeed, even at the Dutch embassy we were told that we could get flyers and brochures in Dutch, but that they did not have correspondence, business letters, or memos in Dutch. The same was true for colleagues we contacted in the appropriate countries. It is therefore very unlikely that correspondence in any of these Scandinavian languages will be sent to Canada, and we have excluded them from this system for the moment.

5.1 Training

The training of IDDOC begins with scanning documents into TIFF images. The scanned documents include magazine articles, books, faxes, and newspaper articles. These images are processed by SCRIPTID in order to obtain a character-coded file for each of the images. These files form the basis for the statistics we compile for word shape 1, 2, and 3-grams, character shape classes, character shape 2, 3, and 4-grams. The training set contains about 10500 words (roughly 20 documents) for each language. The training program stores all the frequencies of word and character n-grams for each character-coded file. All those word shapes or n-grams that contain any of the following character classes are filtered out: {. , : ! ?}. Shapes that begin or end with a {-} or a {’} or that contain two consecutive {-} or {’} are also omitted.

Language models for unique word shapes must be compiled semi-automatically. To create models for character classes, the thirteen character classes for a language are sorted in decreasing order of frequency. To create a model for any one of the remaining characteristics requires storing the top T n-grams, where T is the number of n-grams for the language containing the least number of distinct n-grams.

Adding a new language to our models requires computing the statistics for that lan-

[4] Russian is included in this set because Cyrillic cannot be reliably distinguished from Latin scripts with the procedures we use for SCRIPTID. While we have developed a special procedure to identify Russian at the time of character shape coding, we still include Russian in IDDOC to catch documents that are not correctly classified by that special procedure.

guage and to redo the unique word shape computation for all languages. Removing a language from our models just requires that we recompute the unique word shapes.

As has already been discussed in connection with the shape coding function, the ability to quickly retrain on a more appropriate training set is a major strength of this system, and adds to its portability.

5.2 Testing

IDDOC can be described in five steps: Loading the language models, reading the input, generating statistics for each heuristic, computing language probabilities for each heuristic, and running the expert system. Each is described in sequence below.

Loading the language models

The first step reads in the information obtained from the training session. Each heuristic (word shape 1, 2, and 3-grams, character shape classes, character shape 2, 3, and 4-grams) contains a ranked list of the most frequent occurrences for each of the seven languages.

A *shape percentage array* contains, for each language, the probability that each of the following shapes {x, A, i, g, j, ü, ', -} occurs in the training set.

Reading the input

This step reads in the first K words of character coded text from the test document. Longer texts give better results. Figure 6 shows the relationship between length of input and performance. Note that in an application there is no need to artificially shorten a document, as long as processing time is adequate. Our system takes roughly 120 seconds for 1000 words (see Section 6 for values for different cutoff points.)

Generating statistics for each heuristic

Compile statistics for the test document (word shape 1-grams, word shape 2-grams, word shape 3-grams, character shape classes, character shape 2-grams, character shape 3-grams, and character shape 4-grams).

Compute language probabilities for each heuristic

We compare the frequency of occurrence of instances of each heuristic in the document with the ranking of that instance for each language. For example, if the word shape 1-gram instance "AAx" occurred ten times in the document, and that same instance is ranked as the seventh most frequent word shape 1-gram for the English training set, then the *Word 1-gram* heuristic will add a value based on the frequency and rank values (ten and seven respectively) to the strength value $strength[Word\_1gram, English]$ cell. These values are normalized from zero to 100 to counteract the influence of long documents. The following formula is used to compute the values to be added to the heuristic strengths.

For each particular instance of a $Heuristic$, we compute its frequency in the test document, $Freq$, and its rank $Rank$ in the Language Model L. We increase the vector cell $strength[Heuristic, L]$ by the value H as follows

$$H = (MaxRank_{Heuristic} - Rank)^2 * Freq$$

where $MaxRank_{Heuristic}$ is the rank of the most frequent shape in the $Heuristic$ model. This is performed for all $Heuristic$ instances that occurred in the document, and for all languages.

Normalization of heuristic strength values

At this point, each heuristic will have seven strength values (one for each language). These values are nonnegative and have no upper limit[5]. We normalize these values from zero to 100 for display purposes, and to determine whether the difference between the first and second highest strengths is greater than a predetermined threshold (0.1%). If so, this heuristic will conclude that the document is written in the language with the highest strength. Otherwise, this heuristic will REJECT the document because it cannot come to a conclusion with sufficient confidence.

Calling the expert system

The expert system identifies the language of a document based on the normalized strength vector. The expert system is a multi-rule system containing eleven rules. Once a rule fires (succeeds), the expert system exits with its solution to the language identification problem. All results provide a confidence value between zero and 100. The following rule is an example from our rule base:

Rule 12: If Word 1-gram and Word 2-gram heuristics agree on the same language L, then return L with a confidence of $min(50 + max(SF\_1A, SF\_1B), 99.99) * RuleConfidence$ (%) where SF_1A is the probability for L obtained from the Word 1-gram heuristic, and SF_1B is the probability obtained from the Word 2-gram heuristic for the language L and $RuleConfidence = 0.75$.

6 Experimental Results

The different modules have been tested on different test sets throughout their development. We have a large database of text samples from different scripts; most of these are paragraphs copied from library holdings, which were used during development. The performance of SCRIPTID reported here is, however, on a client test set. It consists of 220 documents (66 Arabic, 91 Ideographic, and 63 Latin), which cover a wide range of style and quality (photocopies of faxes scanned at 200 DPI, for instance). SCRIPTID has 91% correct and 7% incorrect classifications; the remaining 2% are rejected.

IDDOC, too, has been tested on several test sets. The expert system approach was originally developed on ASCII data taken from the WorldWideWeb and shape coded-directly (see [10]). In order to assess performance of the shape coding function and the preprocessing steps, we used two in-house test sets. The first consists of 69 documents taken from the WorldWideWeb which are converted to PostScript images (thus there is neither skew nor deterioration). These images are processed by SCRIPTID before IDDOC begins language identification. The system correctly identifies the language for 96% of the documents; 4% are incorrect, with no rejections.

The second test set consists of 319 documents, which include scanned magazine articles, newspaper clippings, foreign correspondence, and brochures. For the second test set, the system showed a 95% correct identification rate, 5% are incorrectly identified, and there is one rejection. We have not been able to conclude a run of this latest system on client data, which has in the past proven to be of poorer quality than our in-house data (which were scanned at 300 DPI.)

We have also experimented with artificially limiting the number of words of each document. The performance degrades, as expected. Figure 6 shows, however, that the

[5]The values are bounded by the maximum value that the compiler can handle.

results are still viable for shorter documents. The results reported above are for a cutoff of 1000 words per document (usually documents are shorter than this, and are processed in their entirety). For a limit of 50 words per document, we still get over 75% correctness with 20% error (this corresponds to roughly 5 lines of text). For a limit of 20 and 10 words the error rate goes up to roughly one third and the recognition rate drops to 60% and 44% respectively. Thus one and two lines are not sufficient for reliable language identification with the current system. We are working on improving the accuracy for lower limits.

Words:	1000	500	100	80	50	20	10
Cor:	95.65%	94.20%	91.30%	89.86%	75.36%	59.42%	43.48%
Inc:	4.35%	5.80%	5.80%	4.35%	20.29%	30.43%	34.78%
Rej:	0.00%	0.00%	2.90%	5.80%	4.35%	10.14%	21.74%

Figure 6: Relationship between text length and accuracy of language classification.

Sample timing results are an average of 120 seconds for the 1000 word cutoff, 63 seconds for the 500 word cutoff, 14 seconds for the 100 word cutoff and 8 seconds for the 50 word cutoff. A cutoff of 2000 words results in 208 seconds runtime with the same accuracy obtained from the 1000 word cutoff.

7 Conclusion

We have presented a robust system that classifies scanned images into four script types (Arabic, Ideographic, Latin, and Cyrillic). Documents identified as Latin script are then shape coded and classified into seven European languages.

The system uses very basic statistics of the document images, mainly bounding box statistics and horizontal projections to derive the classifications. These statistics have proven to be robust. While most of the basic techniques we employ can be found in the literature, we have refined and complemented them, and combined them into a large, coherent system. We have then tested the system on several large test sets. Thus the reported system is a very good indication of the state of the art.

The language classification relies on a training phase which is not very time-consuming (a specialist can retrain the system in an afternoon), and mostly automatic. Thus the system is portable, and can be used and adapted easily.

We believe that this technology is not only useful on its own, but can be employed for other tasks as well. We will extend the number of classes distinguished (i.e. more scripts and languages), and also investigate how to improve results on sparse data (very short documents) and noisy data (very poor quality). Incorporation of OCR techniques will be another investigation to correct some errors introduced by incorrect bounding box determination.

References

[1] B. Waked, S. Bergler, S. Khoury, and C.Y. Suen. Skew detection, page segmentation, and script classification of document images. In *Proceedings of the 1998*

IEEE International Conference on Systems, Man, and Cybernetics, San Diego, CA, October 1998. forthcoming.

[2] J. Hochberg, P. Kelly, T. Thomas, and L. Kerns. Automatic script identification from document images using cluster-based templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):176–181, 1997.

[3] A.S. Ratner. Simultaneous identification of script and orientation. In *Proceedings of the Symposium on Document Image Understanding*, April-May 1997.

[4] D.-S. Lee, C.R. Nohl, and H.S. Baird. Language identification in complex, unoriented, and degraded document images. In *Proceedings for the International Association for Pattern Recognition Workshop on Document Analysis Systems, Malvern, PA*, pages 76–98, October 1996.

[5] S. Wood, X. Yao, K. Krishnamurthi, and L. Dang. Language identification for printed text independent of segmentation. In *Proceedings of the International Conference on Image Processing*, pages 428–431, 1995.

[6] A.L. Spitz. Determination of the script and language content of document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):235–245, 1997.

[7] P. Sibun and A.L. Spitz. Language determination: Natural language processing from scanned document images. In *Proceedings of the Fourth Conference on Applied Natural Language Processing, Stuttgart, Germany, 1994*, pages 15–21, 1994.

[8] W.B. Cavnar and J.M. Trenkle. N-gram based text categorization. In *Proceedings of the Third Annual Symposium on Document Analysis and Information Retrieval*, pages 161–169, April 1994.

[9] E. Gokcay and D. Gokcay. Combining statistics and heuristics for language identification. In *Proceedings of the Fourth Annual Symposium on Document Analysis and Information Retrieval*, pages 423–433, Las Vegas, April 1995.

[10] N. Nobile, S. Bergler, C.Y. Suen, and S. Khoury. Language identification of on-line documents using word shapes. In *Proceedings of the Fourth International Conference on Document Analysis and Recognition, Ulm, Germany, August 1997*, pages 258–262, 1997.

[11] R. O. Duda and P. E. Hart. Use of the Hough transform to detect lines and curves in pictures. *Communications of the Association for Computing Machinery*, 15:11–15, 1975.

[12] D.S. Le, G.R. Thoma, and H. Wechsler. Automated page orientation and skew angle detection for binary document images. *Pattern Recognition*, 27(10):1325–1344, 1994.

[13] J. Ha, I.T. Phillips, and R.M. Haralick. Document page decomposition by the bounding-box projection technique. In *Proceedings of the Third International Conference on Document Analysis and Recognition, Montréal, August 1995*, pages 1119–1122, 1995.