

A LARGE SCALE HMM-BASED OMNI FONT-WRITTEN OCR SYSTEM FOR CURSIVE SCRIPTS

by

Mohamed Saad Mostafa El-Mahallawy

**A Thesis Submitted to the
Faculty of Engineering, Cairo University
in Partial Fulfillment of the
Requirements for the Degree of**

DOCTOR OF PHILOSOPHY

in

ELECTRONICS & ELECTRICAL COMMUNICATIONS

Faculty of Engineering, Cairo University
Giza, Egypt
April 2008

A LARGE SCALE HMM-BASED OMNI FONT-WRITTEN OCR SYSTEM FOR CURSIVE SCRIPTS

by

Mohamed Saad Mostafa El-Mahallawy

**A Thesis Submitted to the
Faculty of Engineering, Cairo University
in Partial Fulfillment of the
Requirements for the Degree of**

**DOCTOR OF PHILOSOPHY
in
ELECTRONICS & ELECTRICAL COMMUNICATIONS**

Under the Supervision of

Mohsen Abdul Raziq Ali Rashwan

Professor
Faculty of Engineering
Cairo University

Mohamd Waleed Talaat Fakhr

Professor
Faculty of Engineering
Arab Academy for Science and Technology and
Maritime Transport

Faculty of Engineering, Cairo University
Giza, Egypt
April 2008

A LARGE SCALE HMM-BASED OMNI FONT-WRITTEN OCR SYSTEM FOR CURSIVE SCRIPTS

by

Mohamed Saad Mostafa El-Mahallawy

**A Thesis Submitted to the
Faculty of Engineering, Cairo University
in Partial Fulfillment of the
Requirements for the Degree of**

**DOCTOR OF PHILOSOPHY
in
ELECTRONICS & ELECTRICAL COMMUNICATIONS**

**Approved by the
Examining Committee:**

Prof. Dr. Mohsen Abdel-Razeq Rashwan, Thesis Advisor

Prof. Dr. Mohamed Waleed Talaat Fakhr, Thesis Advisor

Prof. Dr. Magdy Fekry Ragae, Member

Prof. Dr. Samia Abdel-Razeq Mashaly, Member

Faculty of Engineering, Cairo University
Giza, Egypt
April 2008

نظام موسع للتعرف الضوئي للنصوص ذات الكتابة المتصلة المختلفة الأبناط والحجوم
باستخدام نماذج ماركوف المخفية

إعداد

محمد سعد مصطفى المحلاوي

رسالة مقدمة إلى كلية الهندسة، جامعة القاهرة
كجزء من متطلبات الحصول على درجة الدكتوراه
في
هندسة الإلكترونيات والاتصالات الكهربائية

كلية الهندسة ، جامعة القاهرة
الجيزة ، جمهورية مصر العربية
أبريل 2008

نظام موسع للتعرف الضوئي للنصوص ذات الكتابة المتصلة المختلفة الأبناط والحجوم
باستخدام نماذج ماركوف المخفية

إعداد

محمد سعد مصطفى المحلاوي

رسالة مقدمة إلى كلية الهندسة، جامعة القاهرة
كجزء من متطلبات الحصول على درجة الدكتوراه
في
هندسة الإلكترونيات والاتصالات الكهربائية

تحت إشراف

محمد وليد طلعت فخر

عميد كلية الحاسبات و التكنولوجيا
الأكاديمية العربية للعلوم و التكنولوجيا
و النقل البحري

محسن عبد الرازق رشوان

أستاذ بقسم هندسة الإلكترونيات والاتصالات
كلية الهندسة ، جامعة القاهرة

كلية الهندسة ، جامعة القاهرة
الجيزة ، جمهورية مصر العربية
أبريل 2008

نظام موسع للتعرف الضوئي للنصوص ذات الكتابة المتصلة المختلفة الأبناط والحجوم
باستخدام نماذج ماركوف المخفية

إعداد

محمد سعد مصطفى المحلاوي

رسالة مقدمة إلى كلية الهندسة، جامعة القاهرة
كجزء من متطلبات الحصول على درجة الدكتوراه
في
هندسة الإلكترونيات و الاتصالات الكهربائية

يعتمد من لجنة الممتحنين:

الأستاذ الدكتور/ محسن عبد الرازق رشوان المشرف على البحث

الأستاذ الدكتور/ محمد وليد طلعت فخر المشرف على البحث

الأستاذ الدكتور/ مجدي فكري رجائي عضوا

الأستاذة الدكتورة/ سامية عبد الرازق مشالي عضوا

كلية الهندسة ، جامعة القاهرة
الجيزة ، جمهورية مصر العربية
أبريل 2008

Table of Contents

| | Page |
|----------------------------------------------------------------------------|-------------|
| List of Tables | vi |
| List of Figures | vii |
| List of Abbreviations | x |
| Acknowledgements | xi |
| Abstract | xii |
| 1. Introduction | 1 |
| 1.1. Character Recognition Systems Capabilities. | 1 |
| 1.1.1. On-Line Systems. | 2 |
| 1.1.2. Offline Systems. | 2 |
| 1.2. Character Recognition Technology Applications..... | 3 |
| 1.3 Arabic OCR Technology | 4 |
| 1.4. Arabic OCR Challenges. | 5 |
| 1.5. Arabic OCR Text Recognition System. | 8 |
| 1.6. Thesis Objectives. | 9 |
| 1.7. Thesis Contributions..... | 10 |
| 1.8. Thesis Outlines. | 10 |
| 2. Methodologies of OCR Systems; Theory and Literature Survey | 12 |
| 2.1. Image Acquisition. | 12 |
| 2.2. Preprocessing. | 13 |
| 2.2.1. Binarization | 13 |
| 2.2.2. Noise Reduction. | 13 |
| 2.2.2.1. Filtering. | 14 |
| 2.2.2.2. Smoothing. | 14 |
| 2.2.3. Normalization. | 15 |
| 2.2.3.1. Slant Normalization..... | 15 |
| 2.2.3.2. Size Normalization..... | 15 |
| 2.2.4. Skew Detection and Correction. | 16 |
| 2.2.5. Thinning and Skeletonization. | 17 |
| 2.2.6. Page Decomposition. | 18 |

| | |
|------------------------------------------------------------------------------|-----------|
| 2.3. Segmentation. | 18 |
| 2.3.1 Explicit Segmentation. | 19 |
| 2.3.2 Implicit Segmentation. | 20 |
| 2.4. Feature Extraction. | 20 |
| 2.4.1. Structural Features. | 21 |
| 2.4.2. Statistical Features | 22 |
| 2.4.3. Global Transformations..... | 23 |
| 2.5. Classification. | 24 |
| 2.5.1. Template Matching. | 25 |
| 2.5.2. Statistical Techniques (Decision Theoretic). | 26 |
| 2.5.2.1 Nonparametric Recognition. | 26 |
| 2.5.2.2. Parametric Recognition. | 27 |
| 2.5.2.3. Hidden Markov Modeling | 27 |
| 2.5.3. Syntactic (Structural) Techniques. | 28 |
| 2.5.4. Neural Networks. | 29 |
| 2.5.5. Hybrid Approaches. | 30 |
| 2.6. Post-Processing. | 30 |
| 2.7. Machine-Printed Arabic OCR Survey. | 31 |
| 3. Arabic Lines and Words Decomposition..... | 34 |
| 3.1. Enhanced Histogram-Based Lines & Words Decomposition Algorithm. | 35 |
| 3.1.1. Step 1; Filtering. | 36 |
| 3.1.2. Step 2; Getting the Darkness & Brightness Centroids. | 36 |
| 3.1.3. Step 3; Writing-Lines Decomposition. | 37 |
| 3.1.3.1. Step 3.1; Physical Writing-Lines Decomposition. | 37 |
| 3.1.3.2. Step 3.2; Logical Writing-Lines Decomposition. | 39 |
| 3.1.4. Step 4; Writing-Lines Grouping. | 40 |
| 3.1.5. Step 5; Part-of-Words Decomposition. | 41 |
| 3.1.6. Step 6; Full Words Concatenation. | 41 |
| 3.1.6.1. Step 6.1; Eliminating Rogue Spaces Among Part-of- Words..... | 42 |
| 3.1.6.2. Step 6.2; Clustering Inter-word/Intra-Word Spaces. | 43 |
| 3.2. Decomposition Algorithm Evaluation. | 44 |

| | |
|--------------------------------------------------------------------------|----|
| 4. Autonomously Normalized Horizontal Differential Features | 46 |
| Extraction | 46 |
| 4.1. ANHDF Basic Idea. | 47 |
| 4.2. ANHDF Design Requirements. | 48 |
| 4.3. ANHDF Vectors Design. | 49 |
| 4.4. Features Normalization. | 52 |
| 5. Vector Quantization and Clustering | 54 |
| 5.1. Training Set. | 55 |
| 5.2. Distance Measure. | 55 |
| 5.3. Centroid Computation. | 56 |
| 5.4. Clustering Algorithm. | 56 |
| 5.4.1. K-Means Algorithm. | 57 |
| 5.4.2. LBG Algorithm..... | 58 |
| 5.5. Clusters Analysis | 59 |
| 5.6. The Adequacy Test. | 61 |
| 6. HMM from ASR to OCR | 80 |
| 6.1. ASR and OCR Problems Analogy. | 80 |
| 6.2. Theory and Elements of Discrete HMM. | 84 |
| 6.3. HMM Topologies. | 85 |
| 6.4. The Basic Problems of HMM's..... | 86 |
| 6.4.1. Model Evaluation Problem. | 87 |
| 6.4.2. Training Problem..... | 88 |
| 6.4.2.1. Baum-Welch Re-Estimation Procedure..... | 88 |
| 6.4.2.2. Initialization Procedures..... | 88 |
| 6.4.3. Decoding Problem..... | 89 |
| 7. Statistical Language Models and Character Recognition | 92 |
| 7.1. Statistical Language Models | 93 |
| 7.2. N-Gram Language Models. | 93 |
| 7.3. The Language Phenomenon from a Statistical Perspective..... | 94 |
| 7.4. Maximum Likelihood Probability Estimation..... | 96 |
| 7.5. “Bayes’, Good-Turing Discount, Back-Off” Probability Estimation.... | 97 |
| 7.6. Arabic Ligature Bigram SLM..... | 98 |
| 7.7. NEMLAR Arabic Written Corpus..... | 99 |

| | |
|------------------------------------------------------------------|-----|
| 8. Arabic OCR System; Implementation and Evaluation | 101 |
| 8.1. HMM-Based OCR System Architecture..... | 101 |
| 8.1.1. Digital Scanning. | 104 |
| 8.1.2. Primary Noise Handler..... | 104 |
| 8.1.3. Words & Lines Decomposer..... | 104 |
| 8.1.4. Features Extraction Module..... | 105 |
| 8.1.5. Vector Clustering and Quantization Modules..... | 105 |
| 8.1.6. Dynamic Range Normalization Parameters Estimator..... | 106 |
| 8.1.7. Characters-to-Ligature Converter. | 106 |
| 8.1.8. Discrete HMM Trainer..... | 106 |
| 8.1.8.1. HMM Initialization | 107 |
| 8.1.8.2. HMM Re-Estimation Embedded Training..... | 108 |
| 8.1.9. Discrete HMM Recognizer..... | 108 |
| 8.1.10. Ligature n-Grams Probability Estimator..... | 109 |
| 8.1.11. Ligature -to-Character Converter..... | 109 |
| 8.2. Arabic Font-Written Database..... | 109 |
| 8.3. Evaluation Experiments Setup..... | 112 |
| 8.3.1. Training Phase Setup. | 112 |
| 8.3.2. Testing Phase Setup..... | 113 |
| 8.3.3. System Parameters Setup. | 115 |
| 8.3.4. System Evaluation..... | 116 |
| 8.4. Experimental Error Analysis..... | 118 |
| 9. Conclusion and Future Work | 121 |
| References. | 124 |

List of Tables

| | Page |
|---------------------------------------------------------------------------------|-------------|
| 2.1: Classification strategies, Holistic vs. Analytic..... | 25 |
| 2.2: Classification approaches comparison..... | 29 |
| 3.1: The proposed decomposition algorithm evaluation results..... | 45 |
| 7.1: NEMLAR Arabic Written Corpus categories..... | 99 |
| 7.2. Correct words versus Corrupted words probabilities computed by the LM.. | 100 |
| 8.1: Visual features description of the fonts used in training and testing..... | 110 |
| 8.2: HMM parameters setting..... | 116 |
| 8.3: Experimental results..... | 116 |
| 8.4: Error analysis of assimilation test regarding font shape/size..... | 118 |
| 8.5: Error analysis of generalization test regarding font shape/size..... | 118 |
| 8.6: Assimilation test most frequent mistakes..... | 119 |
| 8.7: Generalization test most frequent mistakes..... | 119 |

List of Figures

| | Page |
|-------------------------------------------------------------------------------------------------|-------------|
| 1.1: Character recognition capabilities..... | 1 |
| 1.2: Grapheme segmentation process..... | 6 |
| 1.3: Example sets of dotting-differentiated graphemes..... | 6 |
| 1.4: Grapheme “Ein” in its 4 positions; Starting, Middle, Ending & Separate... | 6 |
| 1.5: Some ligatures in the Traditional Arabic font..... | 7 |
| 1.6: Some overlapped Characters in Demashq Arabic font..... | 7 |
| 1.7: Arabic text with diacritics..... | 8 |
| 1.8: Typical Components of an OCR-System..... | 8 |
| 2.1: Results of removing noise from a text image using median filter | 14 |
| 3.1: Sample Arabic text rectangle bitmap..... | 35 |
| 3.2: Sample text rectangle bitmap after median filtering..... | 36 |
| 3.3: Gray values histogram of the pixels population of the sample text rectangle..... | 37 |
| 3.4: Sample text rectangle bitmap after initial lines decomposition. | 38 |
| 3.5: Sample text rectangle bitmap after refined lines decomposition..... | 39 |
| 3.6: Sample text rectangle bitmap after logical writing lines concatenation..... | 39 |
| 3.7: Sample text rectangle bitmap after part-of-words decomposition..... | 41 |
| 3.8: A hypothetical histogram of intra-word, inter-word, and rogue part-of- word spaces..... | 42 |
| 3.9: Sample text rectangle bitmap after full words concatenation..... | 44 |
| 4.1: Word slice and segments | 49 |

| | |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------|-------|
| 4.2: Feature extraction algorithm flowchart..... | 51 |
| 4.3: Feature values density distributions. | 52 |
| 5.1: A two dimensional space showing samples with their representative centroid..... | 56 |
| 5.2: LBG Centroid splitting..... | 59 |
| 5.3: A two dimensional data with its normalized mean square error function..... | 60 |
| 5.4: a) Normalized Features density distributions projected on $d=1$; b) Normalized Code book (size 128) density distribution projected on $d=1$. | 62 |
| 5.5-5.20: Normalized Features density distributions with its Codebooks (of sizes 2048, 1024, 512, 256) density distributions projected on $d=1, \dots, 16$. | 64-79 |
| 6.1: Sliding window over a speech signal and over a text image bitmap..... | 81 |
| 6.2: Noisy channel communication model..... | 82 |
| 6.3: Ergodic Hidden Markov Model..... | 85 |
| 6.4: An Example of Left-to-Right HMMs..... | 86 |
| 6.5: HMM search trellis for graphemes recognition..... | 90 |
| 7.1: The attenuating correlation phenomenon. | 94 |
| 7.2: Zipf's curve; ranking entities by their frequencies..... | 95 |
| 8.1: Discrete HMM-based ASR system block diagram..... | 102 |
| 8.2: Discrete HMM-based OCR system block diagram. | 103 |
| 8.3. Sample text rectangle bitmap. | 105 |
| 8.4. Sample text rectangle bitmap after lines and words decomposition..... | 105 |
| 8.5: Characters-to-Ligatures conversion example..... | 106 |
| 8.6: Viterbi training flowchart..... | 107 |
| 8.7: Baum-Welch training flowchart..... | 107 |
| 8.8: Arabic Ligatures set used in the OCR system..... | 111 |
| 8.9: Samples of fonts MS-Windows used in training..... | 112 |

| | |
|-----------------------------------------------------------------|-----|
| 8.10: Samples of Mac. fonts used in training..... | 113 |
| 8.11: Samples of fonts used in the generalization tests..... | 114 |
| 8.12: System performance using different codebook size..... | 115 |
| 8.13: Number of states per model versus system performance..... | 116 |

List of Abbreviations

| | |
|--------|----------------------------------------------------------|
| AI: | Artificial Intelligence |
| ANHDF | Autonomously Normalized Horizontal Differential Features |
| ASCII: | American Standard Code for Information Interchange |
| ASR | Automatic Speech Recognition |
| B&W | Black & White |
| BBN | Bolt Beranek and Newman's |
| BMP | Bitmap |
| DNRP | Dynamic Range Normalization Parameters |
| EC | European Commission |
| FD | Fourier Descriptor |
| HMM | Hidden Markov Model |
| ICR | Intelligent Character Recognition |
| IF | Information Technology |
| IR | Information Retrieval |
| KM | Knowledge Management |
| LBG | Linde, Buzo, and Gray |
| MAP | maximum a posteriori probability |
| MFCC | Mel Frequency Cepstral Coefficient |
| ML | Maximum Likelihood |
| NEMLAR | Network for Euro-Mediterranean Language Resource |
| NN | Neural Network |
| NLP | Natural Language Processing |
| OCR | Optical Character Recognition |
| pdf | probability density function |
| R&D | Research and Development |
| RDI | Research & Development International company |
| SLM | Statistical Language Model |
| WER | Word Error Rate |
| VQ | Vector Quantization |

Acknowledgements

Most of all, I wish to express my deepest appreciation to my supervisor, Prof. *Mohsen A. A. Rashwan*, for his guidance in the preparation of this work. He was the first to hear about, and comment on, the innovations that constitute the substance of this thesis. I have been able to rely with confidence on his extensive expertise in the field.

Secondly, I would like to express my gratitude to my supervisor, Prof. *Mohamed Waleed Fakhr* who introduced me to the key concepts of the problems involved in this thesis.

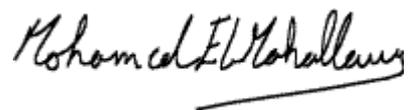
It is my pleasure to acknowledge the support from *The Engineering Company for the Development of Computer Systems; RDI* www.rdi-eg.com where this work has been adopted and supported for the last two years in my Ph.D. study. The continuous support and encouragement from the people in RDI are also greatly appreciated.

I am greatly indebted to Dr. *Mohamed Attia*, the human language technologies consultant at RDI, for his constructive comments and valuable suggestions on my dissertation. It has been a great pleasure to conduct my research under his guidance with his remarkable ideas and his generous efforts and patience.

Above all, I am immensely grateful to my parents, brother and my sister-in-law for their continuous support and encouragement throughout this study.

Finally and the most importantly, this study could not have been accomplished without the unfailing love and dedication of my wife who has shared me in the delights and grieves of this research and writing process. She deserves to be named on the dedication of this dissertation along with my lovely daughters, Rana and Nada.

December 2007



Abstract

The state of the art of automatic recognition of text at the dawn of the new millennium is that as a field it is no longer an esoteric topic on the fringes of Information Technology (IT), but a mature discipline that has found many commercial uses. Automatic font-written (machine-printed) Optical Character/text Recognizers (OCR) are highly desirable for a multitude of modern IT applications.

Reliable font-written OCR systems for Latin scripted languages are readily in use since long. For cursively scripted languages, that are the mother tongues of over $\frac{1}{4}$ of the world population, such OCR systems are however not available at a robust and reliable performance. In this regard, the main challenge is the mandatory connectivity of the characters/ligatures (i.e. graphemes) that is to be resolved simultaneously upon the recognition of these graphemes.

Among the numerous approaches tried over 30 years of Research and Development (R&D), Hidden Markov Model (HMM) based OCR systems seem to be the most promising as they capitalize on the ability of HMM decoders to achieve segmentation and recognition simultaneously as is the case with the widely used HMM-based Automatic Speech Recognition (ASR) systems.

In our dissertation, an omnifont, open vocabulary, HMM-based Arabic text recognition system is implemented with the aid of a robust developed algorithm for lines and words decomposition of multifold and multisize Arabic documents; an example of cursive scripts. Building an open vocabulary system derives us to the usage of a character based HMM system not a word based HMM system.

Unlike ASR systems, what is missing in HMM-based OCR systems is the definition of a rigorously derived features vector capable to robustly achieve minimal omni-font Word Error Rate (WER) comparable to that realized with Latin scripts. The design of such a features vector is fully introduced and discussed in this dissertation.

Based on the OCR-to-ASR analogy, a full fledged Arabic font-written OCR system analogous to HMM-based ASR systems has been also introduced and analyzed in this dissertation with and without the aid of statistical language models (SLM's). The experimental results done among this dissertation puts our HMM-based OCR with the new features vector in the lead among the other open-vocabulary omni font-written OCR systems for cursive scripts , at least for Arabic script.

CHAPTER 1

Introduction

Character Recognition is the branch of pattern recognition that ultimately aims to compete with the human ability to read written text regarding both the speed and accuracy by associating character codes (e.g. Unicode) to images of characters (i.e. graphemes).

The origin of character recognition can be found as early as 1870 by the invention of the scanner, which has first appeared as an aid to the visually handicapped. The first successful attempt was made by the Russian scientist Tyurin in 1900. The modern version of OCR appeared in the middle of the 1940s with the development of the digital computers. [Mantas 1986], [Govindan 1990]

Character recognition has become one of the most successful applications of technology in the field of pattern recognition and artificial intelligence. Many commercial systems for performing character recognition exist for a variety of applications, although the machines are still not able to compete with human reading capabilities.

1.1. Character Recognition Systems Capabilities

Character recognition systems differ widely in how they acquire their input (on-line versus off-line), the mode of writing (handwritten versus machine printed), the connectivity of text (isolated characters versus cursive words), and the restriction on the fonts (single font versus omni-font) they can recognize. [Govindan 1990], [Al-Badr 1995]

The different capabilities of character recognition are illustrated in Figure (1.1).

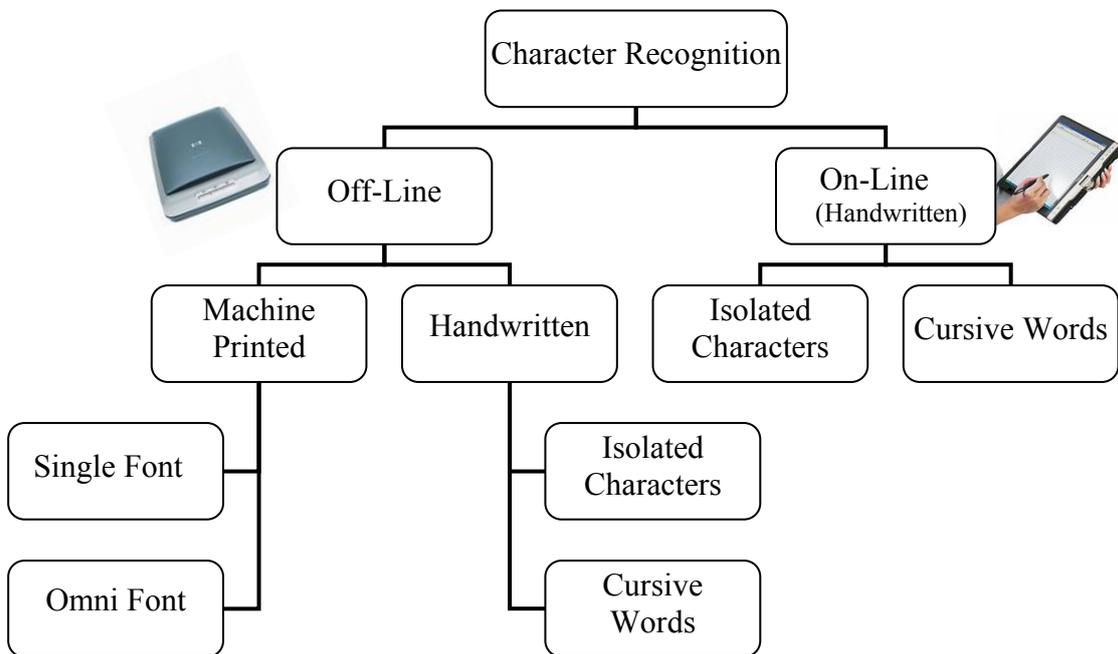


Figure (1.1): Character recognition capabilities

1.1.1. On-Line (Real-Time) Systems

These systems recognize text while the user is writing with an on-line writing device, capturing the temporal or dynamic information of the writing. This information includes the number, duration, and order of each stroke (a stroke is the writing from pen down to pen up). On-line devices are stylus based, and they include tablet displays, and digitizing tablets. The writing here is represented as a one-dimensional ordered vector of (x, y) points. On-line systems are limited to recognizing handwritten text. Some systems recognize isolated characters, while others recognize cursive words.

1.1.2. Off-Line Systems

These systems recognize text that has been previously written or printed on a page and then optically converted into a bit image. Off-line devices include optical scanners of the flatbed, paper fed and handheld types. Here, a page of text is represented as a two-dimensional array of pixel values. Off-line systems do not have access to the time-dependent information captured in on-line systems. Therefore off-

line character recognition is considered as a more challenging task than its online counterpart.

The word optical was earlier used to distinguish an optical recognizer from systems which recognize characters that were printed using special magnetic ink. In the case of a machine-printed (font-written) document image, this is referred to as Optical Character Recognition (OCR). In the case of handprint (Mail Orders, Checks, Credit Cards Applications, ...,etc.), it is referred to as Intelligent Character Recognition (ICR).

Over the last few years, the decreasing price of laser printers has made computer users able to readily create multi-font documents. The number of fonts in typical usage has increased accordingly. However the researcher experimenting on OCR is unhappy to perform the vastly time-consuming experiments involved in training and testing a classifier on potentially hundreds of fonts in a number of text sizes and in a wide range of image noise conditions; even if such an image data set already existed. Collecting such a database could involve considerably more effort.

Although the amount of research into machine-print recognition appears to be tailing off as many research groups turn their attention to handwriting recognition, it is suggested that there are still significant challenges in the machine-print domain. One of these challenges is to deal effectively with noisy, multi-font (including possibly hundreds of fonts) cursive scripts.

The sophistication of the off-line OCR system depends on the type and number of fonts to be recognized. An omni-font OCR machine can recognize most non stylized fonts without having to maintain huge databases of specific font information. Usually omni-font technology is characterized by the use of feature extraction. Although omni-font is the common term for these OCR systems, this should not be understood literally as the system being able to recognize all existing fonts. No OCR machine performs equally well or even usably well on all the fonts used by modern computers.

1.2. Character Recognition Technology Applications

The intensive research effort in the field of character recognition was not only because of its challenge on simulation of human reading but also because it provides widespread efficient applications. Three factors motivate the vast range of

applications of off-line text recognition. The first two are the easy use of electronic media and its growth at the expense of conventional media. The third is the necessity of converting the data from the conventional media into the new electronic media.

OCR technology has many practical applications [Govindan 1990], [Srihari 1992] which include the following, as examples but not limited to,

- Storing, retrieving and indexing huge amount of electronic data as a results of the resurgence of the World Wide Web. The text produced by OCRing text images can be used for all kinds of Information Retrieval (IR) and Knowledge Management (KM) systems, which are not so sensitive to the inevitable Word Error Rate (WER) of whatever OCR system as long as this WER is kept lower than 10% to 15% [Callen 2003]. It might be sufficiently illustrative to mention the gigantic project of the online global cross lingual searchable library being achieved by *Google™* as an example of how OCR is vital in this regard [Feng 2006].
- Office automation for providing an improved office environment and ultimately reach an ideal paperless office environment.
- Business applications as automatic processing of checks.
- Automatic address reading for mail sorting.
- Automatic passport readers.
- Digital bar code reading and signature verification.
- Front end components for blind reading machines by transferring the recognition results into sound output or tactile symbols through stimulators.
- Machine processing of forms.
- Mobile card reader.

1.3. Arabic OCR Technology

Since the mid-1940s researchers have carried out extensive work and published many papers on character recognition. Most of the published work on OCR has been on Latin characters, with work on Japanese and Chinese characters emerging in the mid-1960s. Although almost a billion of people worldwide, in several different languages, use Arabic characters for writing (alongside Arabic, Persian and Urdu are the most noted examples) [Khorsheed 2002], Arabic character recognition has not been researched as thoroughly as Latin, Japanese, or Chinese and it has almost only

started in 1975 by Nazif [Nazif 1975]. This may be attributed to the following [Al-Badr 1995]:

- i) The lack of adequate support in terms of journals, books, conferences, and funding, and the lack of interaction between researchers in this field.
- (ii) The lack of general supporting utilities like Arabic text databases, dictionaries, programming tools, and supporting staff.
- (iii) The late start of Arabic text recognition.
- (iv) The special challenges in the characteristics of the Arabic script as stated in the following section. These characteristics results in the fact that the techniques developed for other writings cannot be successfully applied to the Arabic writing.

In order to be competent with the human capability at the digitization of printed text, font-written OCR systems should achieve an omni-font performance at an average WER $\leq 3\%$ and an average speed ≥ 60 words/min. per processing thread [Al-Badr 1995]. While font-written OCR systems working on Latin script can claim approaching such measures under favorable conditions, the best systems working on other scripts, especially cursive scripts like Arabic, are still well behind due to a multitude of complexities. For example, the best reported ones among the few Arabic omni font-written OCR systems can only claim assimilation WER's exceeding 10% under favorable conditions and not to mention the realistic ones. [Bazzi 1999], [Khorsheed 2007]

1.4. Arabic OCR challenges

The written form of Arabic language while written from right to left presents many challenges to the OCR developer. The most challenging features of the Arabic orthography are [Al-Badr 1995], [Attia 2004] :

i) The connectivity challenge

.Whether handwritten or font written, Arabic text can only be scripted cursively; i.e. graphemes are connected to one another within the same word with this connection interrupted at few certain characters or at the end of the word. This necessitates any Arabic OCR system to not only do the traditional grapheme recognition task but do another tougher grapheme segmentation one (see Figure

(1.2)). To make things even harder, both of these tasks are mutually dependent and must hence be done simultaneously.

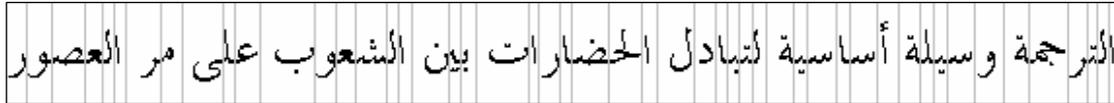


Figure (1.2): Grapheme segmentation process illustrated by manually inserting vertical lines at the appropriate grapheme connection points.

ii) The dotting challenge

Dotting is extensively used to differentiate characters sharing similar graphemes. According to Figure (1.3), where some example sets of dotting-differentiated graphemes are shown, it is apparent that the differences between the members of the same set are small. Whether the dots are eliminated before the recognition process, or recognition features are extracted from the dotted script, dotting is a significant source of confusion – hence recognition errors – in Arabic font-written OCR systems especially when run on noisy documents; e.g. those produced by photocopiers.

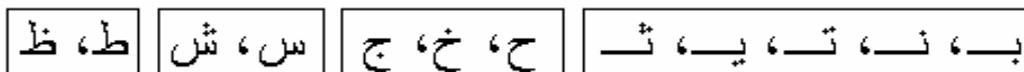


Figure (1.3): Example sets of dotting-differentiated graphemes

iii) The multiple grapheme cases challenge

Due to the mandatory connectivity in Arabic orthography; the same grapheme representing the same character can have multiple variants according to its relative position within the Arabic word segment {Starting, Middle, Ending, Separate} as exemplified by the 4 variants of the Arabic character “ع” shown in bold in Figure (1.4).



Figure (1.4): Grapheme “ع” in its 4 positions; Starting, Middle, Ending & Separate

iv) The ligatures challenge

To make things even more complex, certain compounds of characters at certain positions of the Arabic word segments are represented by single atomic

graphemes called ligatures. Ligatures are found in almost all the Arabic fonts, but their number depends on the involvement of the specific font in use. Traditional Arabic font for example contains around 220 graphemes, and another common less involved font (with fewer ligatures) like Simplified Arabic contains around 151 graphemes. Compare this to English where 40 or 50 graphemes are enough. A broader grapheme set means higher ambiguity for the same recognition methodology, and hence more confusion. Figure (1.5) illustrates some ligatures in the famous font “Traditional Arabic”.

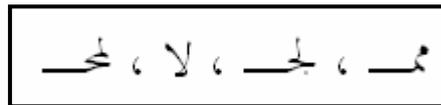


Figure (1.5): Some ligatures in the Traditional Arabic font.

iv) The overlapping challenge

Characters in a word may overlap vertically even without touching as shown in Figure (1.6).

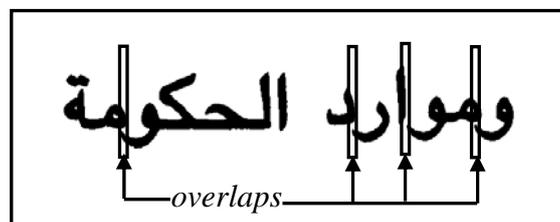


Figure (1.6): Some overlapped Characters in Demashq Arabic font.

v) Size variation challenge

Different Arabic graphemes do not have a fixed height or a fixed width. Moreover, neither the different nominal sizes of the same font scale linearly with their actual line heights, nor the different fonts with the same nominal size have a fixed line height.

vi) The diacritics challenge

Arabic diacritics are used in practice only when they help in resolving linguistic ambiguity of the text. The problem of diacritics with font written Arabic OCR is that their direction of flow is vertical while the main writing direction of the body Arabic text is horizontal from right to left. (See Figure (1.7)) Like dots;

diacritics – when existent - are a source of confusion of font-written OCR systems especially when run on noisy documents, but due to their relatively larger size they are usually preprocessed.

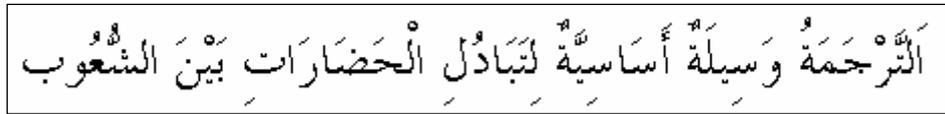


Figure (1.7): Arabic text with diacritics.

Although classic Arabic texts use a relatively limited number of font faces, new typographic systems have led to the proliferation of Arabic font faces which are almost as varied as those for Roman alphabets. Because of these, and the mentioned challenges, the state of the art in Arabic OCR significantly lags that for Roman text. Hence, a countable handful of Arabic OCR products are commercially available (with high price) in the market compared to Latin OCR systems (now much cheaper).

1.5. Arabic OCR Text Recognition System

The process of recognizing Arabic text can be broadly broken down into 5 stages after image acquisition; pre-processing, segmentation, feature extraction, classification, and post-processing as shown in Figure (1.8).

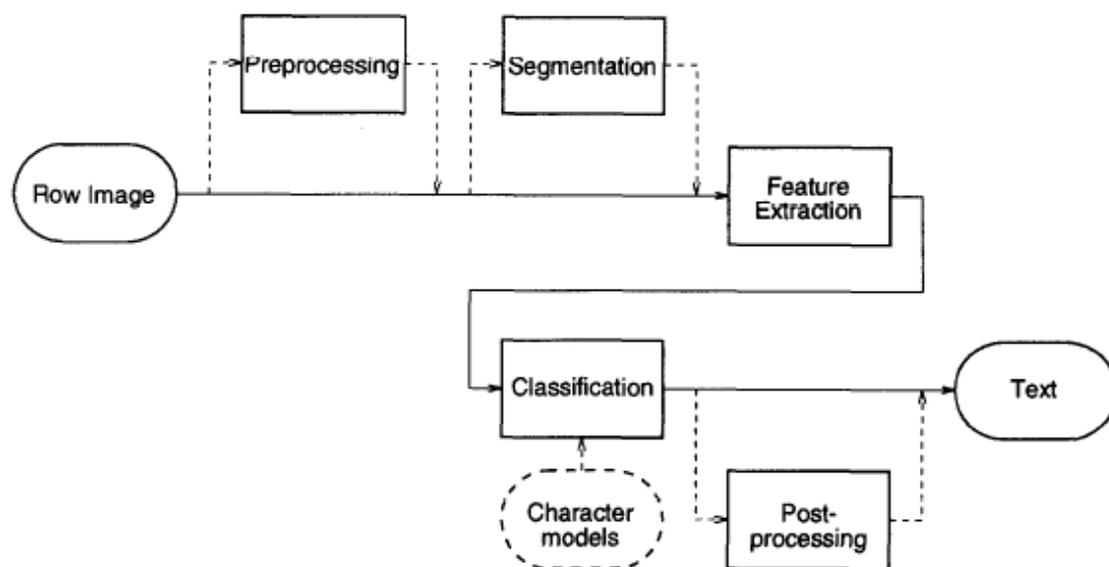


Figure (1.8): Typical Components of an OCR-System

Through the image acquisition process, a digital image of the original document is captured. In OCR, mainly optical scanners are used, which generally consist of a transport mechanism plus a sensing device that converts light intensity into gray-levels.

The preprocessing stage is a collection of operations that apply successive transformations on an image. It takes in a raw image and enhances it by reducing noise and distortion, and hence simplifies segmentation, feature extraction, and consequently recognition. After the preprocessing stage, many OCR systems segment the text into individual characters or strokes before recognizing them.

The feature extraction stage analyzes a text segment and selects a set of features that can be used to uniquely identify the text segment. These features are extracted and passed in a form suitable for the recognition phase

The classification stage, the main decision-making stage of an OCR system uses the features extracted in the previous stage to identify the text segment according to preset rules. This stage may use feature models obtained in an (off-line) training (modeling) phase to classify the test data.

Finally, the post processing stage improves the recognition performance by refining the decisions taken by the previous stage and recognizes words by using context. It is ultimately responsible for outputting the best solution and is often implemented as a set of techniques that rely on character frequencies, lexicons, and other context information.

The preprocessing, segmentation, and post processing stages are not necessarily executed by all Arabic optical text recognition systems. Some systems assume noise-free input. Also, some techniques recognize a character before segmentation, while others recognize isolated characters. [Al-Badr 1995]

1.6. Thesis Objectives

Despite 35 years of Research and Development (R&D) on the problem of OCR, the technology is not yet mature enough for the Arabic font-written script compared with Latin-based ones. There is still a wide room for enhancements as per: lowering the WER, standing robust in face of moderate noise, and working on an omni-font and open-vocabulary basis.

Among the best trials done in this regard so far, Hidden Markov Models (HMM) based ones appear. Elaborating on this Automatic Speech Recognition (ASR)-inspired promising approach, we have significantly refined the basic processes and modules deployed in such architectures (e.g. lines & words decomposition, features extraction, models parameters selection, language modeling, .., etc.) to develop what is hoped to be a truly reliable (i.e. low WER, omni font-written, open-vocabulary, noise-robust, and responsive) Arabic OCR suitable for real-life Information Technology (IT) applications.

1.7. Thesis Contributions

While building an omni-font and open vocabulary Arabic OCR system, the following contributions are evolved:

- Developing an enhanced histogram-based algorithm for line and words decomposition of Arabic text rectangles, so that it performs robustly on documents containing the idiosyncrasies of real-life documents especially as per noise and structural textual complexities.
- The design of a mathematically rigorous lossless differential luminosity coding based features capable to robustly achieve minimal omni-font WER.
- Developing a visual aid for estimating the training data set size and the codebook size of a clustering algorithm, in order to achieve the minimal WER achieved from an OCR recognition system without doing neither computationally expensive nor time consuming measures.
- The design, implementation and evaluations of a real life omni-font, open-vocabulary, HMM-based Arabic recognition system.

1.8. Thesis outline

After outlining the problem we are facing with, the objective, and the main contributions of this thesis, the formal organization of the thesis is illustrated as follows,

Chapter 2 presents a technical review for Arabic machine printed OCR techniques. The main steps in any Arabic OCR system including scanning, pre-processing, segmentation, feature extraction, classification, and post-processing are

extensively reviewed. Finally, a survey on the most recent published research on Arabic OCR is briefed.

Chapter 3 introduces a robust algorithm for lines and words decomposition especially in Arabic, or Arabic dominated, of text rectangles from real-life multifold / multisize documents decomposition. Also an extensive evaluation of the introduced algorithm over different types of document sources with different noise levels is demonstrated.

Chapter 4 defines and introduces the design of a rigorously derived features vector capable to robustly achieve minimal omni-font WER. The nature of the designed autonomously normalized horizontal differential features vector is also discussed.

Chapter 5 gives an overview of the theory of vector quantization and clustering. Also, the LBG clustering technique used in the construction of the codebook used in our proposed system is illustrated. Eventually, a developed visualization aid that helps us in estimating the best training data set size and the best codebook size for a least WER is discussed.

Chapter 6 demonstrates the analogy between OCR and ASR problems, and gives a brief overview of the Hidden Markov Models (HMM's) and its application to OCR.

Chapter 7 is focused on the discussion of use of the Statistical Language Models (SLM) in our specific problem, i.e. the decoding of the Arabic words.

Chapter 8 introduces and analyzes a full fledged open-vocabulary, omni font-written Arabic OCR system analogous to HMM-based ASR systems. Evaluation of the system performance is also illustrated along with the detailed description of the Arabic database used in our experiments

Chapter 9 contains a conclusion and a discussion of the future work to be done to enhance the performance of our Arabic OCR system.

CHAPTER 2

Methodologies of OCR Systems; Theory and Literature Survey

In this chapter, we focus on the methodologies of OCR systems emphasizing on the off-line Arabic character recognition problem. The different modules of a typical OCR system; image acquisition, preprocessing, segmentation, features extraction, training and recognition, and post processing as discussed briefly in chapter one are discussed in more details in the following sections.

2.1. Image Acquisition

The off-line recognition system recognizes the text after it has been written or typed. The system may acquire the text using a video camera or a scanner. The latter is commonly used because it is more convenient, it introduces less noise into the imaging process, extra features such as automatic binarization and image enhancement can be coupled with the scanning process to enhance the resulting image text and, most importantly, it is more relevant to the problem of recognizing written script. [Khorsheed 2002]

Scanning at a higher resolution may be desirable but not practical as it would need too high storage space. Lower resolution and poor binarization can contribute to readability when essential features of characters are deleted or obscured. The resulting image can also be affected by the presence of marking or stains, or if the document has been faxed or copied several times. The latter causes a diminishing of contrast, the appearance of ‘salt and pepper’ noise, and the false appearance of text by becoming either thinner or thicker than the original document. [Khorsheed 2002]

2.2. Preprocessing

The recognition accuracy of OCR systems greatly depends on the quality of the input text and the lack of noise, even more so than with humans. Preprocessing operations are usually specialized image processing operations that transform the image into another with reduced noise and variation. Those operations include

2.2.1. Binarization

Binarization, or thresholding, is a conversion from a grey level image to a bi-level image. A bi-level image contains all of the essential information concerning the number, position and shape of objects while containing less information. The simple and straightforward method for thresholding is to select a fixed threshold, where gray-levels below this threshold is said to be black and levels above are said to be white [Khorsheed 2002]. For a high-contrast document with uniform background, a pre-chosen fixed threshold can be sufficient. However, a lot of documents encountered in practice have a rather large range in contrast. In these cases more sophisticated methods for thresholding are required to obtain a good result.

The best methods for thresholding are usually those that are able to vary the threshold over the document adapting to the local properties as contrast and brightness. However, such methods usually depend upon a multilevel scanning of the document, which requires more memory and computational capacity. Therefore such techniques are seldom used in connection with OCR systems, although they result in better images.

2.2.2. Noise Reduction

The noise, which is introduced by the optical scanning devices causes disconnected line segments, bumps and gaps in lines, filled loops etc. The distortion which includes local variations, rounding of corners, dilation and erosion is also a problem. Prior to the character recognition, it is necessary to eliminate these imperfections. There are many techniques to reduce the noise which can be categorized in two major groups; filtering and smoothing [Pratt 1991], [Gonzalez 2002].

2.2.2.1. Filtering

The aim of filtering is to remove noise and diminish spurious points as salt and pepper noise, usually introduced by the image acquisition device. Various spatial and frequency domain filters can be designed for this purpose. The basic idea is to convolve a predefined mask with the image to assign a value to a pixel based on its neighboring pixels.

A method that is often used is the median filter which is applied to reduce salt-and-pepper noise, see Figure (2.1) [Bunke 1997]. This is a small window which passes through all pixels in the image. The pixel in the centre is replaced by the median value of all the pixels in the region. Median filters are, however, hard deciding and cause considerable erosive distortion to the graphemes of written text which is generally harmful to the recognition performance.

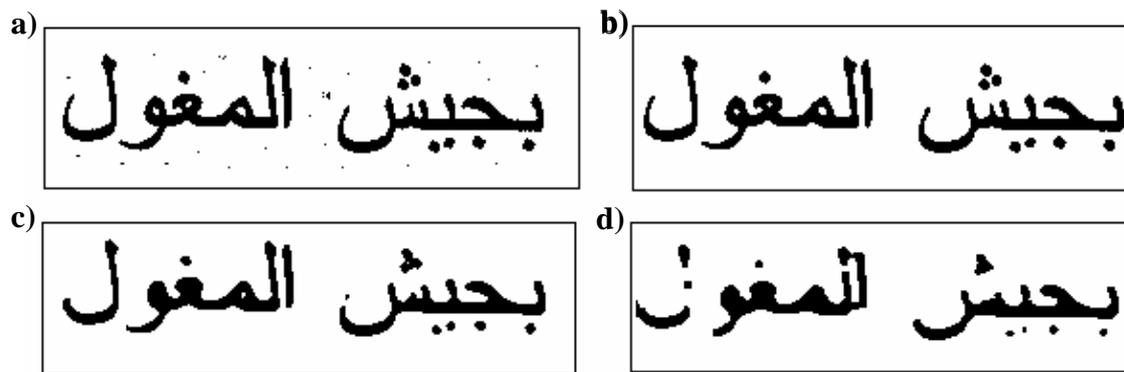


Figure (2.1): Results of removing noise from the original text image in (a) using median filter with window size of 3·3 in (b), 5·5 in (c) and 7·7 in (d).

2.2.2.2. Smoothing

This reduces the noise in an image using mathematical morphological operations. The basic idea behind the morphological operations is to filter the document image replacing the convolution operation by the logical operations. Various morphological operations can be designed to connect the broken strokes, decompose the connected strokes, smooth the contours, prune the wild points, thin the characters, and extract the boundaries. Therefore, morphological operations can be successfully used to remove the noise on the document images due to low quality of paper and ink.

Two operations are mainly used, Opening and Closing [Al-Badr 1995]. Opening opens small gaps or spaces between touching objects in an image; this will break narrow isthmuses and eliminate small islands. In contrast, Closing fills small gaps in

an image; this will eliminate small holes on the contour. Both Opening and closing apply the same basic morphology operations, namely, Dilation and Erosion, but in the opposite order [Gonzalez 2002].

2.2.3. Normalization

Normalization methods aim to remove the variations of the writing and obtain standardized data. The following are the basic methods for normalization.

2.2.3.1. Slant Normalization

This problem may be clearly seen in handwritten words, although machine-printed words with italic fonts suffer from the same problem. The most common method for slant estimation is the calculation of the average angle of near-vertical elements. In [Madhvanath 1999], vertical line elements from contours are extracted by tracing chain code components using a pair of one-dimensional (1-D) filters. Coordinates of the start and end points of each line element provide the slant angle.

Another study [Guillevic 94] uses an approach in which projection profiles are computed for a number of angles away from the vertical direction. The angle corresponding to the projection with the greatest positive derivative is used to detect the least amount of overlap between vertical strokes and, therefore, the dominant slant angle.

In [Bozinovic 89], slant detection is performed by dividing the image into vertical and horizontal windows. The slant is estimated based on the center of gravity of the upper and lower half of each window averaged over all the windows. Finally, a variant of the Hough transform is used by scanning left to right across the image and calculating projections in the direction of 21 different slants. The top three projections for any slant are added and the slant with the largest count is taken as the slant value. On the other hand, in some studies, recognition systems do not use slant correction and compensate it during training stage [Cote 98].

2.2.3.2. Size Normalization

It is used to adjust the character size to a certain standard. Methods of OCR may apply both horizontal and vertical size normalizations. Since the sizes of Arabic characters greatly vary, size normalization is often used to scale characters to a fixed

size and to center the character before recognition [Cowell 2002]. This is useful in recognition methods that are sensitive to variations in size and position like template matching and correlation methods [Al-Badr 1995].

Although, it was natural to normalize an isolated character into a fixed size rectangle, a word cannot be normalized in such a way without losing important temporal information [Cho 1995]. Hence, size normalization is done as a preprocessing step in isolated character recognition systems.

2.2.4. Skew Detection and Correction

Scanning a document so that text lines are within about three degrees of the true horizontal is acceptable. This is feasible if the document is aligned manually on the object glass of the scanner. Recent scanners are equipped with automatic feeders which may cause the document to rotate up to 20 degrees of the true horizontal. One of the first steps in attempting to read this document is to estimate the orientation angle, the skew angle, of the text lines. This process is called *skew detection*, and the process of rotating the document with the skew angle, in the opposite direction, is called *skew correction* [Khorshed 2002]. This skew has a detrimental effect on document on document analysis, document understanding, and character segmentation and recognition. Consequently, detecting the skew of a document image and correcting it are important issues in realizing a practical document reader.

The common, and perhaps the most efficient, approach to estimate the skew angle is to use the Hough Transform [Gonzalez 2002], [Nixon 2002]. This locates fragmented lines in a binary image. Given a binary image with a dominant text area, the detected lines will most probably go along the whole middle zone of the textual lines. Hence these lines have approximately the same skew as the reference lines of the text which define the skew of the whole page. Whenever the Hough transform is used, there is always a trade off between accuracy and speed. The more accurate the angles of the detected lines are, the more computation is required [Yu 1996], [Amin 2000].

Another approach to estimating a skew angle is based on using bounding boxes of *Connected Components*. This is a two step process. In the first step, all 8-neighbour connected pixels are grouped as distinct components, and the centre of gravity for each component is calculated. In the second step, a virtual line is drawn between various centers. The angle between this line and the horizontal represents the

skew angle. However, implementing this approach to Arabic text image may be misled by dots and diacritics located above and below the word characters [Khorsheed 2002]. Hence, removing dots and diacritics is a vital step for skew detection of Arabic text image [El-Adawy 2007].

2.2.5. Thinning and Skeletonization

Thinning is an essential step for many OCR systems whose main task is reducing patterns to their skeletons. Skeleton of a pattern is the set of points that has semi-equal distance from two or more points of the pattern contour. Thinning is often an efficient method for expressing structural relationships in characters as it reduces space and processing time by simplifying data structures [Al-Badr 1995]. Systems that recognize handwriting frequently use thinning to reduce the inherent variation in writing styles. Often, OCR systems traverse the skeleton of a word to simulate an on-line representation of it [Abuhaiba 1993]

Many thinning algorithms are susceptible to noise in that the generated skeletons are sensitive to even small variations in the input pattern. Another problem with many thinning algorithms that is particular to Arabic characters is that it reduces double and single dots (e.g., — and —) to the same shape (a short line) [Mahmoud 1991].

There are particular problems when dealing with Arabic script which have lower structural contents than Latin script, that is fewer stroke intersections and holes. Variations in the thickness of strokes in Arabic character cause further problems in the production of a thinned form which can be used to successfully extract structural information required in the later stages of character recognition. [Cowell 2001]

Two basic approaches are used for thinning; *pixel wise* and *nonpixel wise* thinning [Lam 1992]. Pixel wise thinning methods locally and iteratively process the image until one pixel wide skeleton remains. They are very sensitive to noise and may deform the shape of the character. On the other hand, the nonpixel wise methods use some global information about the character during the thinning. They produce a certain median or centerline of the pattern directly without examining all the individual pixels

Mahmoud et al. [Mahmoud 1991] proposed an algorithm for skeletonization of isolated Arabic characters that was based on clustering the character image into a set of a predetermined number of adjacent clusters and each cluster is a group of adjacent

pixels of the image. The skeleton is then made up of line segments that pass through the centers of the clusters. Altuwaijri and Bayoumi [Altuwaijri 1998] implemented a self-organizing neural network to cluster the Arabic character. Plotting the cluster centers and connecting adjacent clusters generated a straight-line sequence, which formed the skeleton.

2.2.6. Page Decomposition

Page decomposition is a sub-field of document analysis. Document analysis studies the structure of documents and the identification of the different logical parts in documents. Page decomposition is limited to separating the different lines of a text block and extracting the words and subwords.

The classical method for identifying text lines in an Arabic text image is to use a fixed threshold to separate the pairs of consecutive lines [Amin 1989]. This threshold is obtained using the distances between various baselines of the text. The median of different distance values is an appropriate selection. An alternative approach [Khorshed 2002] is to use the horizontal projection and look for the pixel lines that have a density of zero and then consider that every text line is situated between two blocks of zero density pixel lines. This method is enhanced by identifying the lines of pixels that have the largest density in the text. The upper and lower parts are then analyzed with respect to these lines [Sarfraz 2003].

The next phase is to segment a text line into words and sub-words. Words and sub-words are determined by inspecting the vertical projection. An average threshold value computed from all vertical gaps is used to determine whether a spacing is an inter-word spacing or an intra-word spacing [Altuwaijri 1994], [Al-Badr 1995], [Khorsheed 2002], [Sarfraz 2003].

2.3. Segmentation

After the preprocessing stage, many OCR systems isolate the individual characters or strokes before recognizing them. It is one of the hardest, crucial, and time-consuming phases. It represents the main challenge in many Arabic character recognition systems, even more than the recognition process itself. It is considered as the main source of recognition errors. A poor segmentation process produces misrecognition or rejection [Zeki 2005].

As it is so difficult and with a great influence on the final recognition rate, many researchers tried to avoid this stage by two approaches:

- **Isolated Characters Approach** [Amiri 2003], [Amor 2006], where they assume that the characters are already segmented. This approach is useful for academic purposes, and for the testing of certain pattern recognition techniques. This will not lead, however, to a practical machine-printed OCR system. The issue is that the character recognition cannot be dealt with, in isolation from the segmentation process both in the training and the recognition phases. Recognition of artificially segmented characters could never lead to practical results, as the segmented characters should be generated from a real segmentor with all its impurities.
- **Holistic (global) Approach**, in which the recognition is globally performed on the whole representation of words and where there is no attempt to identify characters individually, that is why it is also known as segmentation-free approach .

Some attempts even went to the extent of proposing new fonts to generate Arabic script instead of cursive Arabic script whereby the characters can be segmented with simple vertical white cuts to help in automatic document understanding [Abuhaiba 2003].

Although the Segmentation methods have been developed remarkably in the last decade and a variety of techniques have emerged, segmentation of cursive script into characters is still an unresolved problem [Zeki 2005]. Character segmentation strategies are divided into two categories [Elyacobi 1999]; explicit and implicit segmentation.

2.3.1. Explicit Segmentation

In the explicit segmentation (or dissection segmentation), words are explicitly or externally segmented into characters or primitives (like strokes intersection points, inflection points, and loops) which are then recognized individually. Contextual high level knowledge (lexical, syntactic or semantic knowledge) is then used to ensure the proper word identification. This approach is usually more expensive due to the increased complexity of finding optimum word hypotheses. Projection analysis, connected component processing, and white space and pitch finding are some of the

common dissection techniques used in Arabic OCR systems [Khorsheed 2002], [Zeki 2005].

2.3.2. Implicit Segmentation

In the implicit segmentation, characters are segmented while being recognized, hence, it is also called recognition-based segmentation or straight segmentation. It searches the image for components that match predefined classes. Segmentation is performed by use of recognition confidence, including syntactic or semantic correctness of the overall result. The advantage of the recognition-based segmentation technique is that no accurate character segmentation path is necessary. Thus, it bypasses the character segmentation stage and the recognition errors are mainly due to failures during the classification stage [Zeki 2005]. In this approach, two different methods can be employed; method that make some search process and method that segment a feature representation of the image

The first one attempts to segment words into characters or other units without use of feature based dissection algorithms. Rather, the image is divided systematically into many overlapping pieces without regard to content. The basic principle of this method is to use a mobile window of variable width to provide the tentative segmentations which are confirmed (or not) by the classification [Al-Badr 1995], [Zeki 2005].

On the other hand, the second method segments the image implicitly by classification of subsets of spatial features collected from the image as a whole. In this method, HMM-based approaches represent the word model as a change of sub-HMM's. The object of HMM's is to model variations in printing or cursive writing as an underlying probabilistic structure, which is not directly observable [Gilloux 1993], [Mohamed 1996].

2.4. Feature Extraction

Feature extraction is the problem of “extracting from the raw data the information which is the most relevant for classification purposes, in the sense of minimizing the within class pattern variability while enhancing the between class pattern variability” [Devijver 1982].

The extracted features must be invariant to the expected distortions and variations that character may have in a specific application. Also the phenomenon called the curse of dimensionality [Duda 2001] cautions us that with a limited training set, the number of features must be kept reasonably small if a statistical classifier is to be used. A rule of thumb is to use five to ten times as many training patterns of each class as dimensionality of each class [Jain 1982].

Feature extraction is one of the most difficult and important problems of pattern recognition and an important step in achieving good performance of character recognition system. A feature extraction method that proves to be successful in one application domain may turn out not to be very useful in another domain. In practice, the requirements of a good feature extraction method make selection of a good feature extraction method for a given method a challenging task.

The choice of feature extraction method limits or dictates the nature and output of the preprocessing step. Some feature extraction methods work on gray level sub images of single characters, while other work on solid 4-connected or 8-connected symbols segmented from the binary raster image, thinned symbols or symbol contours. Furthermore the type of features extracted must match the requirements of the chosen classifier. [Trier 1996]

Feature types can be categorized into three main groups: structural features, statistical features and global transformations.

2.4.1. Structural Features

Structural features are the most popular features investigated by researchers [Govindan 1990]. Structural features describe a pattern in terms of its topology and geometry by giving its global and local properties. Structural features can highly tolerate distortions and variations in writing styles (multi-font) but extracting them from images is not always easy. A combination of several structural features could enhance the overall recognition rate. The structural features used in the literature depend on the kind of pattern to be classified [Al-Badr 95].

In the case of characters, the features include strokes and bays in various directions, end points, intersection points, loops, dots, and zigzags. Some researchers use the height, width, number of crossing points, and the category of the pattern (character body, dot, etc.); the presence and number of dots and their position with respect to the baseline; the number of concavities in the four major directions, the

number of holes, and the state of several key pixels; the number of strokes or radicals and the size of the stroke's frame; and the connectivity of the character.

In the case of primitives, the features extracted include the direction of curvature (e.g., clockwise), the type of the feature point at which a curve was segmented (cross point, etc.); the direction, slope, and length of strokes; the length of a contour segment, the distance between the start and end point of the contour projected on the x- and y-axis, and the difference in curvature between the start and end points; and the length of vectors in the four major directions that approximate a curve.

Sometimes the pattern is divided into several zones and several types of geometric features in each zone are registered and counted, with some features constrained to specific zones. Those features include the number of concavities, holes, cross points, loops, and dots; the number and length of the contour segments; the zone with maximum (minimum) number of pixels; and concavities in the four major directions.

2.4.2. Statistical Features

Statistical features can be easily and quickly extracted from a text image; they can tolerate moderate noise and variation, and systems may be trained automatically for new fonts. The main difficulty, though, is to find an optimal set of features that would properly partition the space and are efficient to extract and classify.

The statistical features used for Arabic text recognition include: zoning, characteristic loci, crossings and moments. [Al-Badr 1995]

- **Zoning**

Features are extracted by dividing a character into overlapping or non-overlapping regions and using the densities of pixels in these regions as features.

- **Characteristic Loci**

It counts the number of zero and one segments a vertical line crosses in the pattern and the length of each segment.

- **Crossings and Distances**

It counts the number of times a set of radial lines at different angles (e.g., 16 lines at 0, 22.5°, 45°, etc.) cross the pattern. This method tolerates distortions and small

variations and is fast to calculate. Also, the distance of line segments from a given boundary, such as the upper and lower portion of the frame, can be used as statistical features.

- **Moments**

Moments is one of the most popular statistical approaches used widely in visual pattern recognition ever since they were introduced by Hu in 1961. Abstractly, when a pattern is visualized as an n^{th} degree polynomial, the n^{th} moments are the coefficients of that polynomial. Invariant moments of a pattern about its center of gravity are invariant to translation, rotation and scaling [Gonzalez 2002]. Moments are considered as series expansion representation since the original image can be completely reconstructed from the moment coefficients. Invariant moments are sensitive to any change and multi-font recognition [Khorsheed 2002].

2.4.3. Global Transformations:

The transformation scheme converts the pixel representation of the pattern to an alternate more abstract representation which reduces the dimensionality of features. In general, transformation schemes can be easily applied and tolerate noise and variation. However, they sometimes require the use of additional features, in conjunction, to obtain high recognition rates. [Al-Badr 1995]

The most common transformation techniques used for off-line Arabic character recognition are:

- **Projections**

Projection histograms can be interpreted as horizontal and vertical bar masks. They transform a two-dimensional image into two one-dimensional histograms. Due to their sensitivity to small rotations in the images, projection histograms cannot be used as the only features in classification. If used in combination with other features, histograms may offer useful additional information.

- **Fourier Descriptors**

Fourier descriptors are used for characterizing the outer boundary of a character. The first coefficients describe the most coarse shape of the character. The amount of details is increased as more coefficients are included. Those descriptors are

invariant to translation, rotation, and scaling and can tolerate moderate boundary variations.

- **Coding**

One of the most popular coding schemas used is Freeman's chain code. This coding is essentially obtained by mapping the strokes of a character into a 2-D parameter space, which is made up of codes. [Khorsheed 2002]

- **Hough Transform**

The Hough Transform is a standard technique for detecting features as lines in a given image. Its principal concept is to define a mapping between an image space and a parameter space. This technique is known for its capacity of absorption of distortions as well as noises. These features are used to recognize Arabic printed characters in their different shapes [Amor 2006], [Touj 2003].

- **Gabor Transform**

Gabor transform has been widely used in pattern recognition domain. It is a variation of the windowed Fourier transform. In this case, the window used is not a discrete size but is defined by a Gaussian function. [Hamamoto 1996], [Su 2007]

2.5. Classification:

As in many areas of Image Analysis, Arabic character recognition systems extensively use the methodologies of pattern recognition, which assigns an unknown sample into a predefined class. There are four main techniques used,

1. Template Matching,
2. Statistical Techniques,
3. Syntactic Techniques,
4. Neural Networks.

The above techniques use either holistic or analytic strategies for the training and recognition stages [Amin 1997], [Dehghan 2001] where,

- **Holistic (Global) strategy** employs top down approaches for recognizing the full word, eliminating the segmentation problem. This approach can be used for particular applications where the size of the vocabulary is small (as would be the

case in a typical application such as the recognition of the legal amount in cheques provided the words in the phrase are well isolated).

- **Analytic strategy** employs bottom up approaches. Word is considered as a sequence of smaller components like strokes, characters or graphemes. The word is identified by extracting and recognizing its constituent components. Explicit or implicit segmentation algorithms are required for this strategy. This not only adds extra complexity to the problem, but also introduces segmentation error to the system. However, with the cooperation of segmentation stage, the problem is reduced to the recognition of simple isolated characters or strokes, which can be handled for recognition applications that involve large-size vocabularies

| Holistic Strategy | Analytic Strategy |
|--------------------------|--------------------------------------------|
| Whole word recognition | Subword (strokes or graphemes) recognition |
| Limited vocabulary | Unlimited vocabulary |
| No segmentation | Requires explicit or implicit segmentation |

Table (2.1): Classification strategies; Holistic versus Analytic

2.5.1. Template Matching:

Template matching and correlation methods basically compare a pattern pixel-by-pixel to a set of pattern templates; the pattern is considered to belong to the class of the template to which it is most similar.

A gray-level or binary input character is directly compared to a standard set of stored prototypes. According to a similarity measure (e.g., Euclidean, Mahalanobis, Jaccard, or Yule similarity measures, etc.), a prototype matching is done for recognition. The matching techniques can be as simple as one-to-one comparison or as complex as decision tree analysis in which only selected pixels are tested. Although direct matching method is intuitive and very fast to execute, the recognition rate of this method is very sensitive to noise.

An alternative method is the use of deformable templates, where an image deformation is used to match an unknown image against a database of known images.

In [Jain 97], two characters are matched by deforming the contour of one to fit the edge strengths of the other. A dissimilarity measure is derived from the amount of deformation needed, the goodness of fit of the edges, and the interior overlap between the deformed shapes.

2.5.2. Statistical Techniques (Decision Theoretic)

In the statistical approach, each pattern is represented in terms of d -features or measurements and is viewed as a point in a d -dimensional space. The goal is to choose those features that allow pattern vectors belonging to different categories to occupy compact and disjoint regions in a d -dimensional feature space. Given a set of training patterns from each class, the objective is to establish decision boundaries in the feature space which separate patterns belonging to different classes. In the statistical decision theoretic approach, the decision boundaries are determined by the probability distributions of the patterns belonging to each class, which must either be specified or learned. [Jain 2000], [Duda 2001]

Statistical decision theory is concerned with statistical decision functions and a set of optimality criteria, which maximizes the probability of the observed pattern given the model of a certain class [Devijver 1982]. Statistical techniques are mostly based on three major assumptions,

- 1) Distribution of the feature set is Gaussian or in the worst-case uniform.
- 2) There are sufficient statistics available for each class.
- 3) Given ensemble of images, one is able to extract a set of features $\{f_i\} \in \{1, \dots, d\}$ which represents each distinct class of patterns.

The major statistical approaches applied in the OCR field are listed below.

2.5.2.1. Nonparametric Recognition

This method is used to separate different pattern classes along hyperplanes defined in a given hyperspace. The best-known method of nonparametric classification is the nearest neighbor classifier and is extensively used in OCR [Blue 1994]. It does not require *a priori* information about the data. An incoming pattern is classified using the cluster, whose center is the minimum distance from the pattern over all the clusters.

2.5.2.2. Parametric Recognition

Since a priori information is available about the characters in the training data, it is possible to obtain a parametric model for each character. A common statistical method used in OCR is the Bayesian classification [Al-Yousefi 1992], [Duda 2001]. Bayes' classifier minimizes the total average loss. Given an unknown symbol described by its feature vector, the probability that the symbol belongs to a certain class is computed for all classes. The symbol is then assigned the class which gives the maximum probability.

For this classifier to be optimal, the probability density functions of the symbols of each class must be known, along with the probability of occurrence of each class. The latter is usually solved by assuming that all classes are equally probable. The density function is usually assumed to be normally distributed, and the closer this assumption is to reality, the closer the Bayes' classifier comes to optimal behavior.

Bayes classifier for Gaussian classes is specified completely by the mean vector and covariance matrix of each class. These parameters specifying the classifiers are obtained through a training process. During this process, training patterns of each class is used to compute these parameters and descriptions of each class are obtained.

2.5.2.3. Hidden Markov Modeling (HMM)

Hidden Markov Models have been found extremely efficient for a wide spectrum of applications, especially speech processing [Rabiner 1986, 1989]. This success has motivated researchers to implement HMM's in character recognition. HMM is qualified as a suitable tool for cursive script recognition for its capability of naturally modeling temporal information [Bunke 1995], [Cho 1995].

HMM is a doubly stochastic process, with an underlying stochastic process that is not observable (hence the word hidden), but can be observed through another stochastic process that produces the sequence of observations. The hidden process consists of a set of states connected to each other by transitions with probabilities, while the observed process consists of a set of outputs or observations, each of which may be emitted by each state according to some probability density function (pdf).

Depending on the nature of this pdf, several HMM classes can be distinguished. If the observations are naturally discrete or quantized using vector quantization [Gray 1984, 1998], and drawn from a codebook, the HMM is said to be discrete. If these observations are continuous, we are dealing with a continuous HMM, with a continuous pdf usually approximated by a mixture of normal distributions. Another family of HMM's, a compromise between discrete and continuous HMM's, are semi-continuous HMM's that mutually optimize the vector quantized codebook and HMM parameters under a unified probabilistic framework [El- Yacoubi 1999].

There are many ways that HMM's had been applied to character and word recognition. A state can represent a character class or it can represent a sub character symbol or feature. A model can represent a character, a word or even a complete lexicon. Classification methods therefore vary, a set of character models may be used in which case a maximum likelihood solution can be found by scoring each model as to how close it matches the unknown pattern. Alternatively the classes may be distributed within a model such that the classification depends on the state sequence of the model which best fits the pattern. HMM is covered in more details in chapter 6.

2.5.3. Syntactic (Structural) Techniques

In many recognition problems involving complex patterns, it is more appropriate to adopt a hierarchical perspective where a pattern is viewed as being composed of simple sub-patterns which are themselves built from yet simpler sub-patterns. The simplest/elementary sub-patterns to be recognized are called primitives and the given complex pattern is represented in terms of the interrelationships between these primitives.

In syntactic pattern recognition, a formal analogy is drawn between the structure of patterns and the syntax of a language. The patterns are viewed as sentences belonging to a language, primitives are viewed as the alphabet of the language, and the sentences are generated according to a grammar. Thus, a large collection of complex patterns can be described by a small number of primitives and grammatical rules. The grammar for each pattern class must be inferred from the available training samples. [Al-Badr 1995]

The use of syntax to express structure is a disadvantage of the structural approach, since patterns can have infinite variations and do not always adhere to the strict mathematical constraints set by the formal languages theory [Govindan 1990].

2.5.4. Neural Networks

A neural network (NN) is defined as a computing architecture that consists of a massively parallel interconnection of adaptive “neural” processors. Because of its parallel nature, it can perform computations at a higher rate compared to the classical techniques. Because of its adaptive nature, it can adapt to changes in the data and learn the characteristics of input signal. An NN contains many nodes. The output from one node is fed to another one in the network and the final decision depends on the complex interaction of all nodes. In spite of the different underlying principles, it can be shown that most of the NN architectures are equivalent to statistical pattern recognition methods.

Arabic OCR using neural networks can simply cluster the feature vectors in the feature space, or they can integrate feature extraction and classification stages by classifying characters directly from images. Generally speaking, the common architecture of NN’s used in Arabic OCR is a network with three layers: input, hidden and output. The number of nodes in the input layer varies according to the dimensionality of the feature vector or the segment image size. The number of nodes in the hidden layer governs the variance of samples that can be correctly recognized by this NN. [Khorsheed 2002]

A brief description and comparison of the mentioned classification approaches is summarized in the following table,

| Approach | Representation | Recognition function | Typical criterion |
|----------------------|-----------------------|----------------------------------|--------------------------|
| Template Matching | Pixels | Correlation, Distance measure | Classification error |
| Statistical | Features | Discriminate function | Classification error |
| Syntactic/Structural | Primitives | Rules/ Grammar | Acceptance error |
| Neural Networks | Pixels, Features | Network function | Mean square error |

Table (2.2): Classification Approaches Comparison

2.5.5. Hybrid Approaches

To improve recognition performance, a trend now is to build hybrid systems, which use diverse feature types and combinations of classifiers arranged in layers. Using a fusion method between hybrid approaches, improved system performance can be achieved. [Magdy 2007]

As a result, increasingly many researchers now use combinations of the above feature types and classification techniques to improve the performance of Arabic OCR systems, but most of them applied it on handwritten Arabic scripts. These researches found it a very promising approach in improving the recognition accuracy. [Al-Badr 1995], [Gouda 2004]

2.6. Post-Processing

The review of the recent OCR research indicates minor improvements when only shape recognition of the character is considered. Therefore, the incorporation of context and shape information in all the stages of OCR systems is necessary for meaningful improvements in recognition rates. This is done in the post-processing stage with a feedback to the early stages of OCR.

Post-processing is used to improve the word recognition rate (as opposed to character recognition rate) by correcting the output of the OCR system based on linguistic (Natural Language Processing (NLP)) knowledge. The linguistic level could be either on character level, lexical level (spell checker), morphological level (using a morphological analyzer, to detect all possible word forms), and up to the syntactic, higher level semantic, and discourse levels.

On the character level, statistical information derived from the training data and the syntactic knowledge such as n-grams improves the performance of the matching process.

On the lexical level, spell checking and correction operations are used. Spell checking can be as simple as looking up words in a lexicon. To improve the speed of lookup, a lexicon is sometimes represented as a tree. When spell checking fails, the Viterbi algorithm is used to find alternate words whose characters, with a high probability, can be interchanged with the original ones, using a HMM for each word [Amin 1989].

Morphological analyzer is used to first remove prefixes and suffixes and then reduce the resulting word to its root. This method of representing a word as a root and a pattern (the pattern specifies how the root relates to the original word) can considerably reduce the size of the dictionary (at the expense of some computation) [Al-Badr 1995], [Sari 2002].

In sentence level, the resulting sentences obtained from the output of the recognition stage can be further processed through parsing in the post-processing stage to increase the recognition rates. The recognition choices produced by a word recognizer can be represented by a graph as in the case of word level recognition. Then, the grammatically correct paths over this graph are determined by using syntactic knowledge. However, post-processing in sentence level is rather in its infancy, especially for languages other than English, since it requires extensive research in linguistics and formalism in the field of AI. [Gouda 2004]

As a technique for post-processing, recognition rates confusion matrix is extracted from the OCR output. This information is rewritten in the form of production rules. These rules are later used to detect the substituted characters [Sari 2002].

2.7. Machine-Printed Arabic OCR Survey

Earlier surveys discussed both machine-print and handwriting, with much more discussion of machine-print [Al-Badr 1995], [Amin 1998], [Khorsheed 2002]. Within the past 10 years, many approaches had been developed to produce reliable Arabic OCR systems. The most obvious systems can be briefed as follows.

In 1998, Al-Badr et al. presented a holistic recognition system based on shape primitives that were detected with mathematical morphology operations. They used a single font, and training was on idealized isolated samples. These operators are normally very sensitive to noise generated from the scanner, and could also degrade in low resolutions. They achieved 27% CER for a limited lexicon. [Al-Badr 1998]

BBN developed a script-independent methodology for OCR, which had been tested on English, Arabic, Chinese, Japanese, and other languages [Makhoul 1998] using their HMM-based speech recognizer with statistical features from image frames (vertical strips). Only the lexicon, language model, and training data depended on the language. In 1999, Bazzi et al. presented a modified system for English and Arabic in

which the lexicon was unlimited. The DARPA Arabic OCR Corpus was used for testing. The system achieved a CER of 4.7% for an omnifont database. [Bazzi 1999], [Lu 1999].

In 1999, Khorsheed et al, have presented a technique for extracting structural features from word's skeleton for recognition without prior segmentation. After preprocessing, the skeleton of the binary word image was decomposed into a number of segments in a certain order. Each segment was transformed into a feature vector. The target features were the curvature of the segment, its length relative to other segment lengths of the same word, the position of the segment relative to the centroid of the skeleton and detailed description of curved segments. The result of this method was used to train the Hidden Markov Model to perform the recognition. [Khorsheed 1999] This system depends highly on a predefined lexicon which acts as a look-up dictionary, so it can be applied only to a single font and for a very limited vocabulary size.

In 1999, Gillies et al. adopted an explicit segmentation followed by recognition approach The Arabic cursive word was segmented into basic primitives which could be a part of a character or a whole character. The characters were then reconstructed from the recognized primitives. Using a chain code features and neural network classifier, the system performance applied on multi-fonts Arabic database achieved was about 7% CER at scanning resolution 200dpi and limited lexicon. [Gillies 1999]

In 2000, Khorsheed et al. transformed each word in a certain lexicon into a normalized polar image, then a two dimensional Fourier transform was applied to the polar image. The resultant spectrum tolerates variations in size, rotation or displacement. Each word was represented by a template that includes a set of Fourier coefficients. The recognition was based on a normalized Euclidean distance from those templates. This method showed CER of 10% but for a fixed lexicon size. [Khorsheed 2000]

In 2001, Abdelazim recognized the Arabic word using a segmentation-free sequential algorithm, and an entropy-based stopping criteria for character-to-character recognition. The CER achieved average 3.5 % when the system was tested on 10 different fonts. [Abdelazim 2001]

In 2001, Cheung et al. proposed an Arabic OCR system, which uses a recognition-based segmentation technique. A mobile window of variable width was

used to provide the tentative segmentations which were confirmed (or not) by the classification. Chain codes were used in the feature extraction stage while string matching was used in the classification stage achieving 10 % CER. [Cheung 2001]

In 2002, Hamami and Berkani developed a structural approach to handle many fonts and it included rules to prevent over-segmentation. The developed system can recognize multi-font of Arabic script at about 1.44% CER. [Hamami 2202]

In 2003, Sarfraz et al. presented a technique for the automatic recognition of Arabic printed text using NN's. The main features of the system were preprocessing (noise removal and skew detection and correction) of the text, segmentation of the text to individual characters, feature extraction using moment invariant technique and recognition using RBF Network. CER reached by the system is about 27% for a uni-font database. [Sarfraz 2003]

In 2007, Khorsheed developed a system to recognize cursive Arabic text. The proposed system was based on HMM Toolkit basically designed for speech recognition purpose. The system was segmentation-free with easy to extract statistical features. The achieved system performance was about 5% CER for multi-font Arabic database. [Khorsheed 2007]

It is remarkable that OCR systems in general are being developed since decades, tens of research Arabic OCR pilots have been produced by the academia, and handful Arabic OCR products are even available in the market. However, a reliable Arabic OCR software that works on real-life (multi-font, multi-size, maybe noisy,...) documents at a practically acceptable average WER within 3% is yet away from being available in the market. [Kanungo 1997], [Windows Magazine Middle East, April 2007]

CHAPTER 3

Arabic Lines and Words Decomposition

Given font/type-written text rectangle bitmaps extracted from digitally scanned pages, inferring the boundaries of lines hence complete words is a preprocessing vital to whatever OCR system while the recognition process itself as well as the post-processing necessary for producing the recognized text.

Decomposing the target text block into lines hence into words is perhaps the most fundamental such preprocessing where errors add in the overall WER of the whole OCR system. This can be shown as follows:

$$\begin{aligned} a &= 1 - e = (1 - e_D) \cdot (1 - e_R) = 1 - (e_D + e_R - e_D \cdot e_R) \\ \because e_D \cdot e_R &\ll 1 && \text{Eq. (3.1)} \\ \therefore e &\approx e_D + e_R \end{aligned}$$

where e_D is the decomposition WER, e_R is the recognition WER, and e is the overall WER. It is hence obvious how critical is the reliability of the lines & words decomposer to the viability of the whole OCR system.

Histogram-based methods are commonly used for lines & words decomposition. Despite being commonly used, simple-to-implement, and computationally efficient, the histogram-based approach may be vulnerable to some idiosyncrasies of real-life documents especially as per noise and complex textual formats [Al-Badr 1995].

Based on this approach and to overcome its shortcomings, especially with Arabic script, a more robust algorithm for lines/words decomposition especially in Arabic, or Arabic dominated text rectangles from real-life multifold/multisize

documents is developed and discussed in this chapter. This algorithm has proved robust on realistic Arabic, or Arabic dominated, documents after the extensive experimentation presented in section 3.2.

3.1. Enhanced Histogram-Based Lines & Words Decomposition Algorithm

The enhanced algorithm presented here is simply composed of a sequential series of processing procedures whose overall logical structure may be briefed in the following steps:

1. Filtering.
2. Getting the Darkness & Brightness Centroids.
3. Writing-Lines Decomposition.
 - 3.1. Physical Writing Lines Decomposition.
 - 3.1.1. Initial Lines Decomposition.
 - 3.1.2. Refined Lines Decomposition.
 - 3.2. Logical Writing-Lines Concatenation.
4. Writing-Lines Grouping.
5. Part-of-Words Decomposition.
6. Full Words Concatenation.
 - 6.1. Eliminating Rouge Spaces Among Part-of-Words.
 - 6.2. Clustering Inter-word/Intra-Word Spaces.

These steps are presented in what follows as a sequence of cascaded steps applied to the sample Arabic text rectangle bitmap shown in Figure (3.1) as a running example visualizing the dynamics of that decomposition process.



Figure (3.1): Sample Arabic text rectangle bitmap; Laser-printed, photocopied once, and scanned at 300 dpi & 256 gray shades.

3.1.1. Step 1; Filtering

The target text rectangle bitmap is passed through a median filter in order to clean the salt & pepper noise [Pratt 1991], [Gonzalez 2002] typically occurring in noisy images extracted, for example, from photocopied documents. In our study, a 5×5 median filter has proved effective for cleaning input scanned text images like our running example in Figure (3.1) which is turned after that filtering into the one just below in Figure (3.2).



Figure (3.2): Sample text rectangle bitmap after median filtering.

3.1.2. Step 2; Getting the Darkness & Brightness Centroids

For deciding about the *blankness* of horizontal/vertical pixels-scan-lines in later steps of this algorithm, it is necessary to accurately determine the centroid of darkness c_d , and the centroid of brightness c_b . We apply the famous K-means clustering algorithm on the gray values histogram of the pixels population in the target text image, on the standard scale from 0 (darkest)-to-255 (brightest), to accurately acquire c_d and c_b .

Figure (3.3) shows such a histogram of our running example along with the darkness and brightness centroids resulting from the application of the K-means algorithm. For two-color images $c_d = 0$, and $c_b = 255$.

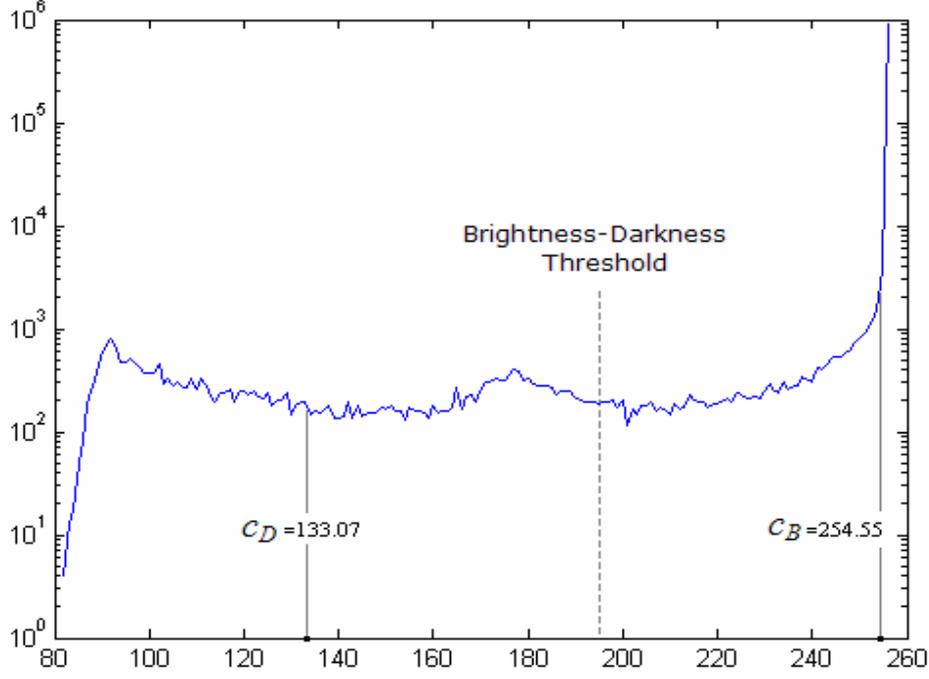


Figure (3.3): Gray values histogram of the pixels population of the sample text rectangle.

3.1.3. Step 3; Writing-Lines Decomposition

Getting the vertical range $[y_{i,bottom}, y_{i,top}]$ of each writing line L_i , whose height $h_i = y_{i,top} - y_{i,bottom}$ is accomplished through the two following sub-steps; 3.1 and 3.2:

3.1.3.1. Step 3.1; Physical Writing Lines Decomposition

This sub-step is trivially done once we get rationally able to decide about whether a given horizontal scan-line l_j is *blank* or *non-blank*. Intuitively, l_j is blank if the following condition is true;

$$\sum_{k=x_{Right_i}-w_i+1}^{k=x_{Right_i}} (255 - p(k, j)) \leq w_i \cdot (255 - c_B) \quad \text{Eq. (3.2)}$$

where w_i is the effective width of L_i to which l_j belongs. This means the decision is tolerable to noise within the amount that locates c_B in step 2. As w_i is yet unknown till the writing-lines decomposition is done, we go around this circular problem by dividing sub-step 3.1 in turn into the following two sub-steps; 3.1.1 and 3.1.2:

A. Step 3.1.1; Initial Lines Decomposition

Here, initial writing lines L_i^* with vertical ranges $[y_{i,bottom}^*, y_{i,top}^*]$ and heights $h_i^* = y_{i,top}^* - y_{i,bottom}^*$, are computed with the blankness condition modified to:

$$\sum_{k=0}^{k=W-1} (255 - p(k, j)) \leq W \cdot (255 - c_B) \quad \text{Eq. (3.3)}$$

Noting that $w_i \leq W$, all the scan-lines decided to be non-blank are truly so, according to Eq. (3.2), while few short scan-lines may be falsely decided blank and have to be reconsidered in the next sub-step 3.1.2. Figure (3.4) shows the result of sub-step 3.1.1 applied on our running text-image example.



Figure (3.4): Sample text rectangle bitmap after initial lines decomposition.

B. Step 3.1.2; Refined Lines Decomposition

Now, for each scan-line l_j judged as blank in the previous sub-step 3.1.1; it gets re-tested in this sub-step 3.1.2 with the blankness condition modified to:

$$\sum_{k=x_{Right\ n(j)}^{x_{Right\ n(j)}}} (255 - p(k, j)) \leq w_{n(j)} \cdot (255 - c_B) \quad \text{Eq. (3.4)}$$

where $n(j)$ is defined as the index of the nearest writing-line, as obtained from the sub-step 3.1.1, to the scan-line l_j . After this second pass of blankness decisions; the writing lines L_i are accordingly re-labeled as shown in Figure (3.5).

| |
|----------------------------------------------------|
| دراسة حول تقسيم النص العربي الممسوح ضوئياً |
| إلى سطور وكلمات آلياً |
| محمد عطية |
| استشاري تقنيات اللغة بشركة RDI |
| محمد المحلاوي |
| مدرس مساعد بالأكاديمية العربية للعلوم والتكنولوجيا |

Figure (3.5): Sample text rectangle bitmap after refined lines decomposition.

For initially decomposed writing-lines L_i^* ; their vertical darkness histograms $g_i^*(k)$ given by

$$g_i^*(k) = \sum_{j=y_{i,bottom}^*}^{j=y_{i,top}^*} (255 - p(k, j)) \quad \text{Eq. (3.5)}$$

are pruned from both tails in a *farther-first* manner to avoid false extensions due to any parasitic signals (e.g. noise) while computing the effective widths w_i of those initially decomposed writing-lines. In our experiments; it has been found best to prune masses beyond $\mu_i \pm 2 \cdot \sigma_i$ of the mentioned histograms.

3.1.3.2. Step 3.2; Logical Writing-Lines Concatenation

Orthography in general, and Arabic script in specific, is full of dots and diacritics over and under the basic bodies of graphemes [Al-Badr 1995], [Attia 2004]. This may, especially with short lines, cause sub-step 3.1 produce false *thin* physical writing-lines which logically belong to thick neighbors. In such cases, these neighboring physical writing-lines have to be unified in logical ones.

To realize that concatenation, a concatenation test is iterating until it states false for all the writing-lines in the target text-image. For a writing-line L_i , the concatenation conditions with its neighbors $L_{i-1}; i > 0$ and $L_{i+1}; i < i_{max}-1$ are respectively;

$$Concat_{Before} = \left(\frac{h_i}{h_{i-1}} < \alpha \right) AND \left(\frac{y_{i-1,bottom} - y_{i,top}}{h_{i-1}} < \beta \right) \quad \text{Eq. (3.6)}$$

$$Concat_{After} = \left(\frac{h_i}{h_{i+1}} < \alpha \right) AND \left(\frac{y_{i,bottom} - y_{i+1,top}}{h_{i+1}} < \beta \right) \quad \text{Eq. (3.7)}$$

If *neither* of the conditions is TRUE, no concatenation occurs and the test advances to the next writing-line. If *either* condition is TRUE, concatenation happens with the corresponding line, and the test resumes iterating at the first writing-line on top of text image. If *both* conditions are TRUE, concatenation happens with the closer writing-line, and the test resumes iterating at the first writing-line on top of text image.

As per the nature of Arabic script, our extensive experiments show that, the best values for α and β are 0.35 and 0.30 respectively. Figure (3.6) shows the resulting logical writing lines of our running example after applying sub-step 3.2.

| |
|----------------------------------------------------|
| دراسة حول تقسيم النص العربي الممسوح ضوئياً |
| إلى سطور وكلمات آلياً |
| محمد عطية |
| استشاري تقنيات اللغة بشركة RDI |
| محمد المحلاوي |
| مدرس مساعد بالأكاديمية العربية للعلوم والتكنولوجيا |

Figure (3.6): Sample text rectangle bitmap after logical writing lines concatenation.

3.1.4. Step 4; Writing-Lines Grouping

Heights of writing lines are usually considered as a rough, never an accurate, indication of the font size they are written in. So, gathering the writing lines in a given text image in groups, where the heights of these lines in the same group are of a sufficiently small variance, may be useful for the recognition process that might tune some parameters for each group according to say its average height.

Far more importantly, this grouping is a necessary auxiliary step towards deciding whether a given space between successive separate word segments is an *inter* or *intra* word space. See step 6 .

We here group the writing-lines so that the standard deviation of the writing-lines heights σ_g in each group and the average of these heights h_g satisfy the condition $\sigma_g \leq \gamma \cdot h_g$.

In our experiments, γ is tuned to 0.2. The 1st and 2nd writing lines from the top of Figure (3.6) are gathered in one group, while the rest are gathered in another one.

3.1.5. Step 5; Part-of-Words Decomposition

In this step, part-of-word segments on each writing line L_i are separated by the vertical spaces among them. This is easily done once we get able to decide whether a given vertical scan-line segment at $x = k$ within the range of L_i is *blank* or *non-blank*. Similar to formula (3.2) in step 3.1; such a blankness condition is formulated as:

$$\sum_{j=y_{i,bottom}}^{j=y_{i,top}} (255 - p(k, j)) \leq h_i \cdot (255 - c_B) \quad \text{Eq. (3.8)}$$

Figure (3.7) shows the result of decomposing the part-of-word segments in our running example, with each marked by a surrounding rectangle.



Figure (3.7): Sample text rectangle bitmap after part-of-words decomposition.

3.1.6 Step 6; Full Words Concatenation

Arabic script is *quasi-cursive*; i.e. words are composed of connectedly written characters, however, the occurrence of some characters mandates putting an end to their part-of-word segments and starting any subsequent characters in a new separate segment. Fortunately, the spaces separating part-of-word segments in different words (i.e. *inter-word spaces*) are typically larger than the spaces among part-of-word segments within the same word (i.e. *intra-word spaces*).

However, there is no static rule to distinguish intra-word spaces from inter-

word ones. Using an adaptive clustering algorithm, like the one of K-means, is hence an obvious resort to do that distinction after considering the following two precautions:

The first precaution is that only spaces coming from the same writing lines group, as obtained in step 4, are clustered together. The second precaution is more challenging as it is concerned with eliminating the *rogue* spaces whose widths are too large with respect to the other normal inter-word and intra-word spaces in their group. If rogue spaces are not eliminated, many spaces may falsely be judged as intra-word spaces!

Rogue spaces - which are eliminated through step 6.1 - emerge due to either unfiltered noise or special editing structures (e.g. tabs) in the target text-image.

3.1.6.1. Step 6.1; Eliminating Rogue Spaces Among Part-of-Words

This problem is in fact a hard one of *outlier's detection* as named in the literature of data mining. Capitalizing on our rich experience about the nature of Arabic/Arabic-dominated script, a much simpler yet effective heuristic solution is introduced here. Consider Figure (3.8) illustrating a histogram of a spaces group where the widths of intra-word spaces, inter-word spaces, and rogue spaces are plotted.

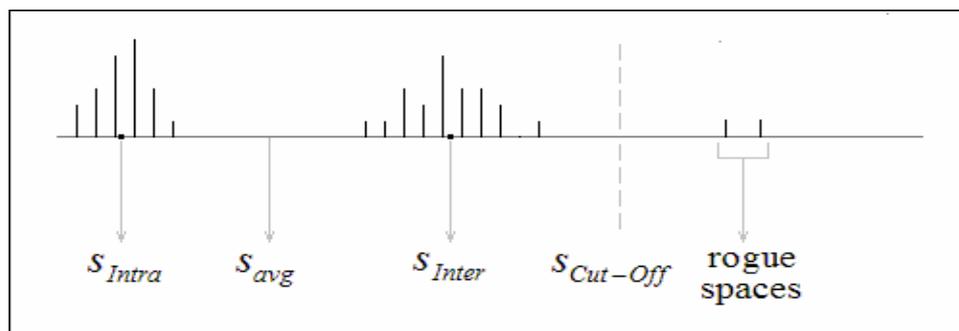


Figure (3.8): A hypothetical histogram of intra-word, inter-word, and rogue part-of-word spaces.

The optimal threshold between the intra-word centroid S_{Intra} and inter-word centroid S_{Inter} is $S_{Intra} + \frac{1}{2} \cdot (S_{Inter} - S_{Intra})$. If symmetry around S_{Inter} is also assumed, the *cut-off* space width for eliminating the rogue spaces can then be expressed as

$$S_{Cut-Off} = S_{Inter} + \frac{1}{2} \cdot (S_{Inter} - S_{Intra}) \quad \text{Eq. (3.9)}$$

While the average of all spaces S_{avg} is easily computable, the centroids S_{Intra} and S_{Inter} are unknown. So, we seek an expression of $S_{Cut-Off}$ in terms of only S_{avg} . Let

$$S_{Intra} = \varepsilon \cdot S_{Inter}; 0 \leq \varepsilon < 1 \quad \text{Eq. (3.10)}$$

Substituting from Eq. (3.10) in Eq. (3.9), we get

$$S_{Cut-Off} = \frac{1}{2} \cdot (3 - \varepsilon) \cdot S_{Inter} \quad \text{Eq. (3.11)}$$

Moreover, with δ is a weighting preference parameter S_{avg} may be expressed as

$$S_{avg} = S_{Intra} \cdot (\frac{1}{2} + \delta) + S_{Inter} \cdot (\frac{1}{2} - \delta); \quad 0 \leq \delta < \frac{1}{2} \quad \text{Eq. (3.12)}$$

Substituting in Eq. (3.12); for S_{Intra} from Eq. (3.10), then for S_{Inter} from Eq. (3.11), we get

$$S_{Cut-Off} = \frac{3 - \varepsilon}{(1 - 2 \cdot \delta) + \varepsilon \cdot (1 + 2 \cdot \delta)} \cdot S_{avg} \quad \text{Eq. (3.13)}$$

Setting $\varepsilon=0$, $\delta=0$ at the extreme, results in $S_{Cut-Off} = 3 \cdot S_{avg}$. However, our empirical knowledge of the Arabic script in general realistic conditions states $\varepsilon \in [0.1, 0.20]$, $\delta \in [0.05, 0.1]$ which produces the simple rogue-spaces cut-off formula that

$$S_{Cut-Off} \in [2.5 \cdot S_{avg}, 3.15 \cdot S_{avg}] \quad \text{Eq. (3.14)}$$

3.1.6.2. Step 6.2; Clustering Inter-word/Intra-Word Spaces

Once the rogue spaces are eliminated – if any - from a spaces group, the rest of these spaces in the group are clustered by the K-means algorithm into an intra-word spaces cluster whose centroid is S_{Intra} and another inter-word spaces cluster whose centroid is $S_{Inter} > S_{Intra}$.

Figure (3.9) shows the result of diagnosing inter-word spaces, hence completing the task of decomposing a text rectangle bitmap into whole words.

| | | | | | | |
|--------------|-----|-------|------|--------|---------|--------|
| دراسة | حول | تقسيم | النص | العربي | الممسوح | ضوئياً |
| إلى | | | | | | |
| سطور | | | | | | |
| وكلمات | | | | | | |
| اليّ | | | | | | |
| محمد | | | | | | |
| عطية | | | | | | |
| RD | | | | | | |
| ستشاري | | | | | | |
| تقنيات | | | | | | |
| اللغة | | | | | | |
| شركة | | | | | | |
| محمد | | | | | | |
| المحلاوي | | | | | | |
| مدرس | | | | | | |
| مساعد | | | | | | |
| بالأكاديمية | | | | | | |
| العربية | | | | | | |
| للعلوم | | | | | | |
| والتكنولوجيا | | | | | | |

Figure (3.9): Sample text rectangle bitmap after full words concatenation.

3.2. Decomposition Algorithm Evaluation

To assess the lines & words decomposition algorithm presented above, we ran it on a total of 1800 real-life documents evenly covering different kinds of sources, different levels of image-quality, and different scanning resolutions and color depths as explained in Table (3.1). This evaluation database has been acquired via a common hardware of HP 3800 flatbed scanner with TMA, HP Laser jet 1100 printer, and XEROX 256XT photocopier.

The WER of this algorithm is $e_d \equiv N_e \div N_{total}$; N_{total} is the total number of contiguous (i.e. un-spaced) strings in the input text block, and N_e is the number of wrongly decomposed full-word rectangles that either contain more than one contiguous strings or contain only part of contiguous strings. Errors due to typing mistakes, tiny isolated noise stains (recognized harmlessly as dots), and wrongly-split contiguous numbers (retrieved contiguous in the OCR post-processing), are all not counted as errors.

| <i>Source of doc.'s</i> → | <i>e_D</i> (%) for LASER printed documents 50 pages, 1081 lines, 12146 words | | | <i>e_D</i> (%) for Book pages 50 pages, 987 lines, 12524 words | | | <i>e_D</i> (%) for News Papers 50 pages, 1309 lines, 12870 words | | |
|------------------------------|-----------------------------------------------------------------------------------------------------------|------|------|------------------------------------------------------------------------------------------|------|------|--------------------------------------------------------------------------------------------|------|-------|
| | <i>Number of photocopying</i> → | PC#0 | PC#1 | PC#2 | PC#0 | PC#1 | PC#2 | PC#0 | PC#1 |
| 300 dpi, B&W | 0.41 | 0.59 | 0.73 | 0.44 | 0.61 | 0.71 | 0.59 | 0.71 | 0.90 |
| 300 dpi, Gray Shades | 0.36 | 0.56 | 0.66 | 0.42 | 0.58 | 0.68 | 2.10 | 2.91 | 3.67 |
| 600 dpi B&W | 0.35 | 0.50 | 0.61 | 0.38 | 0.53 | 0.64 | 0.50 | 0.63 | 0.84 |
| 600 dpi, Gray Shades | 0.32 | 0.46 | 0.57 | 0.31 | 0.45 | 0.59 | 0.85 | 1.10 | 01.43 |

Table (3.1): Results of evaluation experiments on the proposed decomposition algorithm.

From the previous evaluation, it is easily to infer the following:

- The WER tends to get higher with more numbers of photocopying (higher noise).
- The WER also tends to get lower with higher scanning resolutions.
- The WER tends to get lower with higher color depths, except for highly noisy text images (e.g. News Papers) where Black & White (B&W) colors performs better especially at lower scanning resolution.
- The algorithm does not collapse with the complexity (multiplicity of fonts & sizes) of the target text images.
- It is also rationally robust against the noise levels till News Papers photocopied twice.

CHAPTER 4

Autonomously Normalized Horizontal Differential Features Extraction

Feature extraction is the problem of “extracting from the raw data the information which is the most relevant for classification purposes, in the sense of minimizing the within class pattern variability while enhancing the between class pattern variability” [Devijver 1982].

Features extraction is one of the most important factors in achieving high recognition performance in character recognition systems. The extracted features must be invariant to the expected distortions and variations that character may have in a specific application. Also, the number of features must be kept reasonably small if a statistical classifier is to be used (curse of dimensionality) [Duda 2001].

A feature extraction method that proves to be successful in one application domain may turn out not to be very useful in another domain. Despite the existence of several image characterization methods [Trier 1996], most of them can only succeed in integrally identifying an image as a whole. Invariant moments [Gonzalez 2002] is a famous example that may be good for globally characterizing characters as a whole, so it can be effectively used in isolated characters recognition. However, applying this method on the narrow differential frames over words does not lead to an omni-font, open-vocabulary Arabic OCR systems.

Furthermore, the type of features extracted must match the requirements of the chosen classifier. In the published literature on HMM-based Arabic font-written OCR, the underlying features are merely simple time-domain luminosity coded features of the frames inside the sliding window as those described by BBN [Bazzi 1999], Gouda[Gouda 2004], and Khorsheed [Khorsheed 2007]. Such features are mainly ad

hoc combinations of simple partial measurements over the contents of the sliding window like its center of gravity, the number of black/white segments, the extensions of these segments, their relative positions within the same segment and maybe across the adjacent segments, ..., etc. Unfortunately, such ad hoc features realized a limited success in building truly omni font-written Arabic OCR.

As such ad hoc time domain features partially describing the differential progress of cursively written words are able to realize limited omni-font performance hence, neatly derived and normalized features fully and losslessly describing these differentials can be capable to achieving a much superior performance.

One major contribution of the work presented in this thesis is the design of a mathematically rigorous lossless differential luminosity coding based features. Our designed features extraction method computes a features vector for each frame while sliding through each Arabic word from right to left to produce a series of feature vectors that are injected to the vector quantizer.

The basic philosophy of our Autonomously Normalized Horizontal Differential Features (ANHDF) proposed is as follows; given a sequence of such a feature, one can fully retrieve the written script with only a scaling factor as well as the digital distortion error. The feature proposed here is a time-domain differential one where the script contour is totally differentiated along the x-direction.

4.1. ANHDF Basic Idea

Within the HMM-based recognition methodology, the information observed through the signal content of the thin sliding window is differential by definition, which may hence drive us intuitively to think of a differential design for the features vector.

Pursuing this line of thought, let each font-written word be formulated in the x - y plane as a parametrically represented contour; $C(t); x = g_x(t), y = g_y(t)$. Regarded as an analog signal, the width of the sliding window would formally shrink into being infinitesimally differential, and the simplest features vector to be thought of is the total derivative of the contour along the horizontal writing direction so that..

$$\underline{f}(x) = \frac{d_x C(x, y)}{dx} \quad \text{Eq. (4.1)}$$

Such a features vector would be fully representative to the writing contours which are losslessly retrievable via integration. Turning our sight back to the actual digitized form of the writing contours; $C^*(x, y)$, the differential dx is then best approximated by a width of a single pixel and our features vector turns into the total difference of the digitized contour along the horizontal writing direction so that..

$$\underline{F}_i = \Delta_x \underline{C}^*(x, y) \Big|_{x=i} \quad \text{Eq. (4.2)}$$

4.2. ANHDF Design Requirements

In the absence of closed-form expressions of the writing contours, how can this total difference be practically encoded into a manageable features vector? This is the non-trivial question that arises at this point. Any viable answer must satisfy the following requirements:

- I- That vector must have a finite dimensionality.
- II- This dimensionality must also be fixed for any vector extracted from the contents of the sliding window at any $x = i$.
- III- This dimensionality should be kept as compact as possible.
- IV- All the components of that vector should be computable efficiently.
- V- Feature vectors sequences corresponding to the same ligature sequences should remain maximally invariant versus the irregularities irrelevant to the writing concepts in the subject script; e.g. scale.

While failing to satisfy the first two requirements sets such vector designs completely out of the HMM recognition paradigm, the failure at satisfying the second two requirements would produce an unacceptably expensive implementation as per the time and/or storage at both the training and recognition phases.

Meeting the fifth requirement is a primal factor towards realizing an omni (i.e. font and size independent) performance of the sought OCR. One key idea towards producing a features vector design satisfying such a criteria is to consider that each single-pixel window (henceforth called a slice) at $x = i$ contains a limited maximum number of vertical dark segments; symbolized by N_d . For example, setting $N_d = 4$ for Arabic script is appropriately sufficient.

The other key idea is to differentially express each segment; $s_{j,i}$ in a sub-feature vector; $\underline{s}_{j,i}$ composed of definitely 4 components encoding all the agglomerative and topological features relative to the segments in the preceding slice at $x = i - 1$. With a dimensionality of $4 \cdot N_d$, the features vector \underline{F}_i derived from each slice at $x = i$ is then constructed by the union of its sub-feature vectors as follows..

$$\underline{F}_i = [s_1|_{x=i} \cup s_2|_{x=i} \cup \dots \cup s_{N_d}|_{x=i}] = \bigcup_{j=1}^{N_d} \underline{s}_{j,i} \quad \text{Eq. (4.3)}$$

4.3. ANHDF Vectors Design

To explain how to compute the above mentioned sub-feature vectors, consider Figure (4.1) illustrating a hypothetical pixel-level view of two consecutive slices

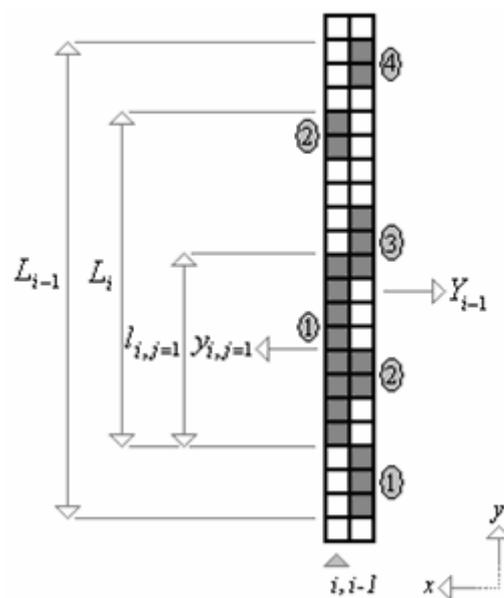


Figure (4.1): Word slice and segments

where the segments of each slice are ordered in a bottom-up manner, $l_{i,j}$ & $y_{i,j}$ are respectively the height and center of gravity of segment $s_{j,i}$, and L_{ij} & Y_{ij} are respectively the height and center of gravity of the whole slice.

A segment is considered disconnected if it is in 8-connection with none of the segments in the preceding slice. A whole slice is considered disconnected, if its preceding slice is empty.

As all feature vectors must have definitely $4 \cdot N_d$ dimensions, the feature vectors derived from slices containing less than N_d dark segments are padded with nullified sub-feature vectors which are assumed to be derived from empty segments.

For any given segment, only 5 distinct topological possibilities can exhaustively be considered. For each possibility, one formula from those numbered 4.4 to 4.8 below needs be applied to compute the sub-feature vectors whose first two components encode the agglomerative properties while the last two encode the topological properties of segments relative to the preceding slice.

Here are the 5 topological possible cases along with the formulae to compute the sub-feature vectors at each case.

Case 1: For non-empty segments in the initializing slice of the whole word,

$$\underline{s}_{j,i=1} = \left[\frac{y_{1,j} - y_{1,1}}{l_{1,1}}, \frac{l_{1,j}}{l_{1,1}}, -1, -1 \right] \quad \text{Eq. (4.4)}$$

Case 2: For non-empty disconnected segments in connected slices,

$$\underline{s}_{j,i} = \left[\frac{y_{i,j} - Y_{i-1}}{L_{i-1}}, \frac{l_{i,j}}{L_{i-1}}, -2, -2 \right] \quad \text{Eq. (4.5)}$$

Case 3: For non-empty segments in disconnected slices other than the initializing one of the word,

$$\underline{s}_{j,i} = \left[\frac{y_{i,j} - Y_{i-k}}{L_{i-k}}, \frac{l_{i,j}}{L_{i-k}}, -3, -3 \right] \quad \text{Eq. (4.6)}$$

Where the slice at $x = i - k$ is the nearest preceding non empty slice to the slice at $x=i$

Case 4: For all empty (i.e. null) segments,

$$\underline{s}_{j,i} = [0, 0, 0, 0] \quad \text{Eq. (4.7)}$$

Case 5: Otherwise,

$$\underline{s}_{j,i} = \left[\frac{y_{i,j} - Y_{i-1}}{L_{i-1}}, \frac{l_{i,j}}{L_{i-1}}, s_{START,j-1}, s_{END,j-1} \right] \quad \text{Eq. (4.8)}$$

Where the last two components of formula (4.8) respectively determine the orders of the most bottom and top segments in the previous slice at $x = i - 1$ which are in 8-connection with $s_{j,i}$.

For example, the features vector at $x = i$ in Figure (4.1) above can be simply calculated, by the aid of the flowchart shown in Figure (4.2), using formulae (4.8), (4.5), and (4.7) respectively as follows,

$$\underline{E}_i = \left[\left[\frac{-1.9}{20}, \frac{8}{20}, 1, 3 \right] \cup \left[\frac{7.1}{20}, \frac{2}{20}, -2, -2 \right] \cup [0, 0, 0, 0] \cup [0, 0, 0, 0] \right]$$

$$= [-0.095, 0.4, 1, 3, 0.355, 0.1, -2, -2, 0, 0, 0, 0, 0, 0].$$

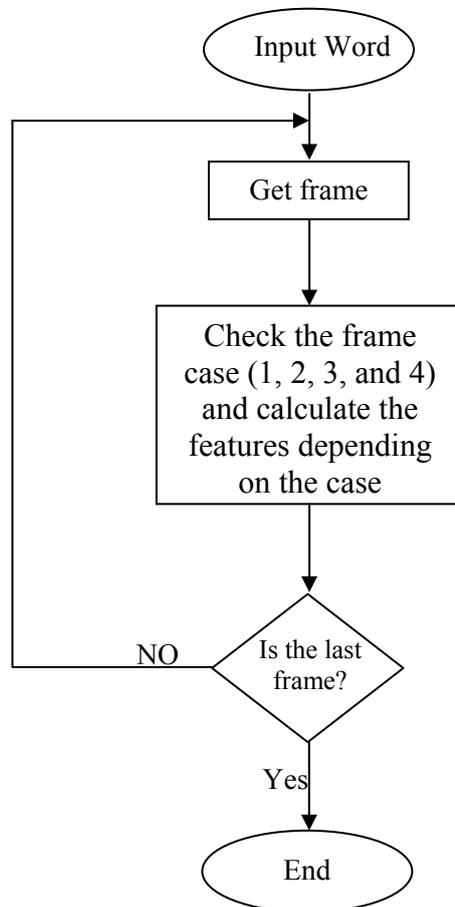


Figure (4.2): Feature extraction algorithm flowchart

It is obvious that the calculations needed to compute the proposed feature vectors are straightforward to implement as it only uses simple arithmetical operations never beyond additions, subtractions, multiplication and divisions. The components of our features vector comprise both continuously analog as well as quantized values.

4.4. Features Normalization

From the aforementioned discussion of the design of the feature vectors, the continuous/discrete hybrid nature of the feature vectors components is obvious. The first and the second components of sub-feature vectors (agglomerative features) of segments have continuous values that vary within different ranges. While the third and the fourth components (topological features) have discrete values that vary within another range.

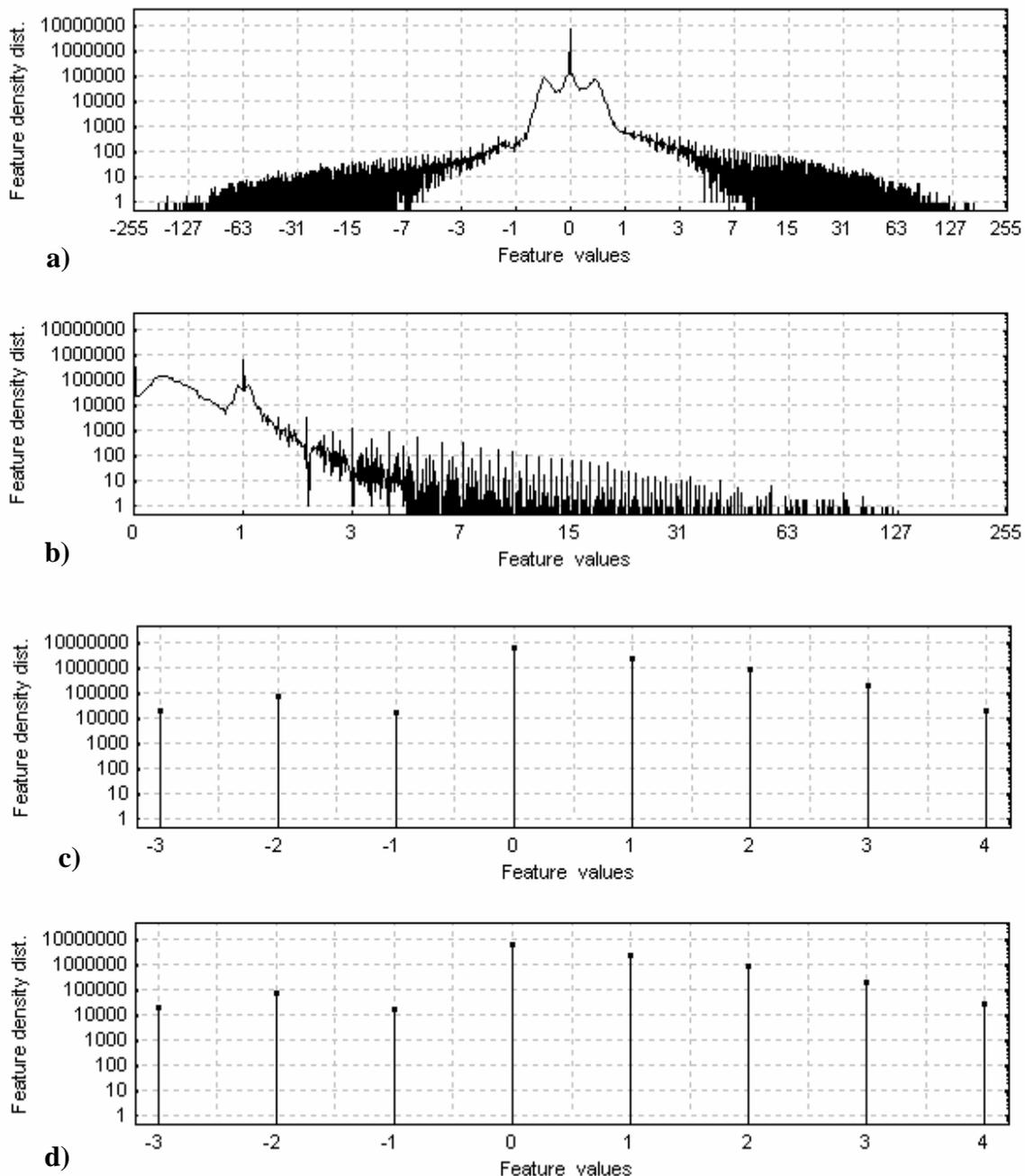


Figure (4.3): Feature values density distributions a) for the 1st component, b) for the 2nd component, c) for the 3rd component and d) for the 4th component.

The first component population tends to follow a natural (Gaussian) distribution with large values extended along the negative and the positive tails of the distribution. The second component distribution coincides with the first component nature in the positive side only, i.e. the distribution has positive tail only and it has no negative tail.

However, the topological sub-feature vectors components (the third and the fourth components) take only 8 discrete values $[-4, -3, \dots, 3]$. Figure (4.3) illustrates the features values distributions for the four sub-feature vectors components of segments.

Due to the wide variation in the ranges of the feature vectors components as shown in Figure (4.3), normalization of the feature vectors components is needed to balance the weights of the different dimensions of the feature vectors while calculating the distances between points in the features space during the clustering and the recognition phase.

Pruning of extreme values for the first and the second components of the sub-feature vectors is essential before normalization, as the extreme values would render the resolution of the effective region of the features indiscriminative, which would in turn severely ruin clustering, hence the training and the recognition phases.

CHAPTER 5

Vector Quantization and Clustering

For using an HMM with a discrete observation symbol density, rather than the continuous vectors, a vector quantizer is required to map each continuous observation vector into a discrete codebook index.

Vector Quantization (VQ) has been widely used in the speech processing field both in the hidden markov modeling of speech for recognition and in low bit-rate speech coding. The process of VQ enables discrete models to represent continuous observations by mapping d -dimensional vectors in the vector space R^d into a finite set of vectors $\underline{Z} = \{\underline{z}_i: i = 1, 2, \dots, K\}$. Each vector z_i is called a code vector or a *codeword* (centroid) and the set of all the codewords is called a *codebook*. Associated with each codeword, \underline{z}_i , a nearest neighbor region called *Voronoi* region, and it is defined by:

$$V_i = \left\{ \underline{x} \in R^d : \|\underline{x} - \underline{z}_i\| \leq \|\underline{x} - \underline{z}_j\| \right\}, \quad \text{for all } j \neq i \quad \text{Eq. (5.1)}$$

The set of Voronoi regions partition the entire space R^d such that:

$$\begin{aligned} \bigcup_{i=1}^K V_i &= R^d \\ \bigcap_{i=1}^K V_i &= \Phi \end{aligned}, \quad \text{for all } i \neq j \quad \text{Eq. (5.2)}$$

The VQ process allows the discrete model with its relatively modest requirement for training data to represent the continuous observations at the expense of the quantization distortion inherent in mapping all the vectors in a cell onto the

same label. Different elements are needed in building a VQ codebook and in implementing a VQ procedure needs which are: a training set, a distance measure, centroid computation and a clustering algorithm.

5.1. Training Set

In order to design the required codebook, a set of vectors (training set) is required which should be representative of the type and distribution of vectors encountered during normal VQ operation. A large training set is the corner stone to building a VQ codebook. The greater the diversity of the training set in covering possible shapes of input vectors later to be processed by the VQ, the smaller the quantization error in representing the information with a fixed-size codebook.

5.2. Distance Measure

This is the cost of reproducing an input vector \underline{x} as one of the K codebook vectors. The most common distance measure is the Euclidian distance:

$$d(\underline{x}_i, \hat{\underline{z}}_j) = \|\underline{x}_i - \hat{\underline{z}}_j\| \quad \text{Eq. (5.3)}$$

The Euclidean distance works well when a data set has “compact” or “isolated” clusters [Jain 1999]. The drawback of the direct use of such Minkowski metric is the tendency of the largest-scaled feature to dominate the others. Solutions to this problem include normalization of the continuous features (to a common range or variance) or other weighting schemes.

The main goal behind the VQ algorithm is to determine the optimum set of K vectors such that the average distortion in replacing each training set vectors (of size N_s) by the closest entry in the codebook is minimized. This can be phrased mathematically as finding the set $\hat{\underline{z}}_j$ for a given K such that

$$D_j = \min_{\hat{\underline{z}}_j} \frac{1}{N_s} \sum_{i=1}^{N_s} \min_{1 \leq j \leq K} d(\underline{x}_i, \hat{\underline{z}}_j) \quad \text{Eq. (5.4)}$$

is minimum, where D_j is the average distortion of the vector quantizer.

5.3. Centroid Computation

The vector space is partitioned into cells where all input vectors yielding a common reproduction are grouped together. Each cell is equivalent to a codebook symbol or entry, and each codebook symbol is represented by its centroid \hat{z}_j . The assignment of an input vector to one of the codebook symbols is based on the minimum distance of that vector from all symbol centroids. The centroid in the case of a squared-error distortion measure is simply the sample mean vector of cluster C_j :

$$\hat{z}_j = \frac{1}{n_j} \sum_{x \in C_j} x \quad \text{Eq. (5.5)}$$

where n_j is the number of samples in cluster C_j .

5.4. Clustering Algorithm

Clustering is the unsupervised classification of patterns (observations, data items, or feature vectors) into groups (clusters). The reason behind this naming is that these techniques are concerned with forming classes from training vectors without benefit of supervision regarding class membership, or the analysis of the statistical structure of the data. There is no systematic approach to defining the most suitable number of classes in the codebook. The clustering problem has been addressed in many contexts and by researchers in many disciplines; this reflects its broad appeal and usefulness as one of the steps in exploratory data analysis [Jain 1999].

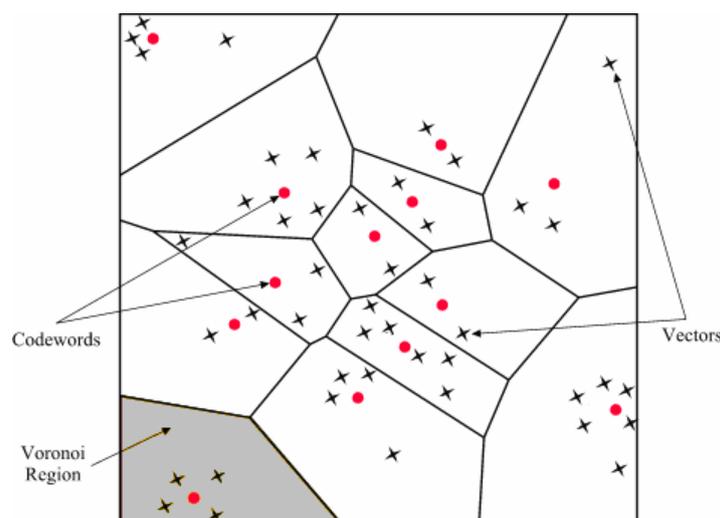


Figure (5.1): A two dimensional space showing samples naturally grouped as thirteen clusters with their representative centroid

The aim of clustering is to find some structure in the distribution of a set of vectors in d -dimensional Euclidian space. In one or two dimensions this can easily be visualized, for example in Figure (5.1) the sample vectors are clearly distributed as thirteen clusters. In many pattern recognition problems the dimensionality is much higher and it becomes impossible to visualize the space.

There are two main types of clustering algorithms; the hierarchical clustering algorithms and the partitional clustering algorithms. In hierarchical clustering algorithms, each vector is initially a separate cluster. Then, at each step of the algorithm the two most similar clusters, based on a predefined distortion measure, are merged until the desired number of clusters is achieved. In the partitional clustering algorithms, a fixed number of clusters or classes is used. In each iteration training vectors are reassigned according to a predefined distortion measure until a stable partitioning of the vectors is achieved. The two most common methods of partitional clustering, the K-means and the LBG algorithms are discussed below.

5.4.1. K-Means Algorithm

The K-means algorithm can be considered as the workhorse of clustering algorithms. The K-Means algorithm is commonly associated with the minimization of a squared error criterion. K refers to the number of classes in the codebook. The algorithm works as follows:

1. Perform an initial selection of centroids.
2. Iterate on samples, for each sample,
 - Find the closest centroid.
 - Assign the sample to the corresponding cluster.
 - Recompute the centroid of that cluster.
3. Repeat step 2 until a convergence criterion is met.

Typical convergence criteria include terminating after an iteration where the cluster memberships of the samples is unchanged, terminating after an iteration where the centroids are left unchanged, terminating after an iteration where the value of the objective function is unchanged, or terminating after a maximum number of iterations is reached. When the data set is large, convergence can be somewhat slow, and the above terminating criteria are substituted by threshold criteria (e.g., terminating after an iteration where the objective function decreases by less than a threshold).

This algorithm systematically decreases the overall distortion, by updating the codebook. The distortion sometimes converges, however, to a local optimum that may be significantly worse than the global optimum. Specifically, the algorithm tends to gravitate towards the local optimum nearest the initial codebook. A global optimum may be approximately achieved by repeating this algorithm for several types of initializations, and choosing the codebook having the minimum overall distortion results.

This algorithm has been experimentally observed that it is dependant on the order in which the patterns of the training set are selected. Hence it is useful for the applications where clustering must be done online (Duda 2001).

5.4.2. LBG Algorithm

A method that closely resembles the K-Means algorithm (to the point that it is sometimes called K-Means) is the clustering algorithm of Linde, Buzo, and Gray (also known as LBG, or as the Modified Lloyd Algorithm) [Linde 1980]. The algorithm works as follows,

1. Perform an initial selection of centroids.
2. Iterate on samples: for each sample
 - Find the closest centroid.
 - Assign the sample to the corresponding cluster.
3. After iterating on all samples, recompute the centroids using the new assignment of points to clusters.
4. Repeat step 2 until a convergence criterion is met.

As the LBG algorithm perform a batch training not an online training as that of the original K-means algorithm, LBG has a distinct computational advantage: the computation of distances between points and centroids can be done more efficiently, because neither points nor centroids change during the computation.

The LBG algorithm compensates for the sensitivity of the of the original K-means algorithm to the initial values of the centroids by using a “splitting” method for designing the quantizer from scratch. This initialization requires that the number of code words is a power of 2. The procedure starts from only one codeword (global mean of the training set) that, recursively, splits it in two distinct codewords (Linde

1980). More precisely, the generic m^{th} step consists in the splitting of all vectors obtained at the end of the previous step.

The splitting criterion is shown in Figure (5.2). It starts from one codeword \underline{z} . It splits this vector into two close vectors $\underline{z} + \xi$ and $\underline{z} - \xi$ where ξ is a fixed perturbation vector.

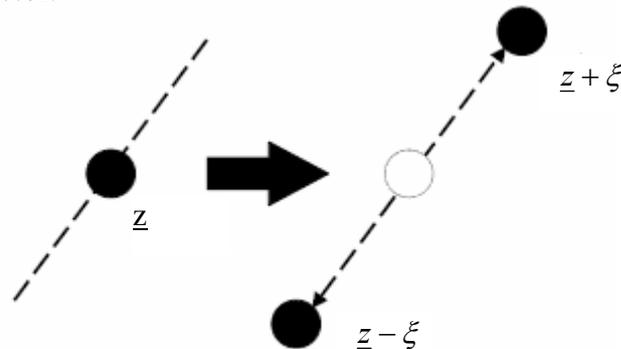


Figure (5.2): LBG centroid splitting

The LBG has a big chance to achieve a better local minimum near the global one and does not stuck to a local minimum that is nearest to the initial randomly chosen codebook as that of the original K-means algorithm.

The reasons behind the popularity of the LBG algorithm are [Jain 1999]:

- (1) Its time complexity is $O(N_s \cdot K \cdot N_{iter})$, where N_s is the number of patterns, K is the number of clusters, and N_{iter} is the number of iterations taken by the algorithm to converge. Typically, K and N_{iter} are fixed in advance and so the algorithm has linear time complexity in the size of the data set.
- (2) Its space complexity is $O(K + N_s)$. It requires additional space to store the data matrix. It is possible to store the data matrix in a secondary memory and access each pattern based on need. However, this scheme requires a huge access time because of the iterative nature of the algorithm, and as a consequence processing time increases enormously.
- (3) It is order-independent; it generates the same partition of the data irrespective of the order in which the patterns are presented to the algorithm.

5.5. Clusters Analysis

An important problem in clustering is how to decide what is the best set of clusters for a given data set, in terms of both the number of clusters and the

membership of those clusters especially in high dimensions. Generally, it has to be estimated from the data and usually selected by some heuristics in practice depending on the application [Duda 2001].

An approach in estimating the optimal number of clusters is to use an objective function such as the normalized mean square error (E_K) between the training data samples and the centroids of a given codebook [Tibshirani 2001]. Figure (5.3) shows a typical plot of an error measure for a clustering procedure versus the number of clusters K employed; the error measure E_K decreases monotonically as the number of clusters K increases, but from some K on the decrease flattens markedly. The location of such an "elbow" indicates the appropriate number of clusters.

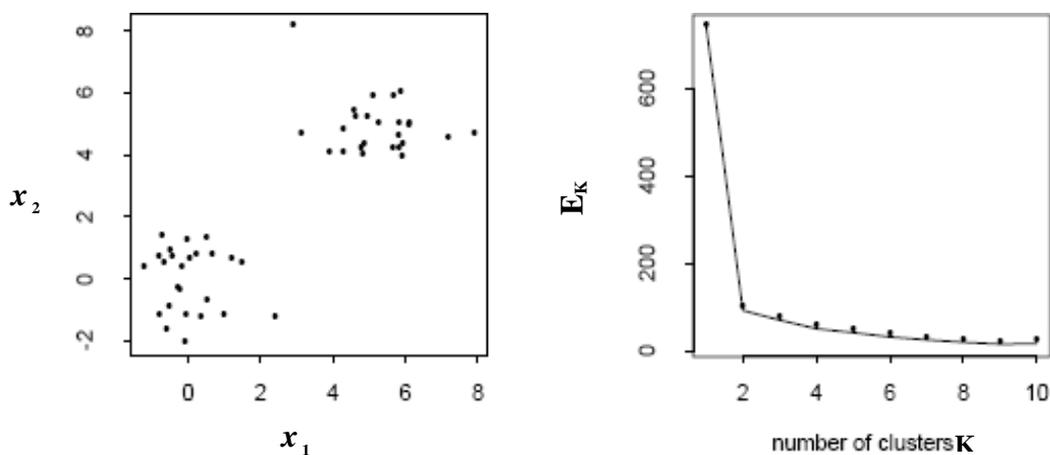


Figure (5.3): A two dimensional data with its normalized mean square error function (E_K) on the right plot

However this elbow represents a global minimum of the error over the centroids and doesn't address the degree of adequacy between the distribution of the features and the distribution of their representative codebook centroids. This measure is also a computationally expensive and time consuming measure.

When clustering is just used as a preprocessing step to reduce the complexity of the data, as in a classification task, K can be chosen pretty large (in order not to loose too much information about the data). However, it must not be too large to be afforded later on (trade-off between computational costs of next steps and number of clusters) and to avoid over clustering.

The final classification results using the testing samples or cross validation is an optimum objective method used to estimate the best number of clusters. However,

using this method over a wide range of K is a time consuming process. Hence, in our work we develop a subjective method for measuring the quality of different sets of clusters by the visualization of the adequacy between the codebook centroid distribution generated and the distribution of the data used in clustering.

5.6. The Adequacy Test

As it is infeasible to visualize the multidimensional feature vectors and their representative codebook distributions if $d > 3$ (as that in Figure (5.1) for the two dimensional feature vectors); we can use the projections of these distributions in the d -dimensions as a good visualization aid for these distributions. Our aims behind the visual representations of the features population density distributions among with the codebook density distributions projected in the d -dimension are:

1. Estimation of the suitable size of the training set that to be used later in the codebook design. This can be done by watching the variations in the features distribution projections in the d -dimensions that occur due to increasing the training data set size and choosing the size when no further variations occur after increasing that size (data saturation occurs).
2. Estimating the appropriate codebook size by choosing the codebook that best matches the distributions of the features in all dimensions.

To achieve our aims, two plots with certain scales are developed. These plots help us to estimate the best training data set size and the codebook size in order to achieve the least WER from our OCR recognition system without doing neither computationally expensive nor time consuming measures. The two plots are described as follows:

- **The first plot:**

It represents the normalized feature density distribution versus the feature values projected on each dimension of the feature vector as shown in Figure (5.4a), where:

- The ***x-axis*** represents the feature values in each dimension of the feature vector. In some dimensions 95 % of the feature values are localized within the normalized range $[-1, 1]$ while the rest of the feature values are expanded outside this compact range. In order to view all the range of the features values a companding scale is developed. For the other dimensions,

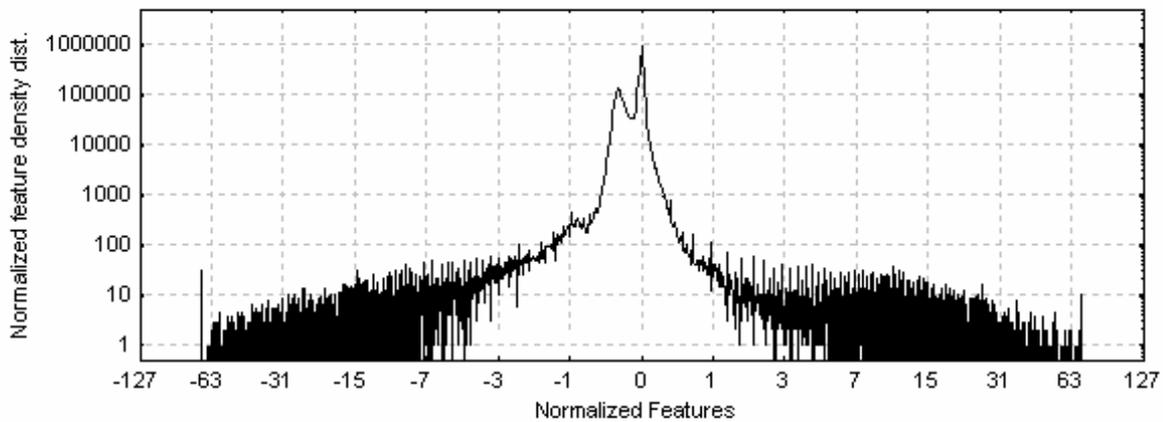
all the features values are localized within the normalized range [-1, 1], so a linear scale is used. The used scaling functions including the companding and the linear functions can be mathematically described as:

$$f(x) = \begin{cases} \text{Sign}(x) \cdot \log_2(1 + |x|) & \text{for } d = 4i + 1, 4i + 2 \\ = x & \text{for } d = 4i + 3, 4i + 4 \end{cases} \quad \text{Eq. (5.6)}$$

where x is the feature value, d is the d^{th} dimension of the feature vector and $i \in \{0 \dots 3\}$.

- The y -axis represents the feature density distribution in \log_{10} scale to cope with the large variations of its values.

a.



b.

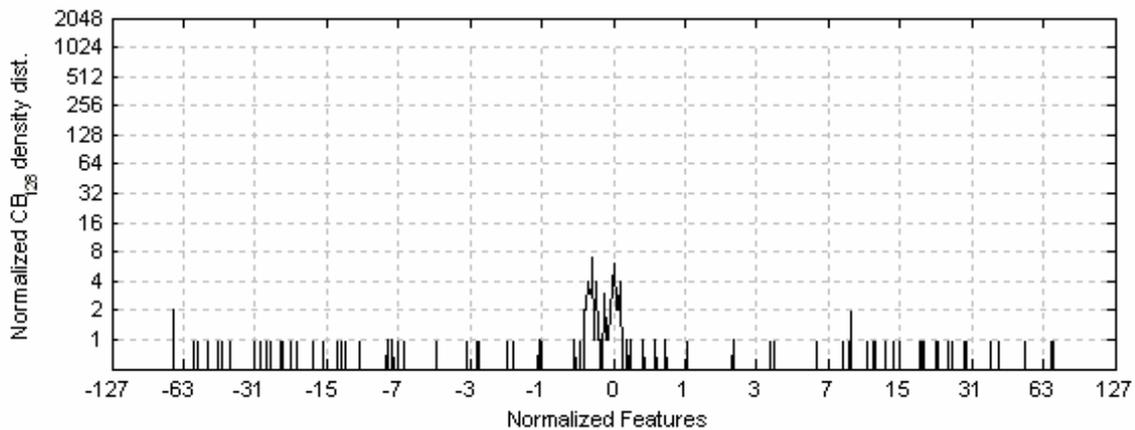


Figure (5.4): a) Normalized Feature density distribution projected on $d=1$,
b) Normalized Code book (size 128) density distribution projected on $d=1$.

- **The second plot :**

It represents the normalized codebook density distribution versus the feature values projected on d^{th} dimension of the feature vector as shown in Figure (5.4 b), where:

- The *x-axis* is the same as that of the first plot.
- The *y-axis* represents the codebook density distribution in \log_2 scale to cope with the large codebook sizes of order power of 2 generated by LBG algorithm.

Figures (5.5-5.20) below show the normalized features and codebook distributions projected on all of the 16 dimensions of our proposed feature vector. In each dimension, the normalized feature density distribution is shown in the first plot. Also, various normalized codebook sizes (codebook sizes =2048, 1024, 512 and 256) density distributions are plotted consecutively below the first plot.

It is obvious from the figures that, in all dimensions, the adequacy degree between the codebook centroid distributions and the normalized feature density distributions increases with the increase in the codebook size, taking into consideration the effect of the cross correlation between different feature vector components . It is observable that the tails of the feature density distribution in each dimension is not well represented by the codebook centroids for the small codebook sizes as most of the centroids are attracted to represent the peaks of the distribution. The tails are well represented with the codebook centroids as the codebook size increases.

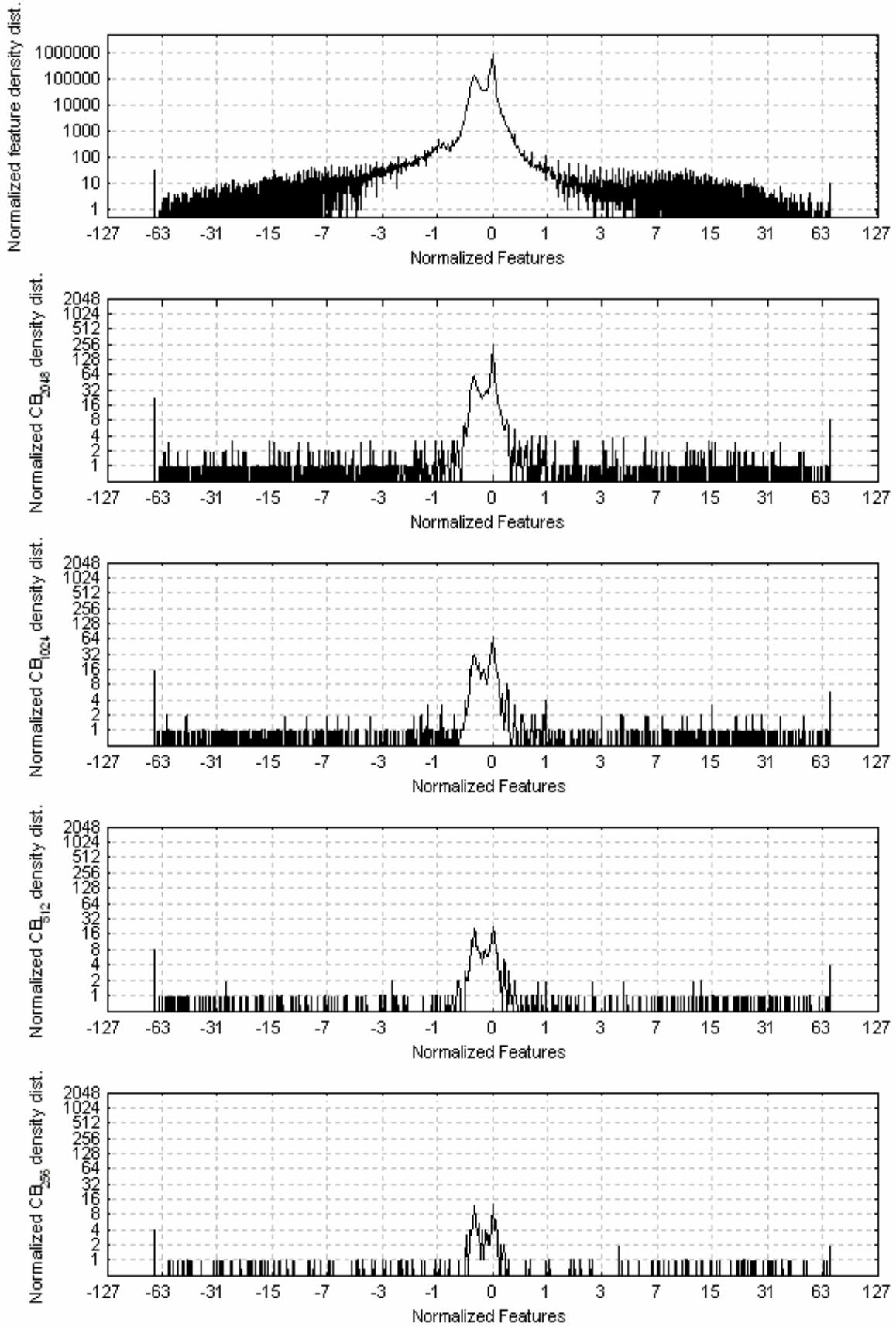


Figure (5.5): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=1$

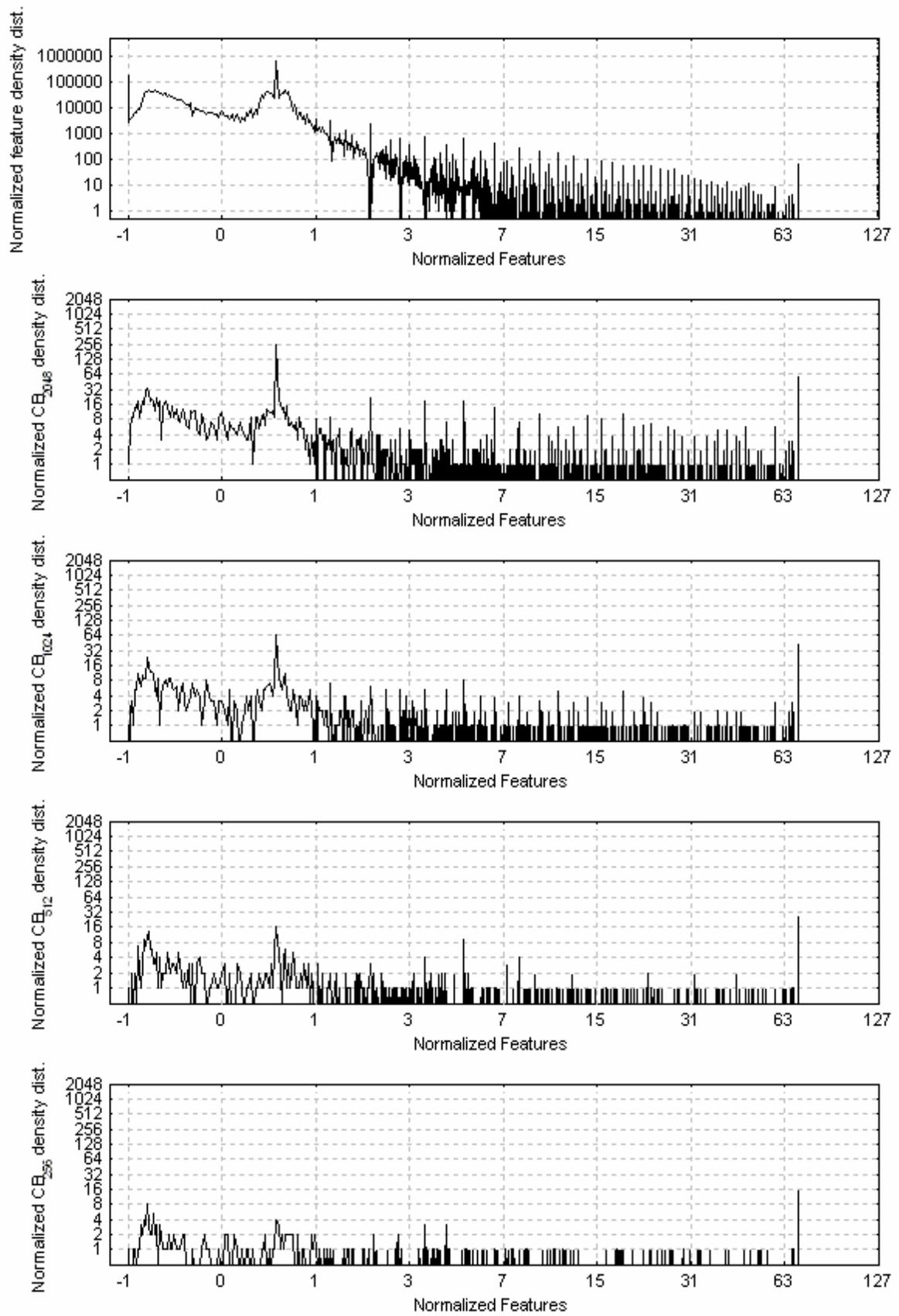


Figure (5.6): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=.2$

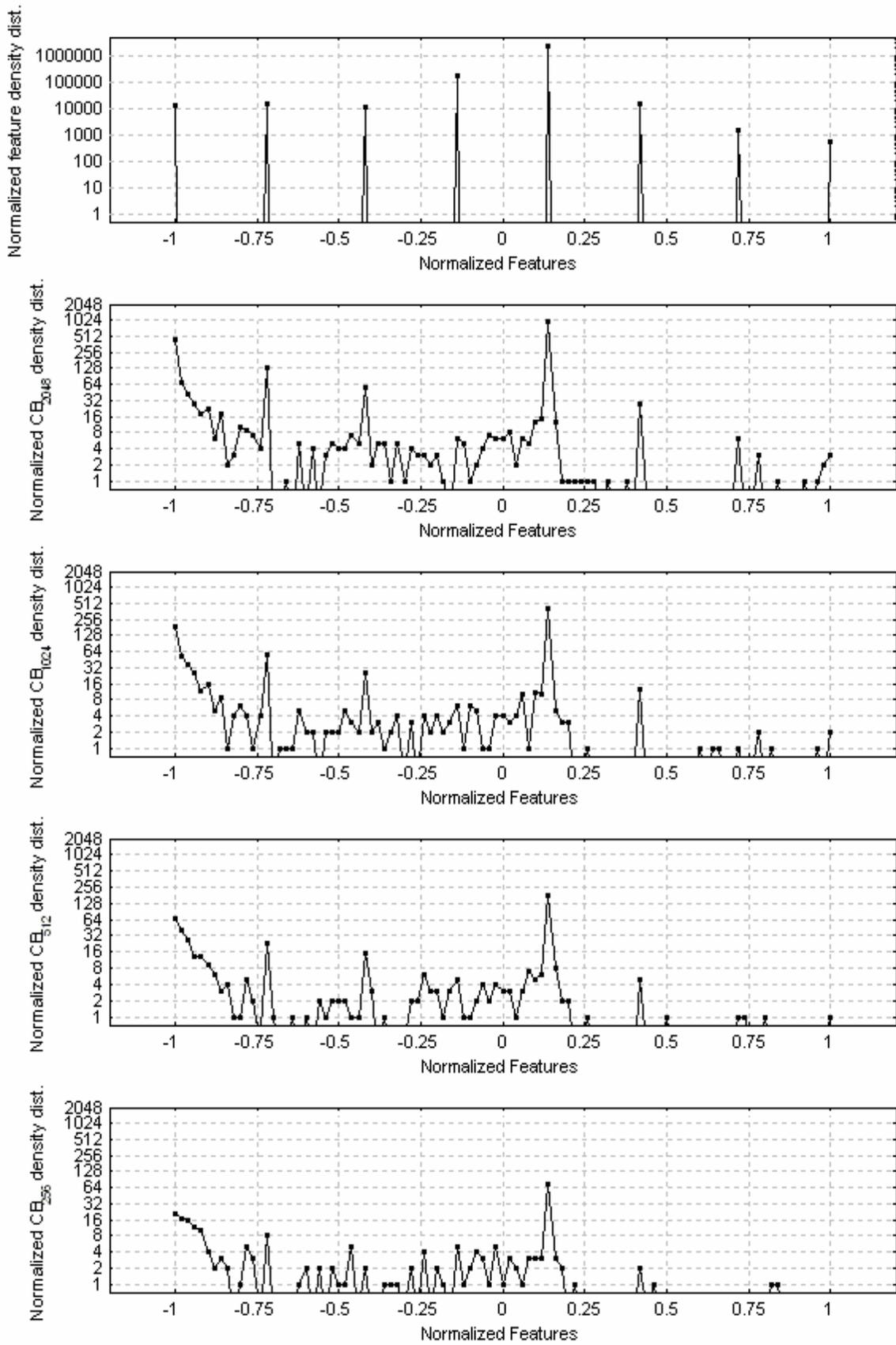


Figure (5.7): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=3$

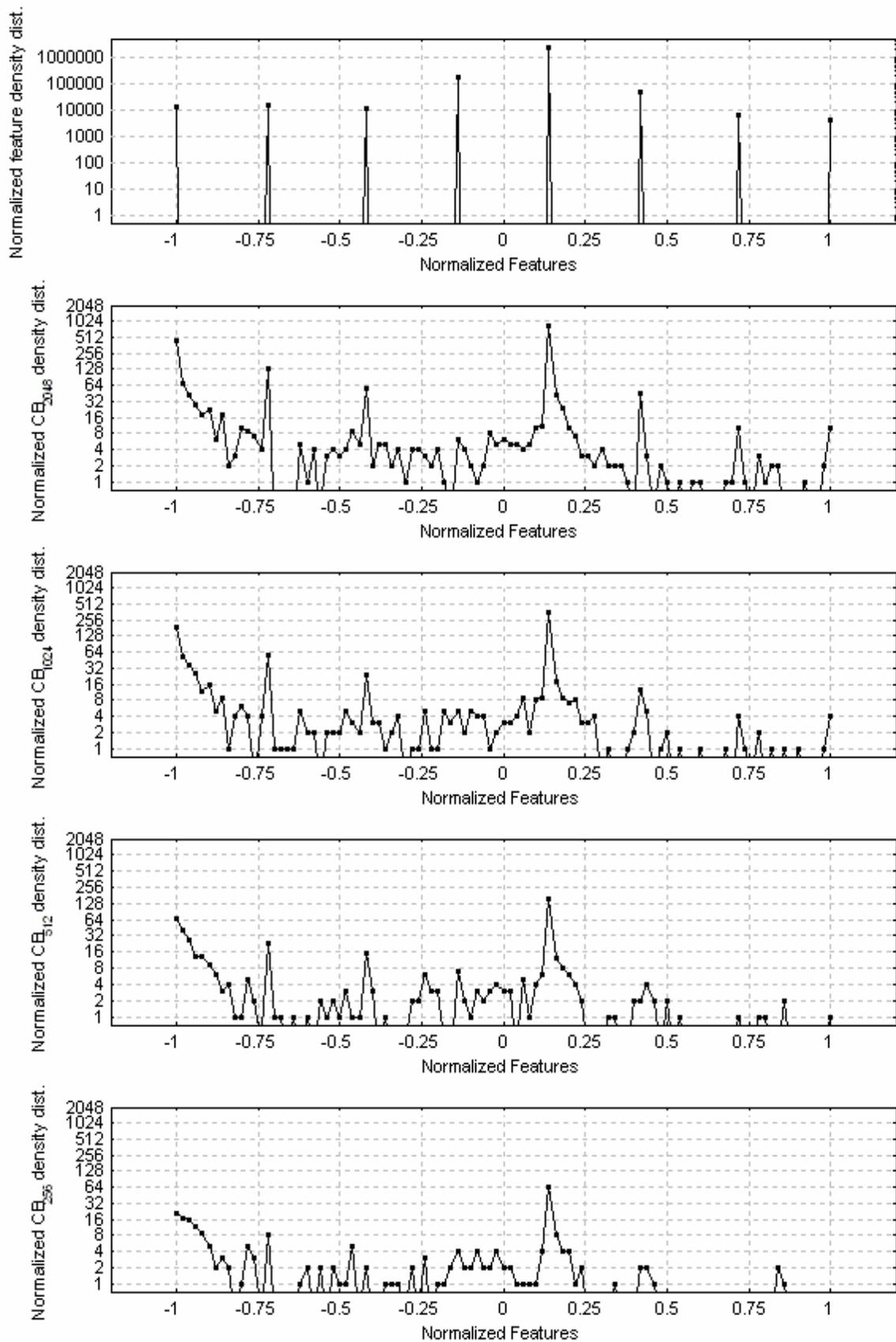


Figure (5.8): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=4$

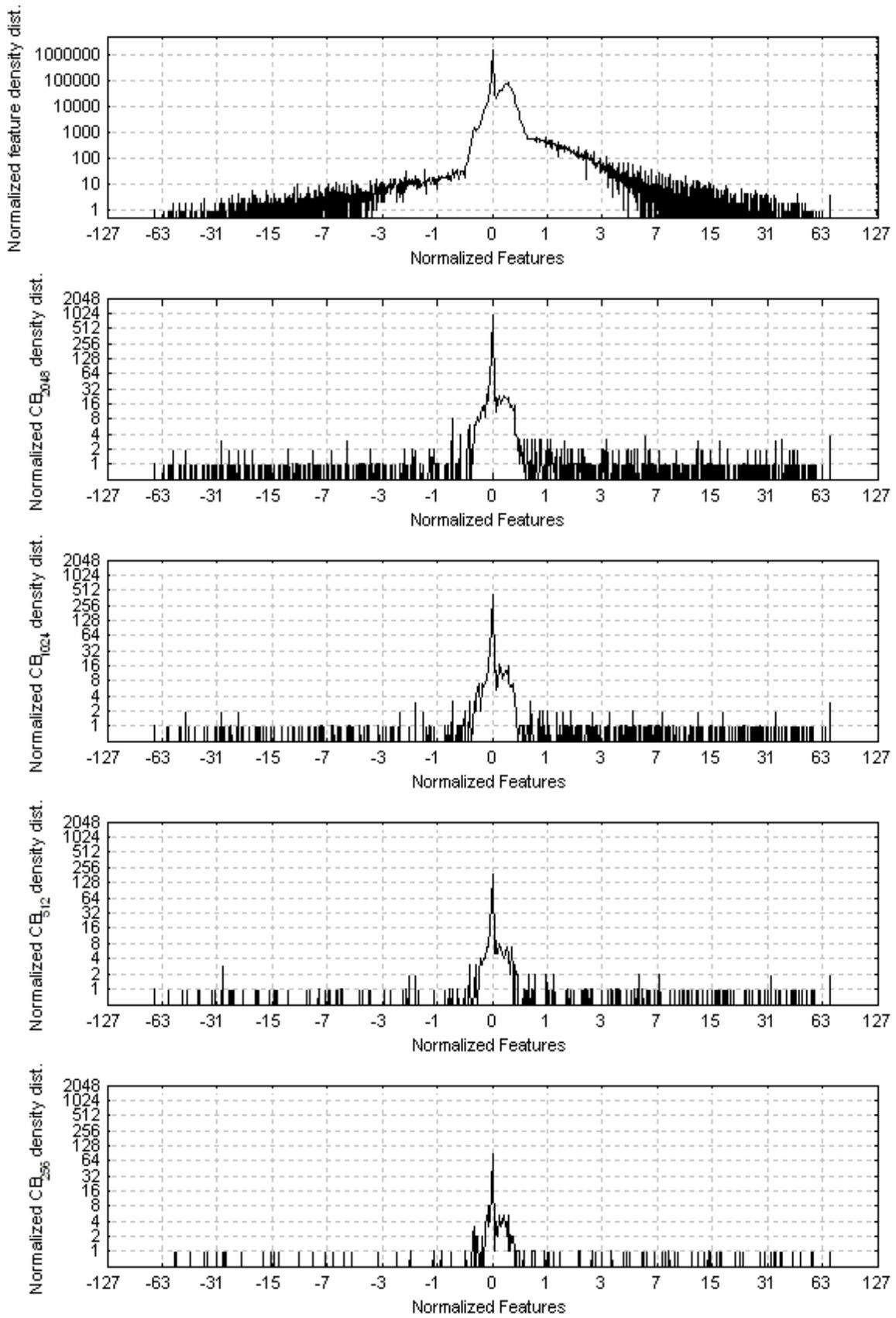


Figure (5.9): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=5$

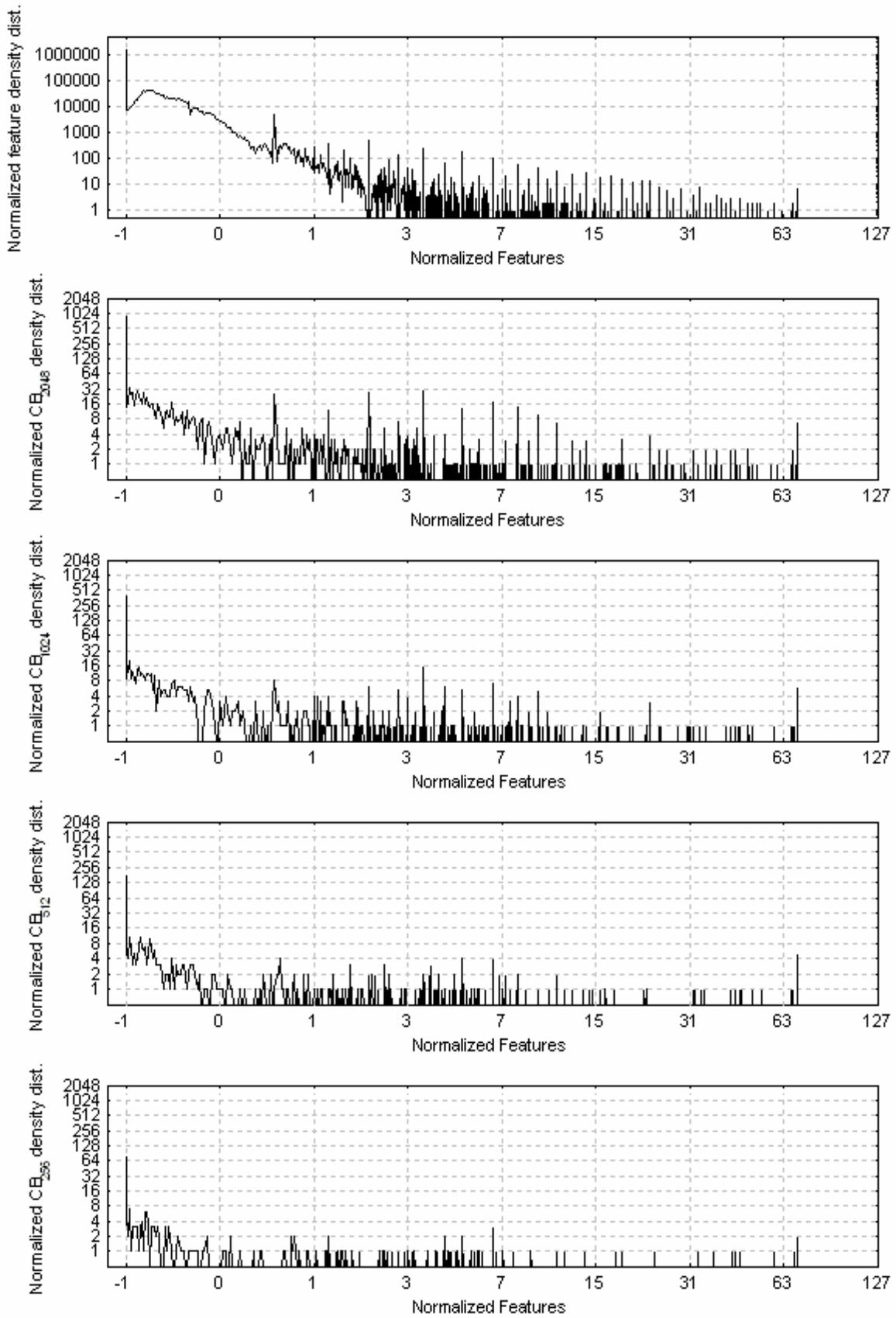


Figure (5.10): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=6$

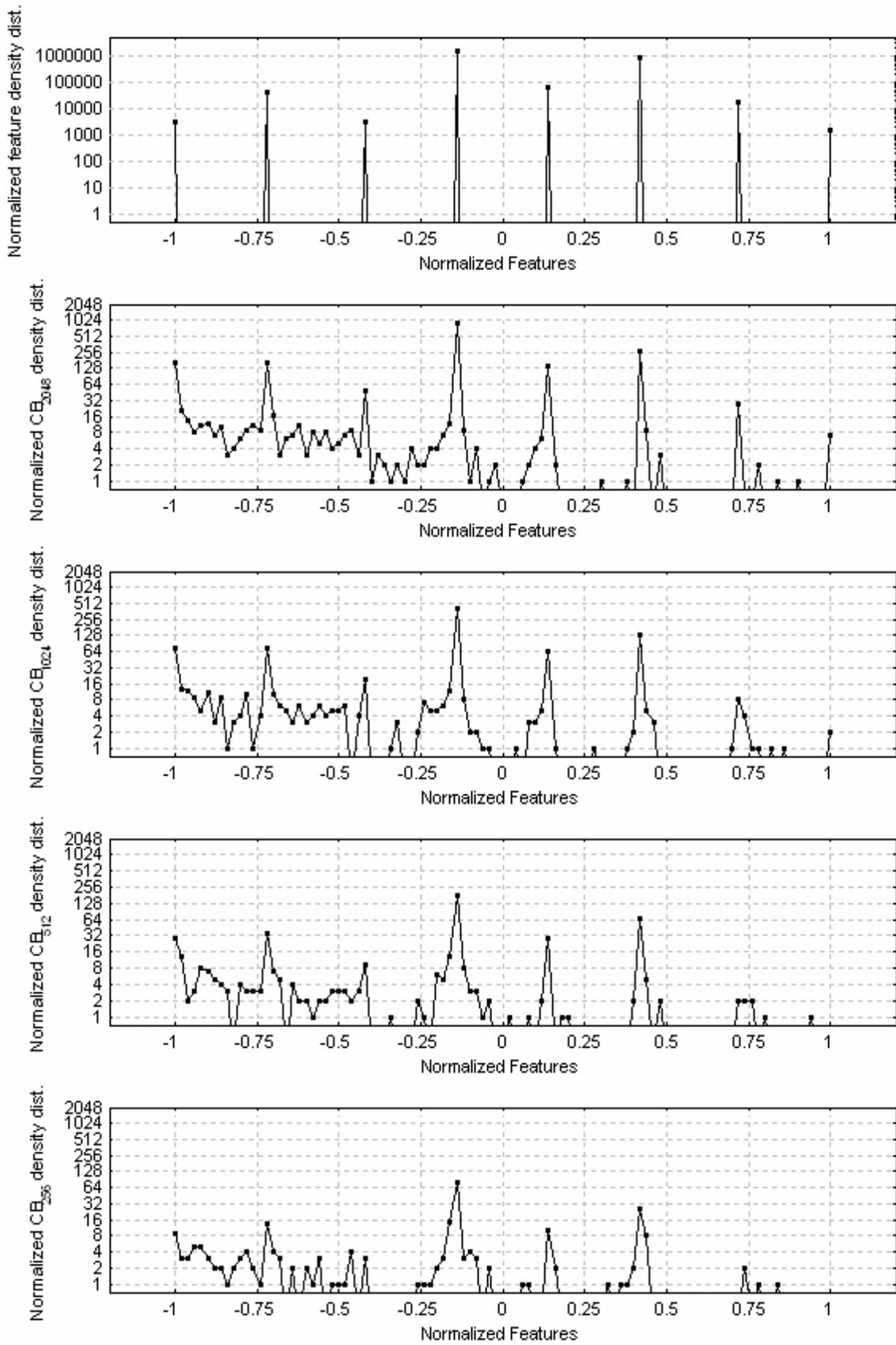


Figure (5.11): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=7$

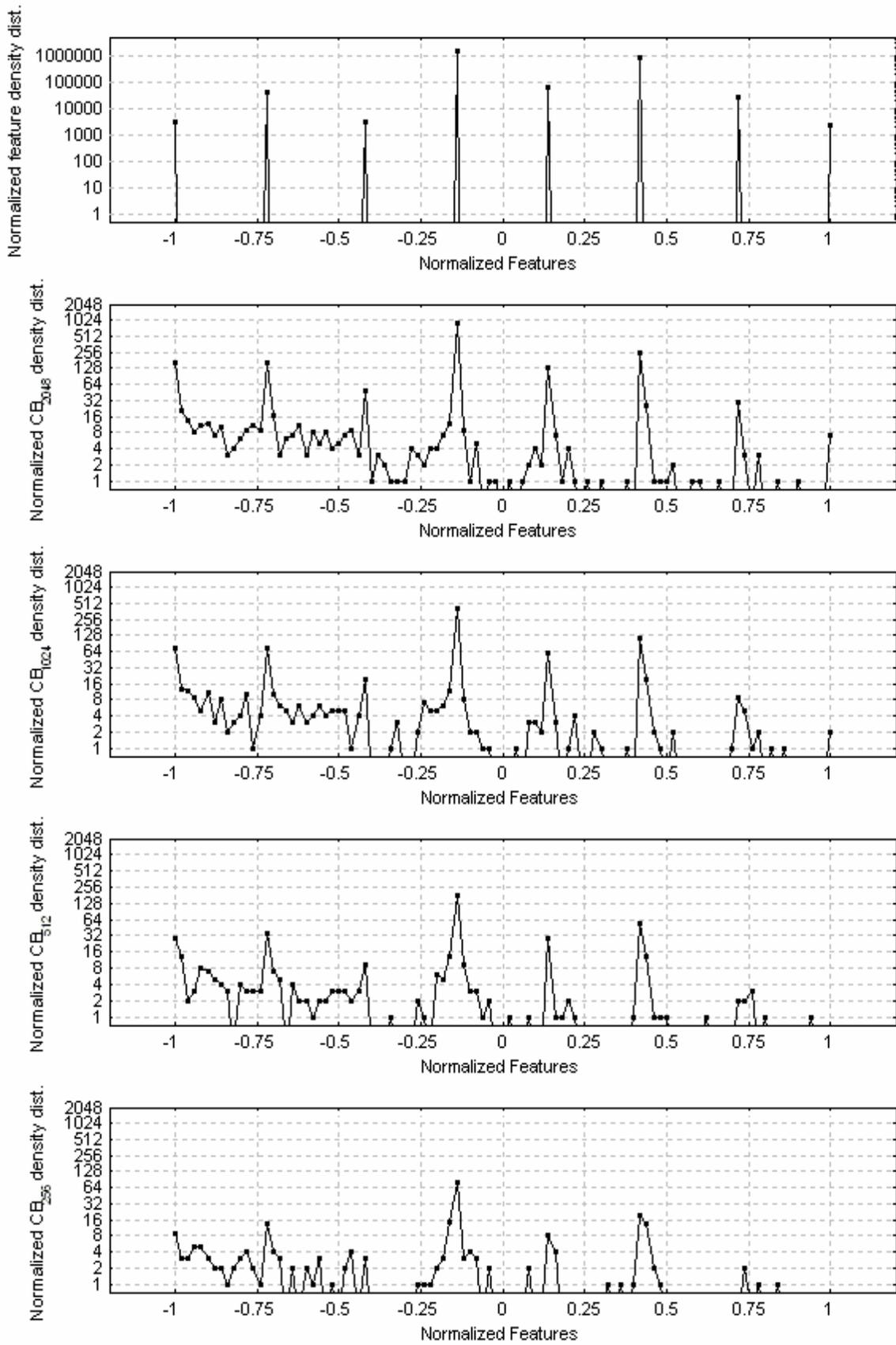


Figure (5.12): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=8$

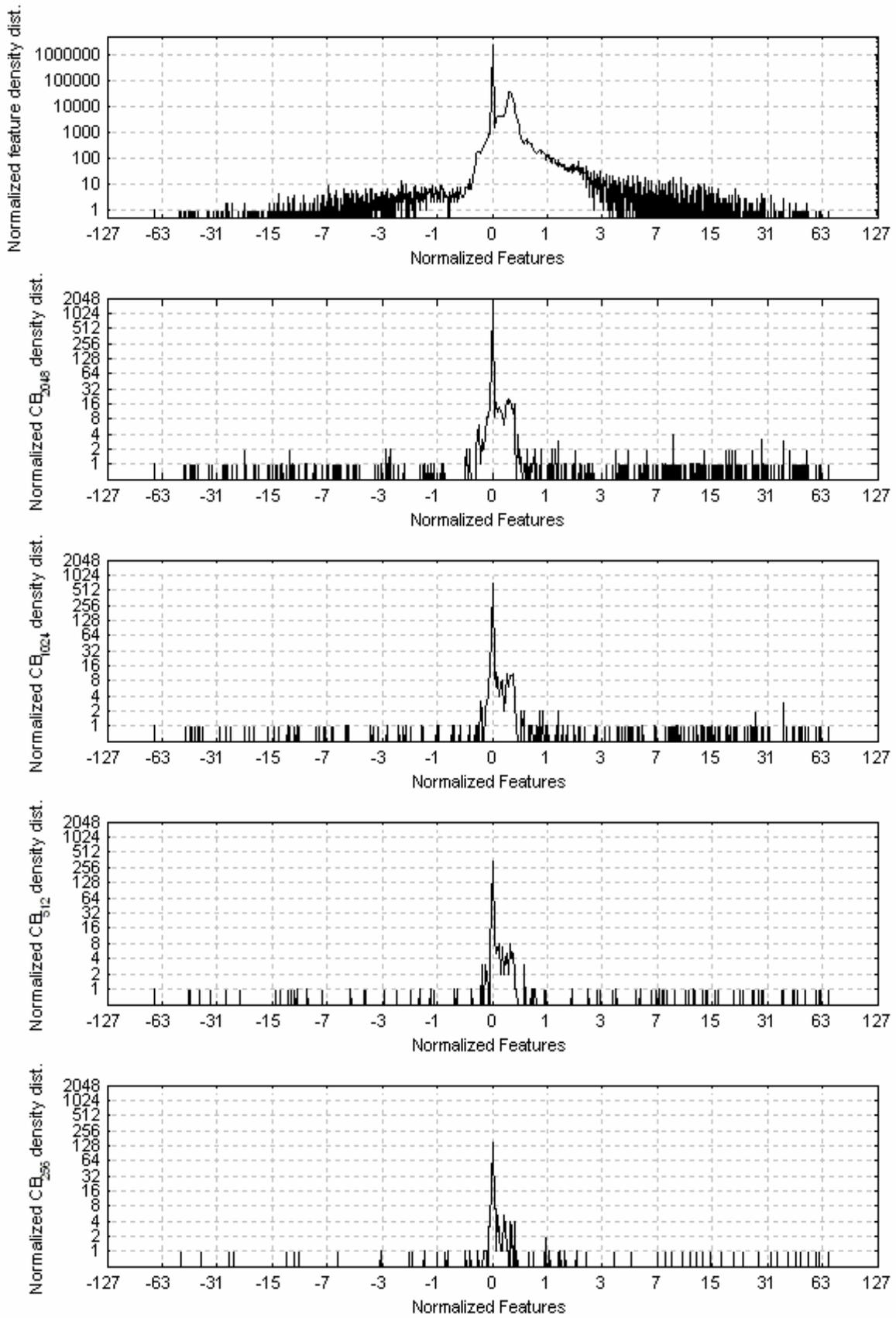


Figure (5.13): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=9$

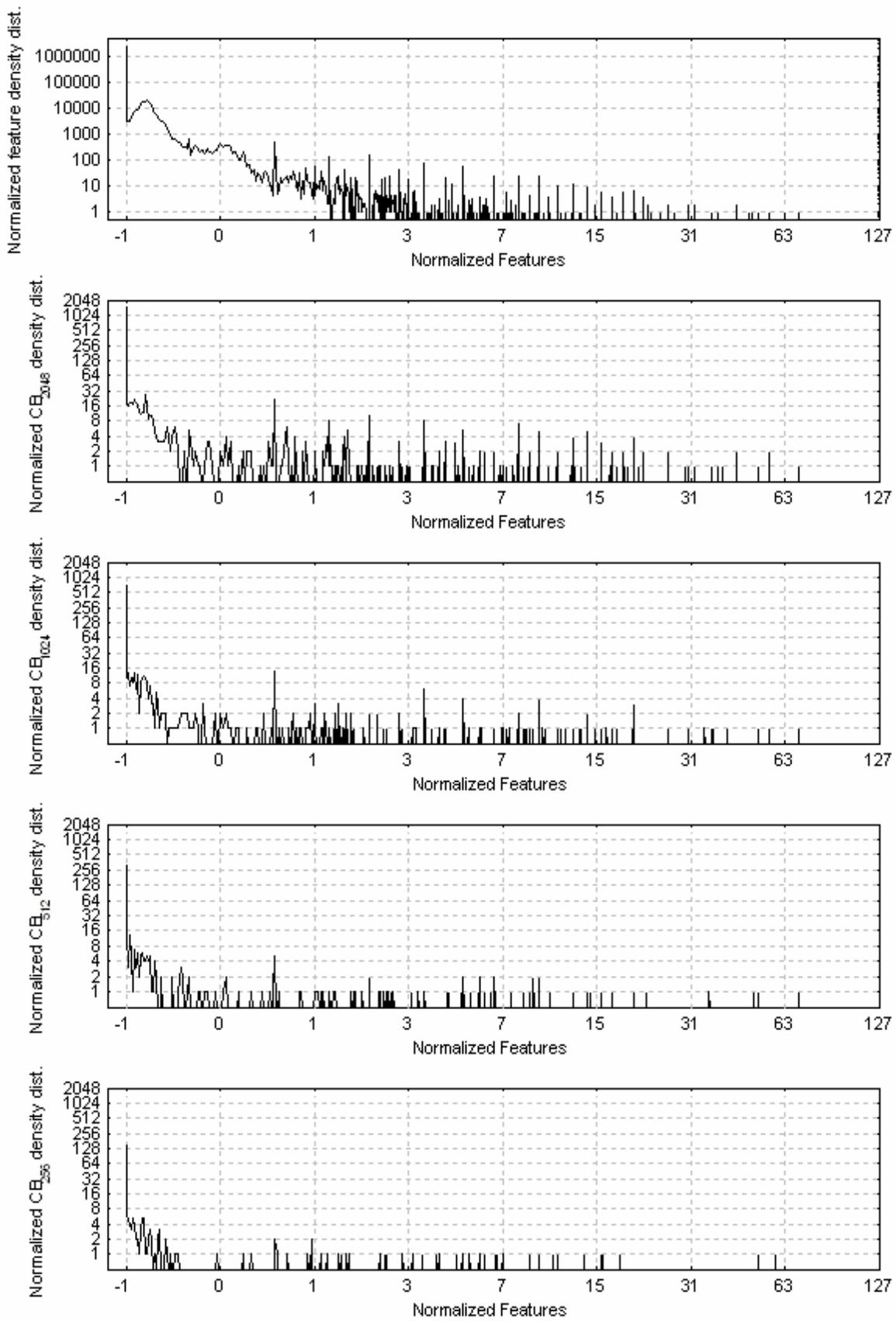


Figure (5.14): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=10$

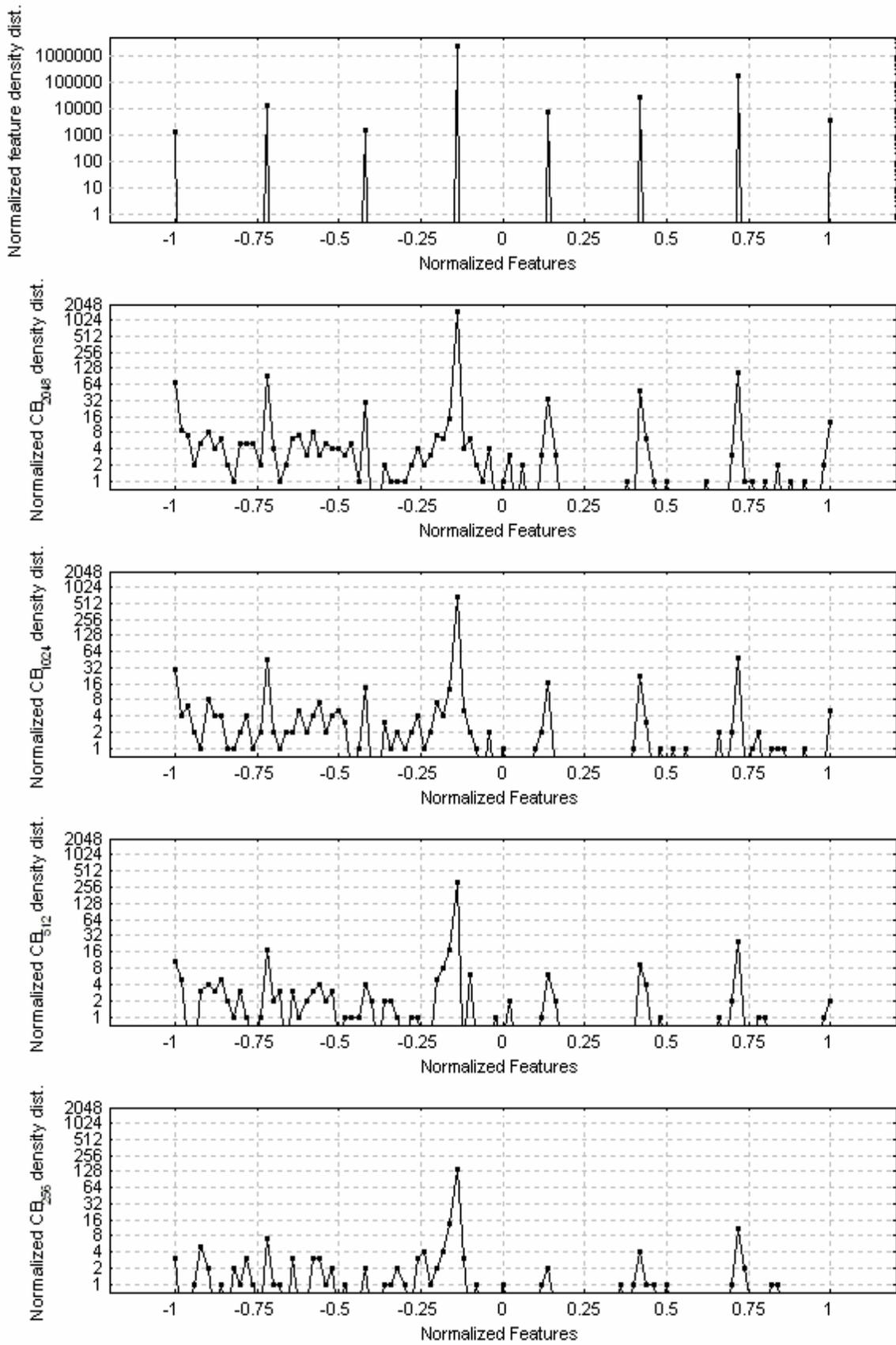


Figure (5.15): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=11$

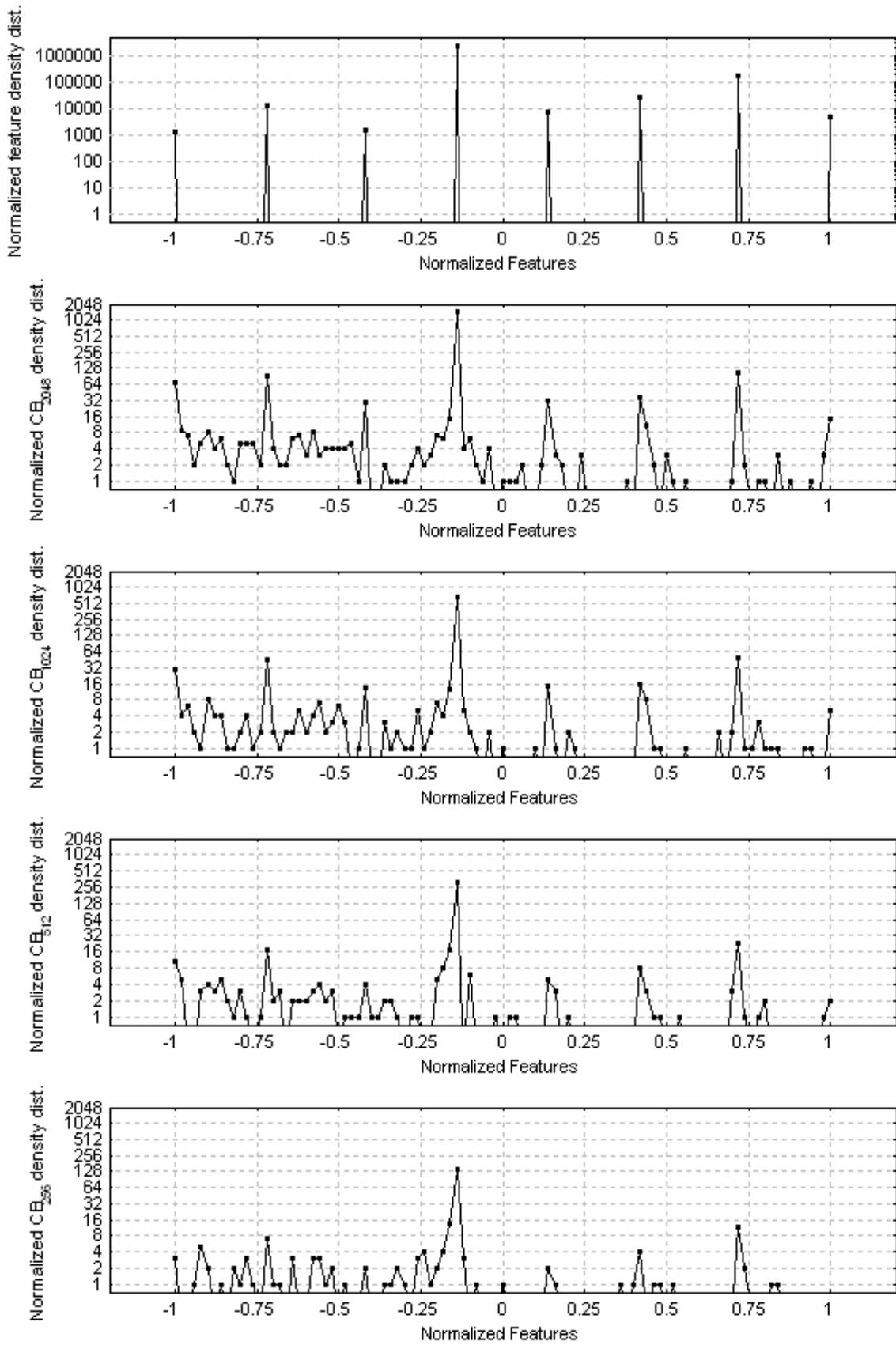


Figure (5.16): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=12$

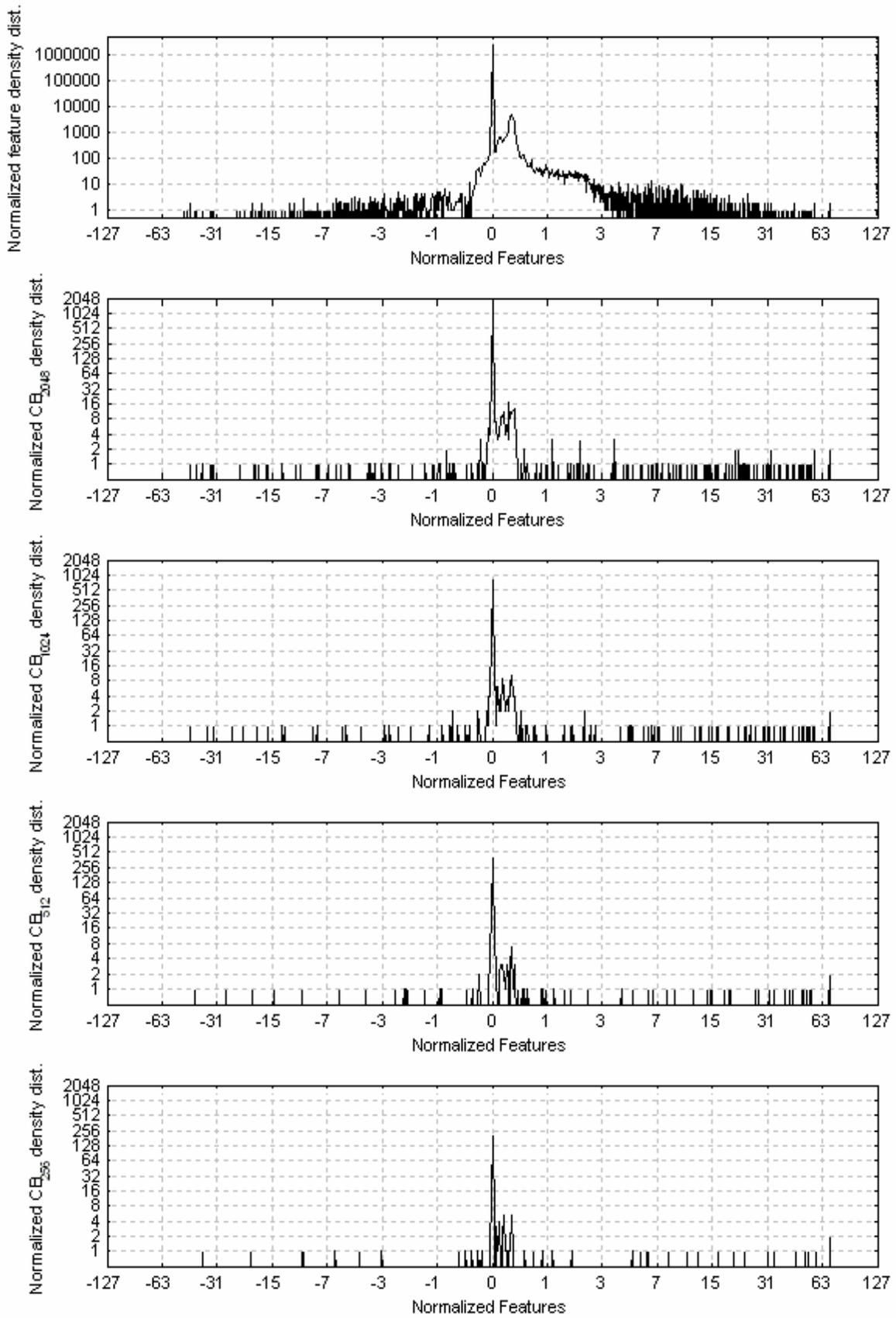


Figure (5.17): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=13$

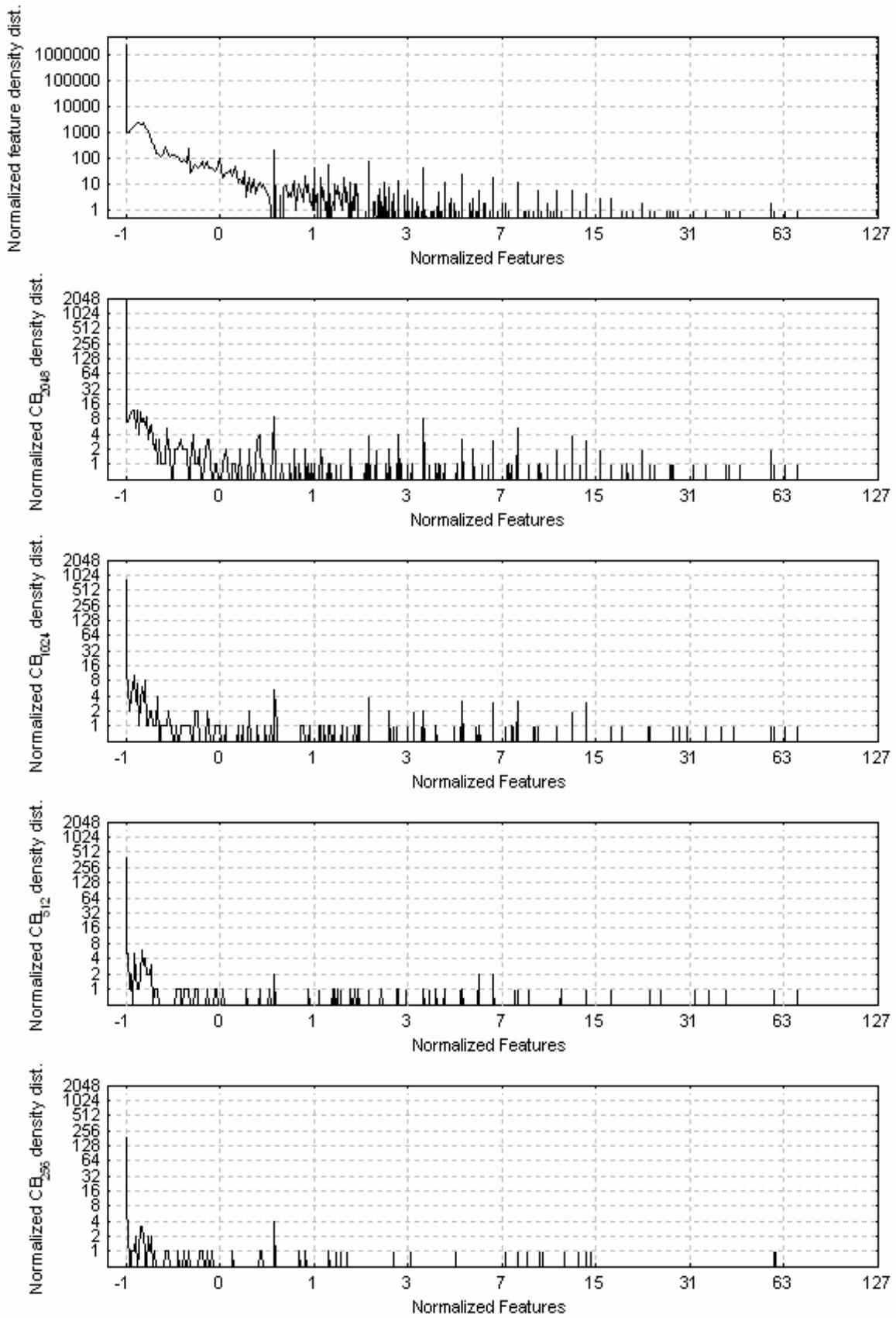


Figure (5.18): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=14$

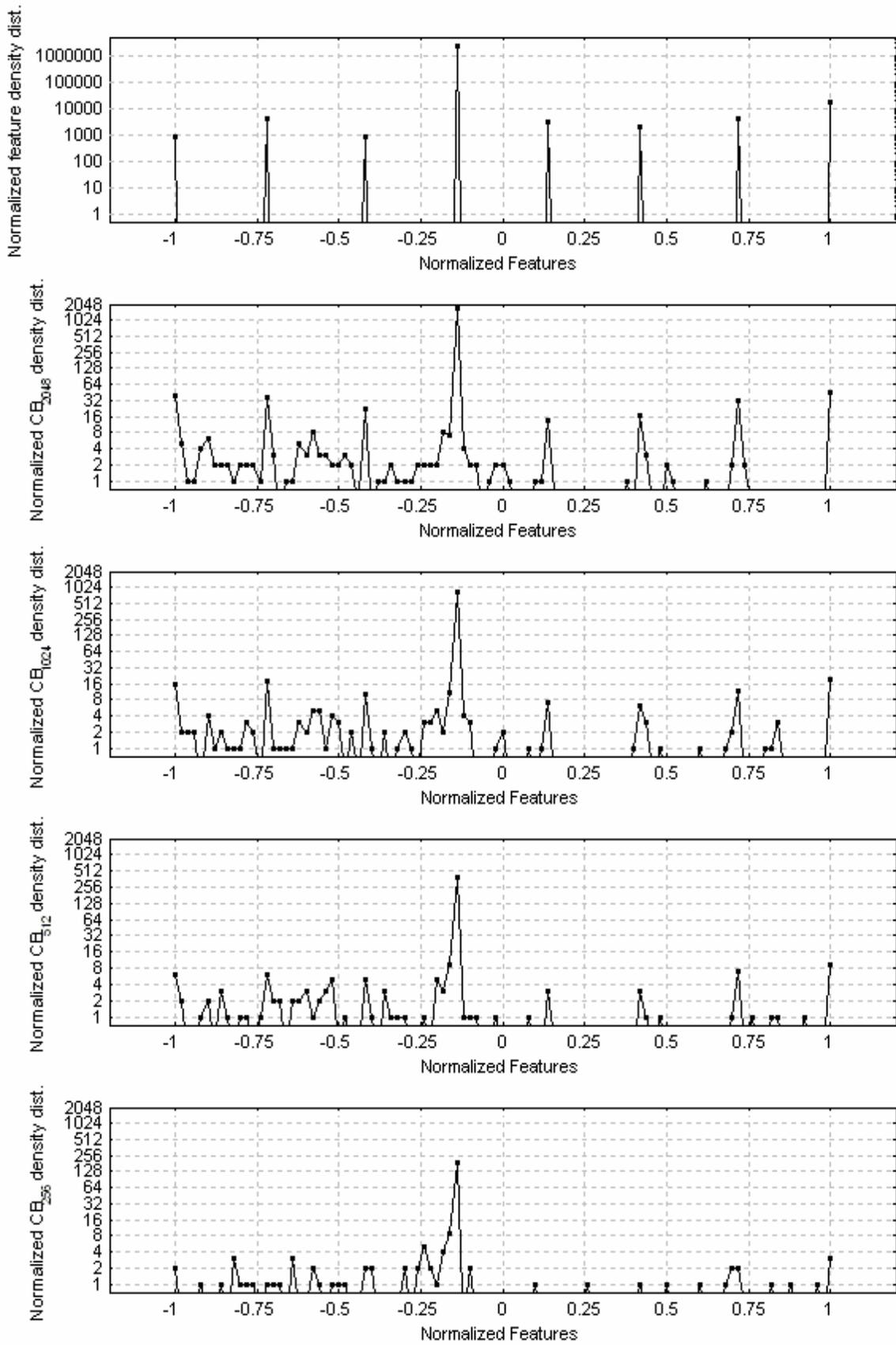


Figure (5.19): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=15$

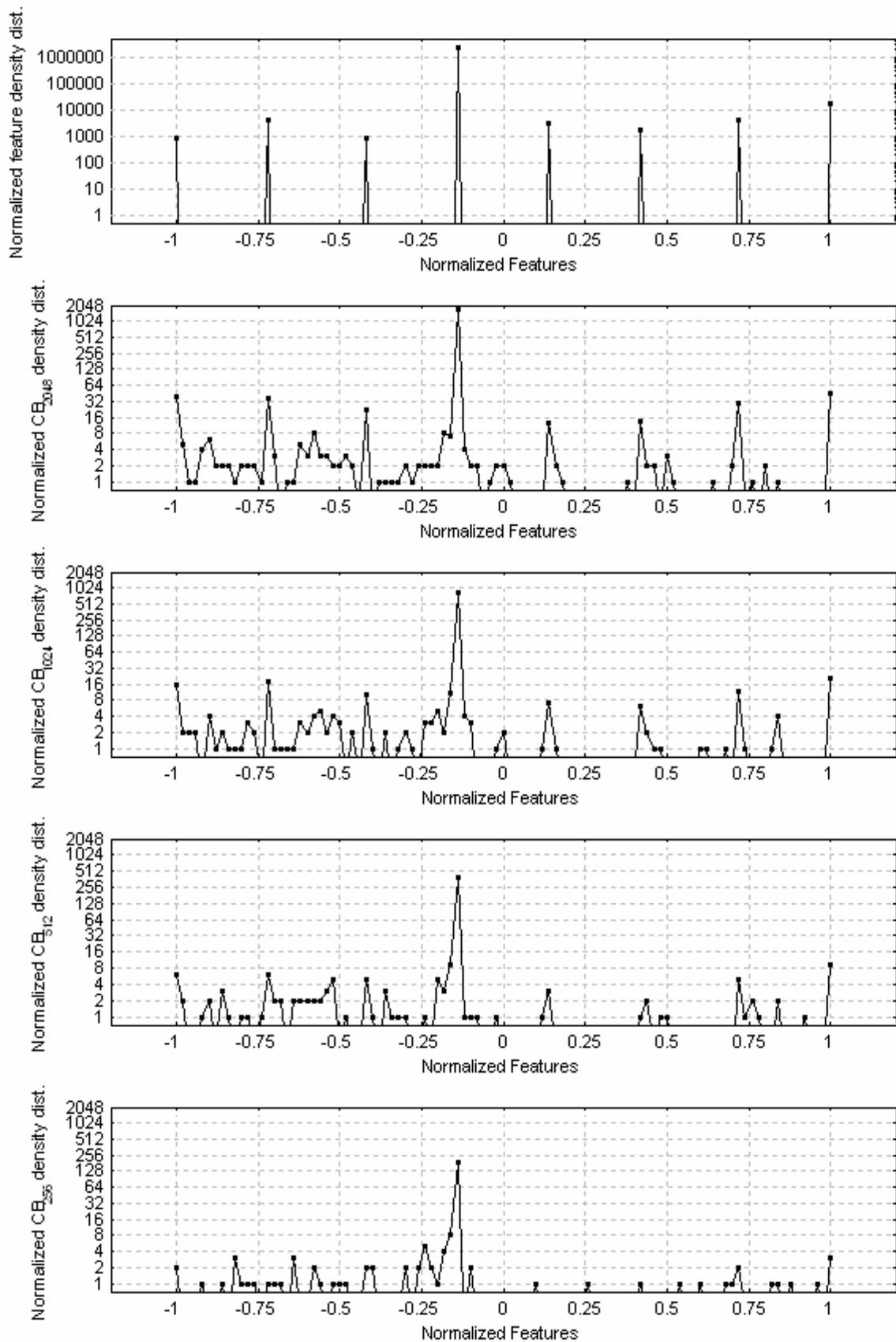


Figure (5.20): Normalized features density distributions with its codebooks, of sizes 2048, 1024, 512, 256, density distributions projected on $d=16$

CHAPTER 6

HMM from ASR to OCR

The output of a real-world process may be observed in a form of a continuous or discrete signal. The objective is to build a signal model that explains and characterizes the occurrence of the observed output. The domain of possible signal models can be dichotomized into: the deterministic models and the statistical models [Rabiner 1989]. The deterministic models exploit some known properties of the signal, and estimate values of the parameters of the signal model. In the statistical models, the signal can be well characterized as a parametric random process, and the parameters of the stochastic process can be estimated in a precise manner. An example of such statistical models is Hidden Markov Models.

HMM is a powerful tool in the field of signal processing. HMM's have been successfully used in speech recognition [Rabiner 1986, 1989]. The application of HMM's has been extended to include text recognition [Chen 1994], [Bunke 1995], [Steinherz 1999]. This is due to the similarities between speech and written words since they both involve co-articulation, which suggests the processing of symbols with ambiguous boundaries and variations in appearance.

6.1. ASR and OCR Problems Analogy

Solutions that tackle ASR and font-written OCR are typically based on the concept of sub-word units (phonemes and graphemes) that are concatenated to form utterances and words respectively.

In ASR; the speech signal, which is a 1D function of time, is sliced into a sequence of windows (called frames) and a features vector for each frame is computed. Frame widths should not be too wide or too narrow [Cho 1995]. When the

former case holds true, a large portion of the computed feature vectors would be taken from inter-phoneme regions leading to the inability to neither build stable models nor recognize the basic units. When the latter condition holds true, the computed feature vectors would be unable to reflect the local characterizing properties of the signal. (See Figure (6.1))

As the phoneme widths typically range between 40-to-250 ms, and as the digital acquisition of speech signals are done at about 44 k bits/s at 256 PCM, a compromise is easily made in speech and frame widths are typically chosen to be within 5-to-10 ms where there are enough samples to reflect the local nature of the signal. In the world of digital speech processing, Mel Frequency Cepstral Coefficients (MFCC's) [Davis 1980] are the well established and widely agreed-upon features to characterize the local identity of the speech signal.

On the other hand, the OCR problem is a one of recognizing a 2D text image signal, with the sliding window moving on the writing direction (from right-to-left in the case of Arabic) and having a fixed height equal to that of the word/line being analyzed. (See Figure (6.1))

Selecting the frame width is subject to the same compromise mentioned just above while talking about speech frames. Taking into consideration the typical printing resolutions around 1200 dpi, the scanning resolutions ranging from 300 to 600 dpi and the most frequently used font sizes ranging from 10 to 24, it is then easy to infer that OCR frame widths should not exceed 4 pixels.

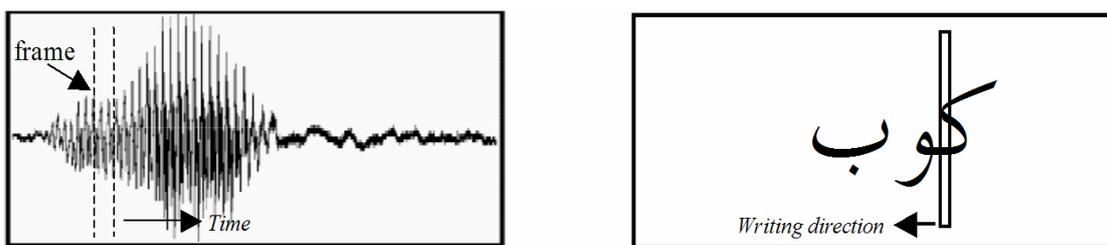


Figure (6.1): Sliding window over a speech signal and over a text image bitmap

Despite the existence of well crafted image characterizers that are proven effective in discriminating images as a whole [Pratt 1991], [Gonzalez 2002], none of them is very successful to describe text images differentially through a sequence of narrow frames inside the sliding window. Here comes the main challenge of applying the OCR-to-ASR analogy; the absence of a standard features set that are able to locally identify text-image signals in a way that reveals the essential writing concepts

so that the smallest variance possible is preserved over the feature vectors obtained from different fonts/sizes of the same sequence of graphemes.(described in chapter 4).

Given the OCR-to-ASR analogy, the widely used approach to solving the ASR problem is then intuitively quoted for solving the Arabic font-written OCR one.

The writing process, like speech and like any other form of language, starts in the mind of some human writer when he/she composes a message to be delivered to his/her intended reader(s). The ultimate goal of this process is to recognize the message as intended by analyzing the message as observed. This is illustrated by the famous noisy channel communication model shown in Figure (6.2).

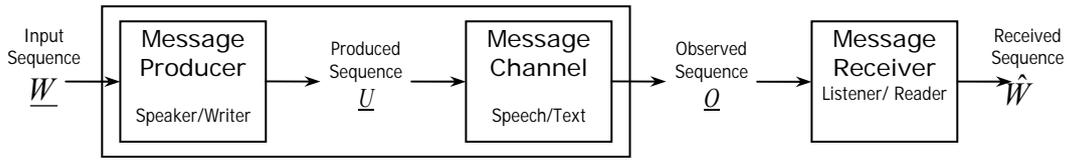


Figure (6.2): Noisy channel communication model where \underline{W} is the intended message composed by the producer, and \underline{Q} is the message as observed by the receiver.

The distortion caused by the noisy channel typically leaves us with the ambiguity problem of having multiple possible perceived sequences from which we have to recognize the intended message sequence. While building a fully rule-based language recognizer is almost impossible due to the randomistic nature of the channel distortion, as well as our typical incomplete knowledge of the laws that govern the linguistic phenomenon, the goal of a language recognizer (like OCR) of finding the intended sequence (graphemes/ligatures in the case of OCR) can practically be achieved stochastically [Attia 2002].

The maximum a posteriori probability (MAP) approach is one of the most widely-used and mathematically well-founded methodologies in this regard. According to this methodology, the elected message sequence $\hat{\underline{W}}$ is selected to maximize the a posteriori probability $P(\underline{W}|\underline{Q})$ over all the possible (producible) messages which is formally expressed as

$$\hat{\underline{W}} = \arg \max_{\forall \underline{W}} \{P(\underline{W} | \underline{Q})\} = \arg \max_{\forall \underline{W}} \left\{ \frac{P(\underline{Q} | \underline{W}) \cdot P(\underline{W})}{P(\underline{Q})} \right\} = \arg \max_{\forall \underline{W}} \{P(\underline{Q} | \underline{W}) \cdot P(\underline{W})\} \quad \text{Eq.(6.1)}$$

Where, $P(\underline{Q}|\underline{W})$ is called the likelihood probability which models the forward conditional stochastic relation between intended/input graphemes and their

consequent observations, and $P(\underline{W})$ is called the language model that gives the a priori marginal probability of any possible sequence of ligatures. The a priori probability of the observations $P(\underline{Q})$ can obviously be omitted from the maximization formula as it is independent of \underline{W} . [Attia 2002]

In the proposed OCR system - like the ASR one - the likelihood probability term $P(\underline{Q}|\underline{W})$ is modeled via the public widely-used stochastic methodology of HMM that has dominated the world of digital speech processing – especially ASR – during the past two decades. The success of HMM in this regard is appealing for researchers to deploy it as a suitable tool for cursive script recognition for the following reasons [Bunke 1995]:

- HMM's are stochastic models that can cope with noise, and pattern variations occurring due to different fonts, sizes, and styles.
- The number of observations representing an unknown input grapheme/ligature or word may be of variable length, which is a fundamental requirement in cursive script as the lengths of individual units may greatly vary.
- The HMM recognition process implemented via search trellis does not only produce the MAP optimal sequence, but also the boundaries of the units comprising this sequence. In Arabic font-written OCR, this eliminates the need for any heuristic graphemes segmentation processes which are typically not only difficult but also error-prone.
- There are rigorously-studied, stable, and computationally efficient algorithms for both the training, and the recognition using HMM's. Moreover, off-the-shelf implementations of such algorithms are easily affordable.

Continuous HMM's model a sequence of continuous-valued observations. In order to do this, the model typically represents the observation pdf as a mixture of Gaussian distributions. This results in a large number of free parameters to be estimated during the training process hence such models typically have a very large requirement for training data. However, discrete HMM's are suitable for modeling processes that emit a sequence of discrete-valued observations with small number of free parameters to be estimated during the training process.

Upon the examination of large populations of our proposed feature vectors, it got clear that it is hard to find mixtures of Gaussians that properly represent that kind

of hybrid data. So, the decision is made to adopt discrete HMM's rather than its other versions.

To build an OCR system that is able to recognize an open-vocabulary Arabic font-written text with M different graphemes, we construct M different HMM's each of them represents one of the Arabic font ligature. To keep HMM search lattices at reasonable widths, which in turn reduces the computational complexity hence the recognition WER [Bunke 1995], the word (not the line) has been chosen in our work as the major unit for training and recognition.

6.2. Theory and Elements of Discrete HMM

HMM is a stochastic process with an underlying finite-state structure. Each one of these states is associated with a random function. Within a state the signal possesses some measurable, distinctive properties. Within a discrete period of time, the process is assumed to be in some state and an observation is generated by the random function of that state. The underlying Markov chain changes to another state based on the transition probability of the current state. The sequence of states is hidden, only the sequence of observations produced by the random function of each state can be seen.

Consider a discrete word recognizer system that at any instance of time may only be in one of its state set. The system undergoes a change from one state to another according to a set of probabilities associated with each state. These transitions take place in regular spaced discrete periods of time. In other words, a system transits from state S_i at time t to a state S_j at time $t+1$, $t = 1,2,3, \dots, N$ (no. of observations) and $i, j = 1,2, \dots, L$ (no. of states/word).

The most important and difficult element to be decided is the number of states, L_g , in the grapheme model, since there is no systematic approach to do so. As mentioned before, at each time unit t the model makes a transition from one state to another or may remain in the same state. This transition is based on a transition probability related to the previous state. This is called a first-order HMM. When a state is entered, an observation output is produced based on an observation probability related to the random function associated with that state. In general, states of an HMM are interconnected based on the topology of the HMM as discussed in section 6.3.

Another important element to be decided is the number of observation symbols K per state. The observation symbols correspond to the physical output of the

system being modeled. For our developed character recognition system, the observations are semi-continuous and are produced as vectors. In this case the vectors are quantized into one of the permissible sets using VQ. Individual symbols are denoted as $\hat{z}_1, \hat{z}_2, \dots, \hat{z}_k$.

The rest of the elements which characterize the discrete observation HMM are:

1. The initial state probability. This is the probability of being in state S_i at $t = 1$.

$$\pi = \{\pi_i = P(S_i \text{ at } t = 1)\} \quad \text{Eq. (6.2)}$$

2. The state transition probability. This is the probability of being in state S_i at time t , then transiting to state S_j at time $t + 1$.

$$A = \{a_{ij} = P(S_j \text{ at } t + 1 / S_i \text{ at } t)\} \quad \text{Eq. (6.3)}$$

3. The observation symbol probability. This is the probability of observing symbol \hat{z}_k while the model is in state j at time t .

$$B = \{b_i(k) = P(\hat{z}_k \text{ at } t / S_j \text{ at } t)\} \quad \text{Eq. (6.4)}$$

Considering the previous elements mentioned, a complete specification of an HMM requires specifying two model parameters, L_g and K , the observation symbols, and three sets of probability measures A , B and π_i . The compact notation to be used to refer to a grapheme HMM is $\lambda_g(A, B, \pi)$. This notation is reduced to $\lambda_g(A, B)$ in case of using the left to right (L-R) HMM topology.

6.3. HMM Topologies

By placing certain restrictions on the structure of the model, the model will tend to behave differently. A model in which every state could be reached (in single step) from every other state of the model is known as an ergodic model, or fully connected model. An example of ergodic model is shown in Figure (6.3).

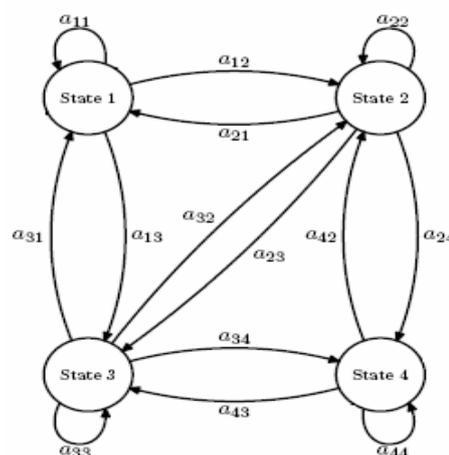


Figure (6.3): Ergodic Hidden Markov Model

For certain types of signals, other types of models have been shown to model that signal's properties better [Rabiner 1989]. In case of time variant signal, the L-R model, also known as Bakis model, is such a model. The underlying state sequence of this type of model is forced to start in state 1 (i.e. the leftmost state) and can only make transitions to higher states (i.e. to the right) or back to the present state at time passes. In addition, the model must end in state L_g (the rightmost state). This allows the sequence of states to represent the passage of time and has become the most popular form of HMM used in speech recognition and character recognition. An example for L-R model is shown in Figure (6.4).

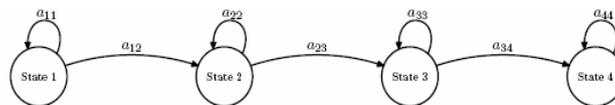


Figure (6.4): An Example of Left-to-Right HMMs

The functional property of all L-R HMM's is that the state transition coefficients have the property:

$$a_{ij} = 0, \quad \forall j < i \quad \text{Eq. (6.5)}$$

i.e., no transitions are allowed to states whose indices are lower than the current state.

Furthermore, the initial state probabilities have the property:

$$\pi_i = \begin{cases} 0, & i \neq 1 \\ 1, & i = 1 \end{cases} \quad \text{Eq. (6.6)}$$

6.4. The Basic Problems of HMM's

Given the form of the grapheme HMM, there are three basic problems of interest that must be solved for the model to be useful in our application. These problems are the following:

Problem 1 (The Evaluation Problem):

Given a word model (λ_w), concatenation of grapheme models λ_g , and a sequence of observations $\underline{Q} = (\underline{Q}_1, \underline{Q}_2, \dots, \underline{Q}_N)$, what is the probability that the model generates the observations $P(\underline{Q} | \lambda_w)$?

Problem 2 (The Decoding Problem):

Given a set of models (λ_g) and a sequence of observations $\underline{Q} = (\underline{Q}_1, \underline{Q}_2, \dots, \underline{Q}_N)$, what is the most likely state sequence in these models that produces the observations?

Problem 3 (The Training Problem):

Given a model (λ_w) and a sequence of observations $\underline{O} = (\underline{O}_1, \underline{O}_2, \dots, \underline{O}_N)$, what should the model parameters be to increase the probability of generating the observations?

The evaluation problem can be viewed as a way of scoring how well a given models matches a given observation sequence. So it is very useful in the case in which we are trying to choose among several competing models.

The decoding problem is one in which we attempt to uncover the hidden part of the model, i.e. to find the best matching state sequence given an observation sequence. So we try to find the best state sequence not the correct state sequence, we need an optimality criterion to solve it. The choice of criterion is a strong function of the intended use for the uncovered state sequence.

The training problem is the most important of the three problems. If we could solve this problem, we would have the mean to automatically learn the parameters given an observation sequence.

6.4.1. Model Evaluation Problem

In order to evaluate models, we must first calculate $P(\underline{O} | \lambda_w)$, the likelihood that a sequence of observations, \underline{O} , were produced by a given word model λ_w . For any observation sequence produced by a model, and a known sequence of states followed to produce this sequence of observations, the probability of the model producing these observations and following this state sequence in case of L-R HMM can be calculated as:

$$P(\underline{O}, S | \lambda_w) = b_{s_1}(\underline{O}_1) \cdot \prod_{i=2}^N a_{s_{i-1}, s_i} \cdot b(\underline{O}_i) \quad \text{Eq. (6.7)}$$

In most practical solutions, however, the state sequence followed is unknown to us. Since there may be many state sequence paths that can produce this observation sequence, the probability that \underline{O} was produced by λ_w is the sum of all the probabilities $P(\underline{O}, \mathfrak{S} | \lambda_w)$ for all state sequence paths, \mathfrak{S} . Computing the probabilities for each of these paths would be computationally very expensive, requiring on the order of $2 \cdot N \cdot G^N$ calculations (where G is the number of states per word). Fortunately, an efficient algorithm using a technique called forward probabilities exist

which can calculate this sum of probabilities in the order of $G^2 \cdot N$ calculations, and can report the state sequence of the most probable path. This is known as the Forward-Backward algorithm [Rabiner 1989] since it can be similarly defined using backward probabilities.

6.4.2. Training Problem

The most difficult problem of HMM's is to determine a method to adjust the models parameters (A, B) to maximize the probability of the observation sequence given a set of models. There is no known way to analytically solve for the model which maximizes the probability of the observation sequence. In fact, given any finite observation sequence as training data, there is no optimal way of estimating the models parameters. We can, however, choose the models parameters (A, B) such that $P(\underline{Q}|\underline{W})$ is locally maximized using an iterative procedure such as the Baum-Welch method (or equivalently the EM (expectation-modification) method).

6.4.2.1. Baum-Welch Re-Estimation Procedure

It is an iterative procedure, using the Forward-Backward algorithm, used for re-estimating the models parameters (A,B). The Baum-Welch re-estimation equations [Rabiner 89] are used to produce a better estimate of the models parameters, from the initialized models. A training observation sequence is used to iteratively improve λ_g until the likelihood that the model λ_w produced the observation sequence reaches a maximum at which point the model parameters have converged. It has been shown that this convergence will be to at least a local maximum of optimization surface. Because this surface is complex and has many local maxima, careful choice of the initialization values of the parameters of the model is necessary to ensure that this local maximum is in fact the global one.

6.4.2.2. Initialization Procedures

The Baum-Welch algorithm is based on creating a new set of parameters using results obtained from the model with previously defined set of parameters. Therefore, some initial set of parameters must be picked in order for the training method to be possible. In theory, the re-estimation equations should give values of HMM parameters that correspond to a local maximum of the likelihood function, but it does

not guarantee finding a global maximum. If the initialization is poor, a poor local maximum may be obtained. For instance, if a probability is initialized to be zero, it will remain zero with every iteration. Therefore, the choice of a good starting point has an impact on how well the model will be.

Basically, there is no simple or straightforward method to choose initial estimates of the HMM parameters so that the local maximum reached is the global maximum of the likelihood function. Instead, Rabiner [Rabiner 89] states that: “an experience has shown that either random (subject to stochastic and non-zero value constraints) or a uniform initial estimate of the state probability distributions (A) is adequate to give useful re-estimates of these parameters in almost all cases”. For the observation probabilities, (B), experience has shown that good initial estimates are helpful in the discrete symbol case. However, a uniform initialization is likely to be sufficient; assuming a reasonable amount of training data exists. If the parameters are allowed too many degrees of freedom with respect to the amount of training more sophisticated forms of initialization are needed.

Such initial estimates can be obtained in a number of ways of ways, including manual segmentation of the observation sequence into states with averaging of observations within states, maximum likelihood segmentation of observations with averaging (Viterbi segmentation), and segmental K-means segmentation with clustering,, etc.

6.4.3. Decoding Problem

Decoding attempts to uncover the hidden part of the model by finding the most likely state sequence associated with the examined observation sequence to obtain the best sequence of ligatures that constitute a word. The discrete HMM-based graphemes recognition problem is rendered to obtaining the path $S_{f_g(t=1),f_s(t=1)}, S_{f_g(t=2),f_s(t=2)}, \dots, S_{f_g(t=N),f_s(t=N)}$ whose sequence of nodes accumulates the maximum sum of probabilities among all the possible paths over the search trellis shown in Figure (6.5); where $N \geq \max_{i=1}^M(L_i)$ is the length of the sequence of observations and M is the number of graphemes. Any legitimate path can start at $t=1$ only at the first state (marked by an inwards arrow) of any grapheme model, and can end only at $t=N$ at the last state (marked by an outwards arrow) of any grapheme model.

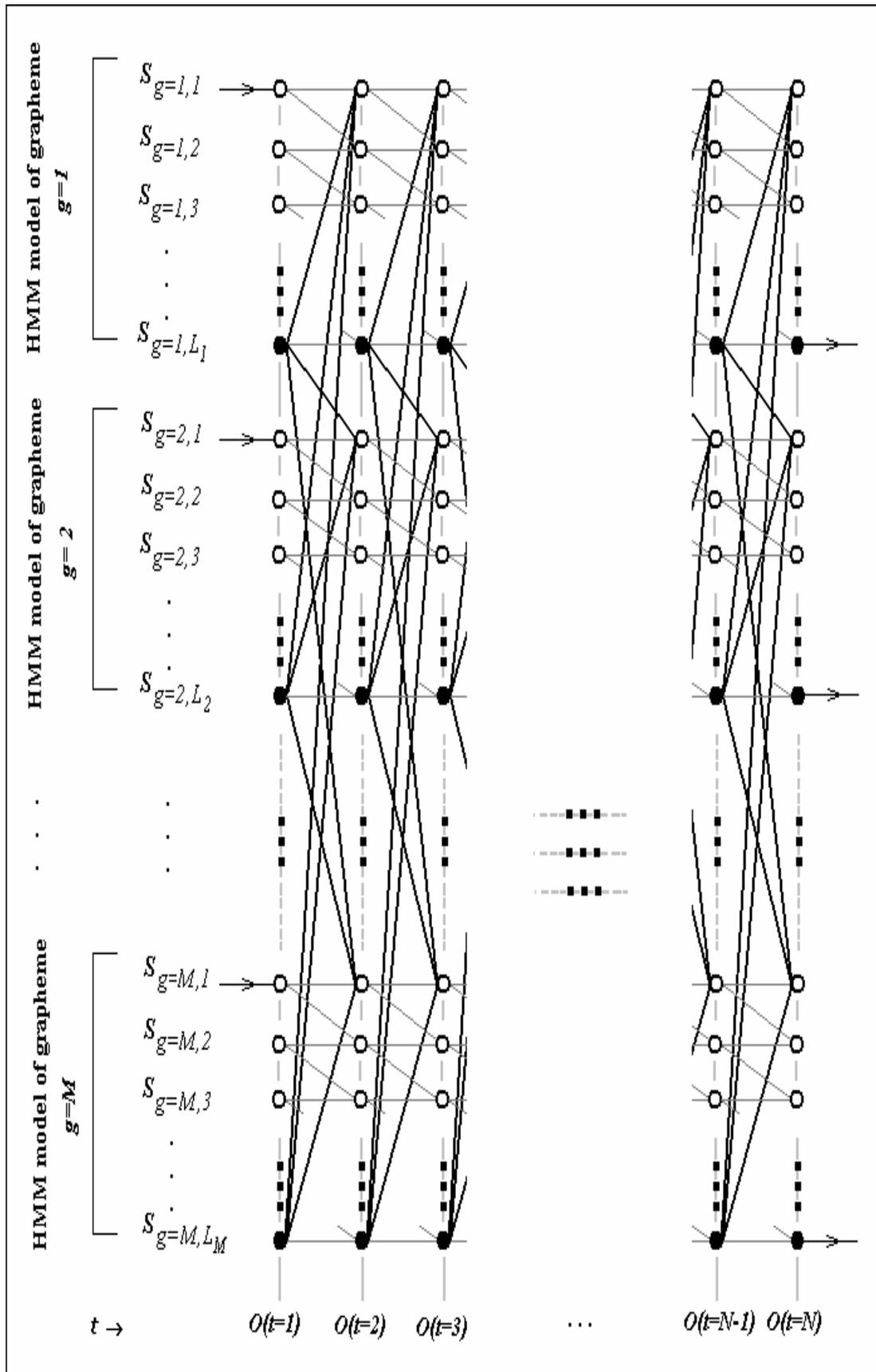


Figure (6.5): HMM search trellis for graphemes recognition.

As 1st order HMM 's are deployed; every state other than the last one in each HMM grapheme model $S_{i,k \leq L_i}$ represented by hollow nodes are connected at the next time instant only to either itself or its subsequent state in the same model by light links whose incremental probabilistic weights are respectively $\log(p(S_{i,k \leq L_i} | S_{i,k \leq L_i}))$ and $\log(p(S_{i,k+1 \leq L_i} | S_{i,k \leq L_i}))$.

The conditional probability of a given observation O_t to occur at a given state $S_{i,k \leq L_i}$ is $\log(p(O_t | S_{i,k \leq L_i}))$. The 1st order approximation of the likelihood probability $p(\underline{O} | \underline{W})$ of the MAP criterion in Eq. (6.1) can then be expressed by the following formula

$$\log(p(\underline{W} | \underline{O})) \approx P_1^* = \sum_{j=1, f_s(j) \neq 1}^{j=N} \left(\log(p(S_{f_g(t=j), f_s(t=j)} | S_{f_g(t=j-1), f_s(t=j-1)})) \right) + \sum_{j=1}^{j=N} \left(\log(p(O_{j=t} | S_{f_g(t=j), f_s(t=j)})) \right) \quad \text{Eq. (6.8)}$$

The last state of each HMM grapheme model S_{i,L_i} represented by filled nodes are connected at the next instant either to itself, or to the first state in each HMM grapheme model $S_{j,1}$ by dark links whose incremental probabilistic weights $\log(p(g_j | g_i))$ constitutes the grapheme-level bi-gram approximation of the language model in the MAP criterion (see Eq. (6.1) above) $p(\underline{W})$ expressed as follows..

$$\log(p(\underline{W})) \approx P_2^* = \sum_{j=1, f_s(j)=1}^{j=N} \log(p(g_{f_g(t=j)} | g_{f_g(t=j-1)})) \quad \text{Eq. (6.9)}$$

Given a sequence of observations, one can use the Viterbi [Rabiner 1989] algorithm to search through the above trellis to obtain the two functions f_g, f_s describing the optimal path whose states sequence correspond to the maximum $P_1^* + P_2^*$ all over the legitimately possible paths in the whole trellis. This leads to the best sequence of graphemes which constitute the recognized word.

CHAPTER 7

Statistical Language Models and Character Recognition

One of the added benefits of HMM's is that they have contextual processing built into them ($P(\underline{W})$) as illustrated in the previous chapter. This can be repeated at many levels (e.g., characters, subword-level, word-level, sentence level) by linking together individual models into some higher level model where the models at the lower level become states for the higher level [Bahl 1983].

As our recognition system can work without any dictionary (open-vocabulary). So, to improve recognition performance we use statistical language models (backoff n-grams), which are well known in speech recognition, on the grapheme level. Statistical Language Modeling attempts to capture regularities of natural languages in order to improve the performance of various natural language applications, e.g. Information Retrieval, Machine Translation and Character recognition.

This chapter is focused on the use of the Statistical Language Models (SLM's) in our specific problem, i.e. the decoding of the Arabic words. As shown by Eq. (6.9), the SLM is supposed to give the a priori probability of a certain word being written. Given the advantage of the statistics available on the likelihood of certain graphemes following a given string of graphemes, impossible paths in the decoding trellis could be pruned out and the less likely hypotheses could get less priority in their expansion. This could increase the accuracy of the recognition.

7.1. Statistical Language Models

A SLM is a statistical model for estimating the probability $P(\underline{W})$ for a given sequence of graphemes

$$\underline{W} \equiv g_1 g_2 \dots g_{L_w} \equiv g_1^{L_w}$$

where L_w is the number of graphemes in a word.

More likely sequences typically have higher probabilities. Finding such adequate information would provide wide language coverage and dramatically constrain the search space for higher language layers. The n -gram model is a well known SLM for its relative simplicity and employed reduction of the problem complexity.

7.2. N-Gram Language Models

The graphemes sequence \underline{W} of length L_w has the probability of occurrence $P(\underline{W})$ that can be decomposed using the chain rule as follows [Schutze 2000]:

$$\begin{aligned} P(\underline{W}) &\equiv P(g_1 g_2 \dots g_{L_w}) \equiv P(g_1^{L_w}) \\ &= P(g_1) P(g_2 | g_1) \dots P(g_{L_w} | g_1^{L_w-1}) \\ &\equiv \prod_{i=1}^{L_w} P(g_i | g_1^{i-1}) \end{aligned} \quad \text{Eq. (7.1)}$$

where $P(g_i | g_1^{i-1})$ is the probability that the grapheme g_i will follow the graphemes sequence g_1^{i-1} . This resultant conditional form of the previous transformation has a remarkable advantage over the marginal one we started with. Natural languages are called a *wide horizon phenomena* since g_i is bonded to earlier and later entities over the whole sequence L_w . Yet the expression elegantly dissects the marginal a priori term into sequences mostly shorter than L_w with causal conditional probabilities dependent only on earlier entities [Attia 2005].

For a vocabulary of size M , there will be more than M^i possible different histories to specify $P(g_i | g_1^{i-1})$ completely. Indeed, this kind of estimation is infeasible even for moderate values of i . It is intuitive that, as text of natural language, the present grapheme g_i mostly has the strongest correlation with its closest past neighbor g_{i-1} and weaker correlations with earlier ones. This is called the *attenuating correlation phenomenon*, which is well described by Figure (7.1). We can then

approximately assume that $P(g_i | g_1^{i-1})$ depends on some previous graphemes g_{i-h}^{i-1} [Attia 2005]. This approximation leads to an n -gram model with

$$n = h + 1 \quad \text{Eq. (7.2)}$$

where h is the effective history length of the sequence g_{i-h}^{i-1} . Thus

$$P(\underline{W}) \cong \prod_{i=1}^{L_w} P(g_i | g_{i-h}^{i-1}) \quad \text{Eq. (7.3)}$$

Hence, for a unigram ($n=1$:no history) this approximation would yield for example

$$P(g_1 g_2 g_3 g_4) \cong P(g_1)P(g_2)P(g_3)P(g_4)$$

and for bigrams ($n=2$: $h=1$)

$$P(g_1 g_2 g_3 g_4) \cong P(g_1 | \langle SOW \rangle)P(g_2 | g_1)P(g_3 | g_2)P(g_4 | g_3)P(\langle EOW \rangle | g_4)$$

where $\langle SOW \rangle$ and $\langle EOW \rangle$ denote a sequence of start and end of word.

Also for trigrams ($n=3$: $h=2$)

$$P(g_1 g_2 g_3 g_4) \cong P(g_1 | \langle SOW \rangle)P(g_2 | \langle SOW \rangle g_1)P(g_3 | g_1 g_2)P(g_4 | g_2 g_3)P(\langle EOW \rangle | g_3 g_4)$$

Recall that the above approximation is valid only under the preliminary assumption that the process is random.

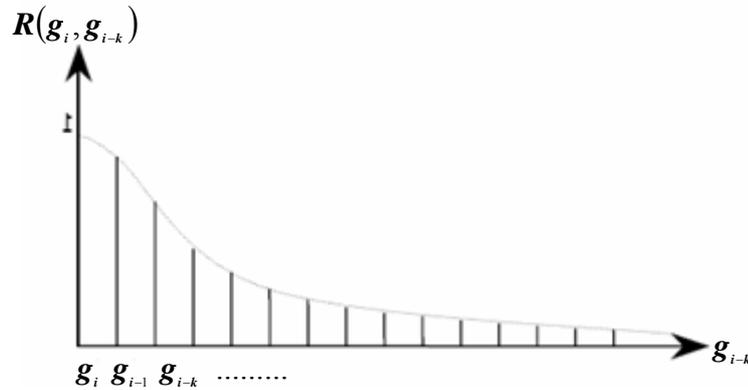


Figure (7.1): The attenuating correlation phenomenon, the correlation between g_i and other previous graphemes.

7.3. The Language Phenomenon from a Statistical Perspective

Let us count the occurrences (frequencies) of single graphemes $C(\underline{W} = g_i)$ (composing the vocabulary of graphemes of size M) in some large fair-text corpus of

size V , assuming that the assemblage sequence is as simple as a single grapheme, and let us rank the graphemes by their frequencies in the corpus. We observe the following:

- 1- Some part, say of size M' , of the graphemes is only covered in the corpus, that is

$$V = \sum_{r=1}^M C(\underline{W} = g_r) = \sum_{r=1}^{M'} C(\underline{W} = g_r), \quad M' \leq M \quad \text{Eq. (7.4)}$$

where r is the rank of grapheme g_r , $C(\cdot)$ is the number of times a certain grapheme appears. We hence define the coverage of the corpus to be

$$I = M'/M \leq 1 \quad \text{Eq. (7.5)}$$

- 2- The rank-frequency relationship roughly follows a hyperbolic curve as shown in Figure (7.2) [Attia 2005]. This curve draws an essential understanding to the language use. Zipf, in his principle of least effort, states that the best compromise of both efforts of speaker and listener while communication results in a use of a few very frequent entities and many low-frequency ones [Schutze 2000]. Also, as a consequence, the coverage increases sluggishly as the corpus gets larger.

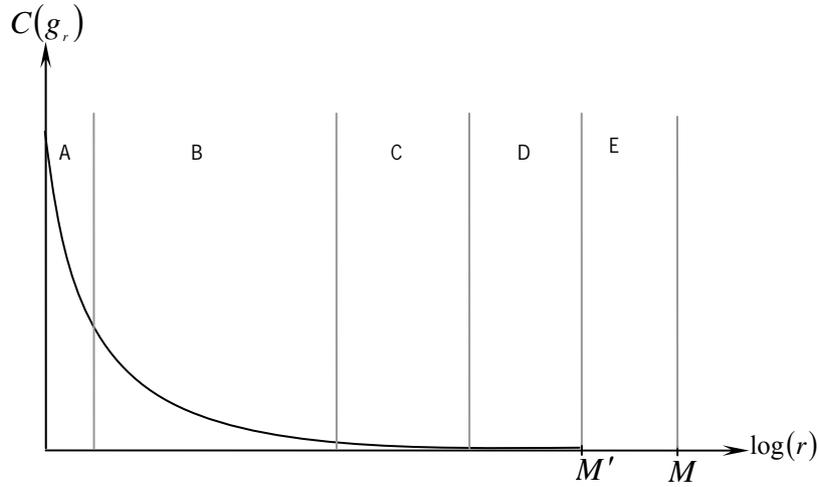


Figure (7.2): Ranking entities by their frequencies. Curve is coarsely considered as of 5 regions. (A) is the stop list, (B) is the frequent list, (C) is the rare list, (D) is the singletons (read-only-once) list, and (E) is the unattested list.

Empirically from the corpus, we have

$$P(\underline{W} = g_r) = \frac{C(\underline{W} = g_r)}{V} \quad \text{Eq. (7.6)}$$

which yields zero for the unattested entities (uncovered part of the vocabulary) of size $M - M'$. We know that this is incorrect since a better coverage would take place if

the corpus is appended some new text. Also for small $C(\underline{W})$ values, $P(\underline{W})$ would be significantly inaccurate [Kapur 1972], [Schutze 2000]. A solution to this problem is addressed by the so-called *probability estimation smoothing techniques* as that of Good-Turing technique.

Ranking longer sequences (assemblages of two graphemes or more) turns out to apply the same phenomenon with greater ratios of uncovered, singletons (read-only-once), and low-frequency sequences, for the same corpus, which affects the model efficiency. An adequately larger corpus is then needed for achieving the same statistical representation as that of single graphemes. On the other hand, a very large corpus may be an overfit and adds unnecessary computation and storage cost.

7.4. Maximum Likelihood Probability Estimation

The probabilities of the graphemes being written (given their histories) are obtained by simply counting the relative frequencies of the grapheme sequences appearing in a text corpus:

$$P(g_i / g_{i-h}^{i-1}) = \frac{C(g_{i-h}^i)}{C(g_{i-h}^{i-1})} \quad \text{Eq. (7.7)}$$

where $C(\cdot)$ is the number of times a certain ligatures sequence appears. This corresponds to a Maximum Likelihood (ML) estimation; the estimated probabilities $P(g_i / g_{i-h}^{i-1})$ maximize the likelihood of the training text. However, this approach gives rise to a serious problem: the model is fitted to the training set and the probability of any n-gram not represented in the training corpus is estimated to be zero.

Also, the method implied by Eq. (7.7) is a good approximation if both counts in the numerator and denominator in the RHS of this equation as well as both the number of h -grams and the number of $(h-1)$ grams in the corpus are large enough. Otherwise this probability should be calculated using a smoothing technique as that described in the following section. This means that the probability mass must be redistributed across all possible n-grams in order to give a nonzero probability to all sequences of n ligatures.

Smoothing allows the extension of an n-gram model trained on a certain text to any other text, but it gives a non-zero probability to n-grams that are impossible from a linguistic point of view. This is the main limit of the n-gram models.

7.5. ‘Bayes’, Good-Turing Discount, Back-Off’ Probability Estimation

Any grapheme in the language vocabulary must have usage in some context, though it seems endless to enlarge some corpus to cover all graphemes. The curve in Figure (7.2) will always, in general, have the regions shown, only with different widths. Presumably, a curve without the uncovered set is smoother than the one shown. The process of biasing the uncovered set on the expense of discounting the other regions is called smoothing [Attia 2005].

A Recognition system that doesn't employ smoothing would refuse the correct solution if any of its input graphemes was unattested in the training corpus and consequently may miss the optimal (most likely) solution. Among the smoothing techniques available in the literature, we selected the so-called Bayes, Good-Turing, Back-off method [Nadas 1985], [Katz 1987]. This technique has the advantage of being based on the Zipf law and is then more robust with respect to a change of data.

In Good-Turing, Back-off method, all n -grams with a nonzero count ($C(g_1^n)$) are discounted according to a discount ratio $d_{c(g_1^n)}$. This discount ratio is predicted based on the Good-Turing estimate [Chen 1998]. The counts subtracted from the nonzero counts are then distributed among the zero-count n -grams via the recursive utilization of lower level conditional distributions, i.e., given the n -gram case, if the n -tuple is not observed frequently enough in the training text then a probability based on the occurrence count of a shorter-context $(n-1)$ -tuple is used instead – using the shorter context estimate is referred to as backing off.

Estimating the n -gram conditional probabilities using the Bayes, Good –Turing, Back-off technique is implemented using the following equation:

$$P(g_n/g_1^{n-1}) = \begin{cases} \frac{C(g_1^n)}{C(g_1^{n-1})} & \text{if } C(g_1^n) > k_1 \\ d_{c(g_1^n)} \cdot \frac{C(g_1^n)}{C(g_1^{n-1})} & \text{if } k_2 \leq C(g_1^n) \leq k_1 \\ \beta(g_1^{n-1}) \cdot \frac{C(g_1^n)}{C(g_1^{n-1})} & \text{if } C(g_1^n) < k_2 \end{cases} \quad \text{Eq. (7.8)}$$

where $\beta(\cdot)$ is the back off weight. For large counts $> k_1$, $C(g_1^n)$ are taken to be reliable, so they are not discounted. Typical values of k_1 and k_2 are suggested to be equal 5 and 1 respectively [Katz 1987], [Chen 1998].

7.6. Arabic Ligature Bigram SLM

In our system, the bigram approximation of the ligatures SLM given by Eq. (7.9) is selected to match the 1st order L-R HMM topology chosen for our system.

$$P(g_j | g_i) = P(c_{j-m_2+1}^{j-m_2} | c_{j-m_1-m_2+1}^{j-m_1}) \quad \text{Eq. (7.9)}$$

where m_1 & m_2 are the number of characters in the ligatures g_i and g_j respectively.

Arabic ligatures may be a compound of one, two, or three characters. By rewriting and expanding Eq. (7.9) using the chain rule for $m_1, m_2 \in \{1, 2, 3\}$, the possibilities of character bigrams, trigrams, ..., 6-grams arise as follows in Eq. set (7.10) below where the shorthand m-grams notation $P(c_{j-m}^j) \equiv P(c_{j-m}, c_{j-m+1}, \dots, c_j)$ is used.

$$\begin{aligned} \log(P(g_j | g_i)) &= & (m_1, m_2) &= \\ \log(P(c_j | c_{j-1})) & & (1, 1) & \\ \log(P(c_j | c_{j-2})) & & (1, 2) & \\ \log(P(c_j | c_{j-3})) & & (1, 3) & \\ \log(P(c_{j-1}^j | c_{j-2})) &= \log(P(c_{j-1} | c_{j-2})) + \log(P(c_j | c_{j-2})) & (2, 1) & \\ \log(P(c_{j-1}^j | c_{j-3})) &= \log(P(c_{j-1} | c_{j-3})) + \log(P(c_j | c_{j-3})) & (2, 2) & \\ \log(P(c_{j-1}^j | c_{j-4})) &= \log(P(c_{j-1} | c_{j-4})) + \log(P(c_j | c_{j-4})) & (2, 3) & \\ \log(P(c_{j-2}^j | c_{j-3})) &= \log(P(c_{j-2} | c_{j-3})) + \log(P(c_{j-1} | c_{j-3})) + \log(P(c_j | c_{j-3})) & (3, 1) & \\ \log(P(c_{j-2}^j | c_{j-4})) &= \log(P(c_{j-2} | c_{j-4})) + \log(P(c_{j-1} | c_{j-4})) + \log(P(c_j | c_{j-4})) & (3, 2) & \\ \log(P(c_{j-2}^j | c_{j-5})) &= \log(P(c_{j-2} | c_{j-5})) + \log(P(c_{j-1} | c_{j-5})) + \log(P(c_j | c_{j-5})) & (3, 3) & \\ & & & \text{Eq. set(7.10)} \end{aligned}$$

The probabilities in Eq. set (7.10) are estimated in our work using Bayes's_Good-Turing_Back-off methodology [Katz 1987], [Attia 2002]. The Arabic text corpus of the written language resource of the NEMLAR project [Yaseen 2006] is used for building the SLM's for our Arabic OCR.

7.7. NEMLAR Arabic Written Corpus

The NEMLAR (Network for Euro-Mediterranean Language Resource and human language technology development and support) is a project supported by the EC with partners from Europe and the Middle East; whose objective is to build a network of specialized partners to promote and support the development of Arabic language resources in the Mediterranean region. The raw textual data of the corpus supported by the NEMLAR project was obtained from Media International the operator of the famous web portal *www.IslamOnLine.net* (75%), RDI internal documents (20%), *Annahar* Lebanese news paper (3%), and other free sources (2%).

The NEMLAR Arabic written corpus consists of 500K words of standard Arabic text compiled from 13 different domains. The Sampling parameters that were taken into considerations in designing the written corpus are [Yaseen 2006]:

- Time span, (mostly recent; i.e. late 1990's till 2005)
- Only Standard Arabic is considered as it is the most commonly accepted variant throughout the native Arabic speakers, and also due to its regularity that can be consistently modeled by the available tools.
- Miscellaneous domains (political, scientific ...) are represented according to their importance weights in potential applications.

The following table represents the size of each of the categories in the selected corpora [Yaseen 2006]:

| Domain | Size | % of the total corpus size |
|--------------------------------|---------------------|-----------------------------------|
| General news | 100,000 | 20 |
| Dictionary entries explanation | 52000 | 10.4 |
| Political news | 51000 | 10.2 |
| Scientific press | 50000 | 10 |
| Sports press | 50000 | 10 |
| Interviews | 49000 | 9.8 |
| Political debate | 35000 | 7.0 |
| Arabic literature | 31000 | 6.2 |
| Islamic topics | 29000 | 5.8 |
| IT Business & management | 20000 | 4 |
| Legal domain text | 20000 | 4 |
| Text taken from Broadcast News | 8500 | 1.7 |
| Phrases of common words | 5500 | 1.1 |
| Total size: | 500000 words | 100 |

Table 7.1: NEMLAR Arabic Written Corpus categories

A comparison between the estimated probabilities of samples of correct words and that of corrupted words (corrupted with errors similar to that occurred by an OCR system) using our built language models is illustrated in Table (7.2). It is easily inferred how the language models, in most cases, can best vote to the correct word hypothesis with higher probability than that of corrupted words. This indicates that our built LM will play a role in enhancing the recognition rate and this will be shown during the evaluation of our built OCR system in the next chapter.

| Correct word | <i>ln prob.(correct)</i> | Corrupted word | <i>ln prob.(corrupt)</i> |
|--------------|--------------------------|----------------|--------------------------|
| خلال | -14.233576 | حلال | -15.770180 |
| فترة | - 14.487489 | قيمه | -16.31970 |
| كثرت | - 16.706138 | كترت | - 20.294384 |
| فيها | - 13.440338 | قيها | - 16.242788 |
| المشادات | - 21.930974 | المساداب | - 22.276129 |
| والكلام | - 19.294052 | والكلام | -33.394045 |
| حول | - 12.816936 | جول | - 13.017899 |
| قانون | - 17.186761 | فايون | - 19.204175 |
| اصول | - 14.845381 | إضول | - 16.812153 |
| المحاكمات | - 23.902800 | المحاكماب | -24.922025 |
| الجزائية | - 20.392440 | الخراتنه | - 27.426248 |
| اللبناني، | - 28.687162 | الليباتي، | - 30.441199 |
| الذي | - 11.734678 | الدى | -12.656464 |
| هو | - 10.178127 | هق | - 14.439598 |
| بصدد | - 18.279730 | يصدد | - 18.460438 |
| التعديل | - 20.576915 | التعديل | -30.538537 |
| والتطوير، | - 27.720399 | والنطوتر، | -29.692344 |
| ارتأينا | - 25.426517 | ارتأنتا | -31.793467 |
| ان | -6.789810 | ان | -19.098538 |
| نعرض | -18.292399 | بعرض | -17.727819 |
| لبعض | - 16.230750 | ليعض | -18.643430 |
| الافكار | - 28.744286 | الافكارز | -34.359380 |
| والمبادئ | | والمنادئ | -26.394882 |
| القانونية | - 23.668309 | القاتونية | -27.600805 |
| والعلمية | - 19.803172 | والعلمته | -27.187210 |
| التي | - 10.621117 | النبي | -11.813260 |
| تتعلق | - 19.707528 | يبعلق | -19.122858 |
| بهذا | - 14.154572 | بهذا | -15.816423 |
| القانون | -19.373087 | القايون | -21.251639 |
| القديم | - 17.222264 | القديم | - 19.131701 |

Table (7.2): Correct words versus Corrupted words probabilities computed by the LM

CHAPTER 8

Arabic OCR System; Implementation and Evaluation

This chapter presents our implemented omni font-written HMM-based OCR system. The characteristics of the used database of Arabic fonts along side a sample of these fonts are also demonstrated. Extensive assimilation and generalization testing experiments have been carried out over the Arabic script to evaluate the system performance. Along with the detailed description of those experiments, this chapter also presents the obtained results manifesting the system superiority over the other ones tried so far with HMM-based OCR systems. Finally an extensive error analysis of the results is also presented.

8.1. HMM-Based OCR System Architecture

Based on the aforementioned OCR-to-ASR analogy discussed over chapter 6, the simplified HMM-based solution architecture of the ASR problem shown in Figure (8.1) can be borrowed to produce an effective solution for the Arabic font-written OCR problem whose architecture is schematized by Figure (8.2).

It is worthy mentioning here that vector quantization is not always a part of ASR systems, as most of them are continuous HMM-based ones like some HMM-based OCR systems. However, our OCR presented herein is based on discrete HMM due to the continuous/discrete hybrid nature of the components of its features vector whose design is detailed in chapter 4.

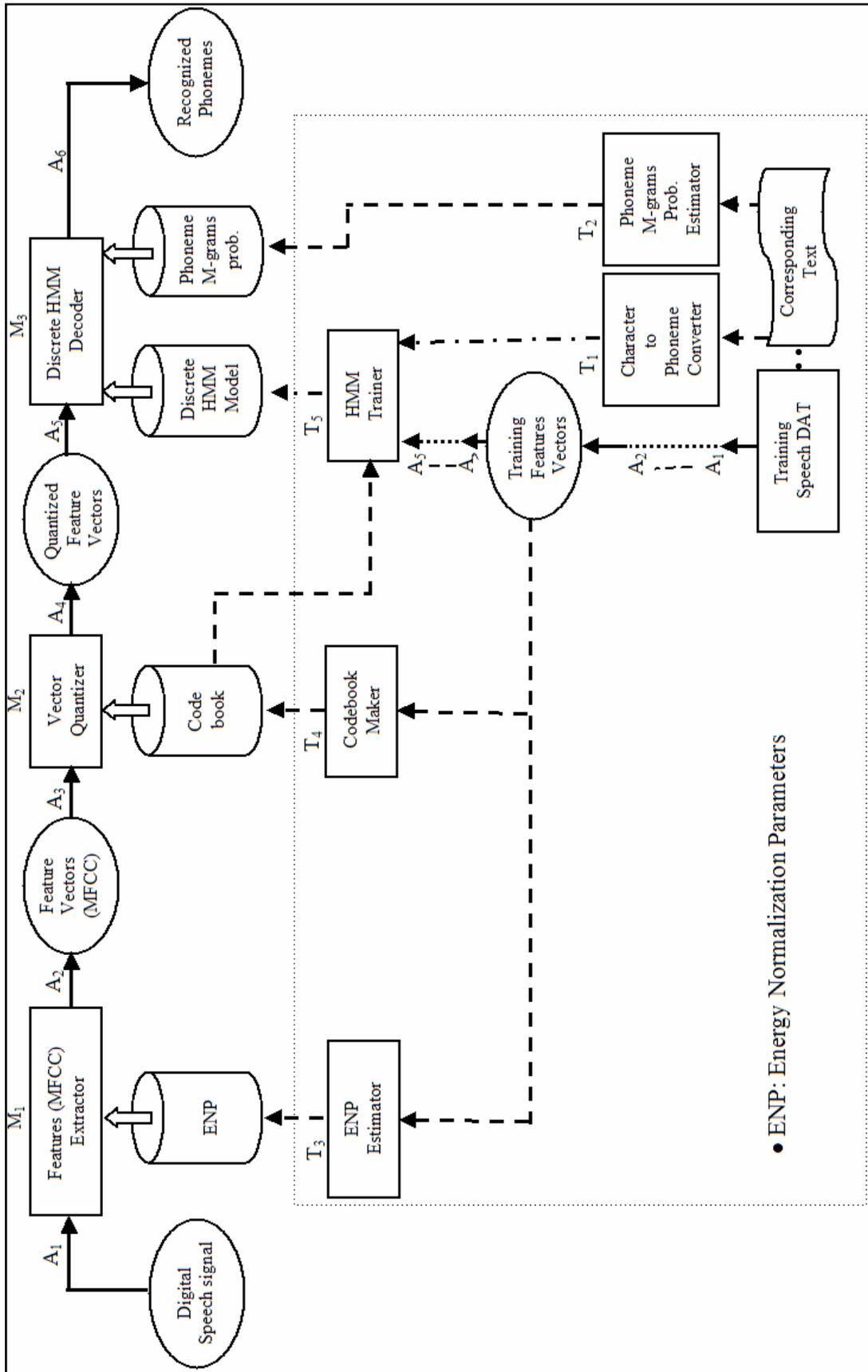


Figure (8.1): Discrete HMM-based ASR system block diagram.

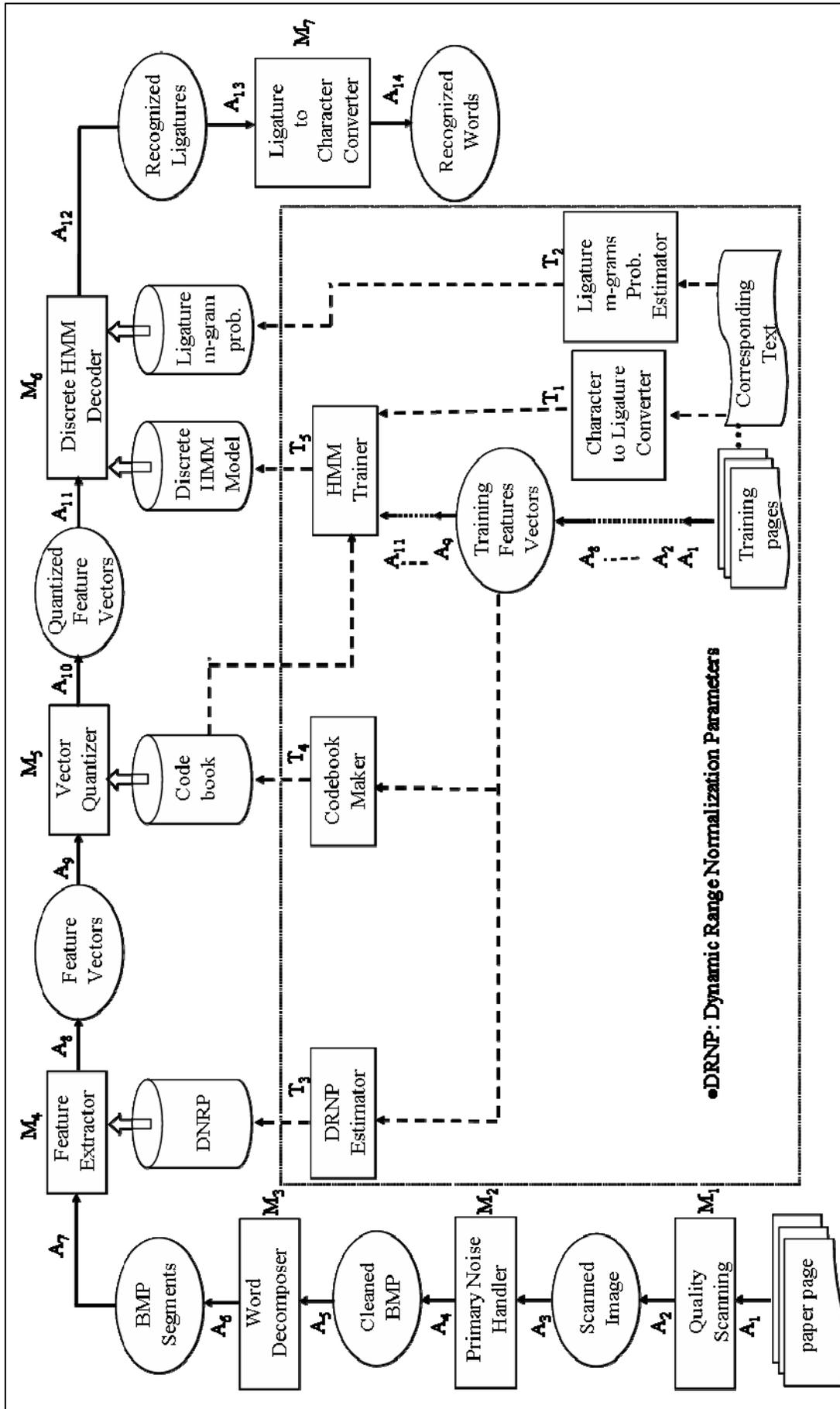


Figure (8.2): Discrete HMM-based OCR system block diagram.

Our proposed HMM-based Arabic OCR system [Rashwan 2008] illustrated in Figure (8.2) is composed of two phases: the initialization & training phase whose paths are drawn in dotted lines, and the recognition phase whose paths are drawn in solid lines.

The modules of this system are implemented using a variety of tools; MATLAB, HTK toolkit [Young 2006], as well as programming in C++. These modules are discussed further in the following subsections.

8.1.1. Digital Scanning

The scanning resolution used is preferred to be 600 dpi as the laser printing quality nowadays is at least 600 dpi. Also, this resolution is demanded for the slim characters of small sizes of compact fonts to have enough features to cope with the number of states of the HMM used.

Scanning at a higher resolution may be desirable but not practical as it would need too high storage space (one A4 page scanned at B&W 1200 dpi would need more than 16M Bytes) and would moreover need more processing time during feature extraction and recognition phases. As demanded by our feature extractor (see chapter 4), input text images to our system should be bi-level (i.e. Black & White) ones which is performed via thresholding by the scanner hardware to save storage and time.

8.1.2. Primary Noise Handler

The target text rectangle bitmap is passed through a median filter in order to clean the salt & pepper noise typically occurring in noisy images. In our study, a 5×5 median filter has proved effective for cleaning input scanned text images [Attia 2007].

Median filters are, however, hard deciding and cause considerable erosive distortion to the graphemes of written text which is generally harmful to the recognition performance. So, they should be used only while the lines & words decomposition, whereas other softer deciding filters – like Gaussian ones – might be used - if necessary - while the recognition process.

8.1.3. Words & Lines Decomposer

Font-written OCR systems need to perform a vital processing on the input scanned text rectangle prior to the recognition phase for decomposing the text rectangle into lines hence into words as illustrated in Figures (8.3) and (8.4).

Our enhanced histogram-based algorithm presented in chapter 3 is used in this kind of decomposition as it performs robustly on documents containing the idiosyncrasies of real-life documents especially as per noise and structural textual complexities.



Figure (8.3): Sample text rectangle bitmap.



Figure (8.4): Sample text rectangle bitmap after lines & words decomposition.

8.1.4. Features Extraction Module

Features extraction is one of the most important factors in achieving high recognition performance in pattern recognition systems. Our features-extractor computes a features vector as discussed in chapter 5 for each frame while sliding through each Arabic word from right to left to produce a series of feature vectors that are injected to the vector quantizer. The maximum number of vertical dark segments per frame (N_d) is set to 4. Hence, the dimensionality of our features vector is 16.

8.1.5. Vector Clustering and Quantization Modules

Taking input feature vectors, the vector-quantizer serially provides quantized observations to the discrete HMM decoder upon the recognition phase using a codebook generated by the codebook maker during the training phase.

Using numerous sample text images of the training set of documents evenly representing the various Arabic fonts and sizes of interest, the codebook maker creates a codebook using the conventional LBG vector clustering algorithm chosen for its simplicity and relative insensitivity to the randomly chosen initial centroids compared with the basic K-means algorithm. The codebook size is a key factor that affects the recognition accuracy and is specified empirically to minimize the WER of the system.

8.1.6. Dynamic Range Normalization Parameters (DNRP) Estimator

This module is used to detect the effective dynamic range of all the feature vectors' components, to calculate the normalization parameters for each component using the population of the feature vectors in the training data after pruning the extreme values. Normalization of the feature vectors injected to the vector quantizer balances the weights of the different dimensions of the features vector while calculating the distances between points in the features space during the recognition phase.

As the 1st component population of the sub-feature vectors tends to follow a natural (Gaussian) distribution, so our estimation of the effective dynamic range is chosen to be within $\mu \pm 3\sigma$ of the distribution. Estimation of the effective dynamic range of the 2nd component is chosen to be within 3σ of the distribution. The dynamic range of the 3rd and the 4th component is within -4 and 3 as illustrated before.

8.1.7. Characters-to-Ligature Converter

This module takes the text files corresponding to the training data to produce the sequences of grapheme tokens of all the ligatures included in each word using the predefined ligatures-set of each used font.

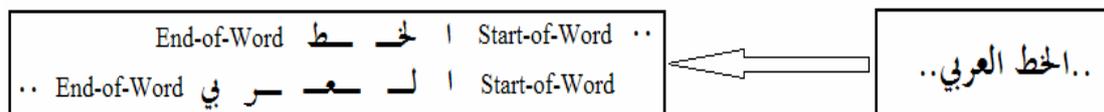


Figure (8.5): Characters-to-Ligatures conversion example

This module is built using a deterministic finite state machine. An example of character-to-ligature conversion is shown in Figure (8.5). The sequence of ligature tokens along with the corresponding sequence of quantized feature vectors (i.e. observations) of each word are then forwarded to the discrete HMM trainer.

8.1.8. Discrete HMM Trainer

Our discrete HMM trainer receives the observations sequence, which is the sequence of quantized feature vectors, of each word along with its sequence of ligature tokens generated by the character-to ligature converter. Training the HMM's are done over two steps: initialization and re-estimation embedded training.

8.1.8.1. HMM Initialization

Initialization of HMM's is done using isolated ligatures bitmaps where HMM parameters are initialized through two steps; the first one uses Viterbi training as a hard deciding algorithm , while the second uses Baum-Welch re-estimation as a soft deciding algorithm.

I. Viterbi training step.

The basic principle of Viterbi training depends on the concept of HMM as a generator of ligature vectors. Every training example can be viewed as the output of the HMM whose parameters are to be estimated. Thus, if the state that generated each vector in the training data was known, then the unknown observation symbol probability could be estimated by averaging all the vectors associated with each state. Similarly, the transition matrix could be estimated by simply counting the number of time slots that each state was occupied. The above idea can be implemented by an iterative scheme as shown in Figure (8.6).

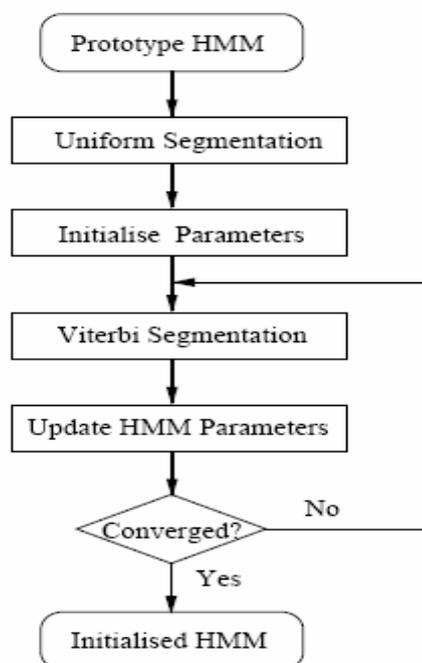


Figure (8.6): Viterbi training flowchart

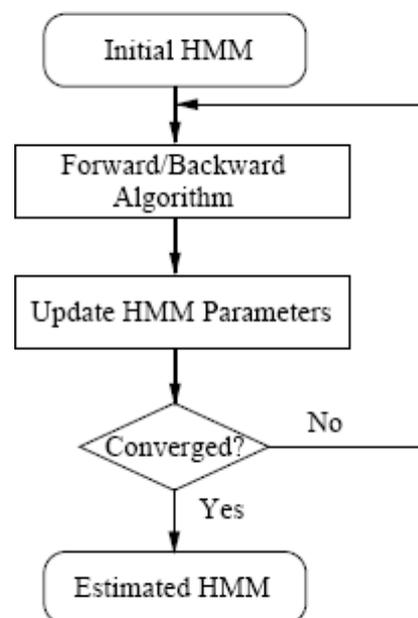


Figure (8.7): Baum-Welch training flowchart

II. Baum-Welch Re-estimation step

Its operation is very similar to the viterbi training except that, as shown in Figure (8.7), it expects the input HMM models to have been initialized and it uses

Baum-Welch re-estimation in place of Viterbi training. This involves finding the probability of being in each state at each time frame using the Forward-Backward algorithm. This probability is then used to form weighted averages for the HMM parameters. Thus, whereas Viterbi training makes a hard decision as to which state each training vector was “generated” by, Baum-Welch takes a soft decision.

8.1.8.2. HMM Re-Estimation Embedded Training (HERest)

Re-estimation embedded training which is the main HMM training step, uses the Baum-Welch re-estimation procedure to train the ligature models using the associated ligature tokens generated by the character-to-ligature converter to iteratively maximize the likelihood probabilities distributions $P(Q|I)$.

In outline, HERest works as follows. On startup, HERest loads in a complete set of HMM definitions. Every training file must have an associated label file which gives a transcription for that file. HERest processes each training file in turn. After loading it into memory, it uses the associated transcription to construct a composite HMM which spans the whole word. This composite HMM is made by concatenating instances of the ligature HMM’s corresponding to each label in the transcription. The Forward-Backward algorithm is then applied and the sums needed to form the weighted averages accumulated in the normal way. When all of the training files have been processed, the new parameter estimates are formed from the weighted sums and the updated HMM set is output.

8.1.9. Discrete HMM Recognizer

The HMM recognizer stage receives a sequence of quantized feature vectors (observations) representing the scanned word and uses the Viterbi algorithm as explained in chapter 6 to produce the most probable sequence of ligature models that produces these observations. Using a statistical ligature bi-grams Probabilities generated by Ligature n-grams probability estimator, the most probable model with the highest score is produced by the Viterbi algorithm.

Our Arabic font-written OCR relies on HMM’s with 1st order left-to-right topology. As one of the key factors that affect the system performance, the number of states per model is empirically fine tuned to optimize that performance.

8.1.10. Ligature n-Grams Probability Estimator

For recognition tasks; the SLM gives the a priori probability of a given hypothesis of ligature sequences. In our system, the bigram approximation of the ligatures SLM, is selected to match the 1st order Left-to-Right HMM topology chosen for our system.

The language models probabilities are estimated using Bayes's_Good-Turing_Back-off methodology. The Arabic text corpus of the written language resource of the NEMLAR project is used for building the SLM for our Arabic OCR.

8.1.11. Ligature –to- Character Converter

This module converts the sequence of recognized ligatures back into their equivalent plain-text character codes (Unicode).

8.2. Arabic Font-Written Database

The Database prepared to assess the performance of the presented HMM-based font-written OCR, has the following characteristics:

- It is chosen arbitrarily from real-life documents which are full of punctuations and special symbols embedded within the text.
- The database has been digitally acquired via a common but standard hardware of HP3800 flatbed scanner with TMA, and HP Laser jet printer.
- All the database pages are scanned at 600 B&W dpi, and losslessly stored as bitmaps.
- It includes 12 visually distinct and widely used Arabic fonts under both the MS-Windows and Macintosh OS. environments. The chosen fonts have different visual features as detailed by Table (8.1).
- For each font, 6 different sizes within the typical document-editing range (10 to 22) are represented by 30 pages per size is included.
- For each size of each font, an extra page of isolated ligatures used for the HMM initialization procedure is included.
- A line boundaries aligned text file (called the text label file) corresponding to each of the training pages bitmaps is also supplied for the HMM re-estimation embedded training procedure.

| Font Name & Type | Visual features | Used for |
|-------------------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------|
| Simplified Arabic (MS font) | Limited no. of ligatures (151), gross graphemes, thin contour, tends to be sharp cornered, clear openings, separate dots, no touching tails, .. | Training and Assimilation testing |
| Mudir (MS font) | Limited no. of ligatures (151), gross graphemes, thick contour, round cornered, clear openings, partially connected dots, no touching tails, .. | Training and Assimilation testing |
| Koufi (MS font) | Limited no. of ligatures (151), gross graphemes, very thick contours, very sharp cornered, small openings, separate dots, no touching tails, .. | Training and Assimilation testing |
| Traditional Arabic (MS font) | Broadest ligatures set (220), minute graphemes, thin contours, round cornered, almost close openings, partially connected dots, no touching tails, .. | Training and Assimilation testing |
| Akhbar MT (MS font) | Limited no. of ligatures (151), minute graphemes, med. contour thickness, tends to be sharp cornered, almost close openings, connected dots, no touching tails, .. | Training and Assimilation testing |
| Tahoma (MS font) | Limited no. of ligatures (151) gross graphemes, thin contours, round cornered, clear openings, separate dots ,touching tails, ..., Some ligatures has odd shapes (e.g. middle <i>Haa</i>) | Training and Assimilation testing |
| Courier new (MS font) | Very broad ligatures set (209), minute graphemes (but very wide), thin contours, sharp cornered, clear openings, connected dots, no touching tails, .. | Training and Assimilation testing |
| Baghdad (Mac font) | Rich ligatures set (167), minute graphemes, thick contours, round cornered, almost close openings, connected dots, touching tails, .. | Training and Assimilation testing |
| Demashq (Mac font) | Limited no. of ligatures (154), minute graphemes, med. contour thickness, tends to be sharp cornered, almost close openings, connected dots, touching tails,.. | Training and Assimilation testing |
| Nadeem (Mac font) | Limited no. of ligatures (154), minute graphemes, med. contour thickness, tends to be sharp cornered, almost close openings, partially connected dots, touching tails, .. | Generalization testing |
| Naskh (Mac font) | Rich ligatures set (167), minute graphemes, thin contours, round cornered, close openings, connected dots, no touching tails, .. | Generalization testing |
| Gizza (Mac font) | Limited no. of ligatures (154), minute graphemes, thin contours, sharp cornered, semi clear openings, partially connected dots, no touching tails, .. | Generalization testing |

Table (8.1): Visual features description of the fonts used in training and testing

8.3. Evaluation Experiments Setup

8.3.1 Training phase Setup

The training data used to experimentally evaluate the implemented OCR system covers 9 visually distinct and widely used Arabic fonts under both the MS-Windows and Mac. OS environments shown in Figures (8.9) and (8.10).

- a. القلقلة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،
- b. القلقلة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،
- c. **القلقلة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،**
- d. القلقلة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،
- e. القلقلة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،
- f. القلقلة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،
- g. القلقلة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،

Figure (8.9): Samples of MS-Windows fonts used in training, a) Simplified Arabic, b) Mudir, c) Koufi, d) Traditional Arabic, e) Akhbar, f) Tahoma, g) courier-new.

a.

القلقلة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط
التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء
الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،

b.

القلقلة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط
التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء
الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،

Figure (8.10): Samples of Mac. fonts used in training, a) Baghdad, b) Demashq

For each training font, 6 different sizes within the typical document-editing range (10 to 22) are represented by 25 pages per size. Each page contains about 200 words selected spontaneously from Arabic websites. Assuming an average of 4 ligatures per Arabic word [Attia 2004], the size of the training data $\approx 9 \times 6 \times 25 \times 200 \times 4 = 1,080,000$ graphemes.

8.3.2. Testing phase Setup

The *Omni* quality of an OCR system is measured by its capabilities at assimilation and generalization tests. The former test measures the performance of the system at recognizing pages (whose text contents are not included in the training data) printed in fonts represented in the training data. While the later test measures the performance of the system at recognizing pages printed in fonts not represented in the training data. The testing data is composed of two parts; assimilation testing data, and generalization testing data:

Assimilation testing data represents all the fonts and sizes used in training using 5 pages (other than the training 25 pages) per each size per font. The size of the assimilation testing data $\approx 9 \times 6 \times 5 \times 200 \times 4 = 216,000$ graphemes.

Generalization testing data covers 3 Mac. OS fonts shown in Figure (8.11) other than the 9 fonts used for training. For each of these 3 fonts, 6 different sizes within the typical document-editing range are represented by 5 pages per size. The size of the generalization testing data $\approx 3 \times 6 \times 5 \times 200 \times 4 = 72,000$ graphemes.

- a. القلقة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،
- b. القلقة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،
- c. القلقة ظاهرة صوتية دار حولها كثير من الجدل والأخذ والرد بين أوساط التجويديين وعلماء اللغة، إلا أنها لم تحظ بهذا القدر من الاهتمام عند علماء الأصوات، وذلك على الرغم من كثرة المسائل الصوتية التي تنطوي عليها،

Figure (8.11): Samples of fonts used in the generalization tests, a) Nadeem, b) Naskh, c) Gizza.

WER is the error rate perceived by the users of the OCR, and is hence the useful error rate used in the evaluation process. On the other hand, CER is the character (or more accurately; the grapheme) error rate which is the one typically reported in the scientific literature as well as the commercials. The relation between the WER and CER, assuming statistically independent multiple errors within the same word and error probability for all ligatures are equal, can be derived as:

$$\begin{aligned}
 a_w &= (a_c)^\ell \\
 1 - WER &= (1 - CER)^\ell \\
 1 - WER &= 1 - \ell \cdot CER + \frac{\ell^2}{2} CER^2 - \dots \dots \dots \quad \text{Eq. (8.1)} \\
 1 - WER &\approx 1 - \ell \cdot CER \quad \text{iff } CER \ll 1 \\
 WER &\approx \ell \cdot CER
 \end{aligned}$$

Where a_w and a_c are the word and character recognition accuracy respectively. ℓ is the average number of ligatures per word which is ≈ 4 for Arabic.

The WER is measured following the speech recognition conventions; i.e. the number of substitutions, deletion, and insertions are summed up and divided by the total number of words in the input textual transcription files.

8.3.3. System Parameters Setup

There is no mathematical method to calculate optimal number of states per model. Alternatively, various values were examined to select the best number of states per model. The first set of experiments done to discover the optimal number of states per model was performed on a single font. Traditional Arabic was selected because it is the most compact and ligatures rich font. Those parameters were then generalized to other fonts in the corpus.

These initial experiments shows that the best no of states regarding the least WER in case of a uni-font system is 14 states for all the ligatures, except for few extremely wide ones where 18 states are used, and other few extremely slim ones where 7 states only are used .

Another two set of experiments are then done, using the assimilation data, to select the two main omni-system parameters; number of states and the codebook size.

The first set of experiments studied the codebook size parameter. Four different codebook size values were examined using the best number of states detected from the first set of experiments. Figure (8.12) shows the improvement in the system performance as the codebook size increases. It shows that the least WER_A is achieved at codebook size =2048 and this was we previously predict using our visual aid discussed in chapter 5.

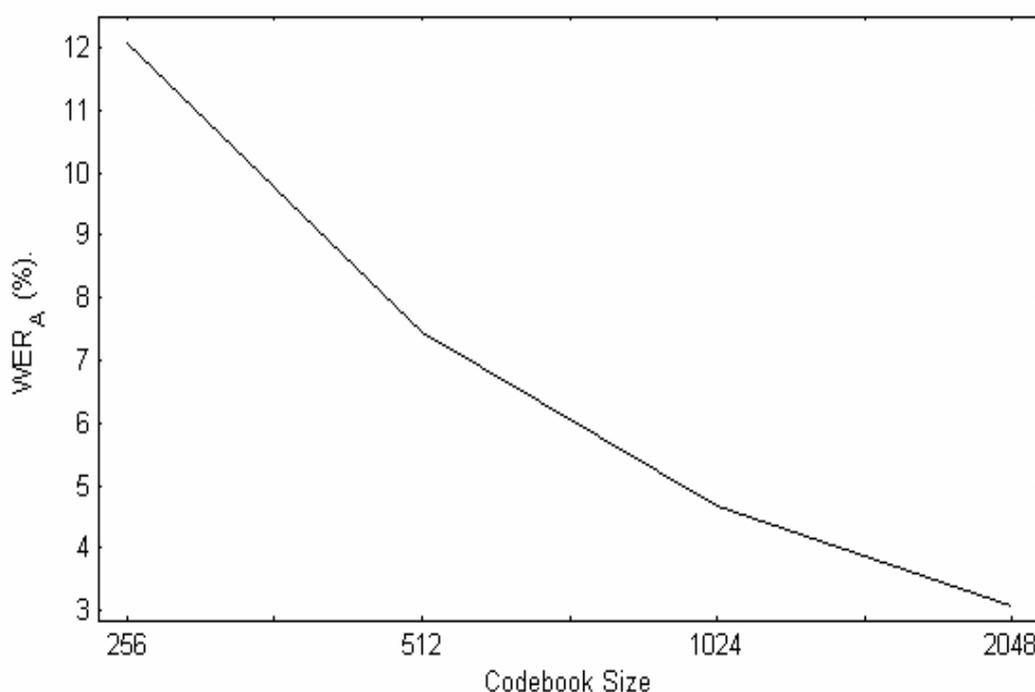


Figure (8.12): System performance using different codebook size

The second set of experiments studied the performance of the system, with codebook size=2048 and using 7 states for slim graphemes, by varying the number of HMM's states for the rest of graphemes. Figure (8.13) shows the improvement in the system performance as the number of states increases. It shows that the WER decreases rapidly until the number of states reaches 14. After that a slight increase is achieved. The improving in the system performance achieved when the number of states =16 is nearly the same when using hybrid number of states for the graphemes as that of the uni-font experiment (18 states for some ligatures)

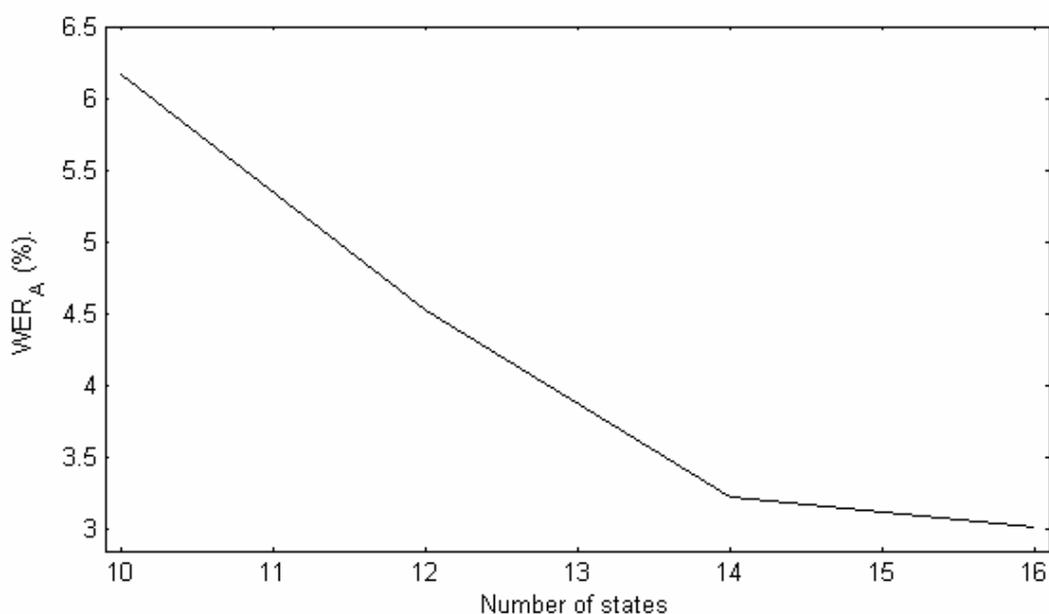


Figure (8.13): Number of states per model versus system performance

8.3.4. System Evaluation

| | |
|-----------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------|
| HMM type | Discrete , 1 st order, L-to-R |
| Codebook size | 2048 |
| HMM states per model | 7 states for very slim graphemes. 18 states for very wide graphemes. 14 states for normal graphemes. |
| HMM algorithms | Initialization: <i>Viterbi</i> , and <i>Baum Welsh</i> . Embedded training: <i>Baum Welsh</i> . Recognition: <i>Viterbi</i> . |
| Training data size | 1,080,000 graphemes |
| Assimilation testing data size | 216,000 graphemes |
| Generalization testing data size | 72,000 graphemes |
| Language Models | Bigram SLM |

Table (8.2): HMM parameters setting

From the previous sets of experiments, the best parameters setting for our HMM-based OCR during both the training and recognition phases are summed up in Table (8.2).

Under the mentioned settings, assimilation and generalization testing have been conducted to obtain the overall recognition error rates. To study the effect of the statistical language models, both experiments are run twice; with the language model enabled on the first the run, and with the language model neutralized on the second run. The obtained error rates of the 4 cases are summed up in Table (8.3), where the assimilation error rates are designated by the subscript A , and the generalization error rates are designated by the subscript G .

| | | |
|-----------------|---------|--------|
| SLM enabled | CER_A | 0.77% |
| | WER_A | 3.08% |
| | CER_G | 2.58% |
| | WER_G | 10.32% |
| SLM neutralized | CER_A | 1.53% |
| | WER_A | 6.13% |
| | CER_G | 3.65% |
| | WER_G | 14.60% |

Table (8.3): Experimental results.

It is valuable to mention that, upon our first trial to run a generalization test, the recognition models are built from the 7 MS-Windows fonts and the testing data was composed of 3 Macintosh Os. fonts. Under these conditions we got the poor results of $WER_G \approx 35\% \approx 11 \cdot WER_A$ ($WER_G \gg WER_A$)

After error analysis and some contemplation, we realized that Macintosh Os. fonts are built with different concepts not covered by the 7 MS-Windows fonts; e.g. connected dots, overlapping of the tails of some graphemes, ..., etc.

After adding 2 Macintosh Os. fonts to introduce those concepts in the training data, we have achieved the dramatic enhancement of $WER_G = 10.32\% \approx 3.4 \cdot WER_A$. This indicates that our OCR system can statistically build font shape concepts.

8.4. Experimental Error Analysis

Error Analysis of both Assimilation and Generalization tests regarding font shape and size are done to show the performance of the system over different sizes and shapes of Arabic fonts. Tables (8.4) & (8.5) show that the system performance is almost good over different font sizes and shapes.

| Shape \ Size | Small | Medium | Large | $\sum WER_A$ % Over sizes per shape |
|-------------------------------------------|-------------|-------------|-------------|-------------------------------------------|
| Simplified Arabic | 0.07 | 0.05 | 0.10 | 0.21 |
| Mudir | 0.05 | 0.05 | 0.03 | 0.13 |
| Koufi | 0.13 | 0.09 | 0.14 | 0.37 |
| Traditional Arabic | 0.19 | 0.11 | 0.15 | 0.45 |
| Akhbar MT | 0.07 | 0.05 | 0.06 | 0.17 |
| Tahoma | 0.16 | 0.11 | 0.18 | 0.45 |
| Courier new | 0.24 | 0.32 | 0.47 | 1.03 |
| Baghdad | 0.13 | 0.05 | 0.02 | 0.20 |
| Demashq | 0.06 | 0.01 | 0.02 | 0.09 |
| $\sum WER_A$ % Over shapes per size | 1.09 | 0.83 | 1.16 | 3.08 |

Table (8.4): Error analysis of Assimilation test regarding font shape/size

| Shape \ Size | Small | Medium | Large | $\sum WER_A$ % Over sizes per shape |
|-------------------------------------------|-------------|-------------|-------------|-------------------------------------------|
| Nadeem | 0.97 | 0.07 | 0.18 | 1.22 |
| Naskh | 2.04 | 1.71 | 1.09 | 4.84 |
| Gizza | 2.53 | 0.93 | 0.80 | 4.26 |
| $\sum WER_A$ % Over shapes per size | 5.54 | 2.71 | 2.07 | 10.32 |

Table (8.5): Error analysis of Generalization test regarding font shape/size

Error analysis of the most frequent recognition mistakes in the Assimilation test is done. The most frequent 17 mistakes that contribute to about 63.15% of WER_A are listed in Table (8.6).

| Original ligature | Replaced By | Frequency % Of the total WER |
|-------------------|-------------|---------------------------------|
| ت ن ن | ن ن ن | 23.02 |
| Decomposition | × | 6.33 |
| ص | ض | 5.03 |
| ب ب | ب ب | 4.74 |
| ع | ل ع | 3.96 |
| ح | ل ح | 3.02 |
| ذ | ل ذ | 2.90 |
| . | ا | 2.60 |
| Insertion | ا | 2.37 |
| و | ر | 2.25 |
| ل ل | ل ل | 1.72 |
| د | ل د | 1.66 |
| ن | ز | 0.95 |
| 5 | ه | 0.77 |
| ف ق | ق ف | 0.71 |
| ل | ا | 0.41 |

Table (8.6): Assimilation test most frequent mistakes

| Original ligature | Replaced By | Frequency % Of the total WER |
|-------------------|-------------|---------------------------------|
| ب ب | ب ب | 15.56 |
| ت ن ن | ن ن ن | 15.06 |
| ل ل | ل ل | 3.81 |
| ع | ء | 3.10 |
| ل | ا | 3.06 |
| و | × | 2.00 |
| ف | × | 2.00 |
| ف | × | 1.56 |
| ف ق | ق ف | 1.44 |
| ت ن ن | × | 1.25 |
| ح | × | 1.12 |
| Decomp .Errors | × | 1.12 |
| ش | × | 1.06 |
| ص | ض | 0.94 |
| ث ث | × | 0.81 |
| ة | ت | 0.81 |
| ذ | ل ذ | 0.50 |
| و | ر | 0.40 |
| ع | ل ع | 0.30 |

Table (8.7): Generalization test most frequent mistakes

Also, the error analysis of Generalization test regarding the most frequent recognition mistakes is done. The most frequent 19 mistakes that contribute to about 55.90% of WER_G are listed in table (8.7), where x means different characters.

The frequency analysis of both the assimilation and generalization errors shown in Table (8.6) and (8.7) shows that over 40% of the errors are due to only the most 5 frequent ones, which maximizes the chances of removing them via simple post-OCR text correction methods.

The error rates obtained after the generalization test with the statistical language model enabled are the most indicative as these conditions are closest to the real life ones. Given that both the scientific and commercial literature in this area typically report about $CER_A < CER_G$, the experimental results of Table (8.3) above puts our HMM-based OCR with the new features vector in the lead among the other open-vocabulary omni font-written OCR's for cursive scripts, at least for Arabic. [Al-Badr 1995], [Bazzi 1999], [Gouda 2004], [Khorshed 2007].

CHAPTER 9

Conclusion and Future Work

Cursiveness represents the main challenge when recognizing Arabic words. Previous research has proved the difficulties in segmenting the Arabic word into its ligatures. The implemented algorithms throughout this dissertation for lines and words decomposition, feature extraction and classification determine the operational characteristics of the recognition system.

Throughout this dissertation, a more robust algorithm for lines/words decomposition especially in Arabic, or Arabic dominated, text rectangles from real-life multifont/multisize documents has been developed. This algorithm has proved robust on realistic Arabic, or Arabic dominated, documents after the extensive experimentation on real-life documents.

After reviewing the inferior status of font-written OCR systems for the challenging cursively scripted languages compared with their Latin scripted peers, the HMM-based methodology that proved to be most promising at approaching this problem has been dissected in this dissertation to reveal its weakest point of lacking a sound features vector design like those defined for HMM-based ASR systems.

A new feature vectors design for HMM-based omni font-written Arabic OCR system based on autonomously normalized horizontal differentials has been rigorously derived and introduced for cursively scripted languages in order to robustly realize WER's as low as those achieved by OCR systems for Latin scripted languages. These features well represent the agglomerative and topological characteristics of the ligatures.

The design and implementation of a real life omni-font, open-vocabulary, HMM-based Arabic recognition system using both our robust algorithm for lines/words decomposition and our new designed feature vectors is fully described in this dissertation.

Based on the continuous/discrete hybrid nature of the components of our designed feature vectors, Discrete HMM is used and hence vector quantization is needed. The LBG well known clustering algorithm is used to build the codebook used for VQ. A developed visualization aid for estimating the training data set size and the codebook size required for the clustering algorithm to achieve optimum codebook construction and hence least WER achieved by the recognition system.

The bi-gram ligature-level statistical language model deployed in the HMM decoder of our recognizer is built via the combined methodology of Bayes', Good-Turing discount, and Back-off probability estimators. Incorporating the SLM in the HMM decoder greatly enhance the performance of the recognition system.

Extensive assimilation and generalization testing experiments have been carried out over the Arabic script which is an extreme case of cursive scripts to evaluate our system. Along with the detailed description of those experiments, the dissertation has finally presented the obtained results

The data set used for training and recognition purposes by the implemented HMM-based recognition system has two main features: the unlimited lexicon size and the data source diversification. The unlimited lexicon size is for the sake of training an open vocabulary recognition system. The data source diversification is essential to test the proposed features on a wide spectrum of Windows and Macintosh Os. fonts.

The experimental results puts our HMM-based OCR system with the proposed features vector in the lead among the other open-vocabulary, omni font-written OCR systems for cursive scripts; at least for Arabic.

Further work in the areas presented in our dissertation is more promising to enhance the performance of our Arabic OCR systems. Future work could be presented in five areas:

1. Using a clustering technique other than LBG to give more flexibility in choosing the codebook size other than order of 2.
2. Applying and testing our system for recognizing noisy text images.

3. Testing the performance of our system using other types of HMM topology.
4. Using Post-OCR Text Correction algorithm to reduce the WER of Arabic OCRed text using character segment correction, language modeling, and Shallow Morphology.
5. Using Fusion between independent Arabic OCR systems to get a total WER of the fused OCR systems lower than the smallest WER in the set of those independent Arabic OCR systems.
6. Training and evaluating our system not only on other major cursively scripted languages especially Indian, Urdu, and Persian, but also on Latin scripted ones to have multilingual system.

References

- [1] Abdelazim, H. Y. "Arabic OCR based on Entropy Measures" Proc. Of ICAV3D, International Conference on Augmented Virtual Environments and 3D Imaging, Mykonos, Greece, June 2001 .
- [2] Abuhaiba, I, Ahmed, P., "Restaion of temporal information in off-line Arabic handwriting", Pattern Recognition, Vol. 26, No. 7, pp. 1009-1017, July 1993
- [3] Abuhaiba, I., "A Discrete Arabic Script for Better Automatic Document Understanding", The Arabian Journal for Science and Engineering, Vol. 28, No. 1B, pp. 77-94, Apr 2003.
- [4] Al-Badr, B., Mahmoud, S.A., "Survey and Bibliography of Arabic Optical Text Recognition", Elsevier Science, Signal Processing, Vol. 41, pp. 49-77, 1995.
- [5] Al-Badr, B. and Haralick, R. "A Segmentation-Free Approach to Text Recognition with Application to Arabic Text", Int'l J. Document Analysis and Recognition, Vol. 1, pp. 147-166, 1998.
- [6] Altuwaijri M, Bayoumi M. "Arabic text recognition using neural networks", Proceedings of IEEE International symposium on Circuits and Systems, London, UK, pp. 415-418, July 1993
- [7] Altuwaijri M, Bayoumi M. "Thinning algorithm for Arabic characters using art2 neural network", IEEE Trans Circuits and Systems, Vol. 45, No. 2, pp. 260-264, 1998
- [8] Al-Yousefi, H., and Upda, S., "Recognition of Arabic Characters", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 14, No. 8, pp. 853-857, August 1992.
- [9] Amin, A., and Mari, J.F., "Machine recognition and correction of printed Arabic text", IEEE Trans. Systems, Man, and Cybernetics, Vol. 9, pp. 1300-1306, 1989.
- [10] Amin, A., "Off-Line Character Recognition; A Survey", Proceedings Of the 4th Int. Conf. Document Analysis and Recognition (ICDAR '97), Ulm, Germany, pp. 596-599, August 1997.

- [11] Amin, A., "Offline Arabic Character Recognition: The State of the Art", Pattern Recognition, Vol. 31, pp. 517-530, 1998.
- [12] Amin, A., Fischer, S. "A Document Skew Detection Method Using the Hough Transform", Pattern Analysis and applications, Springer-verlag, Vol. 3, No. 3, pp 243–253, 2000.
- [13] Amor, N., Amara, N. E., "Multifont Arabic character Recognition using Hough Transform and HMM/ANN Classification", Journal of Multimedia, Vol. 1, No. 2., May 2006
- [14] Attia, M., Rashwan, M., Khallaaf, G., "On Stochastic Models, Statistical Disambiguation, and Applications on Arabic NLP Problems", The Proceedings of the 3rd Conference on Language Engineering; CLE'2002, the Egyptian Society of Language Engineering (ESLE). This paper is freely downloadable at <http://www.RDI-eg.com/RDI/Technologies/paper.htm>.
- [15] Attia, M., "Arabic Orthography vs. Arabic OCR; Rich Heritage Challenging A Much Needed Technology", Multilingual Computing & Technology magazine, USA, Dec. 2004.
- [16] Attia, M., Theory and Implementation Of A Large-Scale Arabic Phonetic Transcriber, and Applications, PhD thesis, Dept. of Electronics and Electrical Communications, Faculty of Engineering, Cairo University, 2005.
- [17] Attia, M., El-Mahallawy, M. "Histogram-Based Lines & Words Decomposition for Arabic Omni Font-Written OCR Systems; Enhancements and Evaluation", Lecture Notes on Computer Science (LNCS): Computer Analysis of Images and Patterns, Springer-Verlag Berlin Heidelberg, Vol. 4673, pp. 522-530, 2007.
- [18] Bahl, L. R., Jelinek, F., and Mercer, R. L., "A maximum likelihood approach to continuous speech recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. PAMI-5, No.2, pp. 179-190, 1983.
- [19] Bazzi, I., Schwartz, R., Makhoul, J., "An Omnifont Open-Vocabulary OCR System for English and Arabic", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21, No. 6, June 1999.
- [20] Blue, J. L., Candela, G. T., Grother, P. J., Chellappa, R., and Wilson, C. L., "Evaluation of Pattern Classifiers for Fingerprint and OCR Applications", Pattern Recognition, Vol. 27, No. 4, pp. 485–501, 1994.
- [21] Bozinovic, R. M., and Srihari, S. N., "Off-line cursive script word recognition", IEEE Transactions of Pattern Analysis and Machine Intelligence, Vol. 11, pp. 68–83, January 1989.
- [22] Bunke, H., Roth, M., and Schukat-Talamazzini, E. G., "Off-line cursive handwriting recognition using hidden Markov models", Pattern Recognition, Vol. 28, No. 9, pp. 1399-1413, 1995.

- [23] Bunke. H., Wang, P. S. P., “Image Processing methods for document image analysis”, Handbook of Character Recognition and Document Image Analysis edited by H. Bunke and P. S. Wang, 1997.
- [24] Cho, W.Y., Lee, S.W., Kim, J.H., “Modeling and Recognition of Cursive Words with Hidden Markov-Models”, Pattern Recognition, Vol. 28, No. 12, pp. 1941-1953, December 1995.
- [25] Callan, J., Kantor, P., Grossman, D., “Information Retrieval and OCR: From Converting Content to Grasping Meaning”, SIGIR conference, 2003.
- [26] Chen, M. Y., Kundu, A. and Zhou, J., “Off-Line Handwritten Word Recognition Using a Hidden Markov Model Type Stochastic Network”, Ieee Transactions On Pattern Analysis And Machine Intelligence, Vol. 16, No. 5, pp. 481-496, May 1994
- [27] Chen, S. F., Goodman, J.T, An Empirical Study of Smoothing Techniques for Language Modeling. Technical Report TR-10-98, Computer Science Group, Harvard University, 1998. <http://www.cs.cmu.edu/~sfc/html/publications.html>
- [28] Cheung, A., Bennamoun, M., and Bergmann, N. W. “An Arabic optical character recognition system using recognition-based segmentation”, Pattern Recognition, Vol. 34 , No. 2, pp. 215-233, 2001
- [29] Cote, M., “Reading of cursive scripts using a reading model and perceptual concepts, the PERCEPTO system”, Int. Journal of Document Analysis and Recognition, Vol. 1, No. 1, pp. 3–17, 1998.
- [30] Cowell, J., and Hussain F., “A fast recognition system for isolated Arabic character recognition”, IEEE Information Visualization IV2002 conference, London, July 2002.
- [40] Cowell J and Hussain, F., “Thinning Arabic Characters for Feature Extraction”, Proceedings IEEE Conference on Image Visualisation 2001, London UK. July 2001.
- [41] Davis, S.B., Mermelstein, P., “Comparison of Parametric Representations for Monosyllabic Word Recognition in Continuously Spoken Sentences”, IEEE Transactions on Acoustics, Speech, and Signal Processing, Vol. 28, pp. 357-366, 1980.
- [42] Dehghan, M., Faez, K., Ahmadi, M., Shridhar, M., “Handwritten Farsi (Arabic) word recognition: a holistic approach using discrete HMM”, Pattern Recognition, Vol. 34, No. 5, pp. 1057-1065, 2001.
- [43] Devijver, P. A., and Kittler, J., “Pattern Recognition: A Statistical Approach”, Englewood Cliffs, NJ: Prentice-Hall, 1982.
- [44] Duda, R.O., Hart, P. E. & Stork, D. G., Pattern Classification, 2nd edition, John Wiley & Sons Inc., 2001.

- [45] El-Adawy, M., Keshk, H., Hamdy, A., and Dawoud, S., "Improvement of the Preprocessing Stage for Arabic OCR System", The Seventh Conference on Language Engineering, Cairo, 2007.
- [46] El-Yacoubi, A., Gilloux, M., Sabourin, R., Suen C. Y., "An HMM-Based Approach for Off-Line Unconstrained Handwritten Word Modeling and Recognition", IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 21 No. 8, pp.752-760, August 1999.
- [47] Feng, S., & Manmatha, R., "A Hierarchical, HMM-based Automatic Evaluation of OCR Accuracy for a Digital Library of Books", JCDL'06, June 11–15, 2006.
- [48] Gillies, A.M., Erlandson, E.J., Trenkle, J.M, Schlosser, S.G., "Arabic Text Recognition System", Proceedings of the Symposium on Document Image Understanding Technology, Annapolis, Maryland, 1999.
- [49] Gilloux, M., Leroux, M., Bertille, J. M., "Strategies for handwritten word recognition using hidden markov models", Int. Conf. on Document Analysis and Recognition, pp. 299–304, 1993.
- [50] Gonzalez, R., Woods, R., Digital Image Processing, 2nd ed., Prentice Hall, 2002.
- [51] Gouda, A., "Arabic Handwritten Connected Character Recognition", PhD thesis, Dept. of Electronics & Electrical Communications, Faculty of Engineering, Cairo University, Nov. 2004.
- [52] Govindan, V. K., Shivaprasad, A. P., "Character Recognition, A review", Pattern Recognition, Vol. 23, No. 7, pp. 671–683, 1990.
- [53] Guillevic, D., and Suen, C. Y., "Cursive script recognition: A sentence level recognition scheme", in Proc. Int. Workshop Frontiers in Handwriting Recognition, pp. 216–223, 1994.
- [54] Gray, R.M., "Vector Quantization", IEEE Signal Processing Magazine, pp. 4-29, Apr. 1984.
- [55] Gray, R.M., Neuhoff, D. L., "Quantization", IEEE Transactions on Information Theory, Vol. 44, No. 6, pp. 2325-2383, October 1998
- [56] Hamami, L., and Berkani, D., "Recognition System for Printed Multifont and Multisize Arabic Characters", The Arabian J. Science and Eng., Vol. 27, pp. 57-72, 2002.
- [57] Hamamoto, Y., "Recognition of hand-printed Chinese characters using Gabor features", in Proc. 13th International Conference on Pattern Recognition (ICPR'96), Vol. 3, , 1996
- [58] Jain, A. K. and Chandrasekaran, B., Dimensionality and sample size considerations in pattern recognition practice. In P. R. Krishnaiah and L. N. Kanal (Eds.), Handbook of statistics, Vol. 2, pp. 835–855, 1982.

- [59] Jain, A. K., and Zongker, D., “Representation and recognition of handwritten digits using deformable templates”, *IEEE Trans. Pattern Anal. Machine Intell.*, Vol. 19, pp. 1386–1391, December 1997.
- [60] Jain, A. K., Murthy, M.N. and Flynn, P.J., “Data clustering, a review”, *ACM Computing Surveys*, Vol. No. 3, pp. 265–323, September 1999.
- [61] Jain, A. K., W., R. P. Duin, and Mao, J., “Statistical pattern recognition: A review”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 22, pp. 4–38, January 2000.
- [62] Kanungo, T., Marton, G., and Bulbul, O., “OmniPage vs. Sakhr: Paired Model Evaluation of Two Arabic OCR Products,” *Proc. SPIE Conf. Document Recognition and Retrieval (VI)*, pp. 109-121, 1999.
- [63] Kapur, J.N., Saxena, .H.C., *Mathematical Statistics*, 7th edition, S. Chand & Co. (Pvt.) LTD, 1972.
- [64] Katz, S.M., “Estimation of Probabilities from Sparse Data for the Language Model Component of a Speech Recognizer”, *IEEE Transactions on Acoustics, Speech and Signal Processing*, Vol. 35, No. 3, March 1987.
- [65] Khorsheed , M. S., and Clocksin, W. F. , “Structural Features Of Cursive Arabic Script”, *Proceedings of the Tenth British Machine Vision Conference*, Sep. 1999.
- [66] Khorsheed, M. S., and Clocksin, W. F. , “Multi-Font Arabic Word Recognition”, *Proceedings of the International Conference on Pattern Recognition (ICPR'00)*, pp.1051-4651, 2000.
- [67] Khorsheed, M. S., “Off-line Arabic character recognition – A review”, *Pattern Analysis .and applications*, Springer-Verlag, Vol. 5, No. 1, pp 31–45, 2002.
- [68] Khorsheed, M.S. “Offline Recognition of Omnifont Arabic Text Using the HMM Toolkit (HTK)”, *Pattern Recognition Letters*, Vol. 28 pp. 1563–1571, 2007.
- [69] Lam, L., Lee, S. W., and Suen, C. Y., “Thinning methodologies; A comprehensive survey”, *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 14, pp. 869–885, Sep. 1992.
- [70] Linde, Y., Buzo, A. and Gray, R. M., “An algorithm for vector quantizer design”, *IEEE Trans. Commun.*, Vol. COM-28, pp. 84–95, Jan. 1980.
- [71] Lu, Z., Bazzi, I, Makhoul, J., Natarajan, P. and Schwartz, R., “A Robust, Language-Independent OCR System”, *Proc. 27th AIPR Workshop: Advances in Computer-Assisted Recognition* , SPIE Proceedings, Vol.3584, 1999.
- [72] Madhvanath, S., Kim, G., and Govindaraju, V., “Chaincode contour processing for handwritten word recognition”, *IEEE Pattern. Anal. Machine Intell.*, Vol. 21, pp. 928–932, Sep. 1999.

- [73] Magdy, W., Darwish, K., and Rashwan, M., “Fusion of Multiple Corrupted Transmissions and Its Effect on Information Retrieval”, The Seventh Conference on Language Engineering, Cairo, 2007.
- [74] Mahmoud S, Abuhaiba I, Green R. “Skeletonization of Arabic characters using clustering based skeletonization algorithm”, Pattern Recognition, Vol. 24, No. 5, pp. 453–464, 1991
- [75] Makhoul, J., Schwartz, R., Lapre, C., and Bazzi, I., “A Script- Independent Methodology for Optical Character Recognition”, Pattern Recognition, Vol. 31, pp. 1285-1294, 1998.
- [76] Mohamed, M., Gader, P., “Handwritten word recognition using segmentation-free hidden markov modeling and segmentation-based dynamic programming techniques”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 18, No. 5, pp.548–554, 1996.
- [77] Mantas, J., “An overview of character recognition methodologies”, Pattern Recognition, Vol. 19, No. 6, pp. 425–430, 1986.
- [78] Nadas, A., “On Turing's Formula for Word Probabilities”, IEEE Transactions on Acoustics, Speech and Signal Processing, Vol. 33, No. 6, Dec. 1985.
- [79] Nazif, A., A System for the Recognition of the Printed Arabic characters, M.Sc. Thesis, Faculty of Engineering, Cairo University, 1975.
- [80] Nixon, M.S., Aguado, A. S., Feature Extraction and Image Processing, 1st edition, Newnes, 2002 .
- [81] Pratt, W.K., Digital Image Processing, 2nd edition, John Wiley & Sons Inc., 1991.
- [82] Rabiner, L.; Juang, B., “An introduction to hidden Markov models”, IEEE Signal Processing Magazine, Vol. 3, No. 1, pp. 4 – 16, Jan. 1986
- [83] Rabiner, L. R., “A tutorial on hidden Markov models and selected applications in speech recognition”, Proc. IEEE, Vol. 77, pp. 257–286, 1989.
- [84] Rashwan, M., Fakhr, W.T., Attia, M., El-Mahallawy, M., “Arabic OCR System Analogous to HMM-Based ASR Systems; Implementation and Evaluation”, Journal of Engineering and Applied Science, Cairo University, www.Journal.eng.CU.edu.eg, Dec., 2007.
- [85] Sarfraz, M., Nawaz, S. N., Khuraidly, A., “ Offline Arabic Text Recognition system”, Proceedings of the 2003 International Conference on Geometric Modeling and Graphics (GMAG’03), 2003 .
- [86] Sari, T. and Sellami, M., “MORpho-LEXical Analysis for Correcting OCR-Generated Arabic Words (MOLEX)”, Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition (IWFHR’02), 2002

- [87] Schutze, H., Manning, C. D., Foundations of Statistical Natural Language Processing, the MIT Press, 2000.
- [88] Srihari, S. N., Character Recognition, in Encyclopedia of Artificial Intelligence 2nd Edition, pp. 138—150, John Wiley, 1992.
- [89] Steinherz, T., Rivlin E., Intrator, N., “Offline cursive script word recognition - a survey”, International Journal on Document Analysis and Recognition, Vol. 2, pp. 90-110, 1999.
- [90] Su, T. H., Zhang, T.W., Guan, D. J, and Huang, H. J., “Gabor-based Recognizer for Chinese Handwriting from Segmentation-free Strategy”, Lecture Notes on Computer Science (LNCS): Computer Analysis of Images and Patterns, Springer-Verlag Berlin Heidelberg, Vol. 4673, pp. 539-546, 2007.
- [91] Tibshirani, R., Walther, G., and Hastie, T., “Estimating the number of clusters in a dataset via the gap statistic ”, J. Royal. Statist . Soc. B, Vol. 63, No. 2, pp. 411-423, 2001.
- [92] Touj, S., Essoukri, N., Amara, B. and Amiri, A. “Generalized Hough Transform for Arabic Optical Character Recognition”, Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR’03), 2003.
- [93] Trier, O. D., Jain, A. K., and Taxt, T., “Feature extraction method for character recognition; A survey”, Pattern Recognition, Vol. 29, No. 4, pp. 641–662, 1996.
- [94] Yaseen, M. et al, “Building Annotated Written and Spoken Arabic LR ’s in NEMLAR Project”, In Proceedings of the Language Resources and Evaluation Conference LREC06, Genoa,. Italy, May 2006
- [95] Young, S., et al., HTK Book, Cambridge University Engineering, 2006
- [96] Yu, B., Jain, A. K., “A Robust and Fast Skew Detection Algorithm for Generic Documents”, Pattern Recognition, 1996
- [97] Zeki, A. M., “The Segmentation Problem in Arabic Character Recognition The State Of The Art”, 1st International Conference on Information and Communication Technologies (ICICT), Aug. 2005.

الملخص العربي

مع فجر الألفية الجديدة نجد أن نظام التعرف الآلي للنص لم يعد موضوعاً مقتصرأ على هامش أعمال تكنولوجيا المعلومات فقط ولكنه أصبح نظاماً متكاملأ وجد له الكثير من الاستخدامات التجارية. إن التعرف الضوئي الآلي للنص المطبوع أصبح له احتياج كبير لعدد وفير من تطبيقات تكنولوجيا المعلومات الحديثه.

فالتعرف الضوئي للنص المكتوب بلغات لاتينية تم استخدامه منذ فتره طويله. أما بالنسبه للغات المكتوبه بأحرف متصله -وهي اللغه الأم لأكثر من ربع سكان العالم- فنجد أن نظام التعرف الضوئي للنص غير متوفر كنظام قوي موثوق في أدائه. فما زال هناك متسع للتحسينات فيما يتعلق بتخفيض معدل الكلمات الخطأ، والقدرة على العمل في وجود قدر معقول من الشوشرة، والتعامل مع مختلف أبناط وحجوم الكتابة بنفس النموذج، فضلاً عن عدم التقيد بحصيلة لغوية معينة. وفي هذا الصدد فإن التحدي الرئيسي في ذلك هو التشابك الإلزامي للحروف والذي يتعين حل هذه المشكلة بالتزامن مع التعرف على هذه الأحرف.

ومن بين العديد من المناهج التي تم تجربتها على مدار ثلاثون عاماً من البحث والتطوير ، يبدو أن التعرف الضوئي للنص القائم علي نماذج ماركوف المخفية (HMM) هو الأكثر وعداً وذلك لأنه يستفيد من قدرة مترجمات نماذج ماركوف المخفية لتحقيق التقطيع والتعرف على وحدات الكتابه في آن واحد كما هو الحال في التعرف الآلي للصوت (ASR) القائم علي نماذج ماركوف المخفية المستخدم علي نطاق واسع حالياً.

في هذه الرسالة نقوم بتطبيق نظام التعرف الآلي على النص العربي متعدد الأبناط و الحجوم مفتوح المفردات القائم علي نماذج ماركوف المخفية مع الاستعانه بنظام قوي ومطور

للتقطيع المبدئي للسطور والكلمات الخاصه بالنصوص متعددة الأبناط و الحجم و دون التقيد بحصيلة لغوية معينة. ينشأ عن وضع هذا النظام المفتوح لمفردات اللغة استخدامنا لنماذج ماركوف المخفية القائم على الحرف وليس القائم على الكلمة.

وخلافاً لنظام التعرف الآلي للصوت ينفصنا في التعرف الضوئي للنص القائم علي نماذج ماركوف المخفية التعريف لمجموعة المميزات المشتقة بدقة والقادرة على تحقيق الحد الأدنى من معدل الخطأ في الكلمات بالمقارنة مع تلك التي تحققت في النصوص اللاتينية. لذلك فنحن في هذه الرسالة نقوم بطرح ومناقشة تصميم هذه المميزات.

استناداً إلي التشابه بين التعرف الضوئي للنص والتعرف الآلي للصوت تم في هذه الرسالة طرح وتحليل نظام متكامل للتعرف الضوئي للنص المكتوب باللغة العربية مستلهما من نظام التعرف الآلي للصوت القائم على نماذج ماركوف المخفية بالاستعانه وبدون الاستعانة بنماذج احصائية لغوية. كما أن النتائج التجريبيه التي تم التوصل إليها في هذه الرساله تضع نظام التعرف الضوئي للنص القائم على نماذج ماركوف المخفية ومجموعة المميزات الجديدة المقترحة في هذه الرساله في المقدمه بين الأنظمة الأخرى للتعرف الضوئي للنصوص المكتوبه بأحرف متصله و المتعدده الأبناط و الحجم و الغير مقيدة بحصيلة لغوية معينة، وذلك بالنسبة للغة العربية على الأقل.