

## **INFORMATION TO USERS**

This manuscript has been reproduced from the microfilm master. UMI films the text directly from the original or copy submitted. Thus, some thesis and dissertation copies are in typewriter face, while others may be from any type of computer printer.

**The quality of this reproduction is dependent upon the quality of the copy submitted.** Broken or indistinct print, colored or poor quality illustrations and photographs, print bleedthrough, substandard margins, and improper alignment can adversely affect reproduction.

In the unlikely event that the author did not send UMI a complete manuscript and there are missing pages, these will be noted. Also, if unauthorized copyright material had to be removed, a note will indicate the deletion.

Oversize materials (e.g., maps, drawings, charts) are reproduced by sectioning the original, beginning at the upper left-hand corner and continuing from left to right in equal sections with small overlaps.

Photographs included in the original manuscript have been reproduced xerographically in this copy. Higher quality 6" x 9" black and white photographic prints are available for any photographs or illustrations appearing in this copy for an additional charge. Contact UMI directly to order.

ProQuest Information and Learning  
300 North Zeeb Road, Ann Arbor, MI 48106-1346 USA  
800-521-0600

**UMI<sup>®</sup>**



# PAGE SEGMENTATION AND IDENTIFICATION FOR DOCUMENT IMAGE ANALYSIS

BOULOS WAKED

A THESIS  
IN  
THE DEPARTMENT  
OF  
COMPUTER SCIENCE

PRESENTED IN PARTIAL FULFILLMENT OF THE REQUIREMENTS  
FOR THE DEGREE OF MASTER OF COMPUTER SCIENCE  
CONCORDIA UNIVERSITY  
MONTRÉAL, QUÉBEC, CANADA

SEPTEMBER 2001  
© BOULOS WAKED, 2001



**National Library  
of Canada**

**Acquisitions and  
Bibliographic Services**

**395 Wellington Street  
Ottawa ON K1A 0N4  
Canada**

**Bibliothèque nationale  
du Canada**

**Acquisitions et  
services bibliographiques**

**395, rue Wellington  
Ottawa ON K1A 0N4  
Canada**

*Your file Votre référence*

*Our file Notre référence*

The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.

0-612-64087-6

**Canada**

# Abstract

## Page Segmentation and Identification for Document Image Analysis

Boulos Waked

The main objective of this thesis is to develop a system to automatically segment and label a variety of real-life documents written in different languages. The main idea is to partition the whole document into different subimages and assign to each of them one of two labels: text or non-text (including graphics); and then identify the text as one of three categories. Roman, Ideographic, or Arabic script.

The whole process consists of several steps. For instance, to detect the skew angle of the document, we use the Hough transform and the most frequently occurring local maximum. Moreover, in order to segment the page into regions, we have developed a novel approach based on diagonal scanning and node-edge orientation. Then, text and graphic components are also isolated using the geometric configuration of the connected components. Next, the textual components are segmented into lines using the projection profile; and finally the script is classified into one of the three categories mentioned above using the bounding boxes and horizontal projection.

The system has been tested on 215 samples of diverse document types from many sources such as journal articles, magazines, newspapers, facsimiles, and office correspondence. These testing samples include low quality document images with different types of distortion: they also contain upside-down, skewed and low resolution images. The system classifies 93.5% of the script type correctly and 6.5% of these documents incorrectly.

# Acknowledgments

I acknowledge and thank many people for their help with this thesis.

First, and foremost, I would like to thank my supervisor, Dr. Ching Suen, for having helped me realize this project and for his guidance during the course of this work. He offered me a wonderful environment at Cenparmi, which is a remarkable research center to work in. I also thank him for the time and effort he put into reading many drafts of this thesis and for his extensive and valuable advice. It is his supervision and support that have made this work possible.

I am also grateful to my co-supervisor, Dr. Sabine Bergler, who provided me with valuable technical feedback and suggestions, as well as pointing out errors and carefully reviewing many versions of this thesis. Hopefully, I learned much from her advice. Also, I am deeply indebted to her for more than four years of scientific help and friendship.

Special thanks to Dr. Mohamed Cheriet, member of the Program Evaluation Committee, who assisted with the review of the final draft of this document. His constructive advice and valuable comments considerably improved the final version. This thesis is much better because of him.

Gratitude is expressed to Dr. Tony Kasvand, who served on the external review committee for this document.

My thanks go to Christine Nadal for collecting and building the document image database. Also I would like to thank Cenparmi staff and fellow graduate students for their friendship: Beverley, Yousef, Lihua, Andrea, Jianxiong, Guiling, Liqiang, Xiao Ping, Soo Hyung, Jun, Danny, Karim and Qizhi. Thanks to Mike, William and Stan for their effort in maintaining a high-quality computing environment.

Finally, thanks to my parents for their support and encouragement. Also, my gratitude and love go to Carmen for her effort and for always having something encouraging to say. Thank you all.

# Contents

<b>List of Figures</b>	<b>vii</b>
<b>List of Tables</b>	<b>xii</b>
<b>1 Introduction</b>	<b>1</b>
1.1 What is Page Segmentation . . . . .	4
1.2 What is Region Identification . . . . .	4
1.3 Main Applications . . . . .	5
1.4 Type and Quality of the Documents . . . . .	6
1.5 Some Assumptions . . . . .	7
1.6 Organization of this thesis . . . . .	8
<b>2 Preprocessing</b>	<b>9</b>
2.1 Introduction . . . . .	9
2.2 Bounding Boxes Generation . . . . .	9
2.3 Large Area and Noise Filtering . . . . .	13
2.3.1 Large Area Filtering . . . . .	13
2.3.2 Noise Filtering . . . . .	13
2.3.3 Advantage of large area and noise filtering . . . . .	13
2.4 Skew Angle Detection . . . . .	14
2.4.1 Our Skew Angle Algorithm . . . . .	17
2.5 Testing Data . . . . .	19
2.6 Comparative Analysis and Discussion . . . . .	22
<b>3 Page Segmentation</b>	<b>24</b>
3.1 Introduction . . . . .	24

3.2	The Technique . . . . .	28
3.2.1	Step 1: Extract Diagonal White Runs . . . . .	28
3.2.2	Step 2: Create Long Vertical Edges . . . . .	31
3.2.3	Step 3: Setting up Vertical White Stream . . . . .	33
3.2.4	Step 4: Setting up Horizontal White Stream . . . . .	35
3.2.5	Step 5: Constructing the Blocks . . . . .	37
3.3	Layout Structure . . . . .	37
3.4	Experiments . . . . .	39
3.5	Comparative Analysis . . . . .	41
<b>4</b>	<b>Block Identification</b>	<b>43</b>
4.1	Introduction . . . . .	43
4.2	Text/Graphics Identification . . . . .	43
4.2.1	Our Text/Graphics Algorithm . . . . .	45
4.2.2	Experimental Results . . . . .	50
4.3	Script Identification . . . . .	51
4.3.1	Our script Identification Algorithm . . . . .	52
4.3.2	Generating Decision Rules . . . . .	58
4.3.3	Experimental Results . . . . .	60
4.3.4	Comparative Results . . . . .	62
<b>5</b>	<b>Conclusion</b>	<b>63</b>
	<b>Bibliography</b>	<b>70</b>
<b>A</b>	<b>Examples of document Images</b>	<b>71</b>



# List of Figures

1	Overview of the main components of the system. . . . .	8
2	a) Three black runs on a scanline and b) an array of runs and their left and right coordinates. . . . .	10
3	The process of generating the bounding boxes from image scanlines. Notice how the boxes are growing in size at each scanline. For example, at step a), box height = 1 pixel, at step b), box height = 2 pixels, and at step c), box height = 3 pixels. . . . .	11
4	Image of a scanned document. . . . .	12
5	Bounding boxes of connected components of black pixels. . . . .	12
6	Collinear points $(x_i, y_i)$ with $i=1, \dots, N$ , are transformed into $N$ sinusoidal curves $\rho_i = x \cos \theta_i + y \sin \theta_i$ in the $(\rho, \theta)$ plane, which intersect in the point $(\rho, \theta)$ . . . . .	17
7	Left) original image, and right) the result of the algorithm: the skew angle is detected as equal to $4.5^\circ$ , and the image is rotated accordingly. . . . .	19
8	Example of a binary document image where the method fails to detect the skew correctly, because of the shaded area in the text frame. Note that the shaded area consists of very small connected components which are created during the digitization process. . . . .	21
9	Embedded image and text between two columns of text. . . . .	26
10	Embedded text in a frame between two columns of text. . . . .	26
11	Oversegmentation occurs between words in the title when white, long rectangle method is applied on the background of the image. . . . .	27
12	Oversegmentation has been reduced by druns (the diagonal boxes). Notice that the druns do not go through the gaps between the words. . . . .	28

13	a) A drun of connected white pixels between two black regions. b) The white pixels are merged to create the white run which is illustrated by the square box. . . . .	29
14	This figure shows the diagonal white runs (druns) and their square boxes. The boxes are drawn to illustrate the size of the druns. Thus, black pixels inside the boxes have not been considered. Note that one drun (see top right) has been removed because it is very large and it overlaps with the one on the next scanline. Also note that the distance between the scanlines should not be very small because vertical edges might go through gaps between words in the headline. . . . .	30
15	A diagonal array with each element representing a white run. . . . .	31
16	This figure simulates the creation of the vertical edges. See the vertical edge between the drun #7 on scanline (0,0) and drun #1 on scanline (0,1). . . . .	31
17	The black, long rectangle illustrates the vertical edge between two druns. . . . .	32
18	a) Vertical runs between two druns, b) A vertical edge is created by merging the runs in a). . . . .	33
19	a) Head to head, b) Head to tail, c) Head to body . . . . .	36
20	The horizontal stream #1 goes directly from vertical stream #1 to vertical stream #3. So it overpasses the vertical stream #2. . . . .	36
21	This figure displays the vertical and horizontal streams, the head and tail nodes inside the streams, and the block extracted. . . . .	37
22	Layout structure: an array of 3 vertical streams where each stream points to 2 nodes and each node is linked to a block. . . . .	38

23	Vertical, horizontal streams and nodes. The geometric structure of this figure can be divided into two relationships: vertical-to-vertical and horizontal-to-horizontal. For example, adjacent nodes on the same vertical stream correspond to adjacent blocks that are related vertically on the same text column, whereas adjacent nodes on two different vertical streams are related horizontally. Each list of nodes has blocks which is in the order from upper to lower node. This is the visual structure of the layout, but for logical structure, a number of generic rules should be extracted from the content of each block and thus we could transform the visual structure into the logical structure. . . . .	40
24	Example of a document image where the method fails to segment the text which is embedded in a circular way. . . . .	42
25	The result of grouping the bounding boxes into the segmented regions.	46
26	The horizontal projection profile of bounding boxes. . . . .	47
27	a) Bounding boxes, b) projection profile and c) array of black runs. .	48
28	Each block has been labeled as either text or graph. . . . .	49
29	A horizontal projection for a line in Roman script. . . . .	53
30	Horizontal projections of Roman, Ideographic and Arabic scripts . . .	54
31	Roman and Ideographic images and their height distributions. . . . .	56
32	Arabic image and its height distribution. . . . .	57
33	The bounding boxes of a Japanese text line. . . . .	57
34	The bounding boxes of Roman, Ideographic and Arabic lines of Figure 30.	57
35	Multifont Roman lines and their horizontal projections. Although the fonts vary, we are still able to see the parallel presence of two dominant peaks which correspond to the baseline and top minuscule line (or x-height line). The type fonts used are <i>italic Shape</i> , <i>Slanted Shape</i> , Sans Serif, Typewriter, SMALL CAPS SHAPE, <b>Boldface</b> and Roman. . . .	59
36	Multifont Arabic lines and their horizontal projections. The unique prominent peak can clearly be shown at the baseline. . . . .	60
37	This document is misclassified as Arabic. It is upside down, contains vertical, horizontal dashes, and two Japanese lines. . . . .	61

38	a) A sample of image document, b) diagonal white runs and their square boxes, c) vertical edges between the columns, d) a head and a tail are shown at the upper and lower parts of each vertical stream, respectively. . . . .	72
39	Vertical, horizontal streams, and nodes. . . . .	73
40	a) English magazine. b) segmented image, c) text/graphics separation. d) script identification. . . . .	74
41	a) An English 2-column page, b) segmented page, c) text/graphics separation, d) script identification. . . . .	75
42	a) A Japanese technical journal, b) segmented page, c) text/graphics separation, d) script identification. . . . .	76
43	a) A Japanese technical journal, b) segmented page, c) text/graphics separation, d) script identification. . . . .	77
44	a) Arabic magazine, b) segmented page, c) text/graphics separation, d) script identification. . . . .	78
45	a) Arabic magazine: a text block is embedded inside a frame, b) segmented page, c) text/graphics separation, d) script identification. . .	79
46	a) Chinese magazine: column with small gap, b) segmented page, c) text/graphics separation, d) script identification. . . . .	80
47	a) French letter, b) segmented page, c) text/graphics separation, d) script identification. . . . .	81
48	a) French memo, b) segmented page, c) text/graphics separation, d) script identification. . . . .	82
49	a) French scientific article, b) segmented page, c) text/graphics separation, d) script identification. . . . .	83
50	a) French scientific article, b) segmented page, c) text/graphics separation, d) script identification. . . . .	84
51	a) A page from a Russian book, b) segmented page, c) text/graphics separation, d) script identification. . . . .	85
52	a) Spanish scientific article, b) segmented page, c) text/graphics separation, d) script identification. . . . .	86
53	a) Spanish scientific article, b) segmented page, c) text/graphics separation, d) script identification. . . . .	87

54	a) Swedish newspaper. b) segmented page, c) text/graphics separation, d) script identification. . . . .	88
55	a) A large black area occupied almost half of the page because a page was scanned randomly with a size smaller than the setting size, which created the dark region outside the paper borders. However, the result showed that our method is not sensitive to this kind of problem and the block is correctly extracted. b) segmented page, c) text/graphics separation, d) script identification. . . . .	89

# List of Tables

1	A brief survey of skew detection methods and their distinct features. .	16
2	Document types and their layout descriptions. . . . .	39
3	The table shows the quantification result of the above line projection. Each row corresponds to a peak, showing its length and position from the top of the line. Note that the two tiny peaks on the left were not considered. . . . .	53
4	Comparative results. . . . .	62

# Chapter 1

## Introduction

Against most predictions, the rapid growth of computer and information technology has not at all reduced the amount of paper documents. Instead, many people still rely on paper as their favorite medium and source of information. This paper usage could be attributed to many aspects such as interactivity, portability or usability. Humans sometimes do not feel the need for a computer to write a letter, or a note; they can simply take a sheet of paper and a pen and start writing. Also, in order to read a book, why should we be overwhelmed by the complexity of hardware and software (e.g. text viewer or a browser to display the text), when we can simply take a book from the shelf, open it and read it. In other words, some people might not feel having to involve special hardware or software to access, read or write document is natural and intuitive.

Since paper is still a necessary form of communication that is well suited for many purposes, a more realistic objective is to integrate it into our computer world. For this purpose, a document analysis and recognition system could play the role of a medium between the human and computer. In this context, the aim of such a system is not to replace the fast visual and cognitive perception of humans, but rather to act as a technical assistant capable of helping us in our daily work with paper documents. This help could be realized by converting documents into electronic form (image) and then further process the image for different purposes such as finding those documents in the archive that belong to a certain category, sorting and distributing according to the language identity of the text, and in other cases directly replying to the person

whose name and address appear on the document (e.g. a fax form). Many more useful applications in this area could be developed to assist humans in document information processing.

Before we send the paper document to the computer and then proceed to design a document analysis system, it might be interesting to know how the computer sees and perceives the paper document compared with humans. Is there a limit to the number of parameters that the computer captures, compared with the unlimited capacity of human vision? To answer these questions, we should know that when a paper document is first converted to an image, what the computer can see is only a set of pixels distributed on a 2-dimensional image. So we are restricted to the coding mechanism of capturing the document in terms of the image format and its resolution. Consequently, these limited resources appear to have some effects on the interpretation and understanding of the document, which makes the document analysis a challenging and difficult task.

On the other hand, human acquisition of information functions differently. The human eye is a remarkable instrument with respect to both sensitivity and resolution. Somehow, a set of parameters in the human vision system is structured in a way to form an image of the scene. For example, in each eye there are 120 million rod cells, 6 million cone cells in each fovea in the region of maximum uniform density, as well as 250 million receptor cells in the two eyes compared to 250,000 independent elements in a TV picture. Given this huge number of cells, one might ask: how then is the information from the eyes coded into neural terms - into the language of the brain - and then interpreted?

When light strikes the retina, the decomposition (bleaching) of pigments results in electrical activity, which is integrated in some cells comprising the sixth and eighth levels of the ten-layer system of the retina. One layer, for instance, preserves the topology and much of the geometry of the imaged scene information.

We do not want to go into details on how the eye and the brain works, but the important point for our discussion is to shed some light on the amazing mechanism by



which organic eyes can easily and quickly detect shape information and colors, which are important attributes for classifying or identifying objects such as classifying the document and identifying its language. For more information on the subject, the book "Intelligence: The Eye, the Brain, and the Computer" [1] is an excellent source of information.

From the human capturing of image, let us return to our computer, or more precisely, to our pixels and examine how they might be explored. For example, can we extract meaning or semantic content from these pixels? Or, can we retrieve information or identify a specific form? Or, can we know the type of document or language of the text?

Those questions are most challenging topics that many researchers have been investigating for more than a decade. Recently, advances have been made in the area of Optical Character Recognition (OCR)<sup>1</sup>. Well, great efforts were focused toward developing a digital library for text and image retrieval for the World Wide Web (WWW). However, the computer is still unable to decompose and label a document image at a satisfactory level, despite all the efforts that researchers dedicated to the image processing area. At present, most of the commercial systems are custom designed to process some specific types of documents such as bank checks, tax forms, postal or mailpieces, but they are only limited to their assigned tasks and cannot be adapted easily to read different types of documents.

Our concern in this thesis is to develop a document analysis system to automatically segment and label a vast variety of real-life documents written in different languages. Thus, throughout this research project, analysis will center around two important aspects: page segmentation and region identification.

---

<sup>1</sup>Optical Character Recognition is the term used to designate devices or systems for automatic text image reading. The system converts a text image into ASCII format.

## 1.1 What is Page Segmentation

Page segmentation is the process of partitioning a document image into a set of regions which contain a unique type of data such as text, graphics, line drawings or pictures. For instance, a region identified as text may be defined as a homogeneous collection of essentially similar components of moderate size, which is usually surrounded by prominent “white space”, and is confined to a single column of text with only one reading order. Also, font type, style and size are mostly homogeneous across one region. A region identified as graphics, however, may be viewed as ambiguous where components might differ greatly in size and are not ordered linearly.

The reason we segment into subregions rather than taking the whole page is that the image may be composed of different entities (graph, text, table, line drawings, etc.), and separating these entities is necessary so that time will not be wasted in attempting to interpret. For instance, the graphics as text if an Optical Character Recognition (OCR) system is going to be used. Or, in a multi-column newspaper, columns of text must be separated so that the OCR can convert the text in the right order rather than in the direction of the scanning process [2]. Thus, page segmentation is a prerequisite for any further analysis.

## 1.2 What is Region Identification

As the name “region identification” infers, the strategy is to identify the regions which are extracted during the segmentation process. Each region could be assigned an identity so that their contents can be understood. For example, a region could be identified as text, graphics, table, halftone, caption, ruling (a box that encloses a section of text), etc. The labels given to regions help in determining the type of treatment to be applied to each block during the analysis and understanding stage. For example, if a region is identified as text, we further need to know in which language this text is written because OCR software is specialized on particular languages. We will see more applications in the next section.

## 1.3 Main Applications

There are significant applications in the area of document segmentation and identification. First, a document layout analysis system could complement the OCR software. Since a document could be composed of both text and graphics, it is necessary to extract only the text regions which can be considered as an input to the OCR system. Otherwise, graphics might be converted into lines of text, which is not tolerable in the face of OCR errors. Therefore, the page segmentation and region identification is an important phase to improve the performance of a document understanding system.

Application is not only restricted to text regions but the graphics could also play an important role. For example, a graphic region can be passed to an object recognition system to locate and extract objects from photographs [3], and a region labelled halftone might also be converted to a continuous tone image (inverse halftone) for storage, interpretation, conversion to different types of halftone (e.g. to improve transmission efficiency [4], or display on a gray-scale device).

More applications on specific types of documents take, for instance, an archived scientific journal where important components (title, author, abstract, references, etc.) extracted by the system, allows one to browse older issues for reference purposes, or to follow threads for particular topics [5, 6]. Satoh et. al. [7], for instance, describes a document image understanding system that is applied to table-of-contents of academic journals to achieve automated generation of electronic library system (called CyberMagazine). One of its functions is to retrieve an article list by author's name, or article title. The basic steps of browsing articles are as follows: (1) Locate target article, (2) Show page image of the article using a page viewer (called PageView). This application can also be linked to the Web for public usage.

A document analysis system could also be applied on the incoming fax cover pages which usually accompany the document being transmitted. This cover page carries information regarding sender, recipient, comments, date, phone numbers, etc. In this fax application, this information needs to be located and recognized to automate the fax distribution so that the system can read an incoming fax cover sheet and store the information in a database. Later on the user can easily retrieve and route the

fax to the receiver's electronic mail box. Moreover, since many personal desktop computers are now equipped to receive data by fax, one of its applications is to recognize tables as well as locating and characterizing the cells of the table sent by fax [8].

Another application which we have developed in our lab is to automatically process international documents. The aim of the system is to categorize documents written in different languages into their script type: Roman, Ideographic, or Arabic (this system will be described later in this thesis). Then the Roman script has been classified into 11 languages. In the literature, several systems for identifying the language (English or French, Chinese or Japanese) of a text region have also been demonstrated [9, 10, 11].

Having mentioned all these applications, now our question is: could we have one unique or a general system for all these different types of applications? Well, it seems to be a trade-off between generality and the amount of logical information that can be extracted for different applications. Let us take a look at what researchers are commenting on the subject. For example, in [12], Jain and Yu mentioned that: *From the point of view of commercial applications, it is very difficult to design an automatic system for document transformation which is efficient for different applications; interactive editing is unavoidable.* Moreover, [13] also mentioned that *in reality, it is very difficult to develop a general system that can process all kinds of documents including financial documents.* These statements show that there are certain limitations in terms of analysis and generalization.

## 1.4 Type and Quality of the Documents

A review on the type of document used for research and commercial applications, as well as on the comments from researchers about having a general system, allows us to specify the type of document so that the system could be applicable in real practical situations. Therefore, the type of our documents has been selected as follows:

- Types of documents are: books, advertisement, contracts, journals, faxes, newspaper articles, editorials, letters and manuals.

- A document could have a complex and irregular layout structure. Text, table, graphics or halftone regions could be present anywhere in the page. Text could have various fonts and their size may vary arbitrarily within the page. A page could contain multi-columns of text. Moreover, various languages may appear on the same page: French, English, Chinese, Arabic, etc.
- A document could be scanned with a skew, up-side down, without any clean-up made to the black border caused by the scanning process. For example, a page might be photocopied many times and then scanned, leading to poor quality, noise and distortion, high or low resolution.

We include documents with large variations to make the task challenging and difficult, rendering analysis and implementation more complex. The Appendix illustrates a variety of image documents we have selected.

## 1.5 Some Assumptions

We observe that a document is not a set of patterns distributed randomly, but rather follows certain universal conventions of legibility, style and printing technology. Thus certain assumptions could be considered as hints based on which we might define some rules for a document model. Some assumptions are as follows.

- **Spacial location** – its purpose is to approximately locate the region in a document image where an object may reside, since it is possible to define a general area in which the object will most likely be located. For example, in a journal paper, the title and the affiliation are always located at the top. Or, in a business letter, the signature is always located at the bottom.
- **Spacial relationship** – the location of an object may strictly depend upon those of its neighboring objects. For example, in the same journal page, the location of the author depends on the location of both the title and abstract.
- **Typographical convention** – the majority of characters in each text line must share a straight reference line, and it is usually called a baseline.
- **Layout delimiter** – white space is always used as a layout delimiter in similar ways by publishers and printers in many languages.

We will see in subsequent chapters how these assumptions have been used in our segmentation and identification system.

## 1.6 Organization of this thesis

This thesis is organized as follows: Chapter 2 discusses the preprocessing phase, which consists of the generation of the bounding boxes, removal of noise, and detection of the skew angle. Chapter 3 shows the segmentation of a page into blocks by vertical and horizontal gap extractions. Chapter 4 identifies the blocks as either text or graphics, and then the text blocks are classified as Roman, Ideographic or Arabic script. At the end, the appendix illustrates the result of the system.

In Figure 1, each rectangle corresponds to a chapter in which we include the main functions listed in the order of processing, as indicated by the arrows.

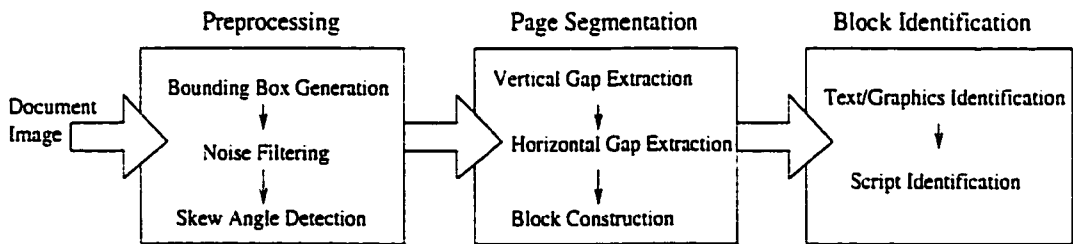


Figure 1: Overview of the main components of the system.

# Chapter 2

## Preprocessing

### 2.1 Introduction

By scanning a hard copy document, we obtain an image represented in a two-dimensional array. A document image might include noise caused by imperfect printing, photocopying, and digitization. Also it might be rotated by some degrees if the paper is not placed or not fed properly on the scanner. Such problems do not bother human readers, but they complicate automated analysis such as layout analysis, page segmentation and so on. So it is necessary to cope with these problems before the image is delivered to our segmentation and identification system.

In this chapter, our task is to preprocess the image by first filtering the noise and then by correcting the skew angle. Since these two operations are based on the bounding boxes of the connected components, we first start by the bounding box generation, followed by noise filtering and then by the skew angle detection.

### 2.2 Bounding Boxes Generation

The bounding box of connected components is defined to be the smallest rectangle which circumscribes the connected component of black pixels. Many authors have used the bounding boxes as a basic unit for different purposes, such as to deskew the image, to segment the page into blocks, to separate text and non-text blocks, and even to identify the language and script of the document.

The main reason for using bounding boxes instead of image pixels is because less

computation is involved and the bounding boxes are easy to manipulate for analysis – two coordinate points are sufficient to represent the box. Second, since a document is composed of columns, paragraphs, titles, figures, lines of text, symbols, etc, a human observer can get the immediate impression of the type of document by looking at its bounding boxes instead of the pixels. For example, from the alignment of the bounding boxes and their heights in Figure 5, we can almost tell that this is a “text” which might correspond to lower-case Roman characters.

In order to generate the bounding boxes, a simple iterative procedure has been used [14]. This consists of scanning the document line by line and checking the connectivity between the runs of black pixels in any pair of adjacent scanlines. If they are connected, a box is created whose size keeps growing, as long as there are more black run pixels on the current scanline joined onto black run pixels of the previous scanline. Figure 2, a) shows a sequence of three runs on one scanline.

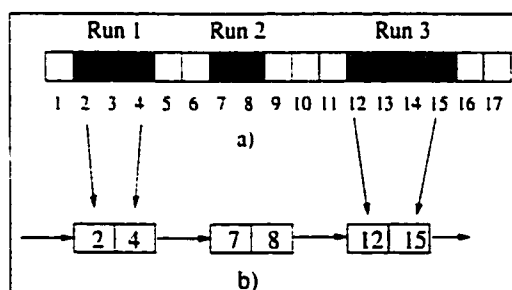


Figure 2: a) Three black runs on a scanline and b) an array of runs and their left and right coordinates.

Briefly, this method involves the following steps:

1. For each row  $r_i$  of pixels, extract black runs by storing beginning  $b_i$  and ending pixels  $e_i$  of each run into an array of runs –  $run[i]$ , (see Figure 2, b).
2. For each black run on row  $r_i$  and for each black run on row  $r_{i+1}$ ,  
if  $(e_i - b_{i+1} \geq -1)$  AND  $(b_i - e_{i+1} \leq 1)$ , the two runs overlap; therefore, merge them vertically into a bounding box (see Figure 3).

The data structure of this algorithm is a linked list of boxes where each element has two points: top left and bottom right. The result of the algorithm is illustrated



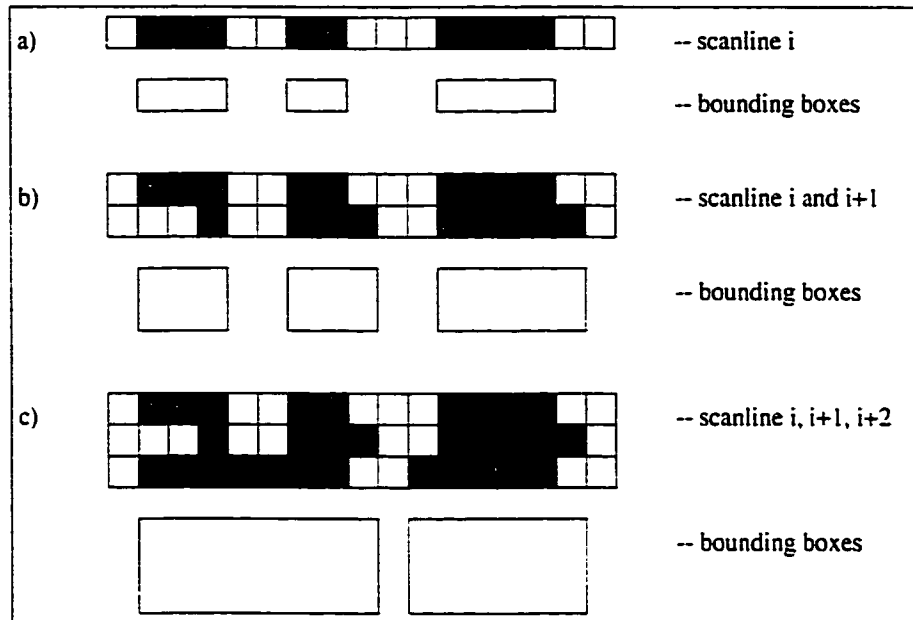


Figure 3: The process of generating the bounding boxes from image scanlines. Notice how the boxes are growing in size at each scanline. For example, at step a), box height = 1 pixel, at step b), box height = 2 pixels, and at step c), box height = 3 pixels.

in Figure 5.

# Death of N.K. defense n likely to accelerate power

The death of Choe Kwang, one of the few remaining members of the first-generation revolutionary group in North Korea, will likely accelerate the power shift inside the leadership of the Communist regime, analysts here say.

Pyongyang announced Saturday that Choe, minister of the People's Armed Forces, died of heart failure Friday. He was 78. A state funeral is scheduled for Tuesday.

Choe ranked second in the North Korean military hierarchy, following only Kim Jong-il, the de facto leader. He ranked fifth or sixth in the political hierarchy.

"The demise of Choe will not likely bring about a big change in the North Korean leadership as Kim Jong-il is firmly in power after the death of his father. Rather, it may help the junior Kim to strengthen his power," said an analyst.

He said Kim may fill the post of defense



Choe Kwang

minister with his own man, not an aide of his deceased father as he no longer wants members of the older generation to serve him, he said.

"It will be important to see who will become the defense minister in order to predict further changes," the analyst said.

North Korea Saturday issued a list of 85 members of the funeral committee, headed by Kim Jong-il and composed of senior party, state and military officials.

Prime Minister Kang Song-san, who is alleged to have been deposed from his post, and Hwang Jang-yop, who is staying at the South Korean Embassy in Beijing, where he is seeking political asylum to the South, were not included in the list.

The funeral committee member list often serves as the list of North Korean power hierarchy.

Following Kim Jong-il in descending order are Vice Presidents Li Jong-ok and Pak Song-chol, Kim Yong-ju, a brother of Kim Il-sung, Deputy Premier and Foreign Minister Kim Young-nam, Li Ul-sol, member of the National Defense Commission, Cho Myong-rak, vice marshal and member of the party's Central Military Committee, Kim Yong-chun, a vice

Figure 4: Image of a scanned document.

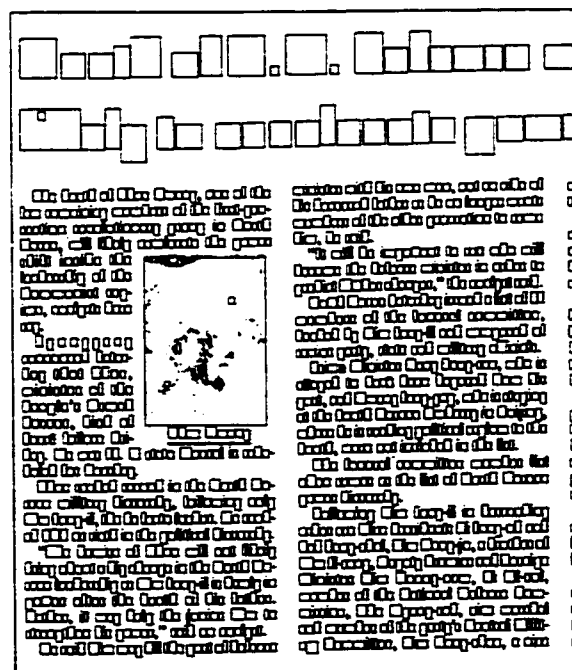


Figure 5: Bounding boxes of connected components of black pixels.

## 2.3 Large Area and Noise Filtering

### 2.3.1 Large Area Filtering

After the bounding boxes are generated, we start to remove the large boxes, such as pictures and photos. Since these boxes are not usually aligned, they are not used to deskew the image. For deskewing, it is more important to have boxes which stand straight in direction, like a text line. In other words, we keep boxes that look like “text” and get rid of what is likely not “text”. Also, since long vertical and horizontal graphic lines are also unlikely to be text characters, they have been discarded on the basis of width/height ratio of the bounding boxes.

The following 3 conditions [15] have been used for large area filtering.

From the first to the last box in the array of boxes, remove the box if :

1. Box height > 20 times box width OR
2. Box width > 20 times box height OR
3. Box Area > Area threshold<sup>1</sup>

### 2.3.2 Noise Filtering

Small boxes such as noise-like dots and background noise are also filtered out. The filtering algorithm works as follows.

1. Find the distribution of horizontal black runs, which is formed by scanning the document row-by-row and counting the neighboring connected black pixels.
2. Locate the most frequently occurring horizontal black run, which corresponds to the highest peak on the distribution. We consider this peak as the thickness of the text font.
3. Filter out the boxes with width or height less than this peak value.

### 2.3.3 Advantage of large area and noise filtering

Removing large and small boxes assist in decreasing the data to be processed by the Hough Transform to deskew the document. Thus, the processing time is reduced.

---

<sup>1</sup>The area threshold is set at five times the larger of the two parameters, the most frequent box size and the average area of the boxes

## 2.4 Skew Angle Detection

During the process of scan-digitization, if a paper is not placed or fed properly on a flat-bed scanner, texts on the document image become inevitably skewed. This will cause serious problems in subsequent stages, such as block extraction, line segmentation and language identification. For example, most of the segmentation algorithms like the horizontal profile or the run-length smoothing are extremely sensitive to even a small skew angle. Therefore, we need a function to automatically detect and correct the skew angle so that the texts are in their up-right orientation, before any further processing.

In the literature, several techniques have been reported and applied to solve the problem of skewed documents. In the following paragraphs, we briefly describe some of those approaches<sup>2</sup>, followed by a description of our method and a comparative analysis.

Baird's algorithm [17] detects the skew angle by using the projection profile technique. First, Baird applies CC (connected components) analysis to the document. The midpoint of the bottom of each CC is then projected onto an imaginary accumulator line perpendicular to different projection angles. Large CCs, like images, are ignored during the projection process; while other CCs, such as characters, character fragments, and margin noise are projected onto the accumulator line. Although the noise and character fragments are sources of skew detection error, Baird claims that they do not affect the detection of skew angle. The method is said to work on a wide variety of layouts, including multiple columns, sparse tables, variable line spacings and mixed fonts.

Hinds *et al.* [18] apply run-length encoding and the Hough transform. An input image is downsampled to 75 ppi and reduced to horizontal and vertical "burst". Every black pixel horizontally adjacent to another black pixel is replaced with white pixel except the one furthest right, which is replaced by the length of the run. An analogous process is done in the vertical direction. The Hough transform accumulator is then incremented by burst values between 1 and 25 pixels. This emphasizes text up to 24 points in size. The skew angle is determined by finding the  $(\rho, \theta)$  accumulator cell that has the largest value. The Hough transform is computed for  $\theta$  between  $\pm 15$  with a resolution of  $0.5^\circ$ .

---

<sup>2</sup>Note that a detailed description of some of the following approaches can be found in [16].

Jiang *et al.*'s algorithm [19] uses nearest-neighbor clustering paradigm. The local clustering process is focused on a subset of plausible feature neighbors. A refinement is achieved by a least-square line fitting using those plausible candidates. The skew angle with the computed straight line is used to build up a histogram. The peak in this histogram is regarded as the skew angle of the input document image. Note that O'Gorman [20] also used the  $k$  nearest neighbors (NNs) for each of the CCs; however Jiang *et al.* demonstrated a significant improvement over O'Gorman's [20] use of multiple neighbors for each CC, which made the connections both within and across text lines, and thus the histogram peak may not be very accurate.

Akiyama and Hagita [21] divide the page into columns (assuming the text is horizontal) and then calculate horizontal projection profiles for each column. Each peak in the profile corresponds to a text line in the column. The skew angle is obtained by calculating the arctangent of the phase shift between adjoining profiles. This method has been shown to be extremely fast, but requires extremely regular text and cannot handle graphics. In addition, the algorithm cannot calculate skew angles of more than about 5 degrees with any sort of accuracy, because the projection profiles will overlap too far and no accurate determination will be possible.

Avanindra *et al.* [24] use the correlation approach which attempts to find the best correlation between two or more profiles of the image, taken from vertical (horizontal) cuts of the image. When the image contains text organized in a few lines, one can deduce from the correlation between a pair of profiles the amount of vertical (horizontal) shift the lines were exposed to. The skew angle of these lines can then be calculated, knowing the distance between the cuts.

A brief survey of skew detection methods is presented in the following table.

Author	Year	Approach	Distinct Features
Baird [17]	1987	Projection of the midpoint of the bottom of each connected component	Work on a wide variety of layouts
Hinds <i>et al.</i> [18]	1990	Hough Transform and Run-Length Encoding	Only small runs are input to the HT, thus negative effect of figures, black margins are avoided
Jiang <i>et al.</i> [19]	1999	Nearest-neighbor clustering, connected components and pass codes	Focus on a subset of plausible neighbors, thus sharper peak
Akiyama <i>et al.</i> [21]	1990	Divide the page into columns and then calculate horizontal projection profiles for each column	Extremely fast but cannot calculate skew angles of more than about 5 degrees
Avanindra <i>et al.</i> [24]	1997	Interline cross-correlation over a small region	No prior segmentation of text and graphics regions
Yan [23]	1993	Interline cross-correlation over the whole image	Accurate but time consuming
Le <i>et al.</i> [28]	1994	Hough Transform applied on the connected components of a square sub-region of a document	Accuracy angle 0.5 degree Tested on 250 medical journals
C. Yu <i>et al.</i> [30]	1995	Modified fractal signature and least squares method	Detect the skew not only on the whole page but in different blocks with different angles
B. Yu <i>et al.</i> [27]	1996	Hough Transform applied on the centroids of connected components	Demonstrate on articles, postal labels, handwritten text, forms and drawings
Chen <i>et al.</i> [25]	1999	Horizontal and vertical cross-correlation over a small region	Apply to grayscale image Deal with Chinese and Japanese documents
Ma <i>et al.</i> [26]	1999	Find lowermost pixels of some selected characters to obtain base lines. Find first-line from the pixels. Cluster pixels and find the angle.	More accurate than the Hough transform but it is not robust for complex document
Okun <i>et al.</i> [29]	1999	Use text row accumulation technique which is based on the 1st and 2nd orders moments where all processing is done on the low resolution (50 dpi).	No restriction on detectable angle range, large text areas are not necessary

Table 1: A brief survey of skew detection methods and their distinct features.

### 2.4.1 Our Skew Angle Algorithm

In the previous section, algorithms that estimate the skew angle are surveyed. Four broad classes of techniques could be identified. These include (1) Projection profile, (2) Nearest neighbor clustering, (3) Correlation, and (4) Hough transform. In the projection profile, an interpolation of the image is necessary because the new position of a pixel may not be represented by integers after a rotation. The nearest neighbor clustering method will not work well for Arabic script because the characters are connected and a complete word can be a single component, unlike in Roman script. Since word length can vary, the nearest neighbor of a word (computed in terms of its centroid) may not be another word to its left or right, and thus the direction of the neighbor may not indicate the skew angle properly.

In this thesis, we use the Hough transform<sup>3</sup> and the most frequently occurring local maximum. The advantage of the Hough transform technique is that it is robust to noise and low resolution, as well as being tolerant to gaps between characters and words.

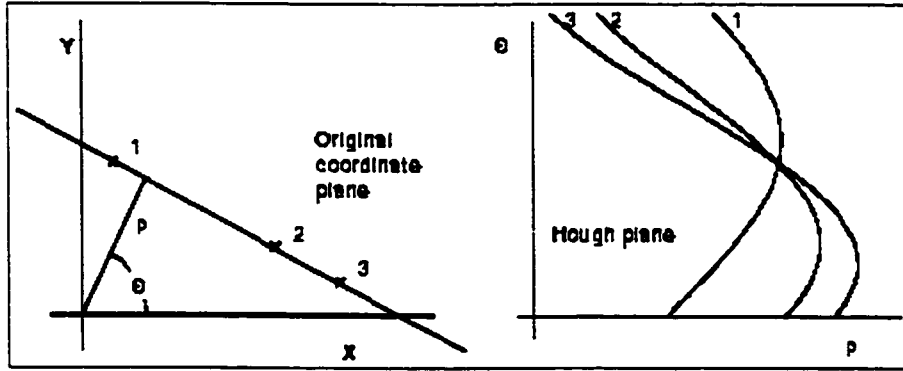


Figure 6: Collinear points  $(x_i, y_i)$  with  $i=1, \dots, N$ , are transformed into  $N$  sinusoidal curves  $\rho_i = x_i \cos \theta_i + y_i \sin \theta_i$  in the  $(\rho, \theta)$  plane, which intersect in the point  $(\rho, \theta)$ .

The algorithm consists of the following steps:

1. We start with the connected components extraction, noise and large area filtering (these steps have been described in previous sections).

---

<sup>3</sup>Hough transform is a popular method for skew detection. It is capable of locating fragmented lines in a binary image. Therefore, given a group of black pixels, one can find the imaginary line or lines that go through the maximum number of these pixels. Given a binary image with a dominant text area, the detected lines will most probably go along the text line.

2. Then we define the Hough space along the  $\rho$  and  $\theta$  axis, where  $\rho$  is the distance along the normal from the origin to the line and  $\theta$  is the angle that the normal to the line makes with the x axis (see Figure 6) – we choose  $\theta$  to be between  $-90^\circ$  and  $+90^\circ$ , and the resolution angle is set at  $0.5^\circ$ . Along the  $\rho$  axis,  $\rho_i$  ranges between  $-\rho_{max}$  and  $+\rho_{max}$ , where  $\rho_{max} = (h^2 + w^2)/R$ ,  $h$  and  $w$  are respectively the height and width of the image, and  $R$  is the resolution which is set equal to 1/5 of the average box height [15].
3. After the Hough space is defined, we use a transformation equation  $\rho_i = x \cos \theta_i + y \sin \theta_i$  where each time a sinusoidal curve intersects another at  $(\rho_i, \theta_i)$ , we increment the value at this location (see Figure 6). In practice, taking each box of the connected component, we transform its corner points - the bottom left and right corners, from  $xy$  space to  $\rho\theta$  space using the above transformation equation. Note that since transforming all the pixels is time consuming, two pixels have proved to be acceptable for our skew detection.
4. In the  $\rho\theta$  space, we search for peaks which correspond to lines in the image. The angle of each line can be found from the coordinates  $\rho$  and  $\theta$  of the peaks. Note that the peak is formed when the transformed points lie along a given line in the image.
5. Then, the maximal peaks are located within a window of 8x8 cells, and the maximum is detected within a neighborhood surrounding each maximal peak.
6. We then search for the skew angle in the region where angles occur most frequently, namely with the range of  $-25^\circ$  to  $25^\circ$  (note that this range is chosen empirically because most of our testing documents fall within this range). Thus, the most frequently occurring local maximum in this range is regarded as the skew angle.
7. Finally, an image rotation is performed relative to the center of the image by separately rotating each pixel. The equations which transforms the location of a pixel of the source image (" $X_{old}, Y_{old}$ ") into its new rotated location in the destination image (" $X_{new}, Y_{new}$ ") are as follows:

$$X_{new} = X_{old} * \cos(\text{angle}_{skew}) + Y_{old} * \sin(\text{angle}_{skew})$$



$$Y_{new} = Y_{old} * \cos(\text{angle}_{skew}) - X_{old} * \sin(\text{angle}_{skew})$$

These equations and their proofs can be found in many elementary trigonometry or computer graphics books under the topic of vector rotations. Figure 7 shows an original image and the result of skew detection and rotation.

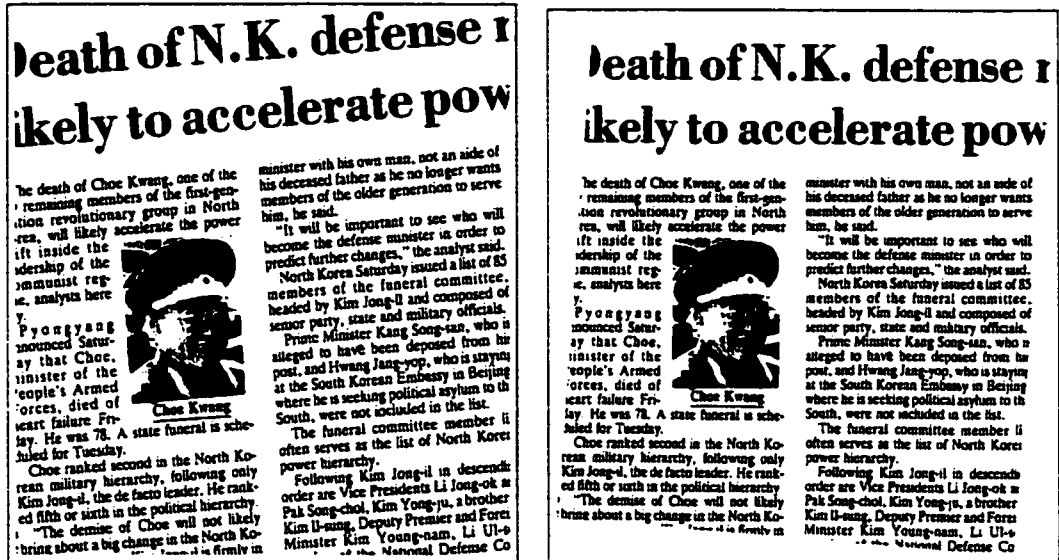


Figure 7: Left) original image, and right) the result of the algorithm: the skew angle is detected as equal to  $4.5^\circ$ , and the image is rotated accordingly.

## 2.5 Testing Data

We have tested the method on more than 200 binary document images ranging from 200 to 300 dpi. Our testing set included various document layouts, such as technical articles, newspapers, drawings, magazine, letters and forms. Also, we included documents written in different languages and scripts, such as European, Ideographic and Arabic. Our data have been randomly chosen from real-life documents which were scanned as they are (black-margins and noise are not filtered out). Some documents have been purposely rotated in different angles to test the performance of the skew algorithm. The system detects 99% correctly, and 1% error. An example of a document where the method fails to detect the skew correctly is presented in Figure 8.

The actual skew of  $-0.5^\circ$  was estimated to be  $17.5^\circ$ . Such a gross error in the estimation can be justified by examining the content of the document. An analysis of the image shows that the small noise in the shaded areas are causing the error. Such noise should have been filtered out in previous steps, but it seems that the second step of noise filtering algorithm (from section 2.3) did not remove it entirely. The reason is that since the threshold for noise removal is based on the thickness of the text font, and since the noise is numerous in this document (almost 50% of the total number of connected components), the threshold is taken as the size of the thickness of the noise, but not the size of the text font. This yielded a very small threshold value for the text font thickness, and since the noise is not filtered out, the average height of the bounding is found to be 1.94. This number is very small compared to the normal size of a character, which should be at least 6 pixels for this type of document. So when those pixels are numerous in the Hough space, the skew angle is randomly created in any direction and not just in the direction of the text lines, leading to a skew angle error.

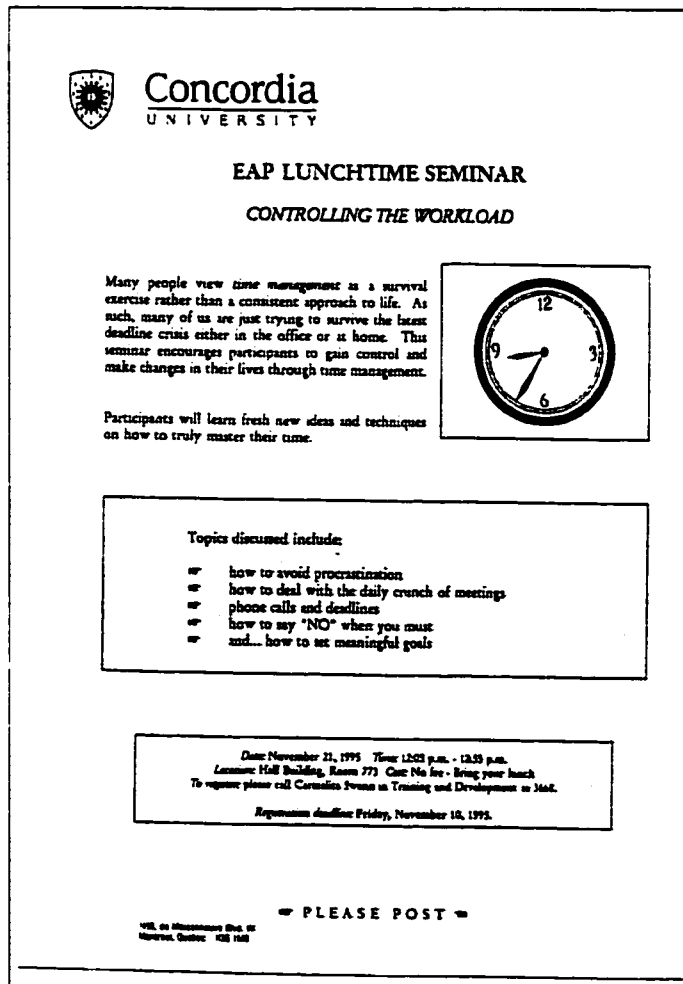


Figure 8: Example of a binary document image where the method fails to detect the skew correctly, because of the shaded area in the text frame. Note that the shaded area consists of very small connected components which are created during the digitization process.

## 2.6 Comparative Analysis and Discussion

From table 1, we notice that the Hough Transform has been adopted by many authors, but each one applied it differently. For example, B. Yu *et al.* [27] applied it on the centroids of connected components, whereas others combined it with the run-length encoding. In our case, we applied the Hough Transform on the bottom left and right corners of each CC. Also, we added the most frequent local maximum to the final step.

The reason for using two corners instead of one is because of the nature of the Arabic language. Since the Arabic language is cursive, characters touch; and thus, we end up with wide bounding boxes. Consequently, the line will have few bounding boxes, where each box is composed of many characters. For our deskew algorithm, taking one point from each box is not enough when the line is composed of less than 5 words. In other words, the more colinear points exist on each text line, the more efficient is the deskewing. Experimental results have shown that two points are sufficient and acceptable to detect the skew angle of Arabic Documents. Note that none of the methods listed in Table 1 experimented on Arabic documents.

Moreover, because of some large peaks that might be encountered on the Hough space, we used the most frequent local maximum, which gave better result than averaging all the angles. Local maximum is a bright spot on the accumulator array. It is local because a relative threshold is applied on the isolated clusters of bright spots in the accumulator array image. In other words, we only consider those local maxima in the accumulator array whose values are equal to or greater than some fixed percentage of the global maximum value. All local maxima are accumulated in a histogram according to their directions – every local maximum is incremented whenever we encounter lines with the same directions (like a paragraph composed of many parallel text lines). However, in some particular cases, we encounter large peaks that do not correspond to text lines. For this, we use the most frequent local maximum to disregard those misleading peaks.

In general, the main advantage of the Hough transform technique is that it is tolerant to gaps between characters and words, and is relatively unaffected by noise (excluding the shaded areas). In our method, we combined the Hough Transform with some proximity criteria to detect the skew angle of various document layouts,

including different script types, and which also include fluctuations, noise, black-margins, etc.

Regarding the shortcoming of this technique, we presented a testing sample with shaded area in the previous section. Besides that, the second problem is the processing time. Many authors from the literature have already mentioned that whenever the Hough transform is used, there is always a tradeoff between accuracy and speed. The more accurate the angles of the detected lines, the more computation is required. Moreover, the more pixels in the image, the greater is the computation time. Most methods make some effort to reduce the number of pixels by image compression, by reducing the resolution or by parallelization of the algorithm for multiprocessors. In our case, we simply eliminated the small and large connected components before applying the Hough transform method. But we could not take less than two points (the bottom left and bottom right corners) from each bounding box, and we did not experiment on multiprocessors.

The average time to deskew and rotate a document image is about 3 seconds (not including I/O in CPU seconds) on Sun Spark 20 workstation, with an accuracy of 0.5 degrees. For example, the image in Figure 38 is of size 1658 by 2263, and it takes about 2.5 seconds for the skew detection and rotation.

# Chapter 3

## Page Segmentation

### 3.1 Introduction

Automatic document segmentation has been explored by many researchers. Sophisticated techniques and algorithms have been proposed and applied in recent years. Many approaches concentrate on processing the foreground of the image while others process the background of a page [37]. Some of these approaches are briefly described in the following paragraphs.

Ittner and Baird [9] use the method of greedy white large rectangles created from the background of the image. These rectangles are pieced together to separate the blocks of texts. But first the rectangles are enumerated, sorted and unified one by one greedily until a stopping rule is satisfied. As a result, the layout is then partitioned into covered and uncovered areas. Disconnected regions in the uncovered area are identified as text blocks.

Pavlidis and Zhou [2] identify wide white spaces on adjacent scanlines and form sequences of such spaces called “white streams”. Columns are then found as the regions between white streams. They also use the projection profile over a block of scanlines rather than the whole page to partition a document into columns.

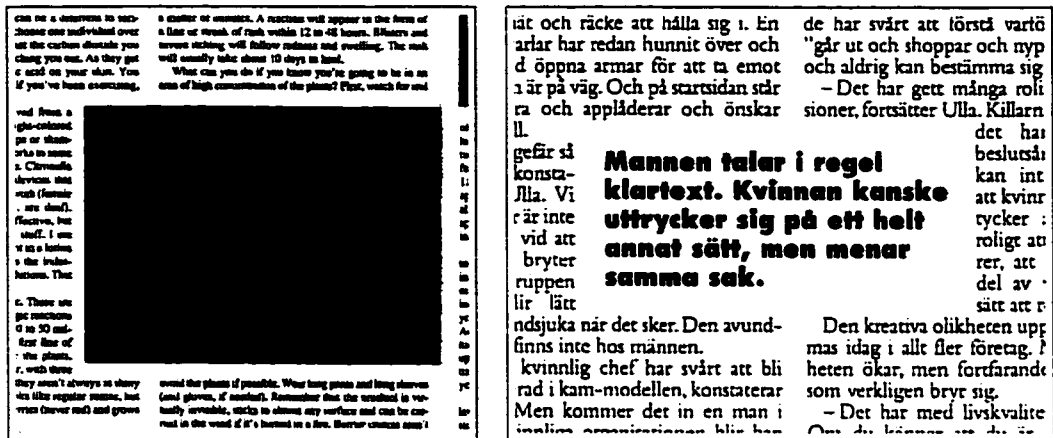
The Run-Length Smoothing Algorithm (RLSA) [32], which is a different approach, consists of merging any two black pixels that are less than a threshold apart, into a

continuous stream of black pixels. The method is first applied row-by-row and then column-by-column, yielding two distinct bit maps. The two results are then combined by applying a logical AND to each pixel location. The output has a smear wherever printed material (text, pictures) appears on the original image.

Nagy *et al.* [33] use the X-Y tree approach, which assumes that a document can be represented in the form of nested rectangular blocks. A “local” peak detector is applied to the horizontal and vertical “projection profiles” to detect local peaks (corresponding to thick black or white gaps) at which the cuts are placed; it is a local operation in that the width is determined by the nesting level of recursion, e.g., gaps between paragraphs are wider than those between lines.

The methods mentioned above could be divided into two categories: foreground (using the black pixels) and background (using the white pixels). RLSA technique, which belongs to the foreground category, is the most powerful for using the black pixels, however, it imposes two smoothing parameters, for vertical and horizontal directions, defined in a heuristic way. Also, it is necessary to train the system with documents having similar fonts. Thus, the method is not robust since the assumption for the parameters are not always satisfied on different types of documents.

Our method ideally belongs to the second category, in which the white spaces are the basis for image segmentation. A more detailed comparison of our work with the works mentioned above is described in section 3.5. However, it is very important to point out that using this method, some challenging problems, on difficult layouts, have been addressed, such as embedded images or text between two columns of text we find in newspapers or technical journals (see the cropped images in Fig. 9 and in the Appendix: Fig. 40 and Fig. 54). Also, we addressed the problem of text surrounded by a frame (see Fig. 10 and in the Appendix: Fig. 45).





We overcome these problems by proposing a new technique which is based on diagonal scanlines and on vertical edges [34]. This technique allows the detection of gaps around the text block, independent of where and how they are located – embedded or inside a frame.

We use the diagonal scanning approach to create the “drun” (a detailed explanation will be presented in the next section, with an illustration shown in Figure 13). Briefly, the “drun” is a set of diagonally connected white pixels along predefined diagonal scanlines. The advantage of this approach is that these “druns” can fall in the gaps between the lines or the columns, whether those gaps are at the borders or in the middle of the page (see Figure 14). Thus they are well suited for embedded layout structures. Moreover, it reduces the oversegmentation created from regular vertical scan. Oversegmentation usually takes place between the inter-word gaps, as shown in Figure 11. The oversegmentation has been reduced by diagonal scanning, as shown in Figure 12.

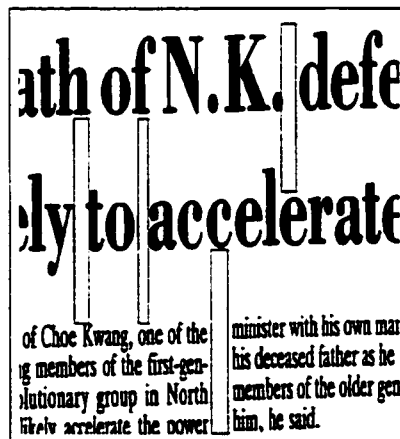


Figure 11: Oversegmentation occurs between words in the title when white, long rectangle method is applied on the background of the image.

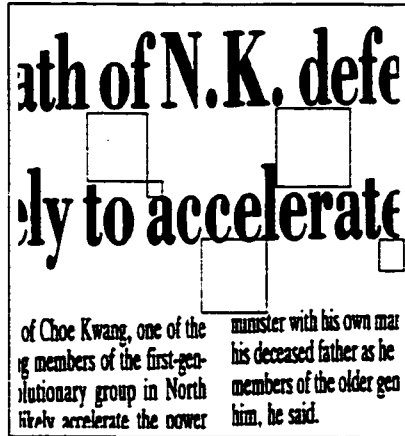


Figure 12: Oversegmentation has been reduced by druns (the diagonal boxes). Notice that the druns do not go through the gaps between the words.

## 3.2 The Technique

The new technique locates the vertical and horizontal gaps in five main steps described below.

### 3.2.1 Step 1: Extract Diagonal White Runs

A diagonal white run (drun) is a set of adjacent white pixels that are diagonally connected (Figure 13.a). Since each run is actually a diagonal, we draw a square box around it so that its size could be clearly shown (Figure 13.b). Figure 14 shows a real document with its diagonal runs represented by square boxes.

In our analysis, we observed that the diagonal runs located between the text lines are smaller than those that lie between the paragraphs or the text columns. Therefore, small druns of size less than 20 pixels are filtered out because they do not really contribute to locate the vertical gaps between the text columns.

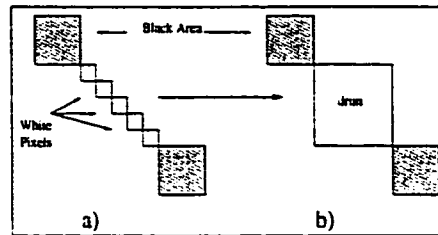


Figure 13: a) A drun of connected white pixels between two black regions. b) The white pixels are merged to create the white run which is illustrated by the square box.

The process of extracting the diagonal white runs consists of:

1. Scan the document at a 45-degree angle as if we are drawing long back-slashes all over the page (see Figure 14). As a result, the scanlines will look like 45-degree parallel lines separated by some distances in-between. Note that if we increase the number of scanlines, the distance in-between will decrease. In our analysis, we set the number of scanlines according to the width of the page. For example, the step to move from one scanline to another was taken as one-sixth of the image width, which proved to be good enough to detect the vertical streams.
2. While scanning along the scanline (from top-down at 45-degree, we stop if we encounter a white pixel and then allocate a run. Now, if we find another white pixel connected to the previous one, the run keeps increasing in size until a black pixel or the border of the page is reached (see Figure 13). The result will be a set of runs created from each scanline, and this is stored in an array of white runs as shown in Figure 15.

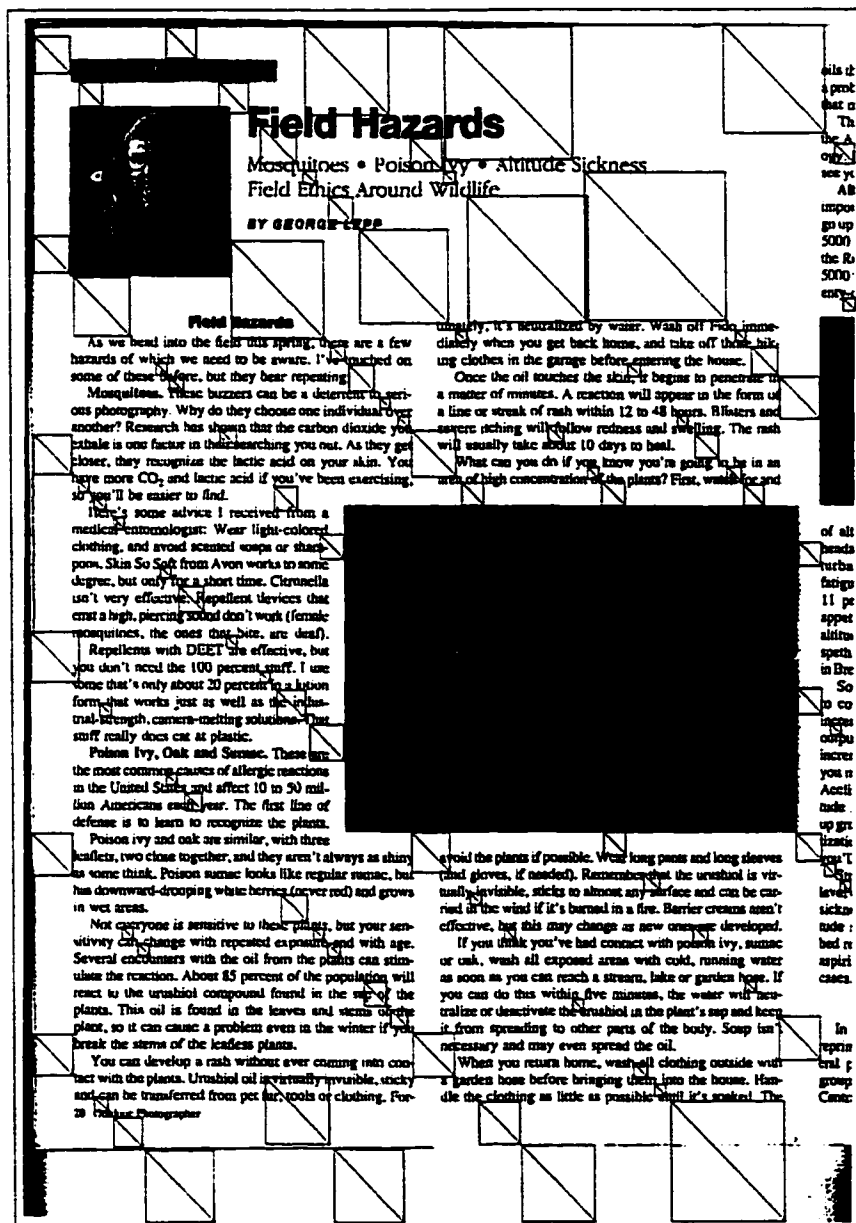


Figure 14: This figure shows the diagonal white runs (druns) and their square boxes. The boxes are drawn to illustrate the size of the druns. Thus, black pixels inside the boxes have not been considered. Note that one drun (see top right) has been removed because it is very large and it overlaps with the one on the next scanline. Also note that the distance between the scanlines should not be very small because vertical edges might go through gaps between words in the headline.

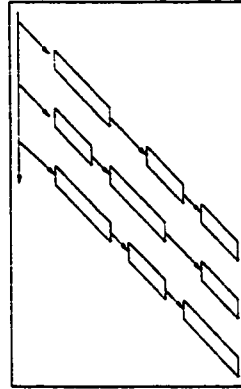


Figure 15: A diagonal array with each element representing a white run.

### 3.2.2 Step 2: Create Long Vertical Edges

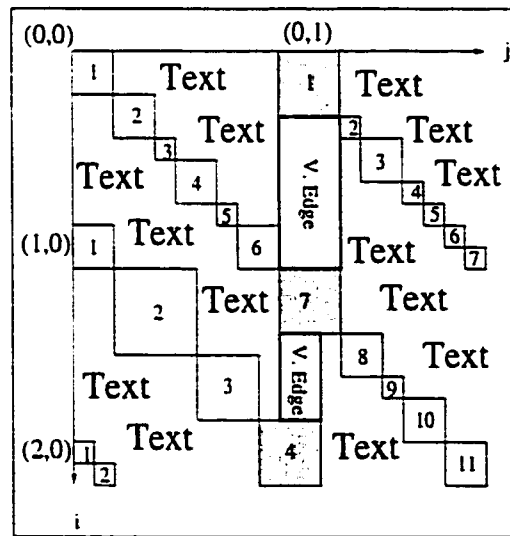


Figure 16: This figure simulates the creation of the vertical edges. See the vertical edge between the drun #7 on scanline (0,0) and drun #1 on scanline (0,1).

By drawing parallel equidistant lines on the image, we are dividing the whole page into sections. This allows the creation of the vertical edges (see Figure 17) for each section, one by one, instead of taking the whole page at one shot<sup>1</sup>.

<sup>1</sup>This enables embedded layouts to be handled correctly. Note also that decomposing the image into small sections makes the detection of vertical edges less sensitive to skew, as mentioned in [2] – “We take the projections only over a short height of scanlines so skew does not obscure column gaps”. In our case, for instance, if the document is skewed slightly, we can still easily detect the short vertical edges between two overlapping runs



Figure 17: The black, long rectangle illustrates the vertical edge between two drums.

The algorithm of creating long, vertical edges can be described as follows.

1. Take two successive scanlines (i.e., (0,0) and (0,1) in Figure 16) and two drums, one from each scanline (i.e., the grey boxes #7 and #1), and check whether they overlap in their vertical projection.
2. If they overlap, scan the region between them at the overlap area and verify whether a connected white area exists or not.
3. While scanning vertically between the two drums, if a black pixel is encountered, we stop and skip the current drum and jump to the next drum.
4. But, if no black pixels are encountered, this means that all the pixels are white. In this case, a vertical edge is created. This vertical edge keeps growing in size horizontally as long as it is connected to another vertical run (see Figure 18, a).
5. If no black pixel is found in the area between the two runs, the edge is then completely formed, and is represented as a white, long rectangle inserted between two drums, see Figure 18, b).

The result of this part of the algorithm is illustrated in Figure 38.c, in the Appendix.

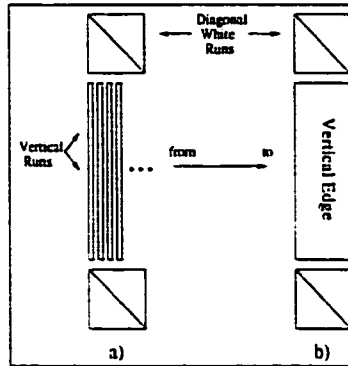


Figure 18: a) Vertical runs between two druns, b) A vertical edge is created by merging the runs in a).

### 3.2.3 Step 3: Setting up Vertical White Stream

A vertical stream is the union of connected vertical edges. It acts like a demarcation line between the text columns. By knowing the top and bottom positions of the stream, we can go from that position left and right to create a horizontal stream. In order to locate the top and bottom coordinates of the streams, we take advantage of the distinctive characteristics of text which make it stand out from other image material. For example, by looking along the vertical gap which separates a 2-column text, one can probably tell quickly where the vertical stream ends without actually looking at the horizontal gap. So the end of the stream might finish at the top border of the page, or it might hit another line of text or an image. Therefore, setting up the vertical stream is a major step in the segmentation process and it should be accurate for further analysis.

In the previous step, we were going diagonally to create the vertical edges, but now once the edges are detected, it is more convenient to access these edges directly without passing through the druns and then the edges. For this purpose, we create a new data structure which consists of an array of vertical streams composed from the vertical edges.

Setting up the vertical stream from the edges consists of the following steps:

1. Create the streams: the streams are created by joining the long, vertical edges located in the previous phase. We first scan the array of runs and look for edges to start the stream. These edges are usually distinguished from others

by having their “up pointers” point to nothing and the “down pointers” point to a drun (denoted as a node). Then going from edge to edge, we join edges vertically till we hit the end of the stream. It forms the union of all the edges which are aligned vertically to create a long, vertical stream.

2. Adjust top and bottom streams: The vertical streams created from the edges do not precisely indicate the top and bottom limits of the stream because it links the edges that are between the scanlines which are drawn randomly. So, in order to find the upper and lower limits of the streams, we need to go beyond the current stream coordinates. For example, to find the upper limit, we scan up vertically from the stream and stop at the closest connected component of black pixels. At this point, we set the upper limit. We do the same to adjust the lower limit.
3. Sort the streams: after the streams have been created, we sort them so that the order will be accurate for connecting a vertical stream to its closest neighbour (the connection process will be described in the next phase: “Set up Horizontal White Stream”).
4. Arrange broken links: a broken link is the area that links two vertical streams if these two fall inside a vertical gap but far from each other. A broken link could be created by an image embedded between two text columns. We deal with this type of document by creating a pointer link from the current stream to the stream that is aligned vertically.
5. Create head and tail nodes for each stream: we set a head and a tail for each stream, which will be used as references to find the horizontal stream. This consists of allocating two nodes: one at the upper part of the stream and another one at the lower part of the stream.



### 3.2.4 Step 4: Setting up Horizontal White Stream

The horizontal stream is the last step before we start extracting the block. We need to complete the white frame which consists of vertical and horizontal streams to demarcate a prospective block. In the previous section, we finished extracting the vertical stream; but here in this section, we will concentrate on locating the horizontal streams, according to the following steps:

1. Take the head node of each stream and scan left till the next stream is encountered. Here we might hit the head, the tail, or the body of the next stream (Figure 19 illustrates the three scenarios). For example, if the head is hit, we allocate a horizontal stream, squeeze it between the two head nodes, and nail it by pointing the nodes to it and from it. We do the same if we hit a tail node. However, if we hit the body, we do not only allocate a stream, but a node will be inserted in the body of the next stream, between the head and the tail (see Figure 19.c)).
2. In the first step, we worked with the head node. Now we take the tail node of the stream and scan left till the next stream. Here, we follow the same procedure as in step 1. One more thing to add is that if the tail does not hit any point on the next stream, we skip the stream and check the one that comes after. This will not only detect the horizontal stream between two successive streams but also the ones that overpass the next stream. For example, Figure 20 consists of three streams, and the horizontal stream that starts from the first one overpasses its neighbour to hit the third stream. In other words, we are creating a bridge that is considered as a path that connects two nodes and along which we jump from one vertical stream to another.
3. As we are going left, we also go right. We apply the same algorithm used in steps 1 and 2 for head, tail and body, but in the opposite or right direction.
4. Once the horizontal stream is created, we adjust its upper and lower coordinates when the stream runs over some black pixels that might be encountered within the stream area. As a result, we are detecting the wide horizontal gap between the blocks.

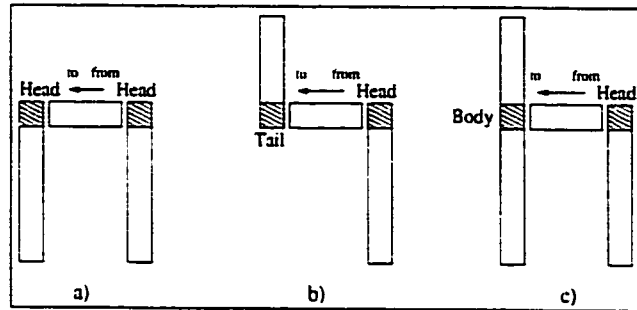


Figure 19: a) Head to head, b) Head to tail, c) Head to body

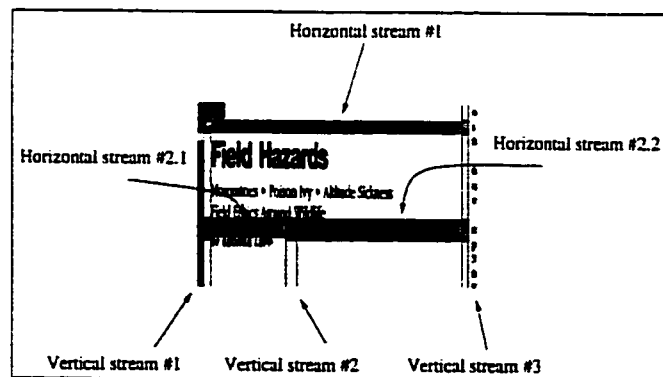


Figure 20: The horizontal stream #1 goes directly from vertical stream #1 to vertical stream #3. So it overpasses the vertical stream #2.

### 3.2.5 Step 5: Constructing the Blocks

By having both the vertical and horizontal streams, and the reference nodes inside the streams, we are now able to extract the blocks. The basic idea is to go from node to node along the horizontal and vertical streams, loop around, and come back to the starting node. Let us illustrate the process in Figure 21: take, for instance, node  $n_1$ , go left to node  $n_2$  of the next stream passing by the horizontal stream, then go down to the nearest node  $n_3$ , from which go right to node  $n_4$ , then return up to the starting node  $n_1$ . At this point, we allocate a block whose coordinates are bounded by the streams and nodes, and then we link the node to the block, (see Figure 21, the pointer from node  $n_1$  to the block (shaded area)).

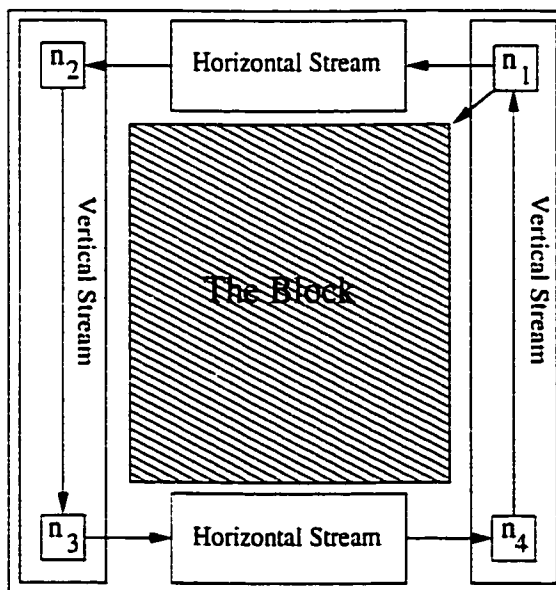


Figure 21: This figure displays the vertical and horizontal streams, the head and tail nodes inside the streams, and the block extracted.

## 3.3 Layout Structure

A new layout structure has been applied to the background of the image, with the purpose of setting up the background into vertical streams, horizontal streams, and

nodes as shown in Figure 23. The vertical streams<sup>2</sup> cut the page into multiple columns whereas the nodes indicate the horizontal positions from which the columns are split by means of cuts along the horizontal streams. Adjacency links among nodes and streams describe local relationships between corresponding blocks. Figure 22 represents the layout structure of a document composed of 3 vertical gaps and 6 blocks. In order to access one of the blocks, we should go first through the stream, the node, and then the block. In this way, each block could have a specific position on the document. The advantage of this structure is that the blocks can easily be located with respect to each other. For example as shown in the figure, *block 6* is below *block 5* and it is close to *block 4*. Note that this layout can also help to classify the document into single or multiple column documents by simply counting the number of vertical streams. Also, since we already captured some information about the position of blocks, the vertical/horizontal streams and their spacial relationships, we can use this structure to model the layout structure of the image. Furthermore, given some information about each extracted block, we can map the layout structure into logical structure where we can associate entities with the type of the contents. For example, a picture in the document can be associated with its captioning, or a title could be linked to a section heading and a paragraph.

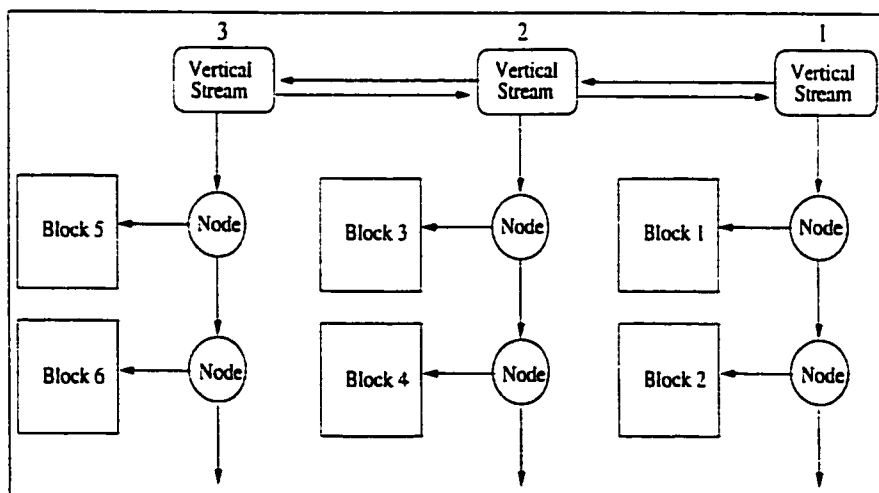


Figure 22: Layout structure: an array of 3 vertical streams where each stream points to 2 nodes and each node is linked to a block.

<sup>2</sup>Note that each vertical stream is the union of connected vertical edges which are formed by diagonal runs, whereas a horizontal stream is created from the nodes that lie on the vertical streams.

### 3.4 Experiments

We have tested the proposed approach on 118 document images with 14 different types and layouts, taken from a wide range of sources such as magazines, newspapers, technical journals, letters, memos, books, manuals, and articles. Furthermore, we have included documents written in different languages: English, Japanese, Arabic, Chinese, French, Russian, Spanish and Swedish. We have also tested documents with embedded texts or images (see Fig. 40, 45, 49, and 54). Some of these documents are described in the table below, in which we include a column of figure's numbers to refer to the Appendix. Note that we evaluate the method visually by comparing the automatically segmented page with the manual segmentation of the original document, because our documents are not ground-truthed.

Document Type	# Col	Layout Description	Refers to
English magazine	2	Embedded large image in the middle, small images, a title and some noise.	Fig. 40
English newspaper	2	A title in a big font, an embedded figure in one column.	Fig. 41
Japanese technical journal	2	Header and some black noise on top and left of the page.	Fig. 42 Fig. 43
Arabic magazine	2	Large image on top and its captioning.	Fig. 44
Arabic magazine	2	Embedded text inside a frame, inserted between the two columns.	Fig. 45
Chinese magazine	2	An image in the right column.	Fig. 46
French letter	1	Date, paragraphs, footnote, etc.	Fig. 47
French memo	1	A title and text in reverse video.	Fig. 48
French scientific article	4	Embedded large image between 2 columns and a big title.	Fig. 49 Fig. 50
Russian book	1	A black area at the right and bottom the lines are a bit tilted.	Fig. 51
Spanish scientific article	4	3 images at different places.	Fig. 52
Spanish magazine	2	A large image at the left column.	Fig. 53
Swedish newspaper	3	Embedded text block between 2 columns	Fig. 54
Swedish book	1	A large black area which occupied almost half of the page.	Fig. 55

Table 2: Document types and their layout descriptions.

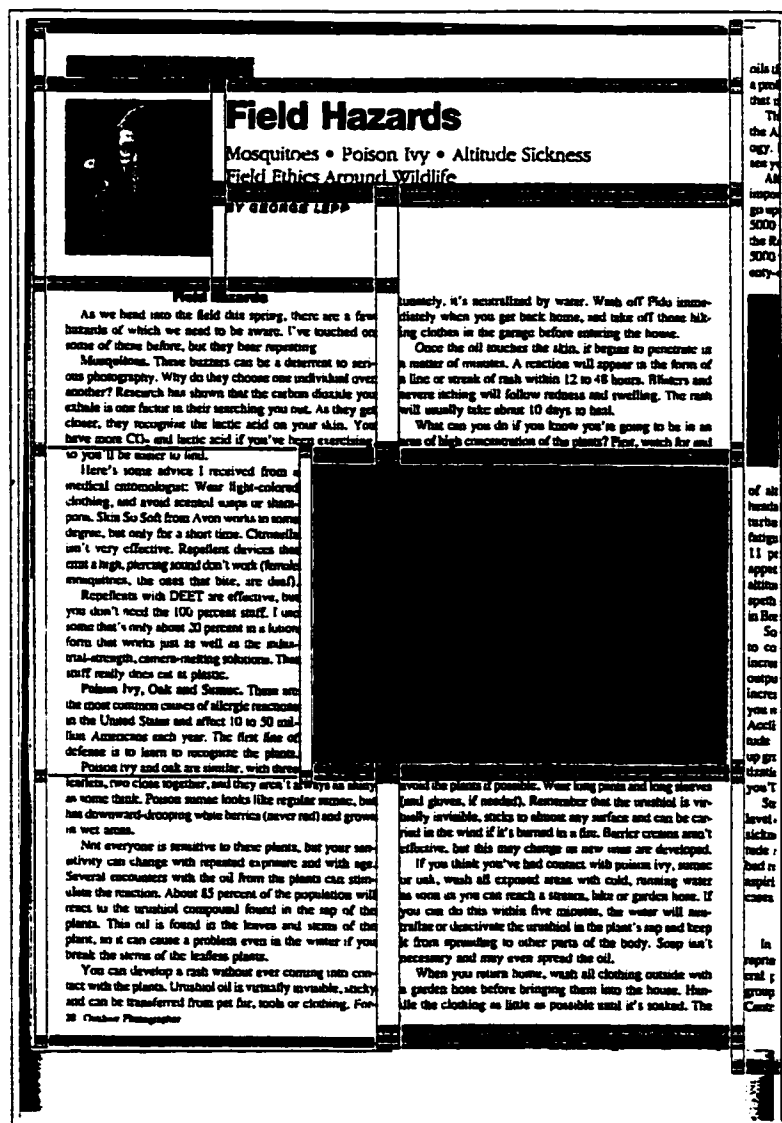


Figure 23: Vertical, horizontal streams and nodes. The geometric structure of this figure can be divided into two relationships: vertical-to-vertical and horizontal-to-horizontal. For example, adjacent nodes on the same vertical stream correspond to adjacent blocks that are related vertically on the same text column, whereas adjacent nodes on two different vertical streams are related horizontally. Each list of nodes has blocks which is in the order from upper to lower node. This is the visual structure of the layout, but for logical structure, a number of generic rules should be extracted from the content of each block and thus we could transform the visual structure into the logical structure.

### 3.5 Comparative Analysis

In this section, we compare our method with the other methods described in section 3.1. This helps the reader to understand the new solution we are proposing, and how we tried to solve the challenging problems that confronted early systems in the field. We will also discuss the advantages and the disadvantages of our approach.

The most important feature of our method is that it is independent of the spacing between columns. Therefore, it does not require the setting of a threshold for wide gaps. Instead, the vertical edges are created by scanning between two successive diagonal scanlines and verifying whether a white area exists (see step 2, page 24). This differs from the method [32] that uses run length smoothing with some predetermined value. This also differs from the XY-tree method [33], where projection profiles are computed along vertical and horizontal directions and the cut between the blocks is made if the gap width is greater than some reasonable threshold. Note that parameter independency is an important feature in segmenting a page in which large size title text has longer inter-character distances than the inter-column distances in the text body. Fig. 41 shows a good performance in the presence of large spacings in the title, with no oversplitting occurrence.

Another distinguishing feature of our method is that it works very well with embedded texts or graphics that are surrounded by a vertical and horizontal white space (see Fig. 49 and Fig. 50), unlike the X-Y cut method where nested tables or text cannot be segmented. Also, it is capable of extracting a text block from within a frame (see Fig. 45). Moreover, since we are using the background as the basic unit, the method becomes independent of text font size and the language of the document. It is also fast, because the diagonal scanlines do not pass through all the image pixels to detect the white space<sup>3</sup>.

Some limitations of our segmentation algorithm should be mentioned. First, the method is relatively sensitive to noise which might randomly be distributed along the gap between the columns. But this is usually not a big problem since we are cleaning the image before the gap is detected. Second, errors occur in documents containing images or texts embedded in a circular way between two columns, or when the vertical stream is not straight but curved (see Fig. 24). For this problem,

---

<sup>3</sup>Thus it solved the problem of white spacing between two descendant (ascendant) letters in English, which yields false column interval (see [2]).

we propose the following method: merge the black pixels of the embedded text using the RLSA<sup>4</sup>, extract the contour of the connected black pixels and then compute the convex hull of points that lie on the contour. In this way, if the block is composed of more than four points, we will have the flexibility of representing a block of different shapes such as rectangles or circles.

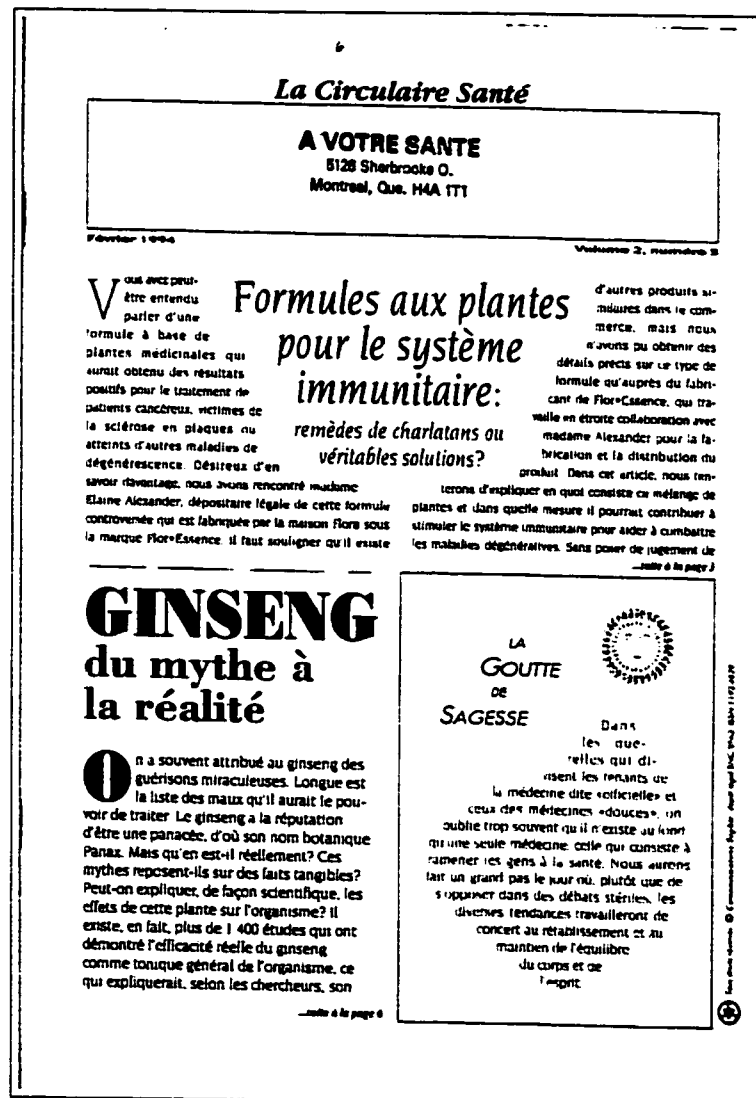


Figure 24: Example of a document image where the method fails to segment the text which is embedded in a circular way.

<sup>4</sup>Note that the smoothing parameter values should be automatically calculated.



# Chapter 4

## Block Identification

### 4.1 Introduction

In the previous chapter, we have determined the layout structure of a document. The binary image has been partitioned into regions, each of which is referred to as a block. The next stage is to label those blocks as either text or graphics. The main reason for this process is that since most documents contain graphics and images in addition to text, image understanding systems such as language or script identifications and OCR engines can not be applied to the graphics regions. So, labeling each region is an essential and a prerequisite step for any further treatment to be applied to each block. In section 4.2, we will start by identifying the regions into texts and graphics, and then in section 4.3, we will identify the text as one of the three script types: Roman, Ideographic or Arabic. Those graphic blocks may be further analysed, but for the scope of this thesis, we keep them for future work.

### 4.2 Text/Graphics Identification

Several algorithms for text/graphics separation have been reported in the literature [35, 39], and [15]. Each of those techniques has its own strengths and weaknesses, but the main difficulty lies in the type of document under study as, for example, when a piece of text lies within predominantly graphics regions, such as a text enclosed in a frame or a table, or when the font style and size of characters change, such as in a title, headings and footnotes. These types of layout may turn the algorithms sensitive

and thus lead to misclassification of text as graphics.

Let us have a look at the different approaches for Text/Graphics separation.

Faure's approach [35] is based on the combination of perception and symbol reading, which are the two processes involved when humans detect the organization of a document. The Text/Graphics segmentation is performed during the association of the OCR text lines (TLs) with the connected components of the filtered image which are called: image lines (ILs). For example, the TLs starting with the sequence of symbols "Fig" and "Tab", capitalized or not, are detected and labelled Caption.

Tan and Yuan [36] use multiple agents working on a pyramid structure to separate texts from graphics in the image. Text strings appear as different groupings of connected components at different resolutions of the images. As such, the pyramid structure, which is a multi-resolution image representation, provides a natural means of identifying and grouping of character strings in the document at different levels of resolution.

Jain and Yu [37] use the connected components analysis – they regard a text region as a regular array of small connected components and a nontext region such as table, halftone image, and drawing as composed of large connected components. They classify a GTL (Generalized Text Line) into the text category if its height  $H$  is (a) less than a threshold and the connected components in it are horizontally aligned, or (b) its width  $W$  is greater than a value  $T_{tw}$  and the connected components in it have roughly the same height.

Belaïd and Akindele [38] do not assume that the blocks are already segmented, nor that they contain homogeneous data. The block labeling method is based on connected component analysis to label the block content as small letter text, medium letter text, large letter text, graphics or photograph. For each set of connected components, they study the classes of spaces between them, as well as their sizes and regularity.

Jain *et al*'s method [39] is based on a multiple channel filtering approach to texture segmentation. The text is considered as a textured region. Nontext contents, such as graphics and pictures, are considered as regions with different textures. Two-dimensional Gabor filters are used to extract texture features for each of these regions.

Fletcher and Kasturi [15] describe the development and implementation of a new algorithm for automatic text string separation which is relatively independent of

changes in text font style and size, and of string orientation. The principal components of the algorithm are the generation of connected components and the application of the Hough transform in order to group together components into logical character strings which may then be separated from the graphics.

#### 4.2.1 Our Text/Graphics Algorithm

Our text/graphics identification approach is based on the projection of the bounding boxes of connected components and their geometric configurations. Since the text region can be regarded as a set of small bounding boxes that are regular in height and are usually aligned horizontally or vertically, we found of particular interest to project those boxes onto the vertical or horizontal lines. As such, this projection creates an array of peaks and valleys at regular intervals. On the other hand, a non-text region such as halftone image and drawing is irregular, the size varies from small to very large components, and they are rarely aligned horizontally or vertically. Therefore the projection of these boxes does not present a regularity between the peaks and the gaps. Based on these observations, the projection could be considered as one feature to differentiate between the graph-like and text-like components. We implement this algorithm by dividing the process into six steps as follows:

1. **Generate the bounding boxes:** Since this step has been used and described in the preprocessing phase in chapter 2, we do not go through it here again. The result of this algorithm can be seen in the graphical representation of the bounding boxes in Figure 5, chapter 2.
2. **Group the block bounding boxes:** In order to group the bounding boxes into the newly extracted blocks, we are using a simple function which consists of connecting the linked list of bounding boxes to the linked list of blocks. This is done by first creating a pointer from each block, and then we trace the list of boxes to examine whether each box falls within the block area. If yes, the pointer of the corresponding blocks will be pointing to the current box. At the end, each block will have a pointer that points to the list of boxes that are inside it. We describe the pseudocode in the algorithm below and the result is illustrated in Figure 25.

```

from the first box to the last box {
  from the first block to the last block {
    if ( (top left corner of a box is less
          than the top left corner of a block) AND
          (bottom right corner of a box is greater
            than the bottom right corner of a block) )
      Insert the box into the block;
    }
  }
}

```

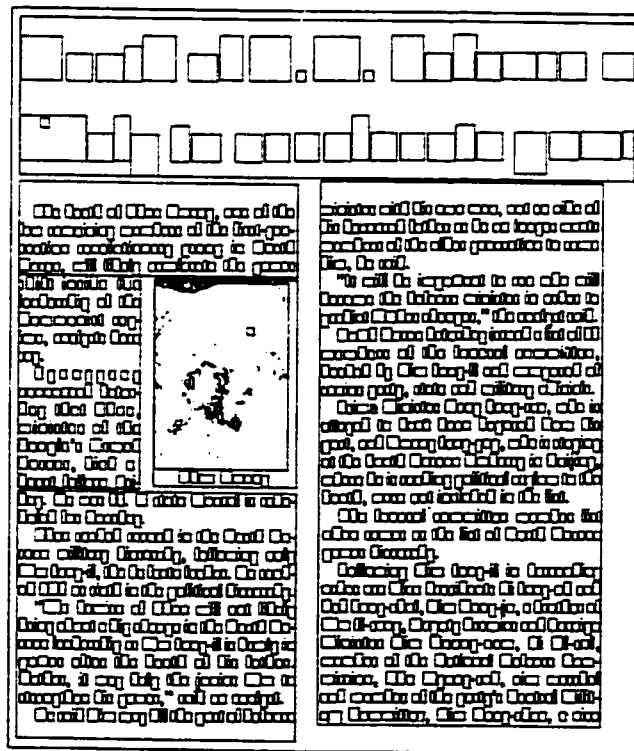


Figure 25: The result of grouping the bounding boxes into the segmented regions.

3. **Project the bounding boxes:** After grouping the bounding boxes into each block, we project the boxes horizontally. The horizontal projections can be formulated as follows: Suppose that we are given a set of bounding boxes  $B = \{b_1, b_2, \dots, b_N\}$ , where each  $b_i$  encloses the region  $R_{b_i}$  where:

$$R_{b_i} = \{(x, y) | x_{min} \leq x \leq x_{max} \text{ and } y_{min} \leq y \leq y_{max}\} \text{ for } i = 1, \dots, N.$$

$x_{min}, y_{min}, x_{max}, y_{max}$  are the boundaries of each bounding box.

The horizontal projection of bounding box  $b_i$  is the association of  $b_i$  with the scalar function  $H_{b_i}$  which is called the height function of  $b_i$  and is defined by:

$$H_{b_i} = \begin{cases} 1 & \text{if } y_{min} \leq y \leq y_{max} \\ 0 & \text{otherwise} \end{cases}$$

So the horizontal projection of  $B$  is a function  $H_B$  which is defined by the sum of the horizontal projection of individual bounding boxes as follows:

$$H_B = \sum_{i=1}^N H_{b_i}$$

Figure 26 shows the result of the projection of bounding boxes for each block.

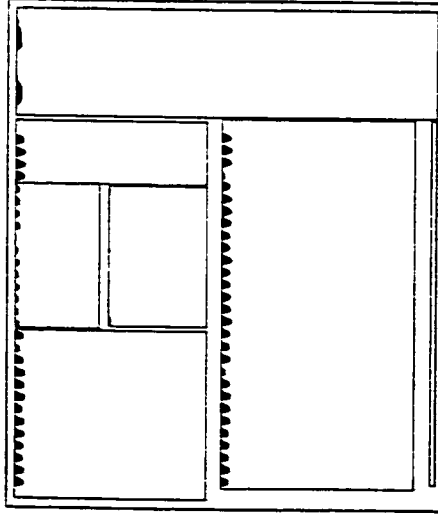


Figure 26: The horizontal projection profile of bounding boxes.

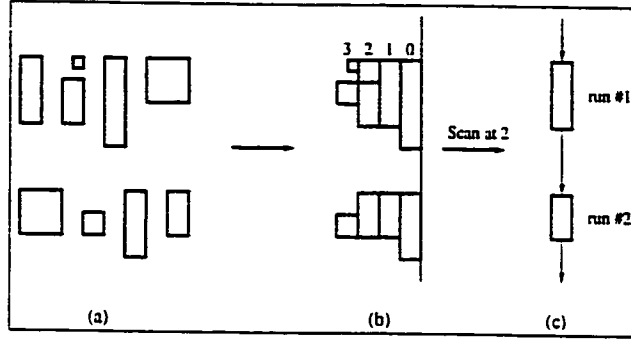


Figure 27: a) Bounding boxes, b) projection profile and c) array of black runs.

4. **Find Black Runs of Projection Profile:** From the previous step, we obtained the box projection for each block. In other words, each projection now corresponds to an array of numbers which represent the number of bounding boxes. In this step, we divide the array into multiple black runs. We do this by scanning the projection at a distance far from the border and accumulate every connected black pixel into one run. The list of the algorithm is shown in the pseudocode below.

```

for (every block)
  for (first pixel in the array to the last pixel)
    if (the pixel is black)
      create a black run.
    if (the pixel is black and connected to the previous one)
      increment the size of the current black run.
    else
      create a new black run.
  }
}

```

This algorithm results in creating an array of black runs for each block (see Figure 27).

5. **Find Average and Standard Deviation:** In order to label each block, we calculate its average run and its standard deviation as follows.

$$average_{run} = \frac{1}{sum_{run}} \sum_{i=0}^{height_{run}-1} run_i \quad (1)$$

$$std_{run} = \sqrt{\frac{(height_{run} - average_{run})^2}{sum_{run}}} \quad (2)$$

6. **Label the blocks:** Based on the height and the sum of runs, as well as its standard deviation, we decide whether the block is text or graphics. Since text regions are characterized by regular size, and graphics are irregular, the standard deviation can be used to measure the irregularity of the height runs in each block. For example, a block with high standard deviation tends to be graphics because of the large variation in the height of runs. Note that graphics components seem to create large runs. Moreover, we have compared the number of runs within each block to the height of the block. For example a block with height equal to 20 and number of runs equal to 20 is far from being a text; Otherwise we will have a line of 1 pixel height, which is not realistic. The illustration below shows a sample result of the algorithm.

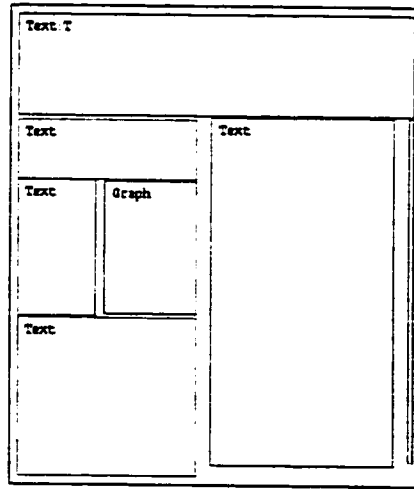


Figure 28: Each block has been labeled as either text or graph.

### 4.2.2 Experimental Results

The system has been tested on 100 isolated blocks, which range from graphics to texts written in different languages (see Appendix). The test set includes blocks typed entirely in big font, as shown in Figure 41. The system classification rates are 93% correct, 3% incorrect and 4% reject. Our classification process is based on visual survey. We compare the blocks that are automatically marked as text or graphic with the blocks of the original document. If they match, we classify as correct. If they do not match (*e.g.* when the result is “text” and the original block is “graphic”), we classify as incorrect. The long thin blocks (horizontal or vertical) have been rejected because they contain a low number of characters in each line (see Fig. 41, where the thin vertical block is not labelled). Most errors occur when the graphic overlaps or touches with text. Fig. 46 illustrates an error where the right block has been misclassified as text. We examined this document and found that this block has a standard deviation of 8.37, which is below the threshold value of 9. This threshold value works well when the block contains graphic and few text lines. However, in this document, it seems that there are many text lines below the image, which reduces its standard deviation to below the threshold. In order to fix this problem, a new heuristic might be added or the threshold value could be adjusted. Note that the evaluation is done manually, but it is necessary to develop some automatic measurement in the future.

Our method shares the connected components analysis with some approaches mentioned above, but it differs in using the horizontal projection of the bounding boxes, which has a lot of beneficial aspects. For example, one could know how many boxes are on the same line. Also, since the labelling is based on the runs of the projection profile, our method proved to be not sensitive to broken characters and page orientation. Also, it is not affected by character densities, slanted characters or large fonts.



## 4.3 Script Identification

In the previous section, we have separated the blocks into two categories: text and graphics. In this section, we proceed with the script<sup>1</sup> identification to be applied only on the blocks which are labelled as “text”. More specifically, we are going to classify the text blocks into three scripts: Roman, Ideographic, and Arabic. Some of the languages that are based on Roman script are: English, French, Italian, German, and Spanish, whereas, the Ideographic script is the set of Chinese-derived characters used in Chinese, Japanese, and Korean languages. On the other hand, the Arabic script is used by numerous languages such as Kurdish, Persian, Pashto, Urdu and Malay.

Script identification has potential applications in complementing the optical character recognition system, retrieving information, indexing of the contents of digital libraries and archiving document image. In the last few years, many papers have been published on the topic and each one attacks the problem in a different way.

Spitz [40], for example, proposes the vertical distribution of upward concavities to distinguish between Asian and European scripts. The positions of these concavities may be derived from the individual runs that comprise the connected components.

Hochberg *et al.* [41] use a cluster-based template. They developed a set of representative symbols (templates) for each script, and for each new document the system compares a subset of symbols from the document to each script’s template; the best match yields the script identification.

Sibun *et al.* [11] examine the use of relative entropy between two probability distributions. A probability distribution is generated by counting particular events, such as character unigrams, bigrams, or trigrams. A test set is assigned to the language which minimizes the relative entropy between the probability distribution of the test set and the distribution associated with the training set for that language. This technique was applied to two types of data: on-line character-coded data to discriminate 27 languages and data derived from documents images to discriminate English, French and German.

Wood *et al.* [42] describe an approach to identify European, Russian, Arabic, Chinese, and Korean text samples taken from facsimile transmissions. The algorithm presented is based on the Hough Transform and the standard morphological filtering

---

<sup>1</sup>Script refers to graphical instantiations of writing systems

operations applied prior to horizontal text line projection to emphasize language-specific features. They also describe the characteristics of the horizontal profile for text written in Roman, Chinese, and Arabic scripts. The method does not require character segmentation and is insensitive to point size.

Peake and Tan [43] observe that a block of text written in any language can clearly be seen as a texture; different scripts produce different textures. They also use both Gabor filters and grey level co-occurrence matrices in independent experiments to extract texture features. Gabor filters proved to be far more accurate than co-occurrence matrices used by Haralick [44].

Lee *et al.* [45] discriminate between Asian and Latin scripts by using a variety of decision rules based on the position of concavity features (as defined in Spitz [40]) and optical density distribution of character height. They assume that the bulk of text is in “Manhattan” layout, that is, with text lines running either horizontally or vertically (after skew and shear correction).

Ding *et al.* [46] present a method which uses several discriminating features based on the horizontal projection and height distribution of connected components to separate between Oriental and European scripts.

With the above overview on the literature, let us describe our script identification method in the next section.

### 4.3.1 Our script Identification Algorithm

Our script identification method [47] uses a set of statistical features extracted from both the horizontal projection of text lines and the bounding boxes of connected components. These statistical features emanate from the distinctive graphical appearance of each script under consideration. In the following sections, we will describe how these features are extracted and how decision rules are generated to decide the script identification.

#### Feature Extraction

For feature extraction, we first highlight the difference in graphical appearance between the three scripts and then demonstrate how the horizontal projection and the bounding boxes reflect the distinctive features extracted from both of them. For example, since the Arabic script is cursive in style (most letters are joined at the

baseline), projecting the line horizontally creates one prominent peak at that level (see Figure 29, third projection). Or in Roman script, since the lower-case letters are usually written between the baseline and the x-height, the height distribution of bounding boxes of these letters also shows one dominant peak (see Figure 31, first distribution). In Ideographic script, since each character is composed of strokes, the horizontal projection shows a random distribution of the number of peaks (see Figure 29, second projection). Details will be described in the following sections.

### The Horizontal Projection

We project each scan line horizontally to get the number of black pixels in the line. We extract the number of peaks, their lengths, heights and positions, by comparing the signs of the successive elements of the projection. For instance, given three neighboring elements  $x_i$ ,  $x_j$  and  $x_k$ , we measure the differences  $d_i = x_i - x_j$  and  $d_j = x_j - x_k$ , and compare the sign of  $d_i$  with the sign of  $d_j$ . This allows us to determine whether the length is increasing or decreasing. Accordingly, a peak vector is created, as shown in each row of Table 3. ‘Steep-up’, a parameter referred to subsequently, is the sum of adjacent positive  $d_i$ ’s; it represents the height of a peak with respect to the previous relative minimum, and is used in both horizontal projections and box height distributions (*v.i.*).

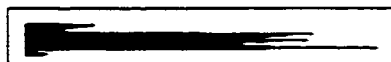


Figure 29: A horizontal projection for a line in Roman script.

Peak	Length	Position
1	341	11
2	334	18
3	418	27

Table 3: The table shows the quantification result of the above line projection. Each row corresponds to a peak, showing its length and position from the top of the line. Note that the two tiny peaks on the left were not considered.

The idea behind using the horizontal projection is to extract the distinguishing characteristics, such as the baseline and the x-height line in Roman script, or the



## The Bounding Boxes

Features are also extracted from the bounding boxes at both page and line levels. At the page level, we use the height distribution of the bounding boxes. Figure 31 and Figure 32 illustrate the distribution of bounding box heights in documents sampled from the three script classes. The Roman script shows a significant and steep peak in the middle, and two smaller peaks on both sides. Since the Roman script has consistent box heights which can be segmented into a small number of distinct class codes [48], the class codes fall into a small set of box heights corresponding to peaks in the diagram. For example, it is obvious that the highest peak shown in the figure corresponds to characters of the height of an 'x', and the 2 peaks on its right correspond to characters of the height of a 't' and an 'A'. The distribution in the Chinese document shows the major peak at the right with hardly any larger boxes in the document, whereas in the Arabic distribution, the highest peak is shown at low values, again with no secondary peak to its right<sup>2</sup>. Most of our analysis is focused on the location along the horizontal axis of MAX1<sup>3</sup> and MAX2<sup>4</sup> (see Figure 31), and their positions relative to the average height of the boxes.

The box height distribution has been dynamically determined during the extraction stage of connected components. For this, the box height distribution is not enough to identify the script because all the boxes have been taken at one shot regardless of their relationships. The distribution can not indicate whether these boxes are set in sequence of lines or randomly distributed. Also, a page might contain boxes which do not represent a text line. For this, we segment the page into lines and proceed by computing the average above-box<sup>5</sup> as follows: horizontally scan the boxes of a line at the middle of the line, stop at each box and add the distances between the top-line and top-box (see Figure 33), then divide the sum by the line height and the number of boxes. A low average above-box indicates that the heights of boxes are close to the line height. This occurs when the document contains capital letters or is Ideographic, as shown in the second line of Figure 34.

---

<sup>2</sup>The pronounced peaks at very low values ( $< 5$  pixels) are considered noise and are not taken into consideration

<sup>3</sup>The highest peak

<sup>4</sup>The second highest peak

<sup>5</sup>Above-box is the distance between the top of the bounding box and the top box of the line

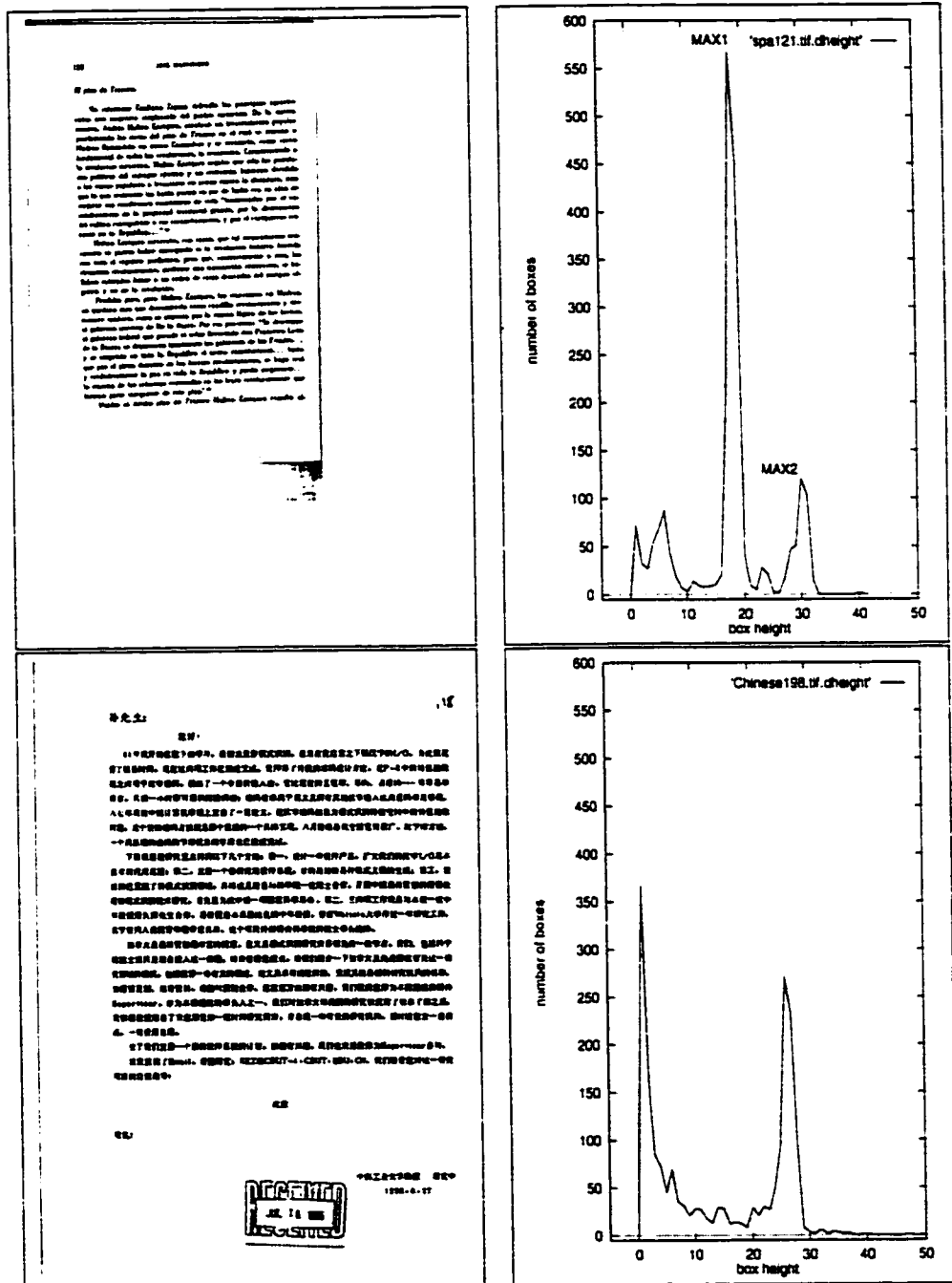


Figure 31: Roman and Ideographic images and their height distributions.

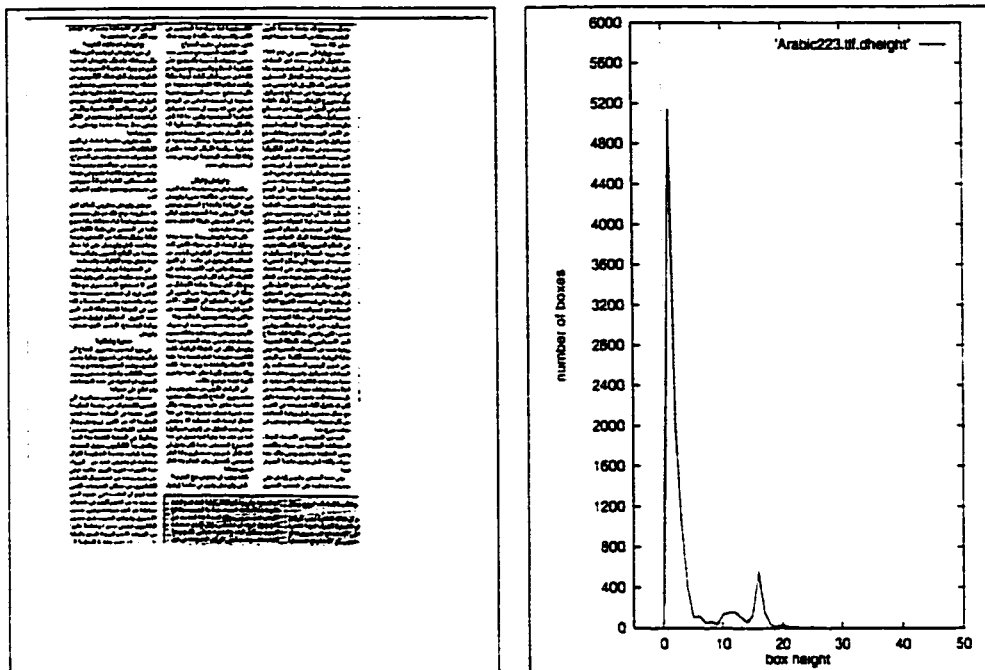


Figure 32: Arabic image and its height distribution.

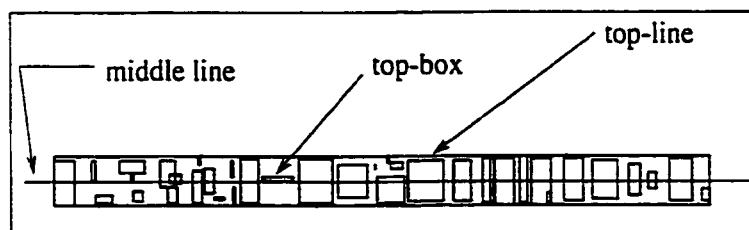


Figure 33: The bounding boxes of a Japanese text line.

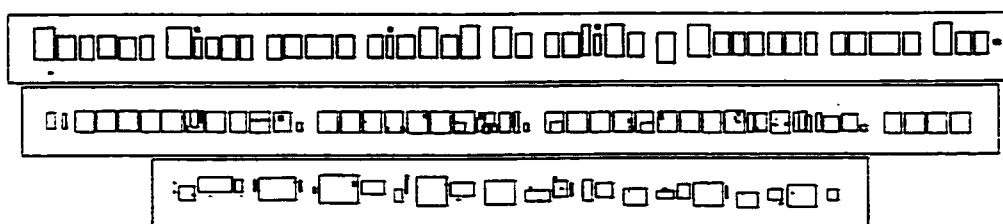


Figure 34: The bounding boxes of Roman, Ideographic and Arabic lines of Figure 30.

### 4.3.2 Generating Decision Rules

The decision to identify the script is based on rules created from the measurements extracted from both the horizontal projection and bounding boxes.

#### Rules to Classify as Roman

- **IF** ( percentage of lines with 2 peaks  $\geq 25\%$  **AND** position of  $MAX1 \geq$  average height **AND** position of  $MAX2 \geq$  average height **AND** position of  $MAX1 <$  position of  $MAX2$  **AND** length of  $MAX1 < 2 \times$  steep-up of  $MAX1$ ) **OR**
- (average top line density of the bounding boxes  $> 52\%$ ) **OR** (percentage of lines with 2 peaks  $\geq 55\%$ ) **AND** percentage of lines with density  $\geq 0.6$  is  $< 80\%$
- **THEN** Script Identity = **Roman**.

#### Rules to classify as Ideographic

- **IF** percentage of lines with density  $\geq 0.6$  is  $> 70\%$  **OR**
- (position of  $MAX1 \geq$  average height **AND** position of  $MAX2 \geq$  average height **AND** percentage of lines with 2 peaks  $\leq 80\%$  **AND** average above-box of the bounding boxes  $\leq 80\%$ ) **OR**
- position of  $MAX1 >$  average height **AND** length of the peak following  $MAX1 \leq$  (length of  $MAX1$ )/20
- **THEN** Script Identity = **Ideographic**.

#### Rules to Classify as Arabic

- **IF** position of  $MAX1 \leq$  average height **OR** position of  $MAX2 \leq$  average height **AND** using the horizontal projection ( $HP$ ), the percentage of one-peak  $HP$  lines is  $> 43\%$  **AND** percentage of lines with density  $\geq 0.6$  is  $< 80\%$  **OR**
- number of peaks = 2 **AND** position of  $MAX1 >$  average height **AND**  $max1Rate \leq 3.5$ , where  $max1Rate =$  percentage of length of  $MAX1 / (\text{number of boxes} \times \text{position of } MAX1)$  **AND** position of  $MAX1 >$  average height **AND** length of  $MAX1 \neq$  steep-up of  $MAX1$
- **THEN** Script Identity = **Arabic**.



In this section, we generated knowledge rules that reflect the geometric characteristics of the three scripts under study. In generating these rules, a number of threshold values were manually selected and adjusted according to the output of the system on tested images. For example, we selected rules that are based on the number of peaks on horizontal projection, line densities and average heights of the connected components. For instance, the line density has been used with threshold value of 70% to classify the script as Ideographic.

Based on these knowledge rules, we are able to discriminate the three scripts at the page and line levels, without any need to identify the letters forming the words, or the words forming the lines.

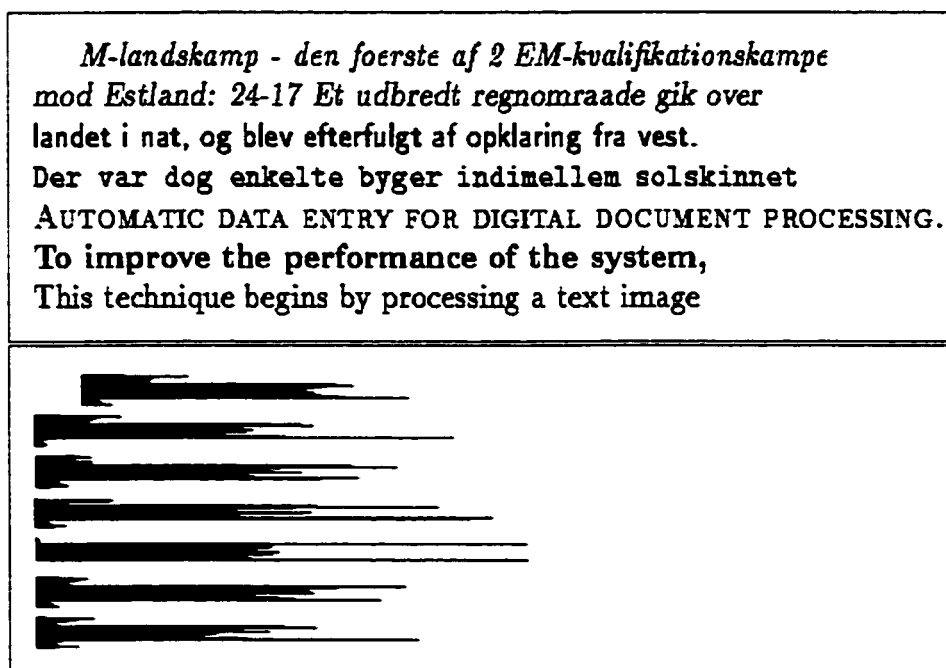


Figure 35: Multifont Roman lines and their horizontal projections. Although the fonts vary, we are still able to see the parallel presence of two dominant peaks which correspond to the baseline and top minuscule line (or x-height line). The type fonts used are *italic Shape*, *Slanted Shape*, Sans Serif, Typewriter, SMALL CAPS SHAPE, Boldface and Roman.



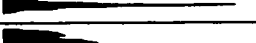


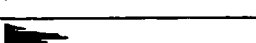


Arabic Script	Horizontal Projection
الحفلات والمسارح الغنائية حتى	
جمال وكمال في خدمة الجالية العربية	
رحب الاتحاد العربي - الكندي في بيان أصدره أمس	
اسعار خاصة للاعراس	
پر اچھوتے ہیں کیا کہ اس بدلے میں سچا پاکستان کرکت ہو گا کہ	
سمرة ياسمرة حلوة ياسمرة	
وَلَمْ يَلْمِزْ أَنْ يَسْتَنْتِ ذَلِكَ الْآخِرُ عَنْ مَوْلَانَا جَلَّ وَعَزَّ، كَيْفَ	
لسمكرة السيارات والدهان	

Figure 36: Multifont Arabic lines and their horizontal projections. The unique prominent peak can clearly be shown at the baseline.

### 4.3.3 Experimental Results

The system has been tested on isolated text blocks from 215 documents (62 Roman, 88 Ideographic, and 65 Arabic) which cover a wide range of styles and qualities at different resolutions. The documents are scanned at 200 and 300 dpi, from business letters/memos, newspapers/magazines, advertisements, flyers, faxes and other sources. The quality of the image ranges from good to poor (i.e., light text or many touching and broken characters). The test set also includes documents typed entirely in uppercase letters and multifont texts, as shown in Figures 35 and 36. The following table presents the experimental results. As shown, 6.5% of those documents were incorrectly classified and 93.5% were correctly classified.

Input Language	# of doc	# of Rom	# of Ideo	# of Arab	# of Error	Script Error	Script Correct
Roman	62	59	1	2	3	4.8%	95.2%
Ideographic	88	9	77	2	11	12.5%	87.5%
Arabic	65	0	0	65	0	0.0%	100%
Total	215				14	6.5%	93.5%

Some documents are misclassified as Arabic because they contain lots of dashes which actually do not represent text lines (see Figures 37). These dashes when projected create one peak on which we rely on to identify the document as Arabic.

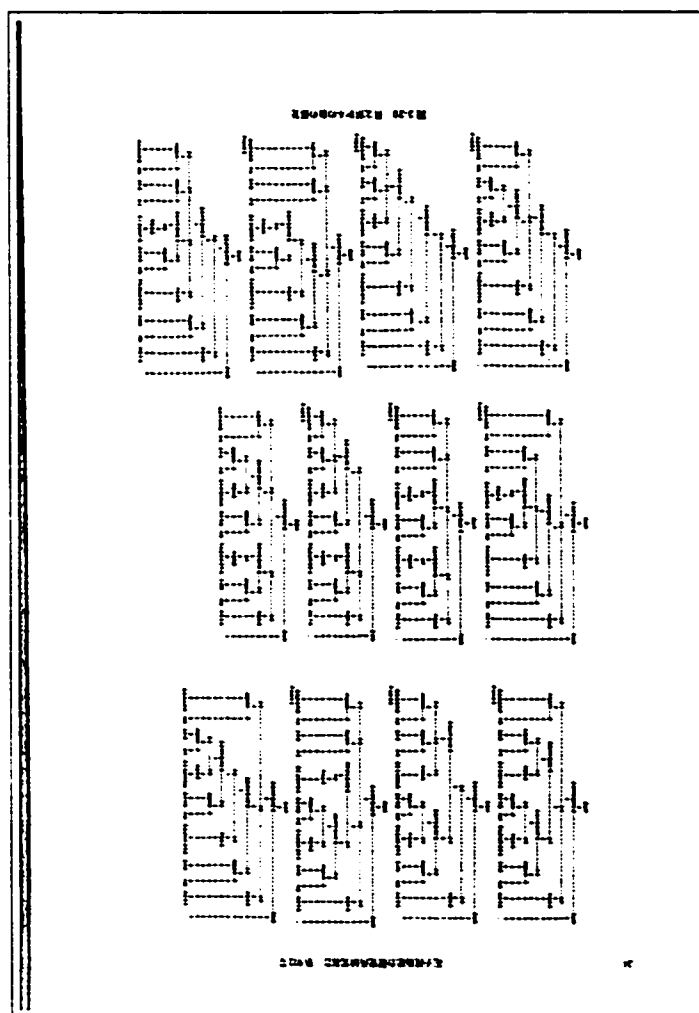


Figure 37: This document is misclassified as Arabic. It is upside down, contains vertical, horizontal dashes, and two Japanese lines.

### 4.3.4 Comparative Results

These results compare well to those reported in the literature. Wood et al. [42] use horizontal projections to distinguish between 5 scripts. They require an additional filtering step before constructing the projections and do not report test results. Ding et al. [46] use very similar techniques to distinguish two scripts. They obtain 99% accuracy rate on documents of somewhat higher quality. They also mentioned that a European document tends to be misclassified as oriental script if the quality of the document is low and many characters are broken. Of the approaches that differ significantly in technique, Hochberg et al. [41] report 96% accuracy in classifying 13 scripts for texts with more than 100 symbols and a reliability threshold of at least 10%. Both Spitz [40] and Lee et al. [45] distinguish only two scripts, and they report 100% accuracy and 98% accuracy respectively. Their system works well within some specific European languages, but not for Cyrillic script in which the upward concavities cluster near the baseline and some positions between the baseline and the x-line.

The table below shows the results of some methods. In our case, although 93.5% recognition rate has been obtained, our method differs from others in several aspects. First, it handles documents in Cyrillic script. Second it deals with complex layouts. Third, it is robust in the presence of noise and broken characters. Finally, no character segmentation is required.

Author	Method	#Script	Rec.(%)	Comment
Wood <i>et al.</i>	Horizontal Projection	5	No test result	Require additional filtering
Ding <i>et al.</i>	Concavity, horizontal projection and height distribution	2	99%	Tested on very high quality images
Hochberg <i>et al.</i>	Cluster Based Template	13	96%	Require character segmentation
Spitz [40]	Vertical Distribution and upward concavity	2	100%	Does not work on Cyrillic script
Lee et al. [45]	Distribution of character height and upward/downward concavity	2	98%	Does not work on Cyrillic script
Waked <i>et al.</i>	Horizontal projection and bounding boxes	3	93.5%	Short lines are not considered

Table 4: Comparative results.

# Chapter 5

## Conclusion

In this thesis, we have presented procedures to segment and identify real-life document images. We divided the system into three major phases: preprocessing, page segmentation and block identification. For each phase, a set of efficient and effective algorithms were proposed, along with experimental results and some comparative analysis.

In the preprocessing phase, we started by filtering noise using the distribution of horizontal black runs from which the most frequently occurring run is extracted. After filtering, the skew angle is detected based on Hough Transform and the most frequent local maximum. Experimental results showed that the method is robust against fluctuation in the text line components.

For page segmentation, we introduced a novel approach which is based on the diagonal scanning and the node-edge orientation. We started extracting nodes from the white runs of 45-degree scanlines, and we extracted the edges by scanning vertically between each pair of nodes. Then the edges are merged to create vertical and horizontal streams, which together define a block surrounded by these streams. Also, a data structure has been introduced to represent the segmented document. Testing the algorithm on different types of documents written in different languages shows comparative and promising results.

Block identification was the last step in our system, in which we isolated the text

and graphic regions using the horizontal projection of the bounding boxes of connected components and their geometric configurations. In this way, we were able to measure the regularity in the setting of text lines, and the irregularity of images or graphic components.

After isolating texts from graphics, we segmented the texts into lines to classify the script type into one of three categories: Roman, Ideographic and Arabic script. For this purpose, both the bounding boxes and the horizontal projections have been exploited, from which we extracted features and generated decision rules to decide the script identification of the document image.

Some directions for further improving the performance of our system can be listed as follows:

- For the preprocessing phase, we want to process the image independent of the skew angle, thus improving the performance and reducing the processing time.
- For page segmentation, we want to optimize the existing algorithms by combining the geometric configurations of the connected components with the vertical and horizontal streams. Also, we want to try to segment block regions that are not necessarily rectangular in shape.
- For script identification, a better result could be attained by identifying the script at the block level even with few text lines.

Finally, we can further extend our work to include the following:

- Identify the type of documents as letter, article, scientific journal, etc.
- Label each text block such as title, abstract, footnote, caption or references.
- Distinguish between printed and handwritten document image.
- Label the graphic regions such as logo, form, table, or signature.

From our work, we conclude that document segmentation and identification is a major and important component of the complete document understanding system which has significant applications, and there is still much room for improvement in almost every aspect of the system.

# Bibliography

- [1] Martin A. Fischler and Oscar Firschein, "Intelligence: the Eye, the Brain, and the Computer," Addison-Wesley, Reading MA., 1987.
- [2] Theo Pavlidis and Jiangying Zhou, "Page segmentation and classification," *Document Image Analysis*, Vol 2, 1995, pp. 226-238.
- [3] Venu Govindaraju, Sargur N. Srihari, and David B. Sher, "Computation model for face location," *Proceedings of the Third International Conference on Computer Vision*, ICCV-90, Japan, pp. 718-721, 1990.
- [4] D. Neuhoff and T. Pappas, "Perceptual coding of images for halftone display," *IEEE Trans. Image Process*, 3, 1-13, 1994.
- [5] M.S. Krishnamoorthy, G. Nagy, S. Seth, M. Viswanathan, "Syntactic segmentation and labeling of digitized pages from technical journals," *IEEE Transaction PAMI*, vol 15, 7, 737-747, July 1993.
- [6] R. Hoch and A. Dengel, "Classification the message in printed business letters," *Proc. Second Annual Symposium on Document Analysis and Information Retrieval*, ISRI, Las Vegas, NV, April 1993, pp. 443-456.
- [7] Shin'ichi Satoh, Atsuhiro Takasu, and Eishi Katsura, "An automated generation of electronic library based on document image understanding," *Proceedings of the Third International Conference on Document Analysis and Recognition*, Montréal, August 1995, pp. 163-166.
- [8] E. Green and M. Krishnamoorthy, "Recognition of tables using table grammars," *Proc. Fourth Annual Symposium on Document Analysis and Information Retrieval*, April 24-26, 1995, Las Vegas, Nevada, pp. 261-277.

- [9] D.J. Ittner, H.S. Baird, "Language-free layout analysis," *Proc. ICDAR93*, pp. 336-340, Tsukuba Science City, 1993.
- [10] A.L. Spitz and M. Ozaki, "Palace: A multilingual document recognition system," *Proc. Document Analysis Systems*, Kaiserslautern, 1994, pp. 59-76.
- [11] P. Sibun and J.C. Reynar, "Language identification: Examining the issues." *In 5th Symposium on Document Analysis and Information Retrieval*, Las Vegas, NV, 1996.
- [12] Anil K. Jain and Bin Yu, "Page segmentation using document model," *Proc. International Conference on Document Analysis and Recognition*, 1997, pp. 34-37.
- [13] Yuan Y. Tang, Ching Y. Suen, Chang De Yan, and Mohamed Cheriet "Financial document processing based on staff line and description language," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 25, No. 5, May 1995.
- [14] D. Drivas and A. Amin, "Page segmentation and classification utilising bottom-up approach," *Proceedings of the Third International Conference on Document Analysis and Recognition*, Montréal, August 1995, pp. 610-614.
- [15] L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 6, November 1988, pp. 910-918.
- [16] A. Amin and S. Fisher, "A Document skew detection method using Hough transform," *Pattern Analysis and Application*, Vol 3, No. 3, 2000, pp. 243-253.
- [17] H. S. Baird, "The Skew angle of printed documents," *Proc. Conf. Society of Photographic Scientists and Engineers*, vol 40, 1987, pp. 21-24.
- [18] S. C. Hinds, J. L. Fisher. and D. P. D 'Amato, "A Document skew detection method using Run-Length encoding and the Hough transform," *Proc. 10th Int. Conf. on Pattern Recognition*, 1990, pp. 464-468.
- [19] Xiaoyi Jiang, Horst Bunke, Dubravka Widmer-Kljajo, "Skew detection of document images by focused Nearest-Neighbor clustering," *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, Bangalore, India, September 1999, pp. 629-632.



- [20] L. O’Gorman, “The Document spectrum for page layout analysis,” *IEEE Transactions on Pattern Analysis and Machine Intelligent*, Vol. 25, pp. 1162-1173, 1993.
- [21] T. Akiyama and N. Hagita, “Automated entry system for printed documents,” *Pattern Recognition*, Vol. 23, pp. 1141-1154, 1990.
- [22] Ming Chen and Xiaoqing Ding, “A Robust skew detection algorithm for grayscale document image,” *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, Bangalore, India, September 1999, pp. 617-620.
- [23] H. Yan, “Skew correction of document images using interline cross-correlation,” *CVGIP Graphical Models and Image Processing*, Vol 55, February 1997, pp. 538-543.
- [24] Avanindra Chaudhuri and Subhasis Chaudhuri, “Robust detection of skew in document images,” *IEEE Transactions On Image Processing*, Vol 6, No. 2, February 1997.
- [25] Ming Chen and Xiaoqing Ding, “A robust skew detection algorithm for grayscale document image,” *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, Bangalore, India, September 1999, pp. 617-620.
- [26] Huiye Ma and Zhenwei Yu, “An enhance skew angle estimation technique for binary document images,” *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, Bangalore, India, September 1999, pp. 165-168.
- [27] B. Yu and A. K. Jain, “A robust and fast skew detection algorithm for generic documents,” *Pattern Recognition*, Vol 29, No. 10, October 1996, 1599-1630.
- [28] D. S. Le, G. R. Thoma and H. Wechsler, “Automated page orientation and skew angle determination for binary document images,” *Pattern Recognition*, Vol 27, No. 10, October 1994, 1325-1344.
- [29] Oleg Okun, Matti Pietikainen and Jaakko Sauvola, “Robust skew estimation on low-resolution document images,” *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, Bangalore, India, September 1999, pp. 621-624.

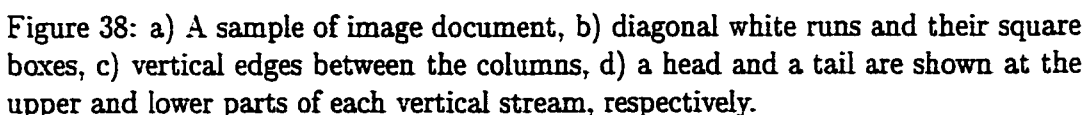
- [30] Chiu L. Yu, Yuan Y. Tang and Ching Y. Suen, "Document skew detection based on the fractal and least squares method," *Proceedings of the Third International Conference on Document Analysis and Recognition*, Montréal, August 1995, pp. 1149-1152.
- [31] R. O. Duda and P. E. Hart, "Use of the Hough transform to detect lines and curves in pictures," *Communications of the Association for Computing Machinery*, Vol 15, 1975, pp. 11-15.
- [32] K. Y. Wong, R. G. Casey and F. M. Wahl, " Document analysis system," *IBM J. Res. Develop.*, Vol. 6, pp. 647-656, 1982.
- [33] G. Nagy, S.C. Seth and S.D. Stoddard. "Document Analysis with Expert System," *Proceedings of Pattern Recognition in Practice II*, Amsterdam, June, 1985.
- [34] Boulos Waked, Ching Suen, Sabine Bergler, "Segmenting document images using diagonal white runs and vertical edges," *Proc. Sixth International Conference on Document Analysis and Recognition*, Seattle, Washington, U.S.A., September 2001 (forthcoming).
- [35] Claudie Faure, "Preattentive reading and selective attention for document image," *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, Bangalore, India, September 1999, pp. 577-580.
- [36] Chew Lim Tan, Bo Yuan, Weihua Huang, Qian Wang, Zheng Zhang, "Text/Graphics separation using agent-based pyramid operations," *Proceedings of the Fifth International Conference on Document Analysis and Recognition*, Bangalore, India, September 1999, pp. 169-172.
- [37] Anil K. Jain and Bin Yu, "Document representation and its application to page decomposition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 3, March 1998, pp. 294-308.
- [38] A. Belaid and O. T. Akindele, "A labeling approach for mixed document blocks," *Proceedings of the Second International Conference on Document Analysis and Recognition*, Tsukuba, Japan, September 1993, pp. 749-752.

- [39] A. K. Jain and S. Bhattacharjee, "Text segmentation using Gabor filters for automatic document processing," *Machine Vision and Applications*, Vol. 5, 1992, pp. 169-184.
- [40] A. L. Spitz. "Determination of the script and language content of document images," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19, No. 3, 1997, pp. 235-245.
- [41] 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*, Vol. 19, No. 2, 1997, pp. 176-181.
- [42] S. L. Wood, X. Yao, K. Krishnamurthi, and L. Dang, "Language identification for printed text independent of segmentation," *Proceedings of IEEE International Conference on Image Processing*, 1995, pp. 428-431.
- [43] G. S. Peake and T. N. Tan, "Script and language identification from document images," *Proc. Third Asian Conference on Computer Vision - ACCV'98*, Hong Kong, China, January 8-10, 1998, pp. 96-104.
- [44] R. M. Haralick, "Statistical and structural approaches to texture," *Proceedings of IEEE*, Vol. 67, 1979, pp. 786-804.
- [45] D.-S. Lee, C. R. Nohl, and H. S. Baird, "Language identification in complex, unoriented, and degraded document images," *Proc. Int. Workshop on Document Analysis Systems*, Malvern, Pennsylvania, 1996, pp. 76-98.
- [46] J. Ding, L. Lam, and C.Y. Suen, "Classification of Oriental and European scripts by using characteristic features," *Proc. Fourth International Conference on Document Analysis and Recognition, ICDAR'97*, Ulm, Germany, August 1997, Vol. 2, pp. 1023-1027.
- [47] 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, pp. 4470-4475.

- [48] P. Sibun and A. L. Spitz, "Language determination: Natural language processing from scanned document images," *Proceedings of the 4th Conference on Applied Natural Language Processing*, Stuttgart Germany, 1994, pp. 15-21.
- [49] C.Y. Suen, S. Bergler, N. Nobile, B. Waked, C. P. Nadal, A. Bloch. "Categorizing document images into script and language classes," In *Proceedings of International conference on Advances in Pattern Recognition (ICAPR98)*, Plymouth, UK, November 1998, 297-306.

# **Appendix A**

## **Examples of document Images**





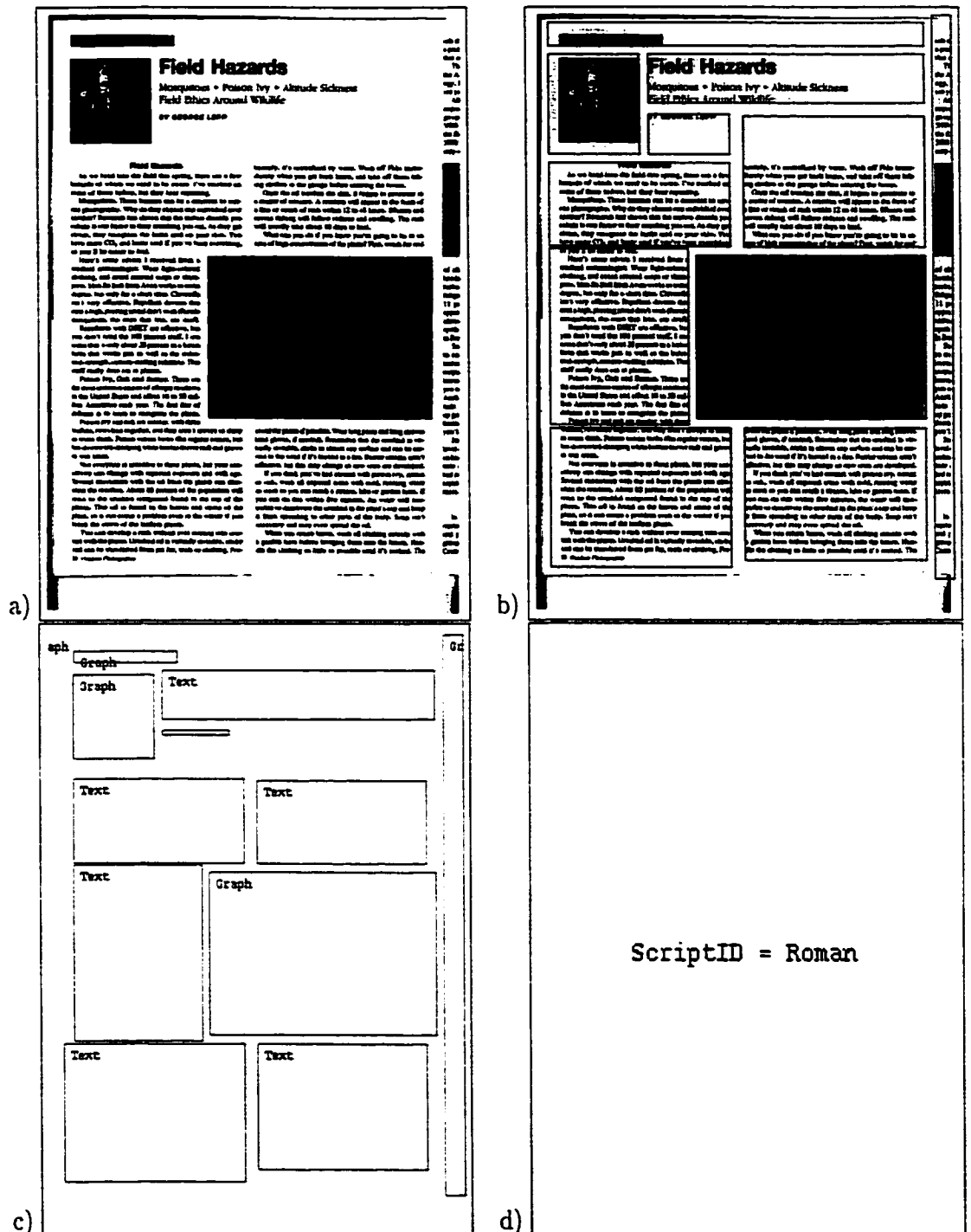


Figure 40: a) English magazine. b) segmented image, c) text/graphics separation, d) script identification.



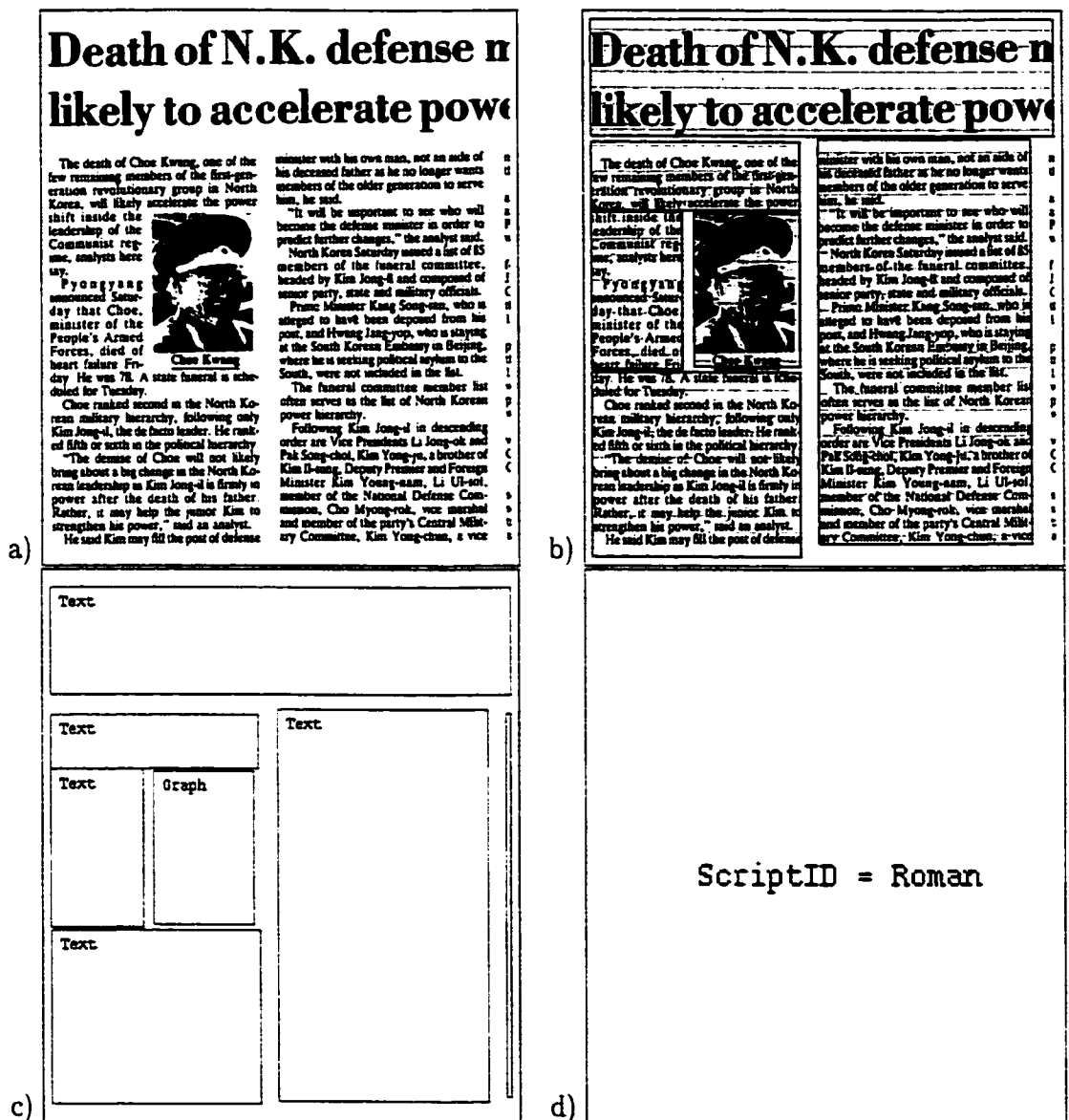


Figure 41: a) An English 2-column page, b) segmented page, c) text/graphics separation, d) script identification.

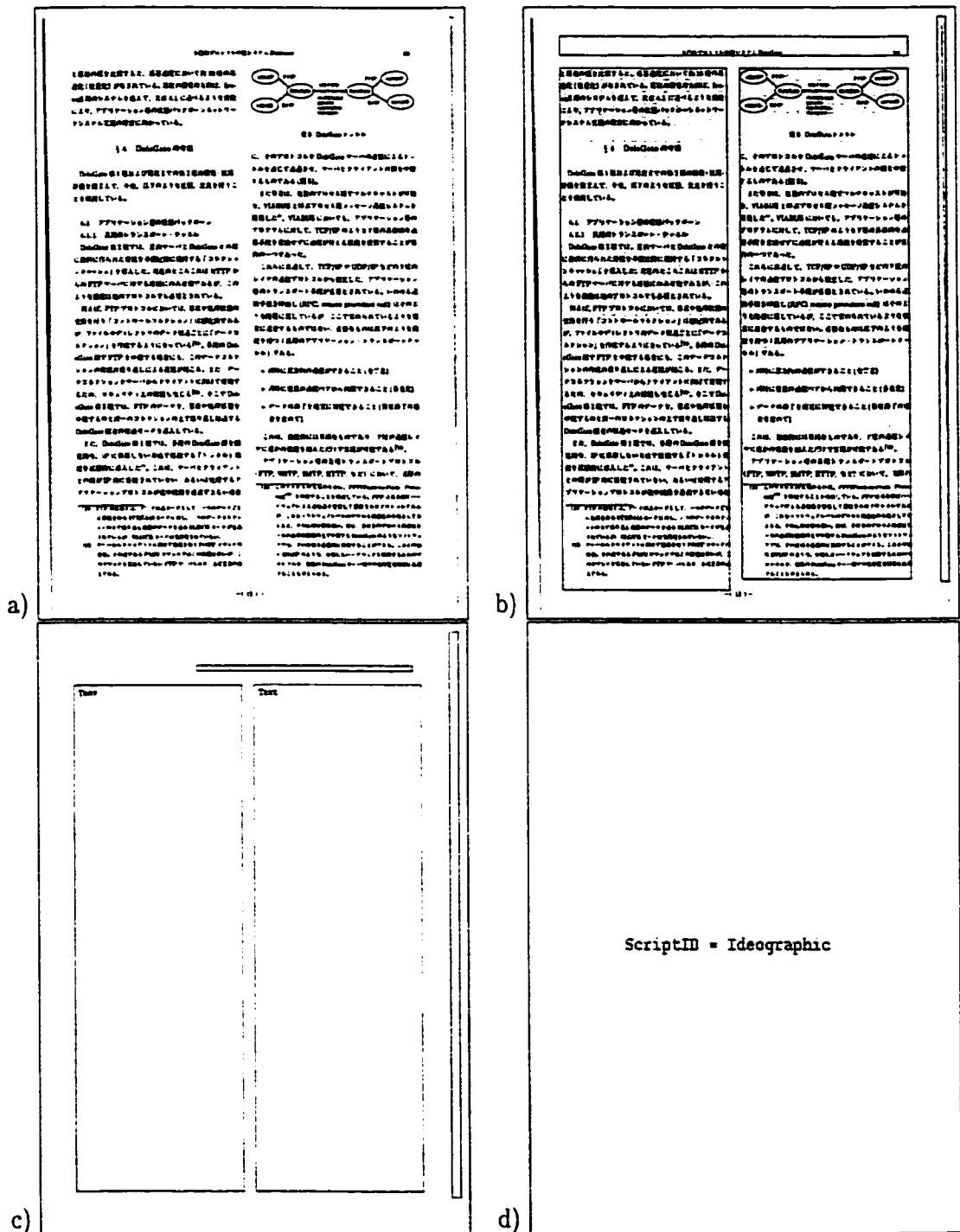


Figure 42: a) A Japanese technical journal, b) segmented page, c) text/graphics separation, d) script identification.

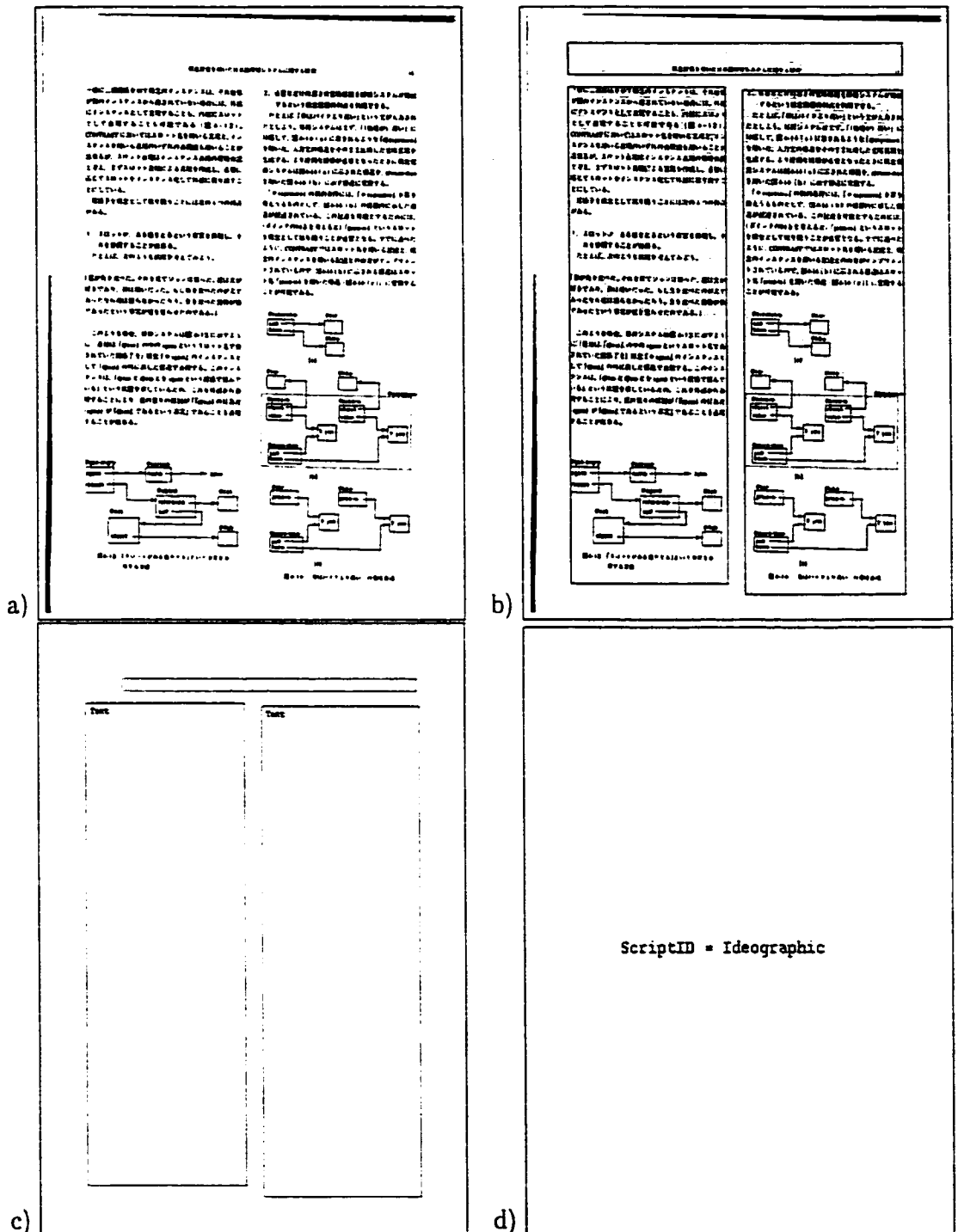


Figure 43: a) A Japanese technical journal, b) segmented page, c) text/graphics separation, d) script identification.



Figure 44: a) Arabic magazine, b) segmented page, c) text/graphics separation, d) script identification.

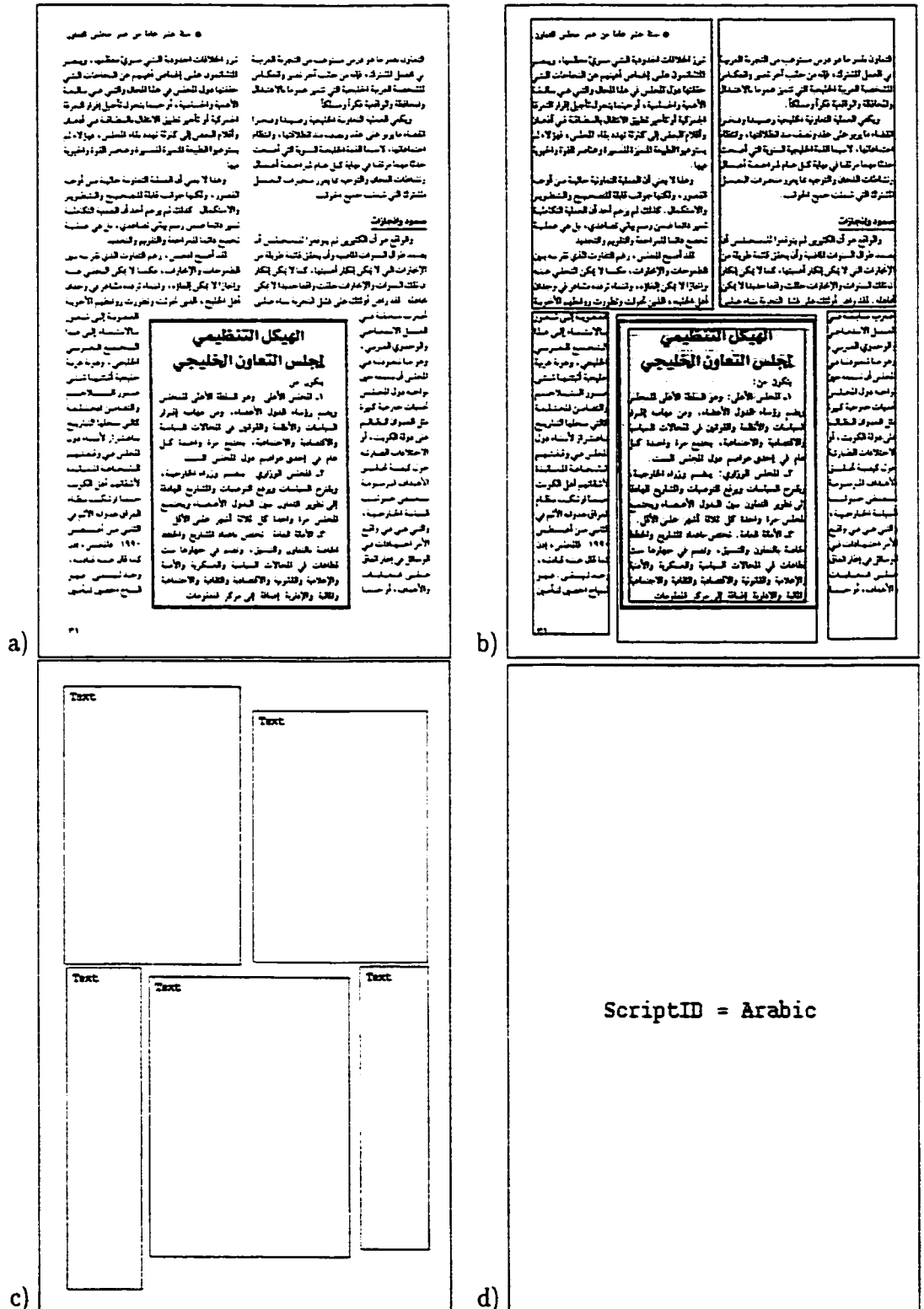


Figure 45: a) Arabic magazine: a text block is embedded inside a frame, b) segmented page, c) text/graphics separation, d) script identification.

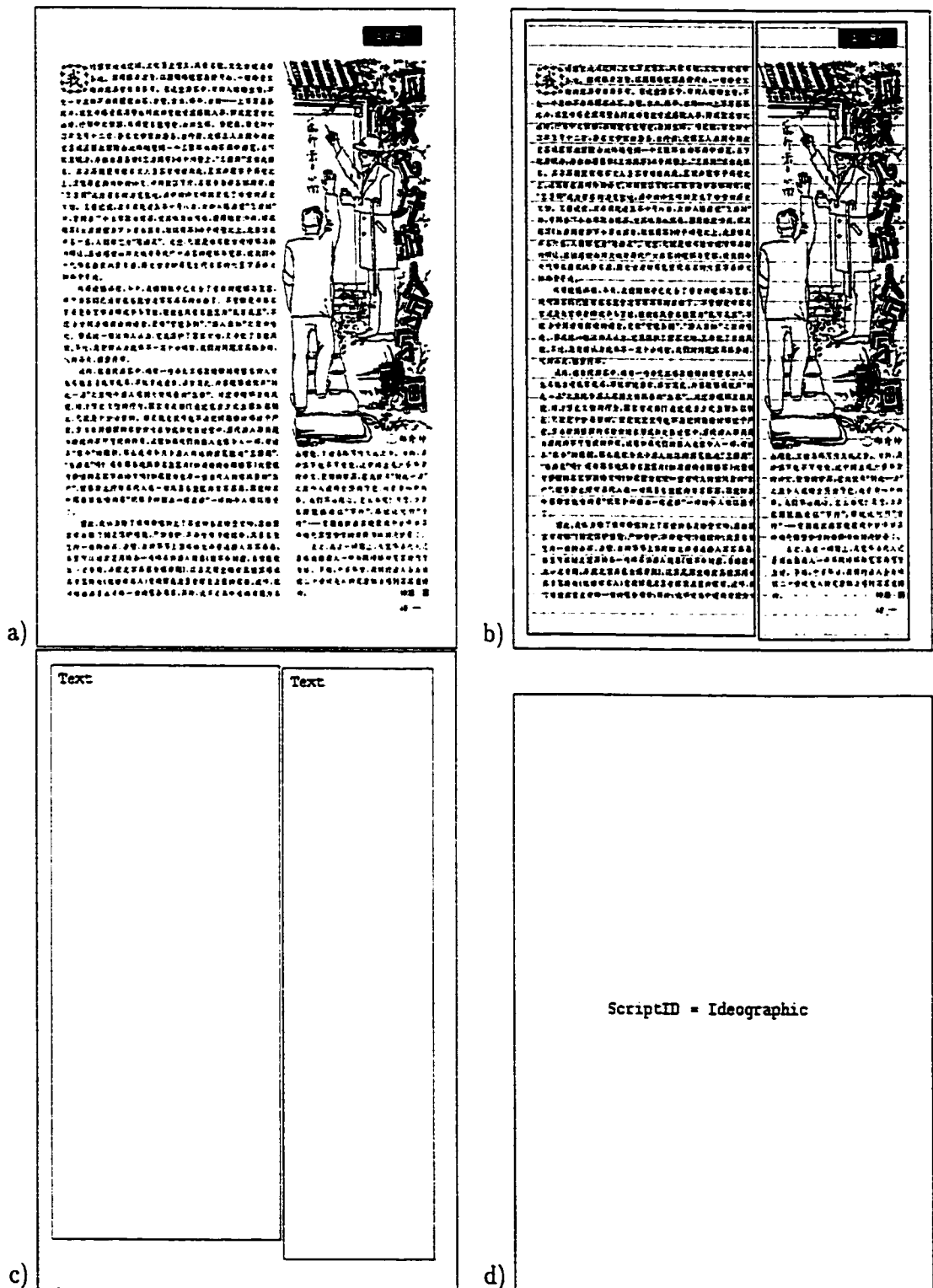


Figure 46: a) Chinese magazine: column with small gap, b) segmented page, c) text/graphics separation, d) script identification.

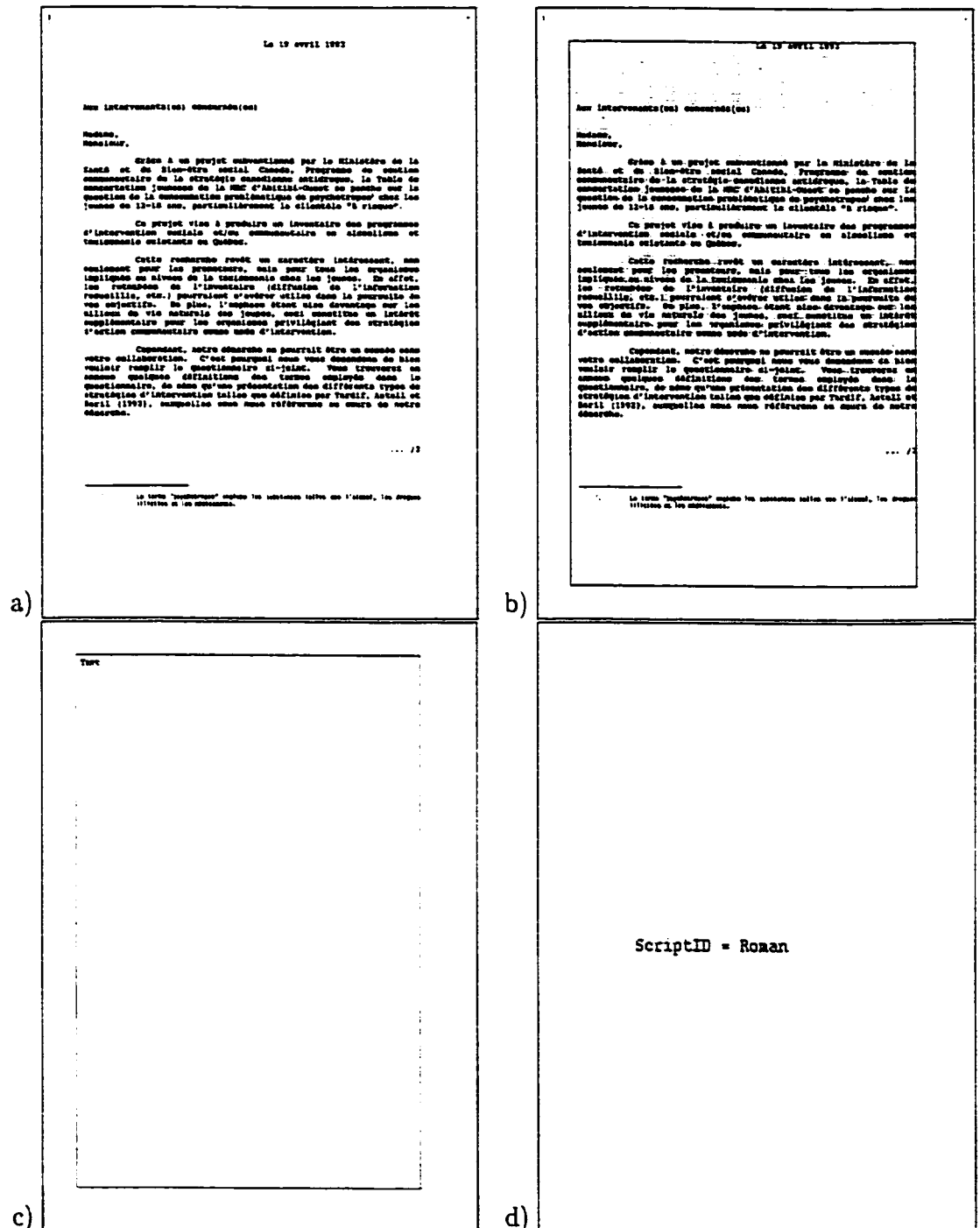


Figure 47: a) French letter, b) segmented page, c) text/graphics separation, d) script identification.

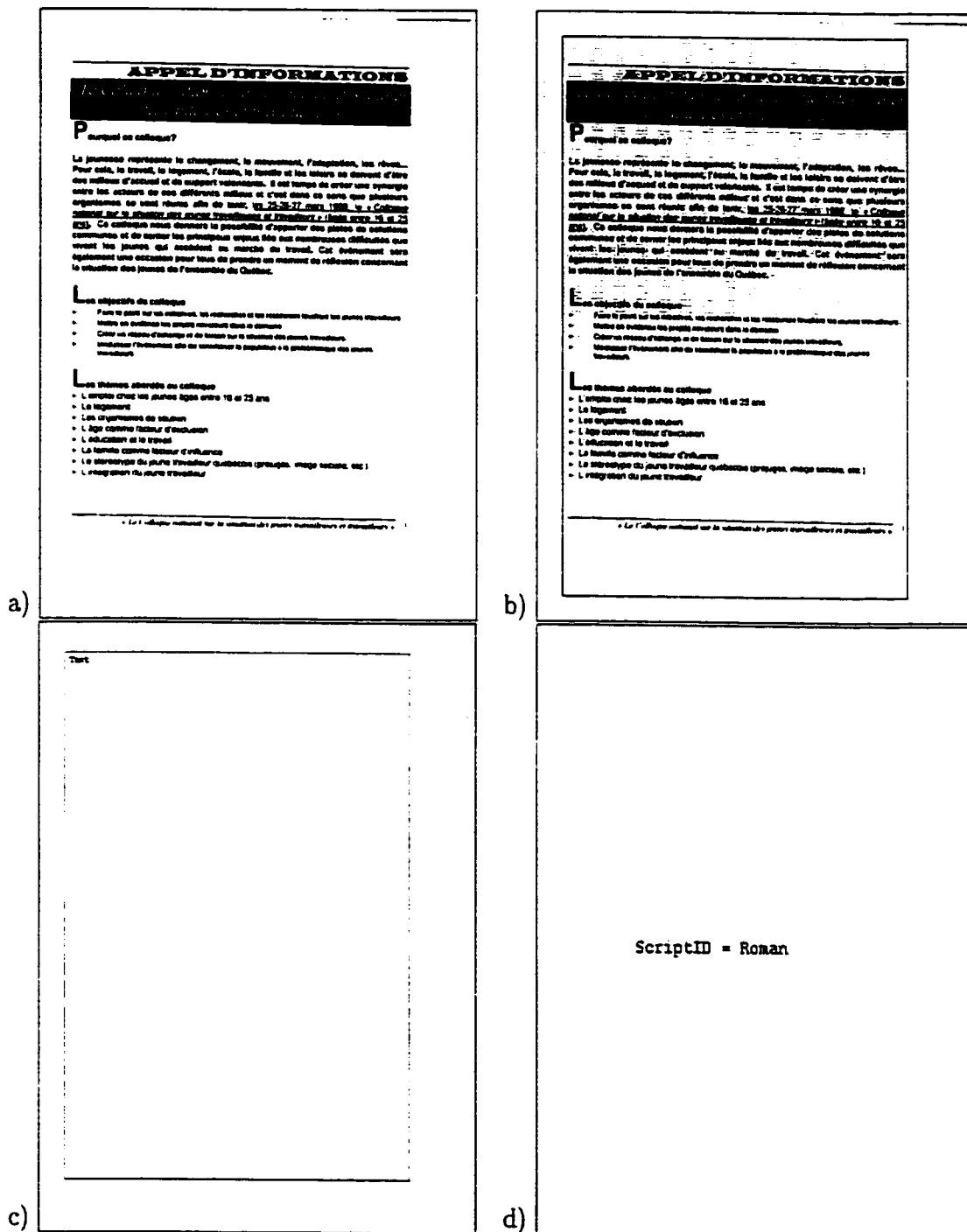


Figure 48: a) French memo, b) segmented page, c) text/graphics separation, d) script identification.



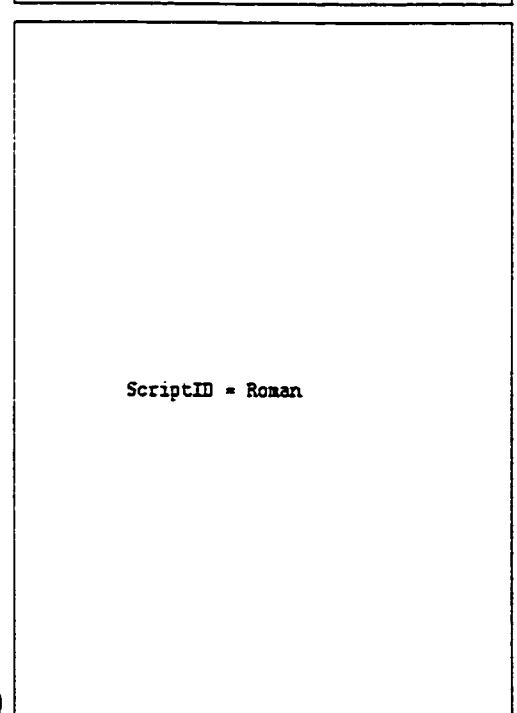
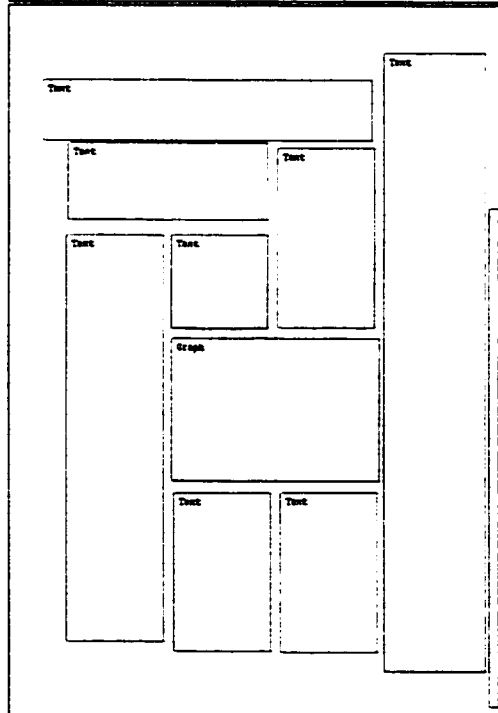


Figure 49: a) French scientific article, b) segmented page, c) text/graphics separation, d) script identification.

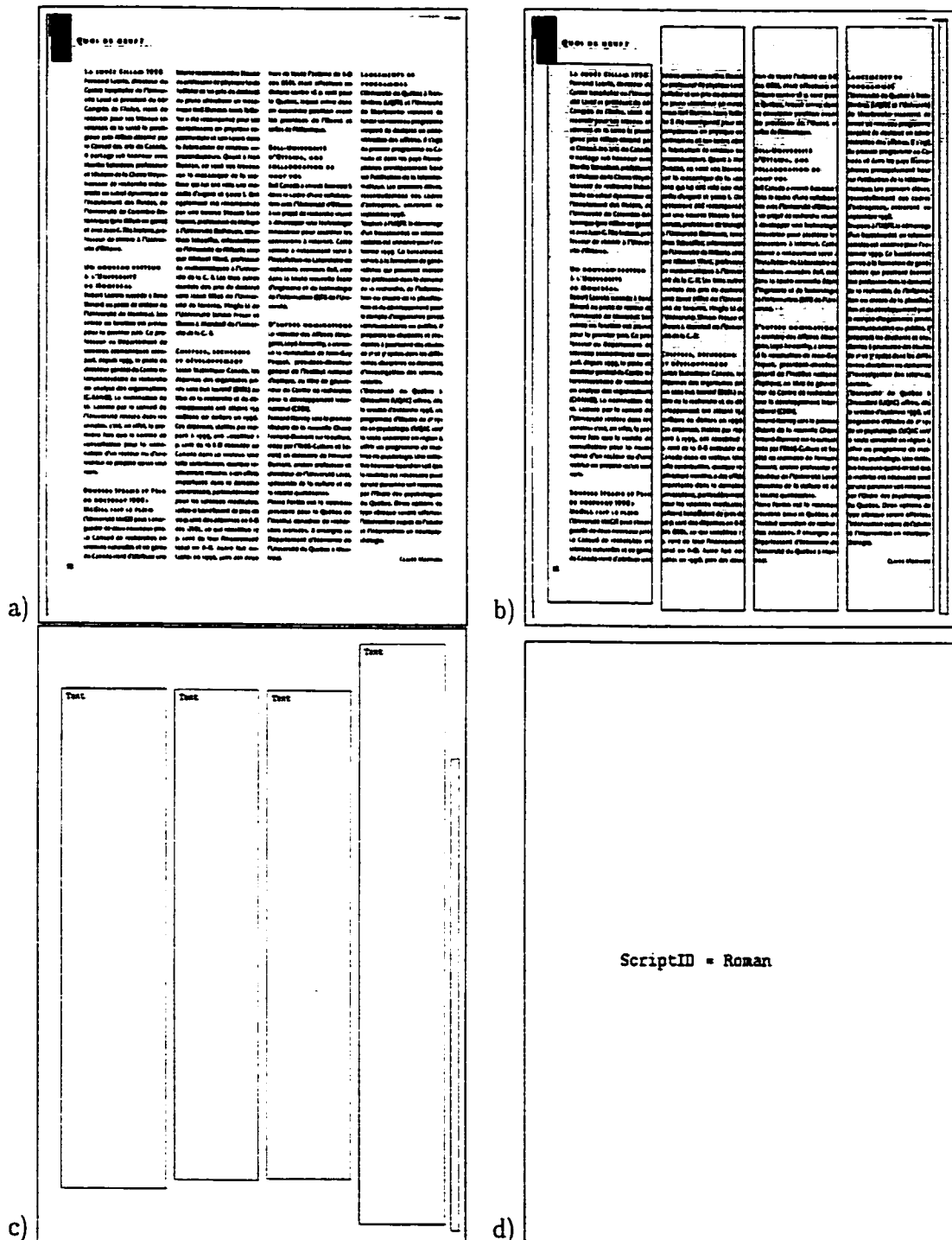


Figure 50: a) French scientific article, b) segmented page, c) text/graphics separation, d) script identification.

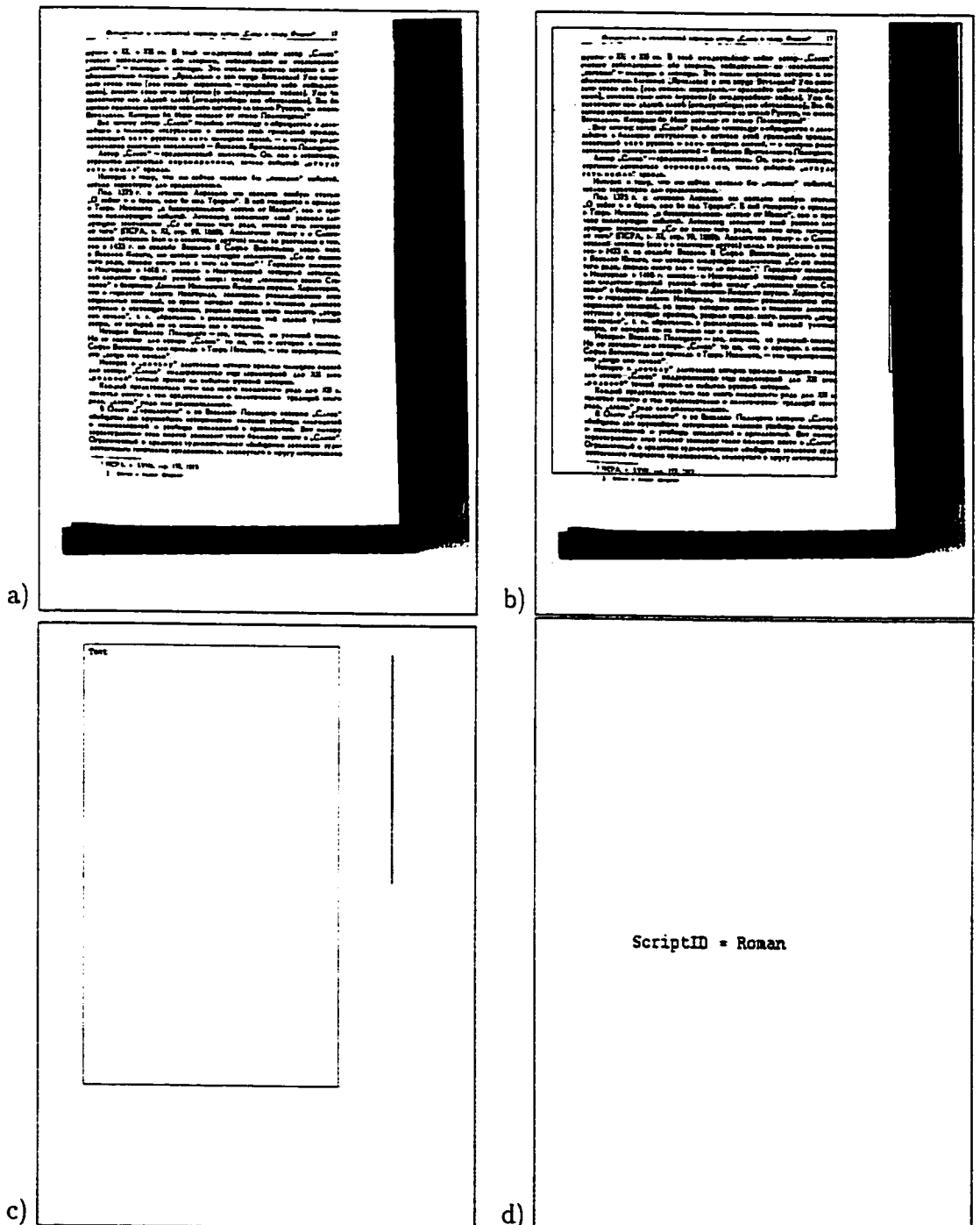


Figure 51: a) A page from a Russian book, b) segmented page, c) text/graphics separation, d) script identification.

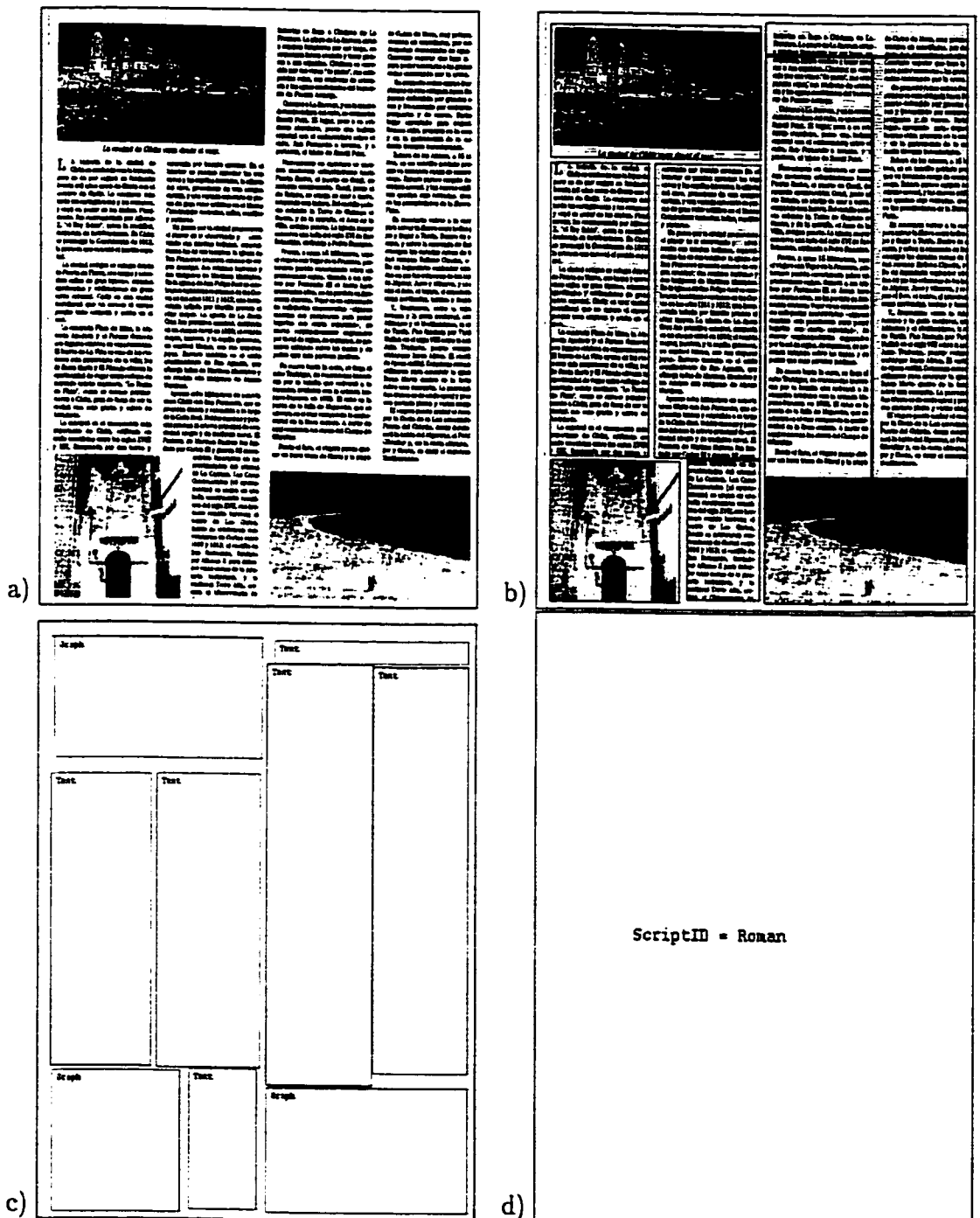


Figure 52: a) Spanish scientific article, b) segmented page, c) text/graphics separation, d) script identification.

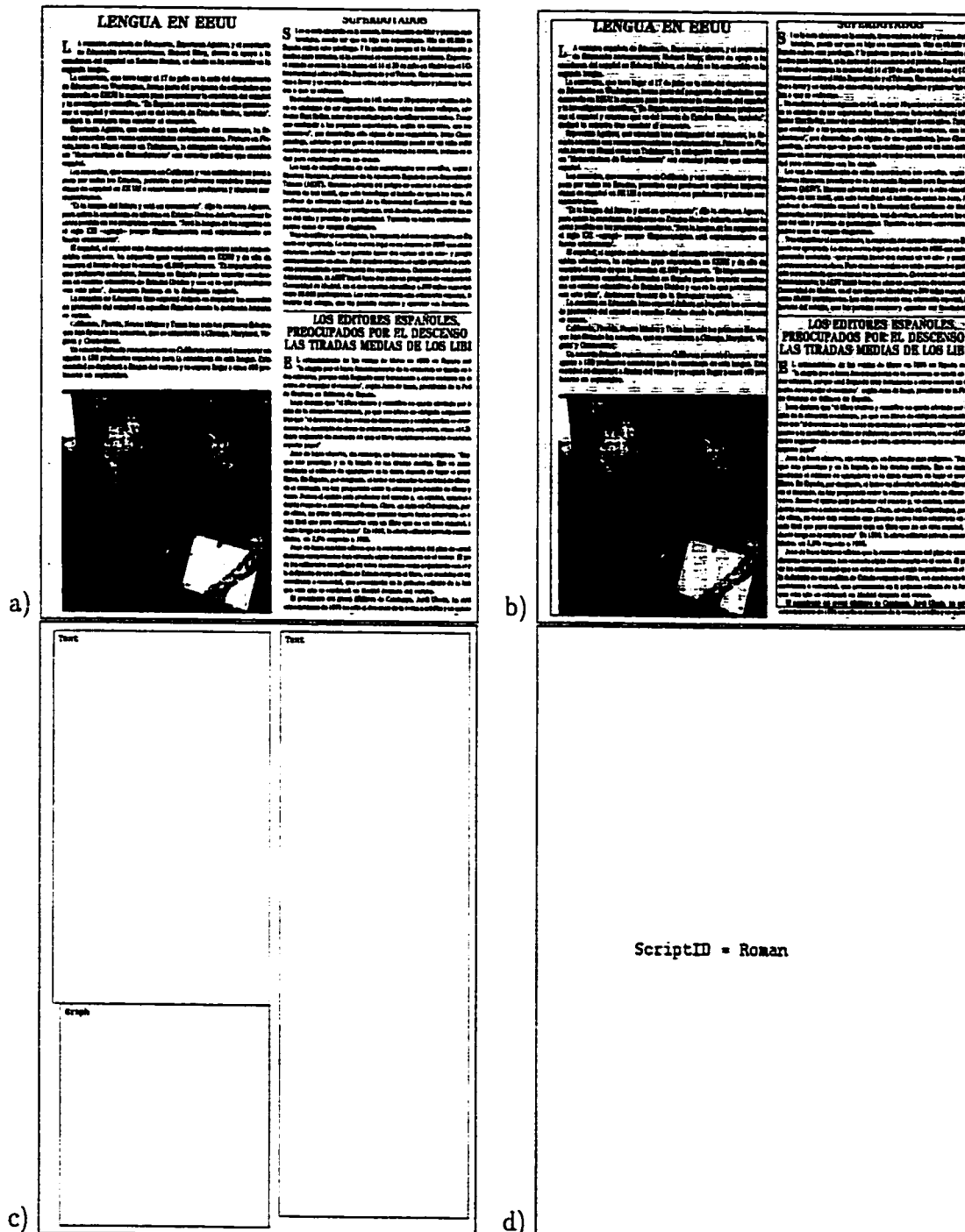


Figure 53: a) Spanish scientific article, b) segmented page, c) text/graphics separation, d) script identification.

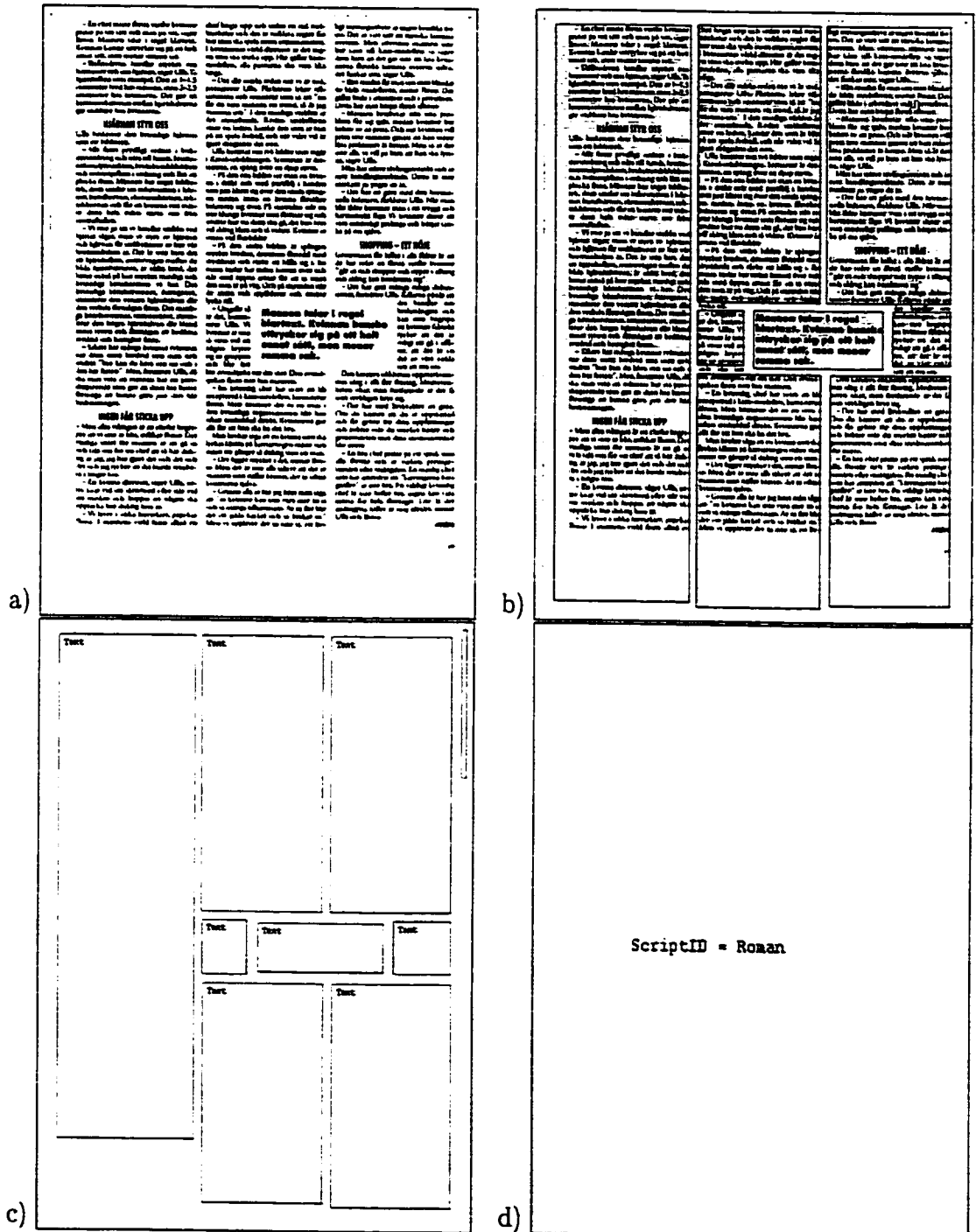


Figure 54: a) Swedish newspaper. b) segmented page, c) text/graphics separation, d) script identification.

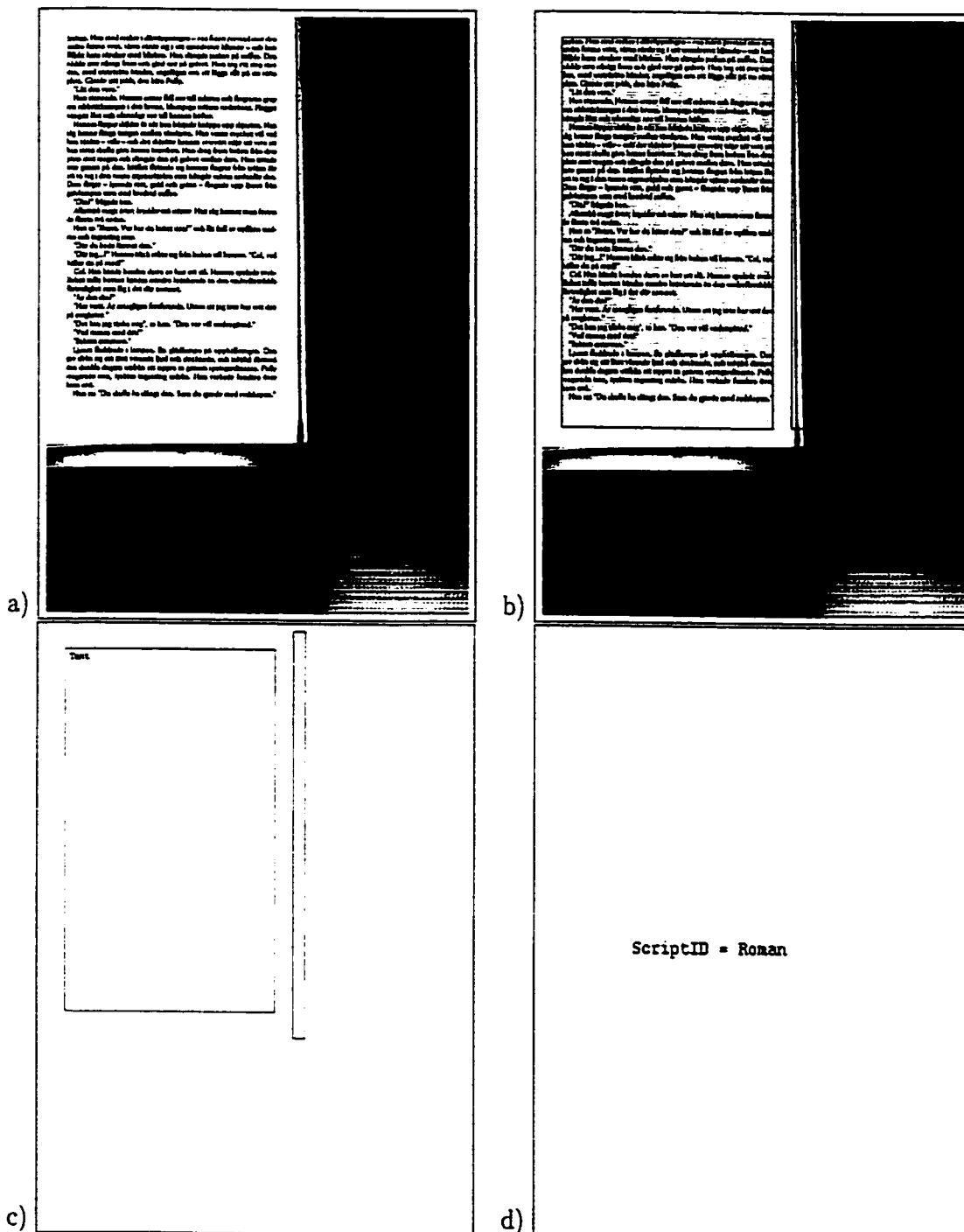


Figure 55: a) A large black area occupied almost half of the page because a page was scanned randomly with a size smaller than the setting size, which created the dark region outside the paper borders. However, the result showed that our method is not sensitive to this kind of problem and the block is correctly extracted. b) segmented page, c) text/graphics separation, d) script identification.