

See discussions, stats, and author profiles for this publication at: <https://www.researchgate.net/publication/2478749>

Automatic Recognition Of Words In Arabic Manuscripts

Article · January 2003

Source: CiteSeer

CITATIONS

24

READS

161

1 author:



Mohammad Khorsheed

King Abdulaziz City for Science and Technology

84 PUBLICATIONS 1,222 CITATIONS

[SEE PROFILE](#)

Some of the authors of this publication are also working on these related projects:



5 Year Strategic Plan for the Technology Development Center [View project](#)



KACST Arabic OCR [View project](#)

Automatic Recognition Of Words In Arabic Manuscripts

**Mohammad S. M. Khorsheed
Churchill College
University Of Cambridge**

**A dissertation submitted for the degree of
Doctor of Philosophy
June 2000**

Abstract

The need to transliterate large numbers of historical Arabic documents into machine-readable form has motivated new work on off-line recognition of Arabic script. Arabic script presents two challenges: orthography is cursive and letter shape is context sensitive.

This dissertation presents two techniques to achieve high word recognition rates: the segmentation-free technique and the segmentation-based technique. The segmentation-free technique treats the word as a whole. The word image is first transformed into a normalised polar image. The two-dimensional Fourier transform is then applied to the polar image. This results in a Fourier spectrum that is invariant to dilation, translation, and rotation. The Fourier spectrum is used to form the word template, or train the word model in the template-based and the multiple hidden Markov model (HMM) recognition systems, respectively. The recognition of an input word image is based on the minimum distance measure from the word templates and the maximum likelihood probability for the word models.

The segmentation-based technique uses a single hidden Markov model, which is composed of multiple character-models. The technique implements the analytic approach in which words are segmented into smaller units, not necessarily characters. The word skeleton is decomposed into a number of links in orthographic order, it is then transferred into a sequence of discrete symbols using vector quantisation. The training of each character-model is performed using either: state assignment in the lexicon-driven configuration or the Baum-Welch method in the lexicon-free configuration. The observation sequence of the input word is given to the hidden Markov model and the Viterbi algorithm is applied to provide an ordered list of the candidate recognitions.

Acknowledgements

This work has greatly benefited from the supervision of William Clocksin of the University of Cambridge Computer Laboratory. He was always there to listen and provide guidance throughout my time at Cambridge.

This work has also benefited from the Computer Vision Group meetings and the discussions with its members particularly John Daugman, Mantej Dhatt, and Thomas Harte. I am also grateful to Ian Lewis, Graham Titmus, and Robin Fairbairns from the Computer Laboratory.

Thanks to the University of Cambridge and the Computer Laboratory for providing an environment conducive to creative research.

My parents, ‘فَاطِمَة - Fatmah’ and ‘سَلِيمَان - Soliman’, the reason of my existence in this life and the two people in this universe who are worthiest of thanks, appreciation, and gratitude. Their sincere prayers and support on a daily basis have played a direct role in forming this dissertation.

I am also indebted to my wife, ‘إِيمَان - Eiman’. Her persistent effort to provide a relaxing atmosphere at home has made this work possible. ‘رَوَّان - Rwan’ and ‘رَزَّان - Rzan’, my two wonderful daughters, have also played their role in this by their patience and their gorgeous smiles which always energised me particularly during difficulties. I cannot forget the magical effect of the continuous phone calls I received from my sister and my brothers.

Declaration

This dissertation is the result of my own work and includes nothing which is the outcome of work done in collaboration. This dissertation is not substantially the same as any I have submitted for a degree or diploma or any other qualification at any other University. No part of this dissertation has already been, or is being currently submitted for any such degree, diploma or other qualification. The text has fewer than sixty thousand words.

Some of the results in this dissertation appear in papers published during the course of research presented here:

- M. Khorsheed and W. Clocksin, "Off-Line Arabic Word Recognition Using a Hidden Markov Model", *Statistical Methods For Image, A satellite conference of the 52nd ISI session in Helsinki*, University of Uppsala, Uppsala-Sweden, August-1999.
- M. Khorsheed and W. Clocksin, "Structural Features Of Curative Arabic Script", *The 10th British Machine Vision Conference*, University of Nottingham, Nottingham-UK, September-1999.
- W. Clocksin and M. Khorsheed, "Word Recognition in Arabic Handwriting", *International Conference on Artificial Intelligence Applications ICAIA '2000*, The American University, Cairo-Egypt, February-2000.
- M. Khorsheed and W. Clocksin, "Spectral Features for Arabic Word Recognition", *The IEEE International Conference on Acoustics, Speech, and Signal Processing ICASSP'2000*, Istanbul-Turkey, June-2000.
- M. Khorsheed and W. Clocksin, "Segmentation-Free Word Recognition For Arabic Handwriting", *International Conference on Pattern Recognition ICPR'2000*, Barcelona-Spain, September-2000.

Contents

1	INTRODUCTION	1
1.1	Research Scope	2
1.1.1	Document Analysis	2
1.1.2	Arabic Text Recognition	3
1.1.3	Arabic Text Transliteration	6
1.2	Motivation	6
1.3	Objectives	7
1.4	Research Approach	8
1.4.1	Segmentation-Free Technique	8
1.4.2	Segmentation-Based Technique	8
1.5	Contributions	8
1.5.1	Extracted Features	9
1.5.2	Classification Algorithm	9
1.6	Dissertation Overview	10
2	THEORY AND LITERATURE REVIEW	15
2.1	Introduction	15
2.2	Arabic Writing Characteristics	16
2.3	Arabic Script Transliteration	20
2.4	Arabic Text Recognition System	22
2.4.1	Image Acquisition	22
2.4.2	Preprocessing	26
2.4.3	Segmentation	34
2.4.4	Feature Extraction	40
2.4.5	Classification and Recognition	42

2.5	Arabic OCR Software	46
2.6	Summary	47
3	METHODOLOGY: USEFUL TECHNIQUES	49
3.1	Introduction	49
3.2	Vector Quantisation	50
3.2.1	Training set	51
3.2.2	Distance measure	51
3.2.3	Centroid Computation	52
3.2.4	Classification Algorithm	53
3.3	Hidden Markov Models	54
3.3.1	Theory and Elements of HMM	55
3.3.2	Problems To Be Solved	57
3.3.3	HMM Types	65
3.3.4	Application To Character Recognition	67
3.4	Summary	69
4	ARABIC MANUSCRIPT SAMPLES	71
4.1	Introduction	71
4.2	Computer-Generated Fonts	72
4.2.1	Simplified Arabic	72
4.2.2	Arabic Traditional	72
4.2.3	Thuluth	73
4.2.4	Andalus	74
4.3	Handwritten Manuscripts	75
4.3.1	UL Moh194.a.4	75
4.3.2	UL Or.172	75
4.3.3	UL Or.1481	77
4.3.4	UL Qq.65	78
4.3.5	UL Qq.91	78
4.3.6	UL Qq.74	80
4.3.7	UL Add.2923	80
4.3.8	UL Add.3257	81
4.4	Summary	83

5	SPECTRAL FEATURES OF ARABIC WORDS	85
5.1	Introduction	85
5.2	Two-dimensional Shape Description	86
5.2.1	Boundary Descriptors	86
5.2.2	Region Descriptors	88
5.3	Normalised Polar Image	90
5.3.1	The Centroid Of The 2D Shape	90
5.3.2	Maximum Distance Computation	91
5.4	Spectral Features	94
5.5	Summary	97
6	TEMPLATE-BASED RECOGNITION SYSTEM	101
6.1	Introduction	101
6.2	Pattern-Comparison Techniques	101
6.2.1	Euclidean distance measure	102
6.2.2	Statistical distance measure	102
6.3	Template Forming	103
6.3.1	A Single-Font Case	103
6.3.2	Multiple Font Case	104
6.4	Evaluation Experiments and Results	105
6.4.1	Typewritten Sample Results	105
6.4.2	Single Handwritten Font Results	109
6.4.3	Four-Font Experiments	112
6.4.4	All-Font Experiments	113
6.4.5	Single-Font Performance	113
6.4.6	Recognised Word Samples	117
6.5	Summary	119
7	MULTI-HMM RECOGNITION SYSTEM	123
7.1	Introduction	123
7.2	VQ Implementation	124
7.2.1	Dimensionality Reduction	124
7.2.2	Segmental VQ	125
7.2.3	VQ Results	127
7.3	HMMs For Word Recognition	129

7.3.1	The Training Phase	129
7.3.2	The Recognition Phase	133
7.4	Recognition and Evaluation	134
7.4.1	Case-I	134
7.4.2	Case-II	137
7.4.3	Case-III	141
7.4.4	Recognised Word Samples	144
7.5	Summary	144
8	STRUCTURAL FEATURES OF ARABIC WORDS	149
8.1	Introduction	149
8.2	Preprocessing	150
8.2.1	Base Line Estimation	150
8.2.2	Thinning	151
8.2.3	Skeleton Modification	152
8.3	Feature Extraction	155
8.3.1	Link Extraction	155
8.3.2	Loop Extraction	156
8.3.3	Feature Set FS_1	157
8.3.4	Feature Set FS_2	159
8.4	Feature Vector Encoding	162
8.4.1	FS_1 Codebook	163
8.4.2	FS_2 Codebook	163
8.4.3	Encoding Examples	164
8.5	Summary	166
9	SINGLE HMM RECOGNITION SYSTEM	169
9.1	Introduction	169
9.2	Lexicon-Driven Scheme	170
9.2.1	System Overview	170
9.2.2	HMM Implementation	172
9.2.3	Evaluation Results	176
9.3	Lexicon-Free Scheme	178
9.3.1	System Overview	179
9.3.2	HMM Implementation	181

9.3.3	Evaluation Results	184
9.4	Summary	189
10	CONCLUSION	191
10.1	What can we learn?	194
10.2	Contributions	196
10.3	Future Work	197

List of Figures

1.1	Two samples of Arabic document images.	4
1.2	Fundamental steps to recognising Arabic text.	5
1.3	A sample of Arabic text, and the transliterated version.	6
1.4	The formal organisation of the dissertation.	11
2.1	A sample of Arabic writing. The two word are 'الْخليفة' - 'The caliph' and 'أَبُو بَكْر' - Abu Bakr'.	19
2.2	A general diagram for the Arabic text recognition sys- tem.	23
2.3	Different text acquisition methods.	24
2.4	Sample results from eight single-threshold selection methods.	25
2.5	Two structuring elements. The origin is surrounded by a frame.	27
2.6	Results of opening and closing using structuring el- ements in Fig 2.5, (a) the original text image, (b) opening (a) with SE_1 , (c) closing (b) with SE_1 , (d) opening (a) with SE_2 , and (e) closing (d) with SE_2	28
2.7	Results of removing noise from the original text image in Fig 2.6 using median filter, window size is (a) 3×3 , (b) 5×5 , and (c) 7×7	28
2.8	Removal of grid lines using a notch filter.	29
2.9	Skew detection, (a) document image rotated 18 de- grees to the left, (b) the Hough Transform of (a).	30

2.10	Document decomposition, (a) original document image, (b) the horizontal projection, (c) the vertical projection of each text line, and (d) the document image after segmenting it into words.	31
2.11	Document connected components.	32
2.12	Sample results of a number of thinning algorithms.	33
2.13	Segmenting a word into its characters	35
2.14	Segmenting a word into primitives.	36
2.15	Segmenting the contour of a word.	37
3.1	Flow chart of K-means clustering algorithm.	55
3.2	The recursion procedure required to compute the forward variable $\alpha_{t+1}(j)$	59
3.3	The recursion procedure required to compute the backward variable $\beta_t(i)$	60
3.4	Sequence of operations required for the computation of the variables: $\xi_t(i, j)$ in equation 3.30, and $\gamma_t(i)$ in equation 3.32.	64
3.5	Two different types of HMMs.	66
4.1	Script image constructed into the shape of a lion.	72
4.2	Sample text image written in Simplified Arabic computer-generated font, and the transliterated text.	73
4.3	Sample text image written in Arabic Traditional computer-generated font, and the transliterated text.	73
4.4	Sample text image written in Thuluth computer-generated font, and the transliterated text.	74
4.5	Sample text image written in Andalus computer-generated font, and the transliterated text.	74
4.6	Sample text image extracted from the manuscript Moh194.a.4, and the transliterated text.	76
4.7	Sample text image extracted from the footnotes of the manuscript Moh194.a.4, and the transliterated text.	76
4.8	Sample text image extracted from the manuscript Or.172, and the transliterated text.	77

4.9	Sample text image extracted from the manuscript Or.1481, and the transliterated text.	78
4.10	Sample text image extracted from the manuscript Qq.65, and the transliterated text.	79
4.11	Sample text image extracted from the manuscript Qq.91, and the transliterated text.	79
4.12	Sample text image extracted from the manuscript Qq.74, and the transliterated text.	80
4.13	Sample text image extracted from the manuscript Add.2923, and the transliterated text.	81
4.14	Sample text image extracted from the manuscript Add.3257, and the transliterated text.	82
5.1	Two boundary descriptors: (a) original character image, (b) the character contour, (c) the skeleton, (d) the polygonal approximation, (e) the chain codes of (b) starting from the pixel marked with the arrow and moving in the clockwise direction, and (f) the chain codes of (d).	86
5.2	Two images of the character <i>Noon</i> . The centroid of each image is marked with a cross ×.	91
5.3	Normalised polar images for a number of Arabic words written in Simplified Arabic computer-generated font. The words are: 'بلدة - Town', 'ذكر - Male'	92
5.4	Normalised polar images for a number of Arabic words written in Thuluth computer-generated font. The words are: 'كبير - Big', 'دينار - Dinar'	93
5.5	Normalised polar images for a number of Arabic words written in Arabic Traditional computer-generated font. The words are: 'الليل - The night', 'رب - Lord'	93
5.6	Normalised polar images for a number of Arabic words written in Andalus computer-generated font. The words are: 'رسول - Prophet', 'مصر - Egypt'	94

5.7	Normalised polar images of the Arabic word 'عثمان' - OTHMAN' written in different handwritten and computer-generated fonts.	95
5.8	Normalised polar images of the Arabic word 'النبي' - The Prophet' written in different handwritten and computer-generated fonts.	96
5.9	Word samples represent the words: (a) 'الله' - God', (c) 'في' - In', (e) 'على' - On', (g) 'عبدالله' - Abdullah', (b),(d),(f), and (h) are the normalised Fourier spectrums of (a),(c),(e), and (g), respectively.	98
5.10	Word samples represent the words: (a) 'من' - From', (c) 'إليه' - To/for him', (e) 'قال' - Said', (g) 'ذلك' - That', (b),(d),(f), and (h) are the normalised Fourier spectrums of (a),(c),(e), and (g), respectively.	99
5.11	Word samples represent the words: (a) 'طالب' - Student', (c) 'إبراهيم' - Abraham', (e) 'دينار' - Dinar', (g) 'الجمعة' - Friday', (b),(d),(f), and (h) are the normalised Fourier spectrums of (a),(c),(e), and (g), respectively.	100
6.1	The Fourier spectrum decomposition into smaller frequency bands, ($N = 64$): (a) $Band_8 \rightarrow \frac{N}{8} = 8$, (b) $Band_{16} \rightarrow \frac{N}{4} = 16$. These are sub-samples of a factor of four of the real illustrations.	104
6.2	A block diagram of a conventional template-based pattern recognition system.	105
6.3	Error rate against lexicon size for typewritten fonts.	107
6.4	Three-dimensional depictions of confusion matrices for two different font sets, subjected to: (a) 252300 recognition tests, and (b) 189225 recognition tests.	109
6.5	Error rate against lexicon size of handwritten fonts. Templates are $Band_8$ size.	110
6.6	Error rate against lexicon size of handwritten fonts. Templates are $Band_{16}$ size.	111

6.7	Error rate against lexicon size. Word templates are formed from all fonts mentioned in Chapter 4.	114
6.8	Font <i>A</i> recognition rate against lexicon size.	115
6.9	Font <i>L</i> recognition rate against lexicon size.	116
6.10	Word templates and samples: (a,b) 'قَرْيَة - Village', (c,d) 'عَبْدُ اللَّهِ - Abdullah', (e,f) 'سَنَة - A year', (g,h) 'عِبْرَة - A lesson', (i,j) 'عُمَر - Omar', (k,l) 'بَعْض - Some', (m,n) 'ذَكَر - Male', and (o,p) 'هَذَا - This'. Word templates are framed.	118
6.11	Word templates and samples: (a,b) 'الَّيْل - The night', (c,d) 'هَذَا - This', (e,f) 'أَخْلَفَاء - The Caliphs', and (g,h) 'مَكَّة - Mecca'. Word templates are framed. . . .	119
6.12	Word templates and samples: (a,b) 'كَانَ - Was', and (c,d) 'كِتَاب - A book'. Word templates are framed. . .	119
6.13	Word templates and samples: (a,b) 'مُحَمَّد - Mohammad', (c,d) 'رَضِيَ - Be pleased', (e,f) 'وَاحِدَة - Single', and (g,h) 'الْعِلْم - The science'. Word templates are framed.	120
6.14	Word templates and samples: (a,b) 'أَلْدُنْيَا - This life', and (c,d) 'بَنِي - Sons of'. Word templates are framed.	120
7.1	The Wedge Ring Detector (WRD) of the Fourier spectrum.	125
7.2	The Fourier spectrum decomposition into six adjacent sectors.	126
7.3	The block diagram of vector quantiser and HMM word recogniser.	127
7.4	Plots of the average distortion $ D_M $ versus size of codebooks.	128

7.5	The HMM with serial constraints, only one skip is allowed.	129
7.6	A block diagram illustrating HMMs competition in assigning the input observation sequence to the winning model.	132
7.7	Recognition rates of typewritten word samples. . . .	135
7.8	The effects of the codebook and the training data set sizes on the recognition rate.	136
7.9	Three-dimensional depictions of confusion matrices for the computer-generated fonts, subjected to 252300 recognition tests. In (b) and (e) 33% of the data set was used for training, while in (c) and (f) 66% of the data set was used for training.	138
7.10	Recognition rates of word sample from fonts: <i>A, B, J</i> , and <i>L</i>	139
7.11	Three-dimensional depictions of confusion matrices for two recognition systems, subjected to 174435 recognition tests. Included fonts are: <i>A, B, J</i> , and <i>L</i> . In (c) and (d) 40% of the data set was used for training the word models.	140
7.12	Recognition rates of word samples from all available fonts: <i>A, B, C, D, E, F, G, H, I, J, K, L</i> , and <i>M</i> . The recognised word appeared as the first option.	142
7.13	Three-dimensional depictions of confusion matrices for HMM-based and template-based recognition systems, subjected to 488650 recognition tests. The data set includes all the thirteen fonts. In (b) and (e) 33% of the data set was used for training the word models, while in (c) and (f) 66% was used for this purpose. . .	145
7.14	Word templates and samples: (a,b) ‘عنه’ - About him’, (c,d) ‘الدِّين’ - The religion’, (e,f) ‘على’ - On’, (g,h) ‘وسلم’ - May peace be upon’, (i,j) ‘عبّاس’ - Abbas’, and (k,l) ‘أمير’ - Prince’. Word templates are framed.	146

7.15	Word samples that HMMs could not recognise correctly: (a) 'حَيْر' - Goodness', and (b) 'الشَّام' - Al-Sham'.	147
8.1	Base line estimation. Each base line is marked with a black straight line.	150
8.2	An illustration of the connectivity number. N_i is one if the pixel is white	151
8.3	Results of thinning algorithms.	153
8.4	Skeleton modification: (a) the word skeleton imposed on the word contour, (b) merging adjacent feature points and calculating new group centres, and (c) reconstructing the word skeleton.	153
8.5	Simple and complex loops that can be seen in some handwritten letters.	156
8.6	Feature $f3$ calculation: link \overline{ab} has feature $f3$ whose value equals 0, while link \overline{cd} has feature $f3$ whose value equals 2.	158
8.7	The importance of deciding the dot positions relative to the base line. The two dots of the first letter are: (a) below the base line, this makes the word 'يبيع' - He buys', and (b) above the base line, this makes the word 'تبيع' - She buys'.	158
8.8	Calculating features $f5 \rightarrow f8$ to the link \overline{ab} which represents the main stroke of the letter 'بي'.	159
8.9	The edge linking process: (a) the original skeleton graph of the word 'الجمعة' - Friday', and (b) the skeleton after being transformed into a sequence of line segments each marked with a number. Feature points are marked with small letters.	160
8.10	Turning points imposed on two word skeletons: (a) 'قال' - Said' and (b) 'تكريم' -	162
8.11	An illustration of transferring a typewritten word image into a sequence of feature vectors	164

8.12	An illustration of transferring a handwritten word image into a sequence of feature vectors	165
9.1	A block diagram of the HMM-based lexicon-driven recognition system.	170
9.2	Single HMM recognition system.	172
9.3	The end forms of letters: (a) 'ب' and (b) 'ش'. The two-letter combination of: (c) 'حا' and (d) 'لي' Elementary units representing letters: (c) 'ب' and (d) 'ش'.	173
9.4	A block diagram of the HMM-based lexicon-free recognition system.	179
9.5	Line segments of: (a) a two-letter combination 'تا', (b) a two-letter combination 'كا', and (c) letter 'ص'. The black mark denotes a turning point.	181
9.6	Three dimensional depictions showing the confusion matrices of: (a) Four-state character-models, (b) Varying-size character-models. Total number of recognition tests is 12960 associated with 405 character samples representing 32 character-models.	183
9.7	Two block illustrations of the word path building through the HMM for: (a) 'مُحَمَّد' - MOHAMMAD', (b) 'حَمَد' - HAMAD'. The two words share the same character-models but in a different order and with different frequencies.	185
9.8	The word paths through the HMM for: (a) 'مُحَمَّد' - MOHAMMAD', (b) 'حَمَد' - HAMAD'. There are four states in each character-model.	185
9.9	The word paths through the HMM for: (a) 'مُحَمَّد' - MOHAMMAD', (b) 'حَمَد' - HAMAD'. Each character-model has a varying number of states relative to the number of line segments the letter has.	187

List of Tables

2.1	The Arabic alphabet set. Each character may have up to four different shapes. The transliterated form of each character is illustrated in the right column.	17
4.1	List of manuscript codes.	83
6.1	Recognition rates (%) of the first set of experiments. Each experiment used a single font to build the template database.	106
6.2	Recognition rates (%) of typewritten word samples.	106
6.3	Recognition rates (%) of the four computer-generated fonts of a 145-word lexicon. Templates are <i>Band</i> ₁₆ size.	108
6.4	Recognition rates (%) of three computer-generated fonts: <i>J</i> , <i>K</i> , and <i>L</i> . Templates are <i>Band</i> ₁₆ size.	108
6.5	Recognition rates (%) of the four selected fonts: <i>A</i> , <i>B</i> , <i>J</i> , and <i>L</i> of a 145-word lexicon. Euclidean distance is used as a dissimilarity measure.	112
6.6	Recognition rates (%) of each of the four selected fonts: <i>A</i> , <i>B</i> , <i>J</i> , and <i>L</i> of a 145-word lexicon.	113
7.1	Recognition rates (%) of typewritten word samples representing 145-word lexicon.	137
7.2	Recognition rates (%) of four font word samples representing 145-word lexicon.	141
7.3	Recognition rates (%) of the four experiments and the template-based recogniser.	142

7.4	Recognition rates (%) of word samples extracted from all available fonts, and representing 145-word lexicon.	143
8.1	Codebook classes for FS_1 feature set.	163
8.2	Codebook classes for FS_2 feature set.	163
9.1	The recognition rates of the single HMM lexicon-driven system.	176
9.2	The system output of the word sample 'قيل - said' from J font. † means not in the lexicon, and ‡ means not a valid Arabic word.	176
9.3	The system output of the word sample 'علي - ALI' from K font. † means not in the lexicon, and ‡ means not a valid Arabic word.	177
9.4	The system output of the word sample 'العرب - The Arabs' from L font. † means not in the lexicon, and ‡ means not a valid Arabic word.	177
9.5	The system output of the word sample 'ثقوب - holes' from L font. † means not in the lexicon, and ‡ means not a valid Arabic word.	178
9.6	The system output of the word sample 'تمامًا - completely' from L font. † means not in the lexicon, and ‡ means not a valid Arabic word.	178
9.7	The system output of the word sample 'مكة - MECCA'.	186
9.8	The system output of the word sample 'أبراهيم - ABRAHAM'.	188
9.9	The recognition rates of the single HMM lexicon-free system.	188

CHAPTER 1

INTRODUCTION

The origin of character recognition can be found as early as 1870. It first appeared as an aid to the visually handicapped, by the Russian scientist Tyurin in 1900 [GS90]. In 1929, Tausheck obtained a patent on OCR in Germany and in 1933 Handel did the same in the United States [MSY92]. OCR machines have been commercially available since the beginning of the 1950s [GS90, MSY92, Nag92].

The spectacular progress of OCR since its commercial debut has been due to advances in process and digitiser technology rather than to improved classification techniques for isolated patterns. New methods for recognising entire documents are currently in more demand than better character-by-character feature extraction and classification algorithms.

Hundreds of papers have been published on alternative line-thinning, contour-following, feature-extraction and classification algorithms. Most of these methods are restricted to isolated characters. It was not until the last three decades that the attention of university researchers turned to hand-printed rather than machine-printed characters, with a few experiments on cursive script.

This dissertation presents two techniques used to recognise words in Arabic manuscripts: the holistic technique and the analytic technique. The holistic technique or word shape recognition treats the word as a whole. Features are extracted from the unsegmented word image and compared against all words in the lexicon. Each word is represented by a single template or hidden Markov model. The analytic technique tends to decompose the word into smaller units which are then transformed into an observation sequence. Two schemes are implemented using this technique. The first scheme represents the whole lexicon using a single hidden Markov model. The model parameters are estimated by incorporating the high-level knowledge from the lexicon, initial-letter and letter-transition probabilities, and the low-level knowledge from the training samples. The second scheme is lexicon-free. It solely depends on the low-level knowledge from the training samples. Each letter in the alphabet is represented by a single hidden Markov model. A global model is composed by interconnecting the character-models. Each word in this technique has one or more paths through the global model.

1.1 Research Scope

There are two major forms of language: written and spoken. The written form, which represents the focus of this work, is contained in printed documents and in handwritten matter. Because of the great importance of the written language to human transactions, recognising different written materials automatically represents a practical significance.

1.1.1 Document Analysis

Document analysis aims to encode any information presented on paper and intended for human comprehension into an appropriate electronic form [SLG⁺92, SBB⁺92, SL95, Sri86]. A document may consist of text, line drawings, tables, and pictures, see Fig 1.1. Doc-

ument analysis is a very important step towards office automation. Another and a more demanding application is to recognise ancient manuscripts. This interest also includes the transliteration of non-English manuscripts such as Arabic and Syriac into the Latin alphabet.

There is an overwhelming number of historical Arabic manuscripts awaiting transliteration into machine-readable form. Currently, this process is carried out manually. The manual task is tedious, time consuming and costly. On the other hand, standard OCR methods are not applicable to even the most carefully written of these texts, because the script is cursive and difficult to segment into meaningful character units.

1.1.2 Arabic Text Recognition

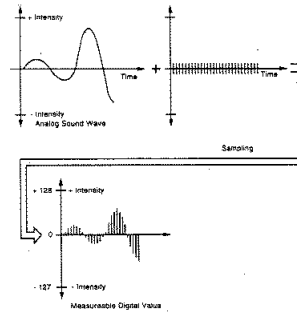
Character recognition, or more generally text recognition, is concerned with studying the automatic reading of a script. It is very important to realise the limitations of technology as the human ability to read a document which has been poorly written is currently unattainable by means of technology. This is due to the fact that the human utilises the ability to understand the context in which the character appears in order to simplify the problem of recognition. This recognition is more difficult when considering cursive text.

Arabic language provides a rich source of technical challenges for recognition and transliteration algorithms. The most obvious characteristic of the Arabic language is that it is written from right-to-left, however this is not a problem. Arabic scripts are inherently cursive: writing isolated characters in ‘block letters’ is an unacceptable and unused writing style. The letters are context sensitive. Certain character combinations form new ligature shapes which are often font dependent. Some ligatures involve vertical stacking of characters. Since not all letters connect, word boundary location becomes an interesting problem, as spacing may separate not only words but also certain characters within a word.

Letters are formed by calligraphic curved strokes that may dis-

أسعار صرف بعض العملات العربية *					
العملة	الدولار الأميركي	اليورو	الجنيه الاسترليني	الدين الياباني	الفرنك السويسري
الدينار الكويتي	٣,٠٣٦	٣,٠٧٣	٤,٩٢٦	٣٣٧,٢	٥,١٩٦
الريال السعودي	٣,٧٥٠	٣,٧٩٥	٦,٠٨٤	٢٧,٣١	٤,٢٠٧
درهم الإمارات	٣,٦٧٢	٣,٧٥٥	٥,٩٥٨	٢٧,٨٨	٤,٢٩٦
الدينار البحريني	٣,٧٦٩	٣,٨١٥	٦,١١٦	٢٧١,٦	٤,١٨٥
الريال العماني	٣,٨٤٩	٣,٨٩٦	٦,٢٤٦	٢٦٦,٠	٤,٠٩٨
الريال القطري	٣,٦٣٩	٣,٦٨٢	٥,٩٠٤	٢٨,١٣	٤,٣٣٥
الليرة اللبنانية	١٥٠١	٦٥١٨,٨	٢٤٣٥	١٤,٦٥	٩٥١,٣
الجنيه المصري	٣,٤١٩	٣,٤٥٩	٥,٥٤٦	٢٩,٩٥	٤,٦١٥
الدينار الأردني	٧,٠٨١	٧,١٦٤	١,١٤٨	١٤٤,٦	٢,٢٢٨
الدينار العراقي	١٢٥٠	١٢٦٤,٨	٢٠٢٧,٨	٠,٨	٠,٠١٣

* أسعار صرف الدولار و الجنيه الاسترليني تشير الي قيمة كل منهما في مقابل العملة العربية المحددة ، والعكس صحيح بالنسبة لل عملات الأجنبية الأخرى.



شكل ١-١٣ عملية تحويل الصوت من الصورة التناظرية إلى الصورة الرقمية

ما معنى أن كارت الصوت يحتاج 16-bit or 8-bit audio!

في الواقع Sound 8-bit أو 16-bit لا تعني أن كارت الصوت له 8-bit expansion أو 16 ولكنهما تعني أن كارت الصوت يستخدم 16-bit أو 8-bit لتوقيع (digitize) كل Sound Sample. وبالنسبة لـ 8-bit فعند القيم الرقمية تصل إلى ٢٥٦ بينما 16-bit تصل إلى ٦٥٥٣٦ قيمة وعموماً فإن 8-bit audio هي الأنسب لتسجيل الكلام (speech) بينما 16-bit audio هي الأفضل في تسجيل الموسيقى (انظر شكل ١-١٣).

Figure 1.1: Two samples of Arabic document images.

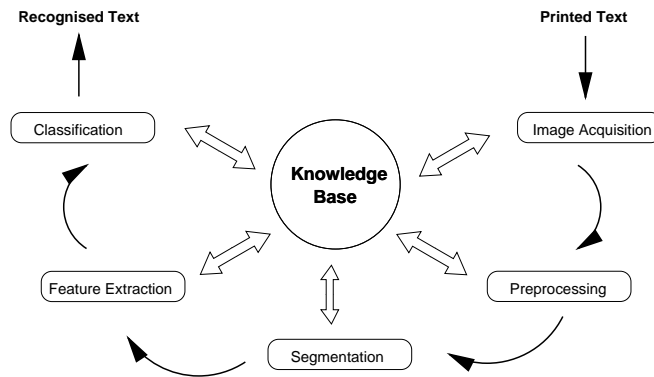


Figure 1.2: Fundamental steps to recognising Arabic text.

play a wide variation of affine and non-affine distortions used for flourishes or other decorative purposes. While variation of individual letter-forms does not arise in typescript, it is normal in handwriting. An important note here is that margins are justified not by adding spaces to the line as in English, but by elongating the baseline strokes of certain letters within the words.

There are five fundamental steps in any Arabic text recognition system: image acquisition, preprocessing, segmentation, feature extraction, and classification and recognition, see Fig 1.2. The image acquisition step acquires the digital image of the text using an imaging sensor. If the output of that imaging sensor is not already in digital form, an analogue-to-digital converter digitises it. The second step is the preprocessing step which aims to improve the text image by enhancing contrast and removing noise. The third step is segmentation in which the text image is partitioned into its constituent parts. The raw text image or its segments are not suitable for the recognition process. The feature extraction step is describing the input image so that features of interest are highlighted. This step results in some quantitative information of interest or features that are basic for differentiating one class of objects from another. Finally, the classification and recognition step assigns a label to an object based on the information provided by its description.

1.1.3 Arabic Text Transliteration

Transliteration aims to write the Arabic text in Latin characters such that both representations are phonetically identical, see Fig 1.3. An important feature of Arabic text which affects the transliteration process is the presence of diacritics. These are marks placed above and below characters to represent vowels and consonant doubling. However, for a fully diacritised Arabic word it is possible to reach one-to-one matching with its English counterpart. On the other hand, for an undiacritised Arabic word there are a number of transliterations. Currently, the transliteration process is carried out manually. In practice, diacritics are optional as Arabic readers are accustomed to reading undiacritised text and inferring the meaning from the context.

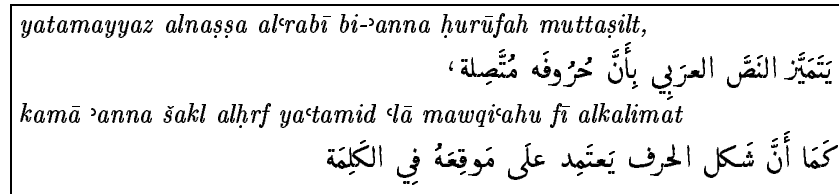


Figure 1.3: A sample of Arabic text, and the transliterated version.

1.2 Motivation

Arabic text recognition is a very important subject for both Arabs and non-Arabs. Arabs consider it as a valuable tool for transforming all printed materials into electronic media, while non-Arabs look at it as an intermediate step towards fully transliterating Arabic text particularly ancient Arabic manuscripts.

There is a very rich collection of Arabic manuscripts around the world, many of which are extant in the UK. The estimated number of these manuscripts exceeds three million [EID98]. Scholars need such texts in machine readable form to prepare critical editions of Arabic sources. Most scholars rely on typists who have the necessary

language skills. However, such individuals are rare, which burdens them with excessive work. The result is a slow and invaluable process.

The cursiveness of the script and the character shape sensitivity to context both confound conventional OCR techniques, which are inclined to treat legitimate variation as noise rather than as inherent characteristics of the data. The effect of legitimate variation on most trainable systems such as neural networks is often to widen classification tolerances rather than to model the context dependency at the root of the variation.

1.3 Objectives

A long-term goal is automatic transliteration of Arabic script. The aim of this dissertation is to make progress towards this goal by designing and implementing a recognition system of typewritten and handwritten Arabic script. This script may be machine-printed or handwritten. There are also underlying goals which harmonise with the main objective:

- Transliterating an Arabic text image requires first recognising that text therefore one goal is to improve upon previous work on typed/hand written optical character/word recognition.
- An important step towards recognising a text image is feature extraction. This dissertation investigates two feature types: global transformation features and structural features.
- The classification algorithm is the decision-making step in the recognition process. This dissertation investigates the implementation of trainable statistical classification models, Hidden Markov Models, to perform the classification and the recognition task.

1.4 Research Approach

The main challenge to any Arabic text recognition system is the cursive nature of the script. This dissertation explores two techniques to solving this problem: a segmentation-free technique, and a segmentation-based technique. Both techniques implement Hidden Markov Models in order to recognise words in Arabic manuscripts.

1.4.1 Segmentation-Free Technique

This technique recognises the entire word as a portion of connected characters without prior segmentation. This involves analysing the word shape with a unique vector of features. The feature vector may then be matched against a database of analogous feature vectors, or represented in attribute/value form to an inductive learning system such as Hidden Markov Models. At some locations of the dissertation, this technique is referred to as the word-model technique, since each word is represented by a separate prototype/model.

1.4.2 Segmentation-Based Technique

This technique attempts to decompose the word image into smaller links which are then described by using the feature space of the problem. The image word is transformed into a sequence of feature vectors suitable for training a hidden Markov model. At some locations of the dissertation, this technique is referred to as the character-model technique, since each word is represented by a separate path through the hidden Markov model and each path is a concatenation of various characters.

1.5 Contributions

The following are the contributions of the research presented in this dissertation, introduced in twofold: extracted features and classification algorithm.

1.5.1 Extracted Features

Two types of features are implemented in this dissertation to describe cursive Arabic words: the global transformation features and the structural features.

1.5.1.1 Global transformation features

This dissertation presents a method to describe cursive Arabic words using the two-dimensional Fourier spectrum of the normalised polar equivalent. The method results in a two-dimensional feature vector that tolerates changes in size, displacement, and rotation. This feature vector can be used by its own as a word template, or divided into wedge-shaped sectors and used as an observation sequence, after vector quantisation, to train the hidden Markov model.

1.5.1.2 Structural features

Two different structural feature sets are presented in this dissertation. Although some of these features are known and have been used before, the combination of various features in a single feature vector can enhance the discriminating ability within the feature space.

1.5.2 Classification Algorithm

Recently, hidden Markov models have been successfully applied to character/word recognition. This dissertation implements two different approaches of hidden Markov models to recognise cursive words in Arabic manuscripts: the model discriminant approach and the path discriminant approach.

1.5.2.1 The model discriminant approach

This approach builds one hidden Markov model for each word in the lexicon. Given an observed pattern, which represents the whole word, the matching score against each model is calculated and the word image is identified with the class/word whose model gives rise to the maximum score.

1.5.2.2 The path discriminant approach

This approach builds only one model for all the words and uses different paths (state sequences) to distinguish one pattern from the others. A pattern is classified to the word which has the maximum path probability over all possible paths. Further within this approach two different schemes are implemented:

- *lexicon-driven scheme*: This incorporates the knowledge from the given lexicon with that from the training samples to estimate the model parameters.
- *lexicon-free scheme*: This is composed from smaller character-models. Each character-model is represented by a separate hidden Markov model. The model parameter estimation is solely based on the low-level pixel-based knowledge of the training samples.

A considerable improvement is made when more than merely the best path is recovered. This is achieved by extending the Viterbi algorithm variables to another dimension which represents the list of choices.

1.6 Dissertation Overview

The formal organisation of the dissertation is illustrated in Fig 1.4, and it is as follows. Chapter 2, entitled “Theory and Literature Review”, provides a review of the theory of Arabic recognition system. The characteristics of Arabic writing are presented with an emphasis on different writing styles. The concept of transliterating Arabic script is also introduced. The process of recognising Arabic text is broken into a sequence of steps. These steps are separately discussed, and previous research work on each step is reviewed. This chapter ends with an overview of available Arabic OCR softwares.

Chapter 3, entitled “Methodology: Useful Techniques”, Introduces two important tools that are used throughout this dissertation. It first shows how Vector Quantisation is used to code a fea-

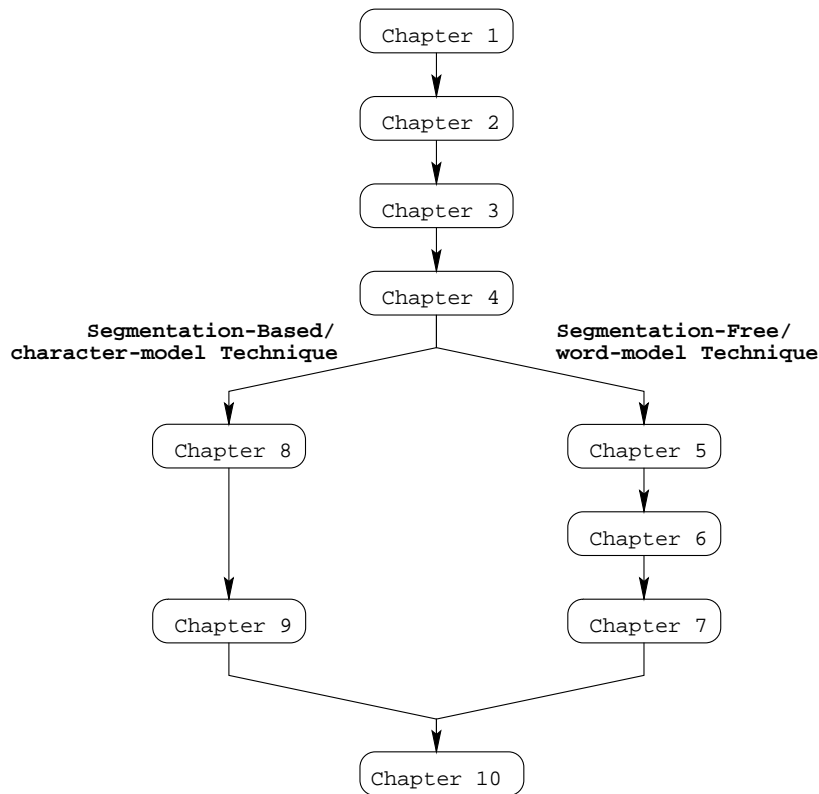


Figure 1.4: The formal organisation of the dissertation.

ture vector into one of a fixed number of discrete symbols in order to reduce the computation required in a recognition system. The chapter also discusses the theory and implementation of Hidden Markov Models to word/text recognition.

Chapter 4, entitled “Arabic Manuscript Samples”, illustrates a number of handwritten manuscripts from the Cambridge University Library (UL), some of these manuscripts are 800 years old. The chapter also presents four typewritten computer-generated font faces which feature ligature and overlap structures. A total of thirteen different typewritten and handwritten fonts participate in forming word samples used in this dissertation for training and recognition purposes.

Chapter 5, entitled “Spectral Features Of Arabic Words”, discusses the two step technique which provides the word feature vector. A normalised Polar Transform followed by an application of the two-dimensional Fourier Transform produces a word Fourier spectrum that tolerates changes in size, displacement, and rotation. This two-dimensional feature vector is utilised as a word template by the recognition system presented in Chapter 6, entitled “Template-Based Recognition System”. The system is a lexicon-based recogniser where each word in the lexicon is represented by a single template. Two different frequency bands are used to investigate the recognition capability of this scheme which is based on the minimum distance measure.

Chapter 7, entitled “Multi-HMM Recognition System”, presents a continuation of spectral feature based recognition system. It primarily divides the Fourier spectrum into six wedge-shaped sectors to transform it into a one-dimensional observation sequence. Observation sequences are used to train word models where there is a separate hidden Markov model for each word in the lexicon.

Chapter 8, entitled “Structural Features Of Arabic Words”, discusses how to extract structural features from Arabic script. The chapter first applies an algorithm to produce a spurious-branch-free skeleton from a word image. It then decomposes the skeleton graph to smaller links from which two different feature sets are extracted. After excluding links that represent loops, the pool of feature vectors is partitioned into various classes using vector quantisation. The chapter ends by showing two examples of how to transfer a word image into a sequence of discrete symbols.

Chapter 9, entitled “Single HMM Recognition System”, utilises the observation sequences produced in Chapter 8 to build a recognition system which is based on a single hidden Markov model. The chapter provides two configurations: the lexicon-driven configuration and the lexicon-free configuration. It shows how the first configuration inherits the initial letter and letter-transition probabilities from the vocabulary to estimate the model parameters. The chapter

also implements the Baum-Welch method to train character-models which form the lexicon-free hidden Markov model. Finally, the modified Viterbi algorithm is applied to provide an order list of the best paths through the hidden Markov model in both configurations.

Chapter 10 summarises the main conclusions of this research. The chapter also highlights some of the opportunities for future work based on the current research.

CHAPTER 2

THEORY AND LITERATURE REVIEW

2.1 Introduction

Among the branches of pattern recognition is the automatic reading of a text, namely, text recognition. The objective is to imitate the human ability to read printed text with human accuracy, but at a higher speed [GS90]. According to [AM95b], a useful target performance is 5 characters/second with a 99.9% recognition rate, with all errors being rejections.

There are three factors pushing toward 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 to the new electronic media. This factor motivates the vast range of applications of off-line text recognition, which includes automatic mail routing [DTLH92, DFV97], machine processing of forms [CDD97], bank cheques [GS98a, GS98b], printed newspapers [GMC⁺97], and signature verification [FB99].

Most optical character recognition methods assume that individual characters can be isolated, and such techniques, although successful when presented with Latin typewritten or typeset text, cannot be applied reliably to cursive script, such as Arabic, where

the shape of the character is context sensitive. This feature, besides other characteristics of the Arabic language, has obliged researchers to examine some obstacles which have only recently been addressed by researchers of other languages. These obstacles have played an important role in delaying character recognition systems for Arabic language compared to other languages such as Latin and Chinese. Additional factors involve: the early start of these systems for Latin (1940s) and Chinese (1960s), the absence of scientific journals and conferences specialised in the field which caused a lack of communication between researchers, and the absence of support utilities such as a language corpus and electronic dictionaries.

Recently, a number of papers which analyse the work done on Arabic character/word recognition have appeared. Some of these papers deal with both on-line and off-line recognition [Sho89, AA93b, AM95b, EN98]. Others focus only on off-line recognition [AK94, AA96, Ami97, Ami98a].

2.2 Arabic Writing Characteristics

The Arabic language has a very rich vocabulary. More than 200 million people speak this language as their native language, and over 1 billion people use the Arabic language in several religion-related activities. The alphabet set used to write this language is the Arabic alphabet, see Table 2.1. There are also a number of languages that use the Arabic alphabet, such as Persian [PT81], Kurdi [G98], and Jawi [MHY98].

Arabic script is written from right to left. As opposed to starting from the left-most position of the page as for Latin, Arabic script starts from the right-most position of the page towards the left in a cursive way, even in machine-printed form *الكتابة العربية متصلة*.

The Arabic alphabet consists of 28 basic letters, which consists of strokes and dots. Ten of them have one dot, three have two dots, two have three dots. Dots can be above; ن ض, below; ب ي,

Character	Isolated	Initial	Middle	End	Transliteration
Alif	أ	أ	ا	ا	<i>a</i>
Ba'	ب	بـ	بـ	بـ	<i>b</i>
Ta'	ت	تـ	تـ	تـ	<i>t</i>
Tha'	ث	ثـ	ثـ	ثـ	<i>t</i>
Jeem	ج	جـ	جـ	جـ	<i>ǧ</i>
Ḥa'	ح	حـ	حـ	حـ	<i>ḥ</i>
Kha'	خ	خـ	خـ	خـ	<i>ḫ</i>
Dal	د	دـ	دـ	دـ	<i>d</i>
Thal	ذ	ذـ	ذـ	ذـ	<i>ḏ</i>
Ra'	ر	رـ	رـ	رـ	<i>r</i>
Zy	ز	زـ	زـ	زـ	<i>z</i>
Seen	س	سـ	سـ	سـ	<i>s</i>
Sheen	ش	شـ	شـ	شـ	<i>š</i>
Sad	ص	صـ	صـ	صـ	<i>ṣ</i>
Dhad	ض	ضـ	ضـ	ضـ	<i>ḍ</i>
T'ah	ط	طـ	طـ	طـ	<i>ṭ</i>
The'ah	ظ	ظـ	ظـ	ظـ	<i>ẓ</i>
Ain	ع	عـ	عـ	عـ	<i>ʿ</i>
Gain	غ	غـ	غـ	غـ	<i>ǧ</i>
Fa	ف	فـ	فـ	فـ	<i>f</i>
Qaf	ق	قـ	قـ	قـ	<i>q</i>
Kaf	ك	كـ	كـ	كـ	<i>k</i>
Lam	ل	لـ	لـ	لـ	<i>l</i>
Meem	م	مـ	مـ	مـ	<i>m</i>
Noon	ن	نـ	نـ	نـ	<i>n</i>
Ha'	هـ	هـ	هـ	هـ	<i>h</i>
Waw	و	وـ	وـ	وـ	<i>w</i>
Ya	ي	يـ	يـ	يـ	<i>y</i>

Table 2.1: The Arabic alphabet set. Each character may have up to four different shapes. The transliterated form of each character is illustrated in the right column.

or in the middle; ج ط of the letter. The shape of the letter is context sensitive, depending on its location within a word, as shown in Table 2.1. A letter can have up to four different shapes: isolated, beginning connection from the left, middle connection from the left and right, and end connection from the right. Most of the letters can be connected from both sides; the right and the left. However, there are six letters which can be connected from one side only; the right. These letters are grouped in the following two words: 'ذِرْوَزَاد'. This characteristic implies that each word may form one unit or more (sub-words), and each sub-word may contain more than one letter. As an example, consider the two word 'محمد - Mohammad' and 'عربي - Arab'. While the word 'محمد' forms a single connected component, the word 'عربي' forms two connected components. The reason is that all letters in the first word can be connected from both sides, while the letter 'ر' in the second word may only be connected from the right side which causes a discontinuity of cursiveness.

Some Arabic letters may have a zig-zag-like stroke called *Hamza*. This additional character can be on *Alif* 'أ', below *alif* 'إ', on *Waw* 'و', on a *Alif-Maqsur* 'ئ', or isolated on the line 'ء'. Another non-basic character is *Ta-Marbuta*. It is a special form of the letter 'ت', and it is always at the end of the word. It can be connected as in *سرعة* or isolated as in *هرة*.

Writing styles may be classified according to their complexity into three categories:

1. *Typewritten or machine-printed*: This is a computer-generated style and it is the simplest among all styles because of the uniformity in writing a word.
2. *Typeset*: This is normally used to print newspapers and books. Typeset style is generally more difficult than the machine-printed style because of the existence of overlaps and ligatures which poses a challenging problem, see Fig 2.1. Ligatures occur when two or more letters overlap vertically and touch. By contrast,

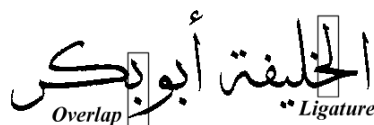


Figure 2.1: A sample of Arabic writing. The two word are 'الخليفة' - The caliph' and 'أبو بكر' - Abu Bakr'.

overlaps occur when two or more letters overlap vertically without touching. Recently, some computer-generated fonts have imitated the typeset style in providing ligatures and overlaps.

3. *Handwritten:* This is assumed to be the most difficult style because of the variations in character shape even if it is rewritten by the same person. Handwritten style may be further classified into: scribe, personal, and decorative. Scribe is more carefully written than the personal handwriting style that represents the daily usage of Arabic alphabet by individuals. Few people are able to perform an exquisite scribe handwritten script. A good example of scribe handwritten scripts is in the Arabic manuscripts which represent a point of interest for this research. Scribe handwriting is certainly different from decorative handwriting which is normally used for adornment purposes.

A *Baseline* is an important aspect of Arabic writing. This is a horizontal line that runs through the connected primitives of a text. In a binary image test, the baseline is assumed to have the maximum number of black pixels. This assumption is no longer valid if the script is skewed, refer to Sec 2.4.2.2. If the script is handwritten the baseline is not usually straight, and may only be estimated.

Arabic characters may have diacritics which are written as strokes and can change the pronunciation and the meaning of the word. Diacritics may appear as strokes above the character as *Fat-ha* in زَرَعَ , *Dhamma* in جُرِرَ , *Sukun* in يَكْتُبُ , *Shadda* in نَوْرَ, or

Maddah in رَآه, or below the character as *Kasra* in اِيل. *Tanween* is a form of diacritising Arabic script but with double *Fat-ha* as in شَيَّاً, double *Dhamma* as in شَيِّء, or double *Kasra* as in شَيِّء. These diacritics perform an essential function in transliterating an Arabic script as described below. In spite of this importance, text may be undiacritic and readers of Arabic are accustomed to inferring the meaning from the context.

2.3 Arabic Script Transliteration

Arabic transliteration into English intends to interpret an Arabic text in English standard machine readable letters such that both representations are phonetically identical. In other words, Arabic transliteration is concerned with rewriting Arabic words in the Latin alphabet. One reason for transliteration is that it is not always possible to render an Arabic text into English. This is because at times there is no corresponding equivalent for an Arabic word in English. As a result, transliteration can serve two purposes: acting as an intermediate step toward translation and transforming the Arabic script into Roman letters so it can be easily read by non-Arabic readers.

Transliteration is affected by diacritics, as mentioned previously, which play an important role in pronouncing a certain word. To illustrate this with an example: consider the following word ‘كَلِيَّة’, which can be pronounced as either ‘كُلِيَّة *kuliyyat* - college’, or ‘كَلِيَّة *kilyat* - kidney’. Thus, for an undiacritic Arabic word there may be a vast number of transliterations whereas for the diacritic Arabic word it is usually a one-to-one table matching process. To remedy the problem of undiacritised words, a two-way Arabic morphological system (analysis/generation) has been developed to deal with vowelized, semi-vowelized, and non-vowelized Arabic text [EH89]. The system consists of three separate modules: computational lexicon, Arabic grammar model, and an analyser/generator. The output of

the system is either a possibly different morphological analysis for a given word and the corresponding vowelisation, or a set of voweled words generated according to specified morphological information given by the user in a natural Arabic form.

Currently, there are five major transliteration schemes used to catalogue printed Arabic materials [Eld99]; each one of them has its own peculiarities. These standard sets are from the Library of Congress (*LC*), the British Standards Institution (*BSI*), the Encyclopedia of Islam, the International Journal of Middle East Studies (*IJMES*), and the International Standards Organisation (*ISO*). Among other differences between these standard sets, it is unnecessary to have one Latin letter for each Arabic peer, e.g. the first four schemes transliterate ‘شِر’ as ‘*shr*’ where ISO transliterates it as ‘*šr*’. When de-transliterating ‘*shr*’ the five schemes give two different answers: ‘شِر and شَهر’. The result is that unless the same transliteration scheme is used while preparing the transliterated version of all manuscripts it is necessary to employ more than a single query to retrieve all possible matches. Another drawback in using multiple transliteration schemes is that two different Arabic words may have the same transliteration, and this is quite common in transliterating names.

In [AFCB94], a hybrid algorithm which automated the transliteration process of Arabic names in real time using neural networks and knowledge-based systems was presented. A more motivating application was presented in [Vas98], where a list of commands were used to convert romanised data in the American Library Association and Library of Congress (*ALA/LC*), and the British National Bibliography (*BNB*) into Arabic script. The author reported that the most common ambiguity was in the conversion of romanised words ending in ‘*ah*’ or ‘*at*’, which could represent اَ , اِ , or اُ in the original Arabic word.

The transliteration scheme must be capable of retrieving the original Arabic script without any loss of information. Most of the standard sets, mentioned above, except ISO, have a certain ambiguity

concerning this issue. For instance, *th*, *sh* and *dh* may be considered as ث, ش, and ض or as ته, سه, and ده. ISO solved the problem by using a diacritical mark above ‘ج ġ - غ ġ’, or below ‘ح ħ - خ ħ’ the letter. Unfortunately, this inhibits using standard word processing. The Encyclopedia of Islam underlines the two letters each time it represents one character such as th for ث, sh for ش, and dh for ض. In this dissertation, the ISO standard set, which is given in [Lag93] and shown in Table 2.1 is used for transliterating Arabic words.

2.4 Arabic Text Recognition System

The Arabic text recognition system can be broken down into a number of stages: image acquisition, preprocessing, segmentation, feature extraction and classification and recognition. Fig 2.2 illustrates these stages according to their order of occurrence.

The techniques described in this section are illustrated by applying my implementation of these techniques to Arabic manuscript samples listed in Chapter 4.

2.4.1 Image Acquisition

This is the first step in the recognition system. The objective is to acquire the text and transform it into a digitised raster image. Fig 2.3 shows two types of character recognition systems in terms of acquiring their input: on-line recognition systems and off-line recognition systems.

The on-line recognition system recognises the text as it is being written [AU90, Ali97]. The preferred input device is an electronic tablet with a stylus pen. The electronic tablet captures the (x,y) coordinate data of pen-tip movement which typically has a resolution of 10 points /mm, a sampling rate of 100 points/s, and an indication of pen-up and pen-down [WMO92]. The on-line recognition system

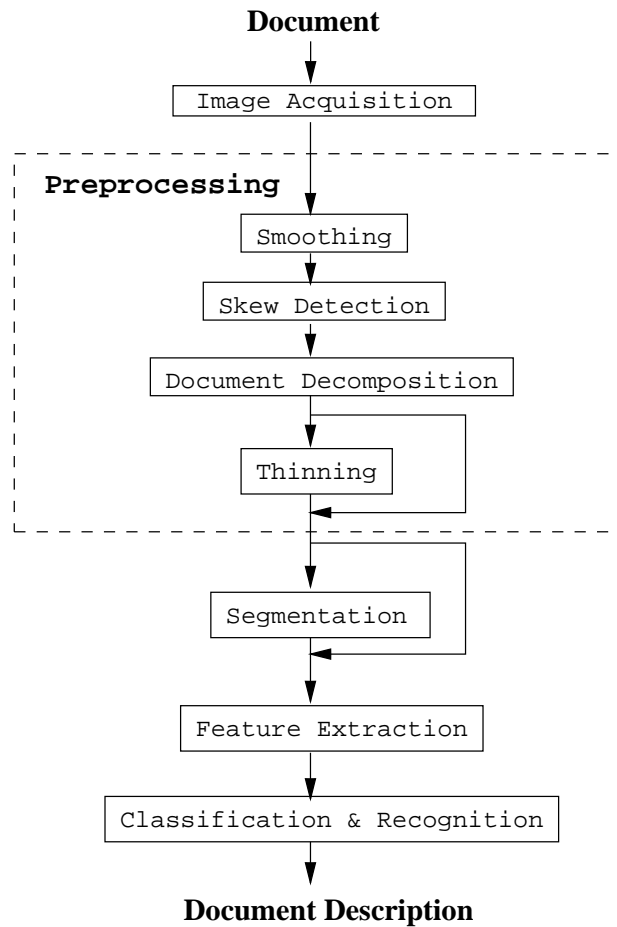


Figure 2.2: A general diagram for the Arabic text recognition system.

has two major advantages: the high-recognition accuracy and the interaction. The first advantage is that on-line recognition captures a character as a set of strokes which are represented by a series of coordinate points. The second advantage is that it is very natural for the user to detect and correct unrecognised characters immediately by verifying the recognition results as they appear. On the other hand, on-line recognition is limited to recognising handwritten text.

The off-line recognition system recognises the text after it has been written or typed. The system may acquire the text using a

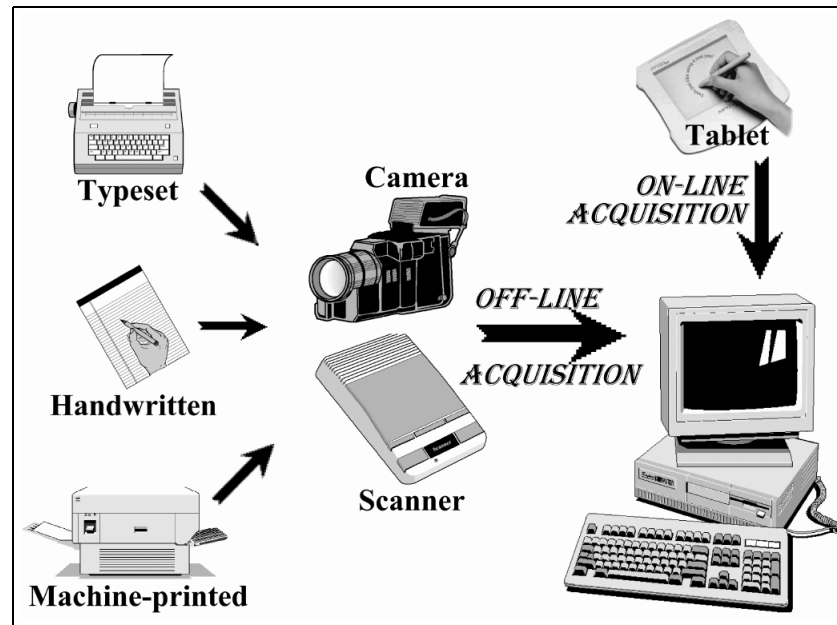


Figure 2.3: Different text acquisition methods.

video camera [AY87, NUN88a, NUN88b, EA90, GUA92] or a scanner [MAG91, AA91, Mah94, YVA96]. The latter is commonly used because it is more convenient, it introduces less noise into the imaging process, extra features such as automatic binarisation 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 recognising manuscripts.

For document management applications the aim is to speed-up the scanning process to maximum speed. The scanner, here, can run for 300 dots per inch (dpi), and they are designed with a high volume document feeder and high-throughput *Small Computer System Interface* (SCSI) that can process 85 pages per minute (ppm).

Lower resolution and poor binarisation can contribute to readability when essential features of characters are deleted or obscured. The resultant 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 the diminishing of contrast, the appearance of salt and pepper noise and false appearance of text by becoming either thinner or thicker than the original document.

Binarisation, or thresholding, is a conversion from a grey-level image to a bilevel image, see Fig 2.4. A bilevel image contains all of the essential information concerning the number, position and shape of objects while containing less information. The simple and straightforward method is to select a threshold value, then all pixels with a grey-level below this threshold will be classified as black, and those above as white.

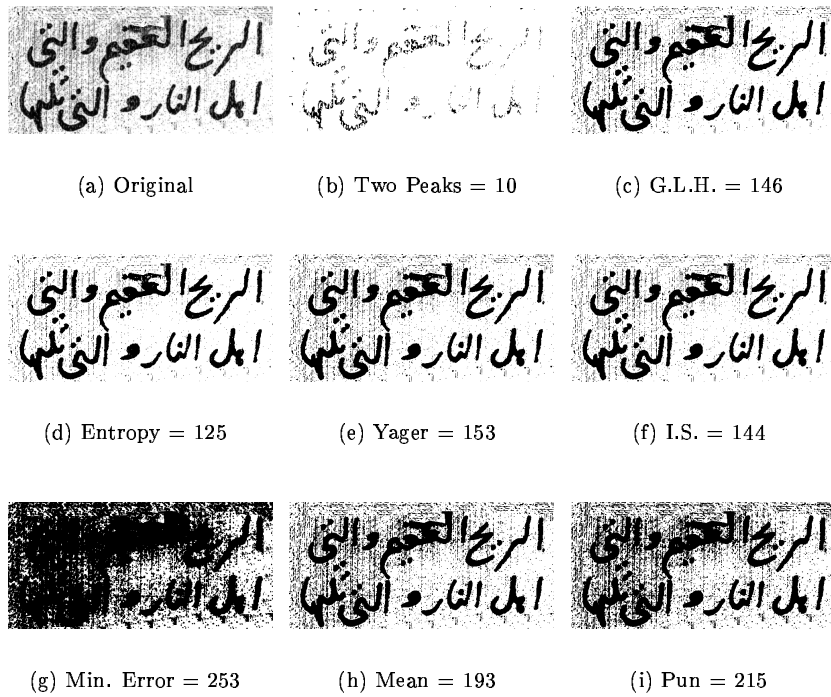


Figure 2.4: Sample results from eight single-threshold selection methods.

The threshold must be determined from the pixel values found in the image, for example the use of the mean grey level in the image as a threshold. Another method is by using the histogram of the grey levels in the image. Given a histogram and the percentage of

black pixels desired, one can determine the number of black pixels by multiplying the percentage by the total number of pixels, then simply count the pixels in histogram bins, starting at bin 0 until the count is greater than or equal to the desired number of black pixels. The threshold is the grey level associated with the last bin counted. Other approaches such as: using edge pixels, iterative selection, the method of grey-level histograms, and using entropy, can also be applied [Pav82, GW92, Par97].

2.4.2 Preprocessing

The OCR system depends upon both the original document quality and the registered image quality. The preprocessing stage attempts to compensate for poor quality originals and/or poor quality scanning. This is achieved by reducing both noise and data variations. All image acquisition processes are subject to noise of some type, therefore there is no ideal situation in which no noise is present. Noise can neither be predicted nor measured accurately from a noisy image. Instead, noise may be characterised by its effect on the image. There are two defined types of noise [Par97]: signal-independent noise and signal-dependent noise. Signal-independent noise adds a random set of grey levels, statistically independent of the image data, to the pixels in the image. In signal-dependent noise the value at each point in the image is a function of the grey level there.

2.4.2.1 Smoothing

This reduces the noise in an image using mathematical morphology operations. Two operations are mainly used, Opening and Closing. 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. Opening applies an erosion op-

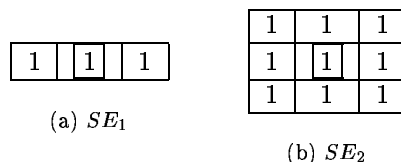


Figure 2.5: Two structuring elements. The origin is surrounded by a frame.

eration immediately followed by a dilation operation using the same Structuring Element, see Fig 2.5. Closing applies a dilation operation immediately followed by an erosion operation, again, using the same structuring element.

In Dilation a small area around a pixel is set to a given pattern. This can be mathematically represented as

$$A \oplus B = \{c | c = a + b, \quad a \in A, b \in B\} \quad (2.1)$$

A represents the image being dilated, and B is a second set of pixels called a structuring element.

The erosion of image A by structuring element B can be defined as

$$A \ominus B = \{c | (B)_c \subseteq A\} \quad (2.2)$$

It is the set of all pixels c such that the structuring element B translated by c corresponds to a set of black pixels in A. Fig 2.6 illustrates the results of opening and closing using the two structuring elements shown in Fig 2.5.

Another approach to reducing salt-and-pepper noise is by applying the median filter, see Fig 2.7. This is a small window, $N \times M$, 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. The median filter is slow, requiring not only a pass through the image of the window, but also needing the pixels' values in that window to find the median. Furthermore, it may reduce the contrast of the edges and close small gaps between different objects in the image.

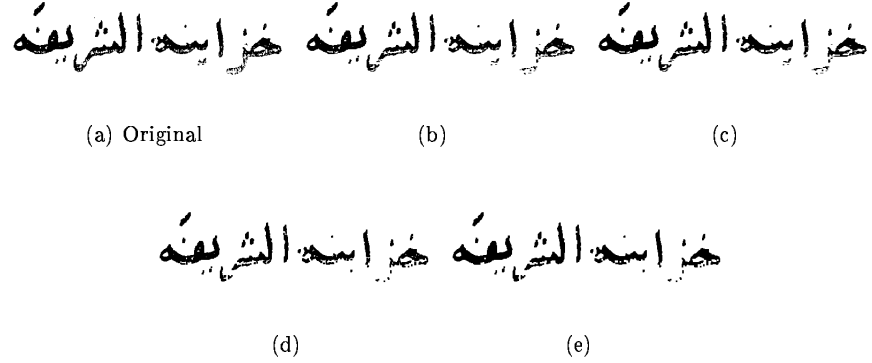


Figure 2.6: Results of opening and closing using structuring elements in Fig 2.5, (a) the original text image, (b) opening (a) with SE_1 , (c) closing (b) with SE_1 , (d) opening (a) with SE_2 , and (e) closing (d) with SE_2

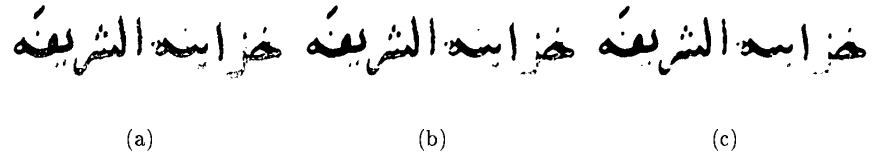


Figure 2.7: Results of removing noise from the original text image in Fig 2.6 using median filter, window size is (a) 3×3 , (b) 5×5 , and (c) 7×7

At times, writing can be on a graph paper thus scanning the original paper will produce an image disturbed by structured or periodical noise. In Fig 2.8, The Fourier Transform [GW92] is used to determine where the peak frequencies are. The noise will correspond to one of these and can be virtually eliminated by clearing those regions in the frequency domain image that correspond to the noise frequencies, and then back-transforming it into a regular image.

Mahmoud [Mah94] implemented a statistical smoothing algorithm which modifies each pixel according to its initial value and to the values of its 8-neighbours.

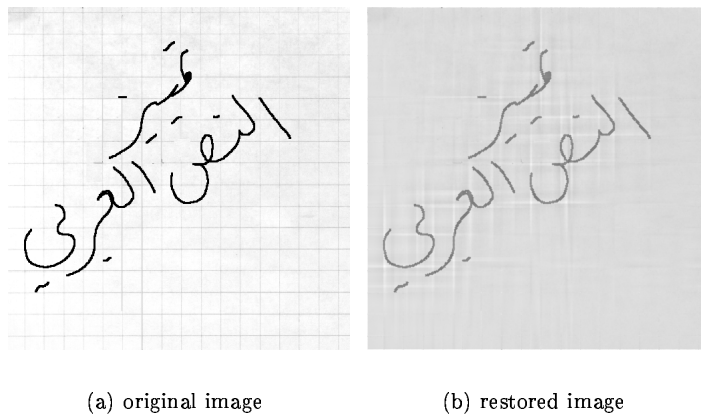


Figure 2.8: Removal of grid lines using a notch filter.

2.4.2.2 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.

The common, and perhaps the most efficient approach to estimate the skew angle is to use the Hough Transform [FS93, Par97, JHF97], refer to Fig 2.9. The Hough Transform is a method for detecting straight lines in a raster image. Each black pixel in the image may be represented by infinite straight lines that pass through this pixel and satisfy the following equation:

$$y = mx + c \quad (2.3)$$

where (x, y) are the co-ordinates of the pixel, m is the slope of the line and c is the intersection of the line with the Y axis.

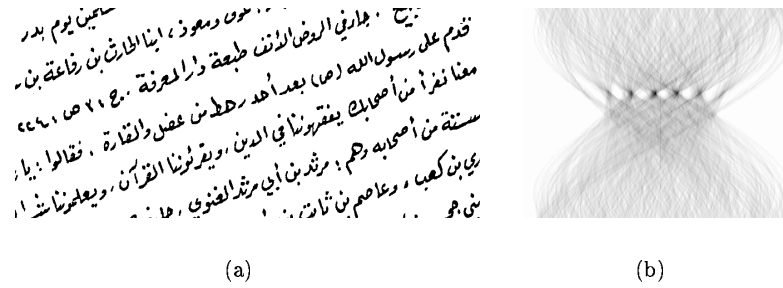


Figure 2.9: Skew detection, (a) document image rotated 18 degrees to the left, (b) the Hough Transform of (a).

Another approach to estimating a skew angle is based on using bounding boxes of connected components [Bai92]. However, implementing this approach to Arabic text image may be misled by dots and diacritics located above and below the word characters.

2.4.2.3 Document decomposition

A document image is a visual representation of a printed page. Typically, this page consists of blocks of text that are interspersed with tables and figures. Methods of deriving the blocks can take advantage of the fact that the structural elements of a document are generally laid down in rectangular blocks aligned parallel to the horizontal and vertical axes of the page. The document decomposition and structural analysis task can be divided into three phases [SLG⁺92]: Phase one consists of block segmentation where the document is decomposed into several rectangular blocks. Each block is a homogeneous entity containing one of the following: a text, an image, a diagram or a table. Phase two consists of block classification. Each block is assigned a label (title, regular text, picture, table, etc) using properties of individual blocks from phase one. Phase three consists of a logical grouping and ordering of the blocks. For OCR the concentration is focused on text blocks.

Work on Arabic has been limited to text documents, thus the

تأثر فريق سدوس في جولة بثلاثة
 صالح وريكارديو ومحمد دابو
 مستواه المعروف وإن كانت خبرة
 ذلك جاليا في وقت مبكر وقد استغل
 أهدافه الثلاثة التي جاءت نتاجا
 لأهلي الذي خاض اللقاء على أرضه

تحت إشراف المدرب من هراسه مختلف السجلات فريق سدوس في حرمه بملعبه
 أهداف بطلان بطلان حلف بتمثيلها حرمه بملعبه وريكارديو ومحمد دابو ،
 ورجح الفوز الكبير أنه لن يتمكن من لقاء مستواه المعروف وإن كانت خبرة
 سدوس لن تكون بالصورة الجيدة ، أم حلفه بثلث حلفه في وقت مبكر وقد استغل
 أهدافه الثلاثة حرمه بتمثيلها من بتمثيل أهدافه الثلاثة التي جاءت نتاجا
 بتمثيلها بطلان حرمه بتمثيلها أهدافه الثلاثة حرمه بتمثيلها حلفه بتمثيلها
 وريكارديو ومحمد دابو

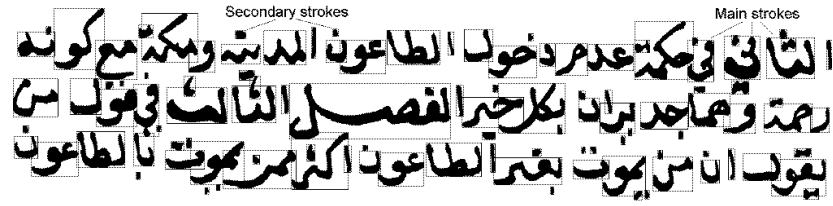


Figure 2.11: Document connected components.

notation of *document decomposition* means the separation of text lines and the segmentation of words and sub-words. 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 [AM86, AM89]. An alternative approach [FA94] is to use the horizontal projection and look for the pixel lines that have a density of zero, then consider that every text line is situated between two blocks of zero density pixel lines, see Fig 2.10. This method is enhanced by identifying the lines of pixels that have the largest density in the text [FA98]. The upper and lower parts are then analysed with respect to these lines.

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 [AM86, AM89, AB94]. 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 as shown in Fig 2.10.

Another attempt at decomposing the Arabic script into words is based on the *Connected Components* [Par97] of that script, shown in Fig 2.11. Abuhaiba et al [AHD98] automated the process of combining secondary strokes with appropriate main strokes. This process depends on finding a perfect bipartite graph with minimum cost, and it is known as *Assignment Problem*.

2.4.2.4 Slant normalisation

This problem may be clearly seen in handwritten words, although machine-printed words with italic fonts suffer from the same problem. Kim and Govindaraju [KG96, KG97] proposed an algorithm to correct slant angle in which vertical and near vertical lines are extracted by tracing chain code components using a pair of one dimensional filters, each being an eight element array of different weights. A convolution operation between the filter and five consecutive components was applied by sliding the filter one component at each iteration.

2.4.2.5 Thinning and skeletonisation

These are the operations that produce the skeleton. A skeleton is presumed to represent the shape of the object in a relatively small number of pixels, all of which are structural and have semi-equal distance from two or more contour points.

Thinning algorithms may be classified into parallel and sequential. The parallel algorithms [GH92, JC92] operate on all pixels simultaneously. In contrast, the sequential algorithms [YT90, Lar98] examine pixels and transform them depending on the preceding processed results. The approach in both cases is to remove the boundary pixels of the character that are neither essential for preserving the connectivity of the pattern, nor for representing any significant geometrical feature of the pattern. The process converges when the connected skeleton does not change or vanish even if the iteration continues, see Fig 2.12. The literature contains many thinning algorithms most of which are susceptible to noise in that the generated skeletons are sensitive to even small variations in the input pattern [Jai89, LLS92, GW92, RW96, Par97].

Most of the existing thinning algorithms operate by iteratively stripping contour pixels. The main problem associated with this approach is that the computation speed is very low due to its layer-to-layer stripping nature. An alternative is to implement a vectorisation-based algorithm [FCW98] which operates directly on the run length

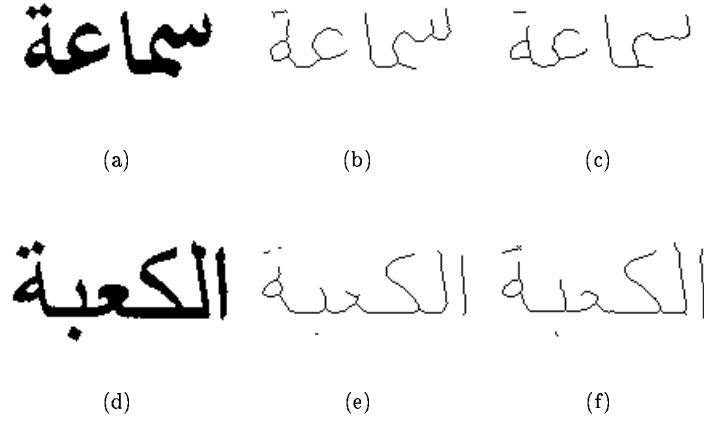


Figure 2.12: Sample results of a number of thinning algorithms.

encoding of a binary image. Another alternative is driven by Euclidean distance mapping [MFV98] which guarantees the invariance under isometric transformation of the results. The computational complexity of these algorithms have been considerably reduced with respect to other thinning algorithms. Moreover, the results generated by these algorithms are better than those generated by traditional thinning algorithms.

Mahmoud et al [MAG91] proposed a new algorithm for skeletonisation of isolated Arabic characters that was based on clustering the character image. The skeleton was then generated after finding the adjacent matrix of different clusters. Finally, they refined the skeleton by removing insignificant vertices.

2.4.3 Segmentation

By now the document image has been enhanced, decomposed into blocks and text lines. In addition, words and sub-words have been extracted from text blocks. At this stage, the system focuses on how it prepares the words and/or the sub-words for the feature extraction. As mentioned previously, cursiveness is the main obstacle

facing any Arabic text recognition system whether it is on-line or off-line.

Segmenting on-line Arabic handwriting [AM82, ES87, ES89, EE89, EE90a, EE90b] is much simpler than segmenting off-line machine-printed Arabic words. This simplicity motivated the work carried out in developing an algorithm to restore the temporal information in off-line Arabic handwriting so that on-line recognition systems may be used [AA93a].

Arabic text recognition systems can be categorised, relative to their approach in tackling word segmentation, into segmentation-based systems and segmentation-free systems.

2.4.3.1 Segmentation-based systems

These can be divided into four categories:

1. *Isolated/pre-segmented characters:* Researchers here recognise numerals [AH89, SYS99], isolated characters [NST84, AAF96], or assume that these characters result from a reliable segmentation algorithm [AKHM80, EE90a, Mah94, AM95a]. These systems are not practical except if we consider mathematical formulas or the indexing of diagrams.
2. *Segmenting a word into characters:* This is the first approach used for segmentation [Ami85, SY85, Ami88]. The system attempts to segment a word into its characters then recognise each character separately. The reason behind the emergence of this approach is the simplicity of the recognition afterward since the cursiveness obstacle is not present and the problem is now similar to Latin OCR.

Some research [Ami88, AM89, AA91] has proposed segmentation algorithms that are based on the vertical projection of the word image. The connectivity points show the least sum of the average summation over all columns. This results in a number of segments which are then connected together to form the basic shape of the character, see Fig 2.13.

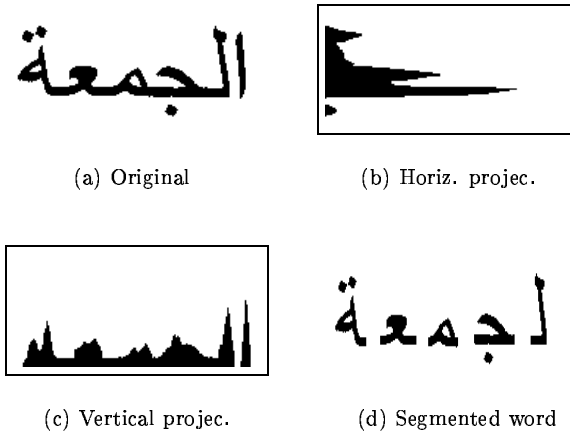


Figure 2.13: Segmenting a word into its characters

Another technique uses a two level segmentation scheme [AU92]. After segmenting a word into its characters using horizontal and vertical projections, a lower level of segmentation is applied to isolate the dots and zigzag-like shapes.

In [YVA96] the segmentation stage partitions a sub-word into its character segments. The algorithm is based on extracting a subset of key feature segments in the sub-word and identifying cut points in each segment. The subset of the key feature segments is obtained by tracing the contour in the clockwise direction. Dots and diacritics are not considered during segmentation. This type of segmentation is a cause of recognition errors and hence a low recognition rate.

3. *Segmenting a word into primitives:* This segments a sub-word or connected component into symbols where each symbol may represent a character, a ligature, or possibly a fraction of a character, see Fig 2.14.

Abdelazim and Hashish [AH88] calculated the vertical projection for each column, then obtained significant primitives by traversing the resulting curve with a selected threshold value.

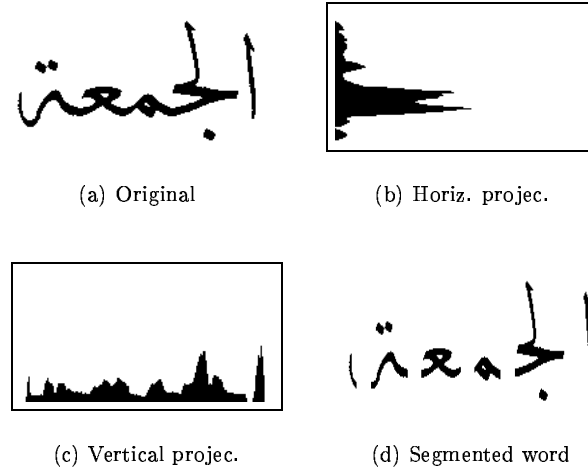


Figure 2.14: Segmenting a word into primitives.

The baseline represents an important feature for Arabic writing. Parhami and Taraghi [PT81] identified a sequence of connection points on the baseline of Farsi script, which uses the Arabic alphabet. The connection point is where the baseline changes from/to its normal thickness. The alphabet was then divided into three major groups. Again the baseline was utilised in an indirect manner when Tolba and Shaddad [TS90] slid a window along the direction of writing and at each instant a segmentation parameter was calculated to match the content of the window with a predefined set of primitives. If the segmentation was less than a certain threshold the region was marked as a “Silence Region”.

Motawa et al [MAS99] applied Morphological operations, *opening* and *closing*, to a word image to allocate *singularities* and *regularities*. Singularities represent the start, the end, or the transition to another character. Regularities contain the information required for connecting a character to its successor. Accordingly, these regularities are excellent candidates for segmentation.

Some researchers extracted the dots and the diacritics in advance. The connected components were then segmented into meta character glyphs [Has94], principal strokes [GUA92], or character segments [All94].

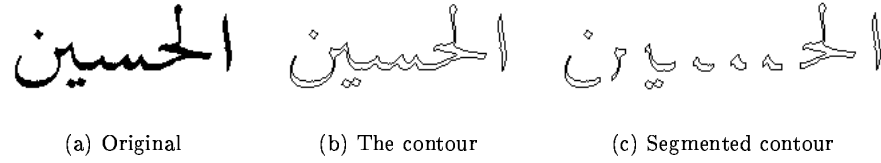


Figure 2.15: Segmenting the contour of a word.

The set of boundary pixels or the contour includes important information of an object which can be segmented into primitives [EAIK94], see Fig 2.15. This may be done by finding points on the contour where there is a transition from a column, which has all its black pixels within the baseline boundaries, to another column, which does not [All95]. Another approach is completely based on tracing the outer contour [EG88, AA96] of a given word and calculating the distance between the extreme points of the intersection of the contour with a vertical line.

Kavianifar and Amin [KA99] divided the contour into three classes: main body or stroke, complementary character, and noise. They set two thresholds to help find the equivalent contour class based on the contour length.

A skeleton is a compact representation of a word image. It can be traced in the same way that Arabic writing is taught to segment the word into a stroke sequence [AY87, GU94], structural features such as loops and branches [ZTF91], or principal and secondary strokes [YVA96]. Alternatively, a character skeleton can be converted to a tree structure and each character is then represented by a single fuzzy constrained character graph model [AMG94].

Al-Muallim and Yamagushi [AY87] segmented a word into strokes. The extraction of a stroke was made by finding out its start point and then following the curve to a point which was inferred to be the stroke endpoint.

4. *Integration of recognition and segmentation:* This claims that the procedure resembles, to a great extent, the human recognition process. The segmentation here is performed after recognition. The approach is to scan the word starting from the far right and at each step either cluster a column to one of the codebook entries [AMSH90] or calculate accumulative moment invariants [REK88, ERK90]. The system is not always able to recognise all characters which implied that all succeeding characters in that sub-word would not be processed. To remedy this drawback, El-Dabi et al [ERK90] developed a backup scanning algorithm that was triggered when such a blockage happened. Mosfeq [Mos96] presented a segmentation algorithm as a centering operation involving a focus of attention. A character was correctly segmented if it appeared in the centre of a large window regardless of what else appeared in that window.

2.4.3.2 Segmentation-free systems

This scheme of text recognition is motivated by discoveries in psychological studies of the human reading process [ZS94]. It attempts to recognise the whole representation of a word without trying to segment and recognise characters or primitives individually. This approach was originally introduced for speech recognition [RJ93].

One approach of the word level Arabic recognition was to analyse the word shape with a unique vector of features, then this feature vector might be matched against a database of analogous feature vectors [ETV96], or represented in attribute/value form to an inductive learning system [Ami98b].

Another approach implemented the morphological Hit-or-Miss Transform [KD94] which was based on marking the location at which a structuring element fits within a pixel set corresponding to a shape

of interest and another structuring element lies outside the pixel set. Shape primitives located on the whole page were then combined into characters [AH96].

A third approach was based on choosing a text line as the major unit for training and recognition [MLR⁺96, MSLB98, BSM99]. When a page was decomposed into text lines the horizontal position along each line was selected as an independent variable. Hence, a text line was scanned from right-to-left and at each horizontal position a set of features was extracted from a narrow vertical strip. The system was based on hidden Markov models where each character was represented by a separate model. The output was a sequence of characters that had the highest likelihood.

2.4.4 Feature Extraction

A feature is a measurement made on a glyph, and combining it into a vector is a simple way of collating multiple measurements. Ideally, the features extracted from an image capture the essential characteristics of the character or the word by filtering out all attributes which make a character/word in one font different from the same character/word in another. At the same time they preserve the properties that make one character/word different from another character/word. Feature types can be categorised into:

2.4.4.1 Structural features

Structural features are the most popular features investigated by the researchers [GS90]. Structural features describe geometrical and topological characteristics of a pattern by representing its global and local properties [GW92, Sim92, Par97].

The extracted structural features depend on the category of the pattern to be classified. For Arabic character, word and text recognition, the features include strokes and bays in various directions, endpoints, intersection of line segments, loops, stroke positions relative to the baseline, dots and their positions relative to the baseline, and zigzag [AH89, AA94, All94, GU94, FA94, AAF96, Ami98b].

Feature space can be divided into more than one independent division and the extracted features can be different in each division [YVA96]. Sometimes, based on the preliminary features, a character/word image may be assigned to a certain group where further feature extraction is carried out [AY87, ESEA91, ZTF91].

Structural features can tolerate distortion and variations in writing (multi-fonts and handwriting), however, they are not easy to extract.

At times, it is very important to locate pixels with certain properties on the skeleton of a word to act as delimiters helping extracting strokes [AY87, GU94, AAF96]. Alternatively, the word image may be scanned vertically, column by column, in order to produce a structural feature vector for each column [All94].

2.4.4.2 Statistical features

Statistical features are derived from the statistical distribution of pixels and describe the characteristic measurements of a pattern. These include zoning [Kon94, MSLB98, BSM99], characteristic loci [AMSH90], the ratio of pixel distribution between two parts of the image [MEF87, Bou96, FA98] and moments [REK88, EA90, AB94, San96].

In [MLR⁺96, MSLB98, BSM99], the text line was partitioned into a sequence of narrow vertical overlapping frames with a width that was a small fraction of the height of the line. Each frame was divided into 20 equal overlapping cells. Features extracted from each cell were: the intensity of a cell, the vertical and horizontal derivative of intensity, and the local slope and correlation across a window of two cells. The frame could have a one-pixel width as in [AMSH90]. The system extracted a feature vector from each column of pixels in the word image where each feature represented the length of black/white pixel run.

Moment invariants refer to certain functions of moments [Jai89], which are invariant to geometric transformations such as translation, scaling and rotation [Hu62, GW92]. Moment invariants are sensitive

to any change and multi-font recognition. To remedy this, Al-Khatib et al [EAIK94] integrated three feature extraction methods: moment invariants, border transitions and perimeter-area ratio.

Generally speaking, statistical features are easy to extract nonetheless they may be misleading due to a fraction of noise brought forth haphazardly according to the binarisation process.

2.4.4.3 Global transformation

Global transformation technique reduces the dimensionality of the feature vector and provides feature invariants to global deformation like translation, dilation and rotation. Zaki et al [ZEEE86] used the projection transform to convert a character image of order $M \times N$ into a projection vector of order $M+N$. Amin et al [AM86, AA91] implemented projection transform to represent the character image as a string of primitives.

Fourier descriptors (FDs) [ZR72] were successfully implemented for Arabic OCR [EG88, Mah94]. FDs use the co-ordinates of the contour pixels so the character's closed boundary can be represented by a periodic function. Mahmoud [Mah94] integrated another global transformation technique with FDs which is the boundary line encoding technique. This technique is based on tracing the contour of the character to generate directions, direction lengths and curvature features.

Other researchers applied a chain-code transformation. Features extracted from the freeman code [Fre61] may be described as directional vectors [AM89, Mah94] and they may be combined with other features [SY85].

2.4.5 Classification and Recognition

A major task after feature extraction is to classify the object into one of several categories. There are a number of various classification techniques applied in text recognition.

2.4.5.1 Minimum distance classifier

Given K different classes where each class is characterised by a feature vector prototype, the problem is to assign an input feature vector to one of these classes according to a predefined discriminant function. The features can be geometrical [PT81], statistical [ERK90, Mah94] or structural [SY85].

Abuhaiba and Mahmoud [AM95a] designed a set of fuzzy constrained character graph models, then an input character, which has been converted to a tree structure, was assigned to the character model which had the maximum degree of acceptance. This process can be repeated on more than one level as presented in [ESEA91] where a character was first assigned to the nearest group among the 96 native groups. The character was then recognised as one of the characters in the native group based on the minimum distance between the input character and the template character. The common distance measure is the Euclidean distance [Pav82], but Hamming distance and Ascher et al scores were also applied to recognise Arabic characters [NUE88]. K-means clustering algorithm [RJ93] was implemented successfully in Arabic OCR for one-dimensional [ZEEE86] and two-dimensional [AH88] feature vectors.

2.4.5.2 Decision tree classifier

This classifier splits the N -dimensional feature space into unique regions by means of a sequential method. The algorithm is such that every class need not be tested to arrive at a decision. This becomes advantageous when the number of classes is very large, as in [Ami88, AM89, EA90, AA91], when the dictionary was composed of a tree and the nodes were labelled with character names. Each node of the dictionary was associated with a Boolean variable indicating if the path joining the root to the terminal node corresponded effectively to an existing word. If during the ongoing sequential identification process several models are candidates then the last mentioned attribute is calculated to make the final decision [ZTF91]. As in the previous category, classification here can be a

two-step process [AY87, EE90a, GUA92, GU94]. In the first step, an input character is assigned to one of the main groups according to some syntactic rules. Then, and relative to a more detailed feature vector, the input character is matched with one of the group members.

Abdelazim and Hashish [AH88] implemented a template correlation matching and a tree classifier to discriminate among primitives in the same cluster group. The recognition process of an unknown pattern was propagated sequentially by following a path through the decision tree from the root node to the leaf node. The leaf node was either labelled with an identified primitive or could be a reject node. It was found that the tree method was faster than a template matching against idealised reference patterns.

Amin [Ami98b] used the *C4.5* learning algorithm to create decision trees to represent classification rules. A node in a tree represented a test on a particular attribute, thus when an object reached a leaf node, it was classified according to the name of that leaf node.

2.4.5.3 Statistical classifier

This classifier assumes that different classes and the feature vector have an underlying joint probability. One approach is to use the Bayes classifier [AU92]. The Bayes classifier minimises the total average loss in assigning an unknown pattern to one of the possible classes. The probability density function can be cumulative [MEF87] therefore, at the end, the assignment is to that class with majority samples.

Al-Badr and Haralick [AH94, AH96] implemented a three-step recognition process: first they found instances of a set of shape primitives on a text image, then they took the detected primitives of a word and hypothesised a number of alternative strings as the recognition of the word. The choice would be on the one with the maximum posteriori probability. Finally, the probability of a match was computed between the symbol model and the word image.

Hidden Markov Models are statistical models which have been

found extremely efficient for a wide spectrum of applications, especially speech processing [JR85, EH88]. This success has motivated researchers to implement HMMs in character recognition. Abdelazim and Hashish [AH89] first applied HMMs to recognise Hindu numerals; ‘٠ ١ ٢ ٣ ٤ ٥ ٦ ٧ ٨ ٩ - 0123456789’. Each numeral was represented with a separate HMM. The observation sequence was passed to all the ten models and assigned to the numeral with the highest model probability of the observation sequence $P(O|\lambda)$. Recently, HMMs have been implemented to recognise Arabic text [All95, MLR⁺96, MSLB98, BSM99]. The approach was to represent each character by a left-to-right HMM model [RJ86].

In [MSLB98] and [BSM99], each character model had 14 states. The model for a word was then a concatenation of the different character models. The training algorithm was similar to that which was used in the Byblos speech recognition system [MLR⁺96]. They used the ground truth transcriptions which specified the sequence of characters given the sequence of input features, the lexicon and the language model.

2.4.5.4 Neural network classifier

OCR is one of the most successful applications that has been proposed for neural networks. A Neural Network (NN) [Pao89] is a non-linear system which may be characterised according to a particular network topology. This topology is decided by the characteristics of the neurons and the learning methodology. There are three main advantages behind implementing NNs in OCR: NNs have faster development times, they have an ability to automatically take into account the peculiarities of different writing/printing styles, and they can be run on parallel processors. On the other hand, introducing a new shape to the NN requires that the network be retrained, or even worse, that the network be trained to a different architecture. An intensive work can be found in the subject of Arabic OCR using neural networks [AB94, HB94, Has94, AAF96, AJAA96, Mos96, San96, Bou96, AM97, FA98]. NNs can simply cluster the feature vectors

in the feature space [AB94, AAF96, Bou96, San96, AM97, FA98] or they can integrate feature extraction and classification stages by classifying characters directly from images [HB94, Mos96, Aud99]. NNs were also applied to recognise Arabic words on-line [Ala96].

Generally speaking, the common architecture of NNs 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 govern the variance of samples that can be correctly recognised by this NN [EAIK94, SYS99].

In [San96], two different NN architectures were employed: architecture one was a network of 13 input nodes, 20 hidden nodes and 28 output nodes. In architecture two, the letters were divided into six groups according to their similarities, e.g. غ, ع, ج, خ, ح all fall in one group. The problem was solved in two stages. In stage one, the letter was classified into one of the six groups using a single NN with 13 input nodes and six output nodes. In stage two, the letter was classified by one of six different NNs, giving one network per group.

2.5 Arabic OCR Software

Currently, there are four different software packages for recognising Arabic script. These packages are: TextPert 3.7 Arabic produced by CTA, ICRA 4.0 produced by Arab Scientific Software & Engineering Technologies, OmniPage produced by Caere Corporation, and Al-Qari' al-Ali 2.0 produced by al-Alamiah Software Company. All these packages are only used for recognising typeset and typewritten Arabic script. A number of critical evaluations for these packages can be either obtained from the Internet [BZ94, Hoo95, Bel95, Zem96, Hoo96a] or found on the proceedings of ICEMCO'96 [BMZ96, Hoo96b]. Here they are recounted briefly:

1. *TextPert*: This runs on the Macintosh Arabic system, and it is easy to use. However, training new fonts is not possible. The

recognition rate was approaching acceptable standards when the program was tested on very good simple texts, however it virtually recognised nothing with more complicated fonts [BZ94, Bel95]. As a consequence, until the training feature is introduced to the software, its usage will be limited to those who only want to scan certain kinds of computer-generated documents.

2. *ICRA*: This runs under Microsoft Windows Arabic, and every typeface needs to be learned. The training process takes about one hour for each typeface. An experiment training this software with a number of Arabic magazines [Hoo95] showed that using these texts in their ordinary size gave disappointing results. Enlarging the text about 20% improved the recognition rate which ranged between 90% and 99.7%.
3. *OmniPage*: This runs under standard Arabic Windows 3.1 and Arabic Windows for Workgroups 3.11, without customising. It integrates with standard Arabic word processors including Microsoft Word, Microsoft Write, and Accent. OmniPage does not require any training of fonts [Zem96]. This is not essentially an advantage, since the program commits the same systematic error repeatedly without being able to learn from its failure.
4. *Al-Qari' al-Ali*: This program was first developed by Dr. Rezvan of the Russian Academy of Science at the beginning of the 1990s [BMZ96]. It is a segmentation-based system which combines vector and bitmap analysis. The program is delivered with a standard set of modern computer fonts which can be recognised automatically. The main problem with this program is the considerable amount of time it takes to train for new fonts, especially typeset fonts with many ligatures. Versions 1.0 and 1.1 of this program run on al-Nawafidh al-Arabiya which is an equivalent environment to Microsoft Windows but in Arabic, and version 2.0 runs under Microsoft Arabic Windows.

2.6 Summary

In this chapter I have reviewed the previous work done in the field of Arabic text recognition and transliteration. I have first discussed the characteristics of Arabic writing that influence the process of recognition. Then I have presented a general model for an Arabic text recognition system. This model was divided into five stages: image acquisition, preprocessing, segmentation, feature extraction, classification and recognition. Research methods at each stage were discussed and analysed. I have also discussed different standard sets used for transliterating Arabic text. Finally, I have reviewed the performance of four existing Arabic text recognition programs.

CHAPTER 3

METHODOLOGY: USEFUL TECHNIQUES

3.1 Introduction

A character recognition system is composed of a collection of algorithms drawn from a wide variety of disciplines such as signal processing and statistical pattern recognition. Different recognisers rely to varying degrees on the signal processing front end to convert the character/word image to some form of parametric representation for further analysis and processing. This chapter presents the techniques of *Vector Quantisation* (VQ) and *Hidden Markov Models* (HMMs). VQ is a procedure for encoding feature vectors, extracted from the character/word images, in a finite codebook of symbols. This technique significantly reduces computation in the recognition process. HMM is a statistical method of characterising features extracted from character/word images. The underlying assumption of the HMM is that the character/word image can be well characterised as a parametric random process, and that the parameters of the stochastic process can be determined in a precise, well-defined manner.

3.2 Vector Quantisation

Any analogue quantity has to be converted into a discrete form proportional to its amplitude before being processed by a digital system. The straightforward conversion process may result in a very high rate of discrete data. The requirement then is to convert a stream of analogue or a very high rate of discrete data into a stream of relatively low rate data for storage in digital memory.

Vector Quantisation (VQ) [Gra89] produces an approximation of the distribution of a signal class in a codebook. Each incoming signal is mapped to the nearest codebook vector, and the index of that vector is processed instead of the original signal. This method reduces the storage required for data analysis, it reduces the necessary computation to determine the similarity between various vectors. On the other hand, coding the incoming signal using VQ will inherit any distortion in representing the actual analysis vector. This may lead to not assigning the incoming signal to that vector which best represents the signal. This is quite common in character recognition with poor quality input. Another disadvantage of using the VQ codebook is that the storage required for codebook vectors may be costly, particularly for codebook sizes of 1000 and more.

There are two main VQ paradigms: the vector quantiser without memory, and the vector quantiser with memory. The vector quantiser without memory encodes successive input vectors without depending on any previous encoder input vectors or their coded outputs. In speech processing, this type is suitable for utterances of limited duration in order to provide efficient discrimination and simple computation. The vector quantiser with memory is a decoder with feedback. This can be used when utterances are excessively lengthy. Therefore, in order to capture the temporal characteristics of these utterances, a different codebook is used for each input vector.

In this section, the discussion is focused on the vector quantiser without memory. I shall inspect different elements essential to build-

ing a VQ codebook and implementing a VQ procedure which are: a training set, a distance measure, centroid computation, and a classification method.

3.2.1 Training set

A large training set is the corner stone to building a VQ codebook. Assume an n -dimensional VQ which assigns each of the U input vectors, $\mathbf{x} = (x_0, x_1, \dots, x_{n-1})$, to one of its M symbols, where M represents the codebook size. This process is based on a predefined distance measure. 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 quantisation error in representing the information with a fixed-size codebook.

3.2.2 Distance measure

This is the cost of reproducing an input vector \mathbf{x} as one of the M codebook vectors. It can also be referred to as the distance between two vectors \mathbf{x} and $\hat{\mathbf{x}}$

$$d(\mathbf{x}, \hat{\mathbf{x}}) = \begin{cases} = 0 & \text{if } \mathbf{x} = \hat{\mathbf{x}} \\ > 0 & \text{Otherwise} \end{cases} \quad (3.1)$$

The main goal behind the VQ algorithm is to determine the optimum set of M vectors such that the average distortion in replacing each training set vectors by the closest entry in the codebook is minimum. This can be phrased mathematically as finding the set $\hat{\boldsymbol{\mu}}_m$ for a given M such that

$$D_m = \min_{\hat{\boldsymbol{\mu}}_m} \frac{1}{U} \sum_{i=1}^U \min_{1 \leq m \leq M} [d(\hat{\boldsymbol{\mu}}_m, \mathbf{x}_i)] \quad (3.2)$$

is minimum, where D_m is the average distortion of the vector quantiser.

To define a distance function d on the vector space χ as a real-valued function on the Cartesian product $\chi \times \chi$, this function should fulfill certain conditions [RJ93]

1. $0 \leq d(\mathbf{x}, \mathbf{y}) < \infty \quad \forall \mathbf{x}, \mathbf{y} \in \chi$
2. $d(\mathbf{x}, \mathbf{y}) = 0 \quad \iff \mathbf{x} = \mathbf{y}.$
3. $d(\mathbf{x}, \mathbf{y}) = d(\mathbf{y}, \mathbf{x}) \quad \forall \mathbf{x}, \mathbf{y} \in \chi$
4. $d(\mathbf{x}, \mathbf{y}) \leq d(\mathbf{x}, \mathbf{z}) + d(\mathbf{y}, \mathbf{z}) \quad \forall \mathbf{x}, \mathbf{y}, \mathbf{z} \in \chi$

The distance measure to be used in this research is the un-weighted Euclidean distance measure. This is the simplest and the most common used distortion measure. It is mathematically represented as

$$\begin{aligned}
 d(\mathbf{x}, \hat{\boldsymbol{\mu}}) &= \|\mathbf{x} - \hat{\boldsymbol{\mu}}\| \\
 &= \sqrt{\sum_{i=0}^{n-1} (x_i - \hat{\mu}_i)^2}
 \end{aligned} \tag{3.3}$$

3.2.3 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, $\hat{\boldsymbol{\mu}}$. 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 ordinary vector sum of all input vectors encoded into a given codebook symbol. This is calculated as follows

$$\begin{aligned}
 \hat{\boldsymbol{\mu}}_m &= \frac{1}{N_m} \sum_{i=1}^{N_m} \mathbf{x}_i \\
 \hat{\mu}_{mj} &= \frac{1}{N_m} \sum_{i=1}^{N_m} x_{ij}
 \end{aligned} \tag{3.4}$$

where m represents the current codebook symbol, and N_m is the number of vectors assigned for the codebook symbol m .

3.2.4 Classification Algorithm

This is also called a clustering algorithm, and it automatically separates the training vectors into groups representing VQ codebook symbols. There is no systematic approach to defining the most suitable number of classes in the codebook. The process of clustering is part of a wider group of techniques referred to as unsupervised learning. 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 are two main clustering algorithms: the dynamic clustering algorithms and the hierarchical clustering algorithms. In the dynamic clustering algorithms, a fixed number of clusters or classes is used. At each iteration, training vectors are reassigned according to a predefined distortion measure until a stable partitioning of the vectors is achieved. In hierarchical clustering algorithms, each vector is initially a separate cluster, then at each step of the algorithm, the two most similar clusters, based again on a predefined distortion measure, are merged until the desired number of clusters is achieved.

3.2.4.1 K-means clustering

This is a dynamic clustering algorithm, and it is widely used particularly in speech processing. K refers to the number of classes in the codebook. The procedure is straightforward: training vectors are continuously reassigned to clusters, and the cluster centroids updated until no further reassignment is needed. The flow chart of this procedure is shown in Fig 3.1. The algorithm is decomposed into two stages:

- *Initialisation*
Choose M vectors randomly out of the training vector set. Let these M vectors be the initial centroids of the codebook symbols.

- *Recursion*

1. For each training vector use Eq. 3.3 to assign the closest codebook entry based on the minimum Euclidean distance measure. This step is known as the nearest-neighbour search.
2. For each codebook entry use Eq. 3.4 to calculate the centroid of all training vectors that fall in this class.
3. Check if the average distance falls below a predefined threshold. If yes proceed to the following step otherwise repeat step 1.
4. The VQ codebook is trained and ready for usage.

Another version of the K-means clustering is to calculate the centroid after each assignment of one of the training vectors. This may be referred to as a running centroid.

3.3 Hidden Markov Models

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 characterises the occurrence of the observed output. The domain of possible signal models can be dichotomised into: the deterministic models and the statistical models [Rab89]. 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 characterised 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.

Hidden Markov Models (HMM's) are a powerful tool in the field of signal processing [Rab89, MS97]. HMMs have been successfully used in speech recognition [RJ86]. Recently, the application of HMMs has been extended to include word recognition [CKZ94,

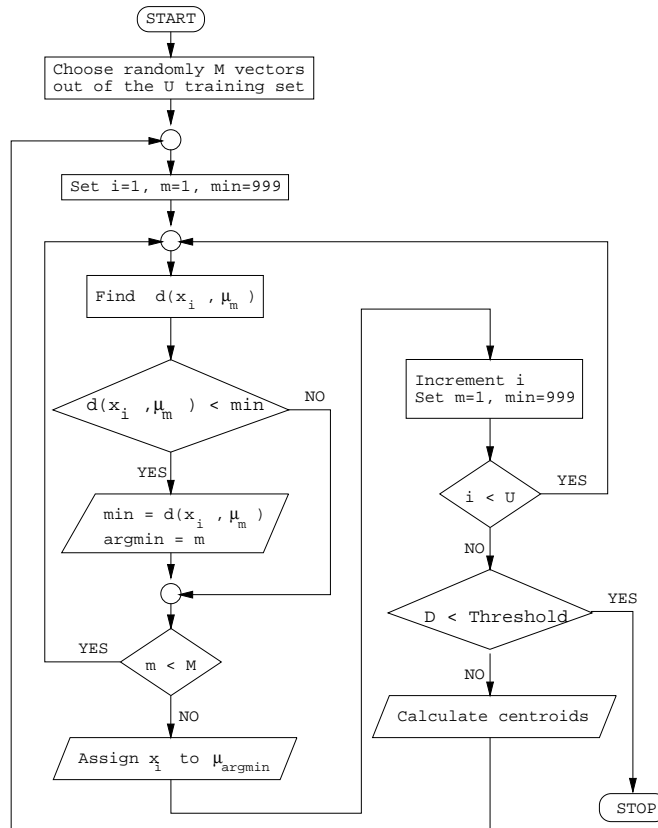


Figure 3.1: Flow chart of K-means clustering algorithm.

BRS95, MLR⁺96]. 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.

3.3.1 Theory and Elements of HMM

An 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 system that at any instance of time may only be in one of the N 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 q_i at time t to a state q_j at time $t + 1$, $t = 1, 2, 3, \dots$ and $i, j = 1, 2, \dots, N$.

The most important and difficult element to be decided is the number of states, N , in the 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 in a way that each state may be reached from other states in one transition, and this is called an ergodic model. Other types are also possible, and they will be illustrated later in this section. Individual states are labeled as $Q = \{q_1, q_2, \dots, q_N\}$.

Another important element is the number of observation symbols M per state. The observation symbols correspond to the physical output of the system being modelled. For some applications like speech recognition and character recognition, the observations are continuous and are produced as vectors. In this case the vectors are quantised into one of the permissible sets using VQ as described in the previous section. Individual symbols are denoted as $V = \{v_1, v_2, \dots, v_M\}$.

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

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

$$\pi = \{\pi_i = P(q_i \text{ at } t = 1)\}$$

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

$$A = \{a_{ij} = P(q_j \text{ at } t + 1 | q_i \text{ at } t)\}$$

3. The observation symbol probability. This is the probability of observing symbol v_k while the model is in state i at time t .

$$B = \{b_i(k) = P(v_k \text{ at } t | q_i \text{ at } t)\}$$

Considering the previous elements mentioned, a complete specification of an HMM requires specifying two model parameters, N and M , the observation symbols, and three sets of probability measures A , B and π . The compact notation to be used to refer to this HMM is $\lambda(\pi, A, B)$.

3.3.2 Problems To Be Solved

Given an HMM previously defined, an observation sequence $O = (o_1 o_2 \dots o_T)$ can be generated using this HMM, where T is the length of the observation sequence and each observation o_t is one of the discrete set of possible symbol observations V . Following are the steps the HMM uses to generate the observation sequence:

1. Set $t = 1$. This means the first observation.
2. Select the initial state $q_i(1)$ based on the initial state probability π_i , where $1 \leq i \leq N$.
3. Select the observation o_t as v_k as one of the symbols in V according to the symbol probability distribution $b_i(k)$, where $1 \leq i \leq N$.

4. Transit from the current state $q_i(t)$ to another state $q_j(t+1)$ according to the state transition probability a_{ij} , where $1 \leq i, j \leq N$.
5. Increment t .
6. If $t \leq T$ go back to step 3 otherwise stop.

Given this formal description of an HMM and the procedure used to generate the observation sequence, Rabiner [Rab89] illustrated three key problems that need to be resolved before applying the HMM to a real-world application:

- Given an HMM, $\lambda(\pi, A, B)$, and a sequence of observations $O = (o_1, o_2 \dots o_T)$, compute the most efficient probability of the observation sequence giving the model $P(O|\lambda)$.
- Given an HMM, $\lambda(\pi, A, B)$, and a sequence of observations $O = (o_1, o_2 \dots o_T)$, find the optimal path (states sequence) $q = (q_1, q_2 \dots q_T)$.
- Given an HMM, $\lambda(\pi, A, B)$, adjust these parameters to maximise the probability of observation giving the model $P(O|\lambda)$.

Following are these three problems and the proposed solution to each one of them.

3.3.2.1 Observation sequence evaluation

This deals with the evaluation of the observation sequence given the HMM. It computes the probability that the observed sequence was produced by a given model. This is extremely important particularly when a decision to choose among several competing models is being made. Hence, the target is to find the model that best matches the observations, in other words, the model with the highest probability. Computing this probability through enumerating every possible state sequence of length T is tedious and time-consuming since there are N^T state sequences which require $(2T-1)N^T$ multiplications and N^T-1 additions. Two alternatives may be used instead: the

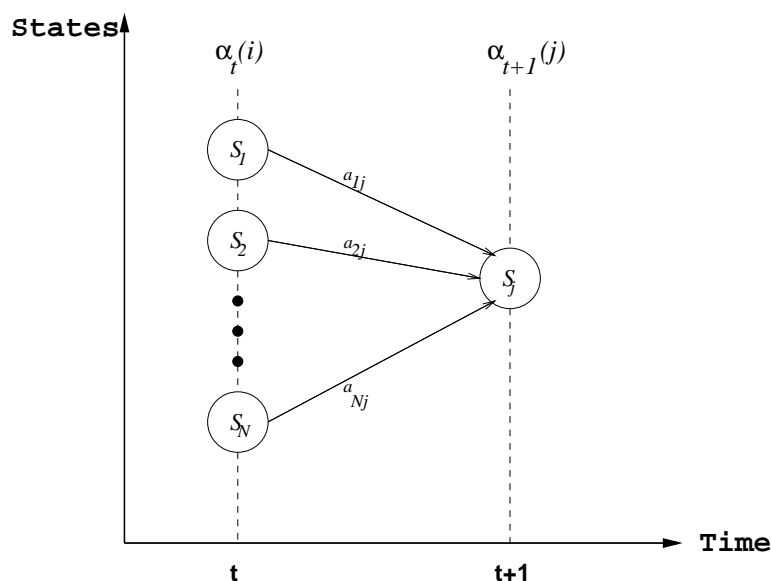


Figure 3.2: The recursion procedure required to compute the forward variable $\alpha_{t+1}(j)$.

forward procedure and the backward procedure. The forward procedure calculates the forward variable at each instance of time t for each state i , i.e. $\alpha_t(i)$, see Fig 3.2. It is a three step procedure:

1. *Initialisation:* This initialises the forward probabilities as the joint probability of state i and initial observation o_1 .

$$\alpha_1(i) = \pi_i b_i(o_1), \quad 1 \leq i \leq N \quad (3.5)$$

2. *Induction:* This calculates the forward probability of state j at time $t+1$ based on the joint probability of previous forward variables from all states at time t and the transition probabilities from each of those states to state j , as well as the observation probability at time $t+1$.

$$\alpha_{t+1}(j) = \left[\sum_{i=1}^N \alpha_t(i) a_{ij} \right] b_j(o_{t+1}) \quad (3.6)$$

$$1 \leq j \leq N, 1 \leq t < T$$

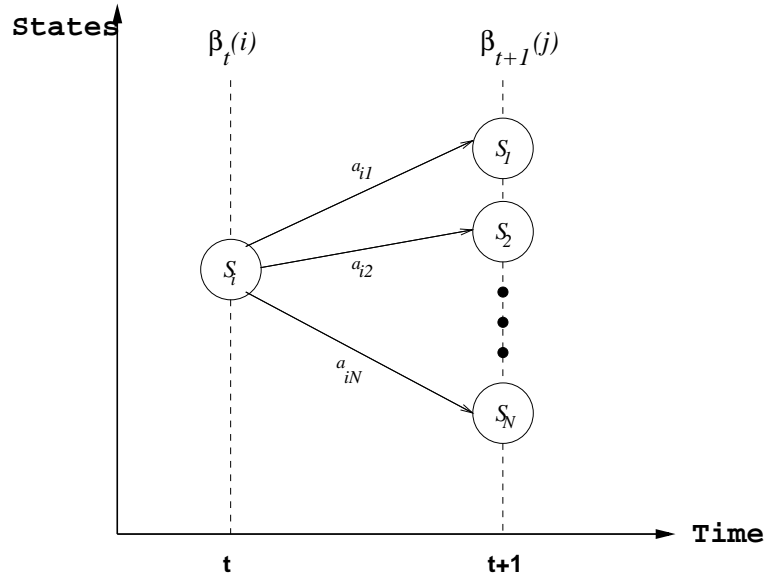


Figure 3.3: The recursion procedure required to compute the backward variable $\beta_t(i)$.

3. *Termination:* This gives the desired calculation of $P(O|\lambda)$ as the sum of the terminal forward variables $\alpha_T(i)$.

$$P(O|\lambda) = \sum_{i=1}^N \alpha_T(i) \quad (3.7)$$

When examining the computation involved in calculating the forward variables over all states at all instances of time, there are $N + N(N + 1)(T - 1)$ multiplications and $N(N - 1)(T - 1)$ additions.

The backward procedure follows the same approach as the forward procedure however in the opposite direction from $t = T$ down to $t = 1$, see Fig 3.3. This is a two step procedure:

1. *Initialisation:* In this step, the backward variable of each state at time T is set to one.

$$\beta_T(i) = 1, \quad 1 \leq i \leq N \quad (3.8)$$

2. *Induction:* This calculates the forward probability of state i at time t based on the joint probability of backward variables from all states at time $t + 1$ and the transition probabilities from each of those states to state i , as well as the observation probabilities of those states at time $t + 1$.

$$\beta_t(i) = \sum_{j=1}^N a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad (3.9)$$

$$1 \leq i \leq N, t = T - 1, T - 2, \dots, 1$$

When examining the computation involved in calculating the backward variables over all states at all instances of time, there are $2N^2(T - 1)$ multiplications and $N(N - 1)(T - 1)$ additions.

3.3.2.2 Optimal path retrieving

This attempts to uncover the hidden part of the model by finding the most likely state sequence associated with the examined observation sequence. It is not a matter of correct sequence, rather it is the outcome of using an optimality criterion to solve this problem as best as possible. An example of a possible optimality criterion is selecting the state q_t which is individually most likely at each time t . A problem with this approach is that when the HMM has state transitions which have a zero probability this implies that the state sequence is not valid. In order to remedy this, an alternative optimality criterion is selected to maximise $P(O|\lambda)$ over a complete state sequence. A formal technique to finding this optimum path is based on dynamic programming methods, and is called the *Viterbi Algorithm* [For73].

The Viterbi algorithm is a four step process where the first three steps are similar to those in a forward procedure however it replaces summation with maximisation. Assume the same HMM, $\lambda(\pi, A, B)$, discussed above. For a given observation sequence $O = (o_1, o_2, o_3, \dots, o_T)$, the steps of the algorithm are as follows:

1. *Initialisation:*

$$\delta_1(i) = \pi_i b_i(o_1) \quad (3.10)$$

$$\psi_1(i) = 0 \quad (3.11)$$

$$1 \leq i \leq N$$

2. *Recursion:* This is to actually retrieve the state sequence by keeping track of the argument that maximises $\delta_t(i)$, for each t and i , and save it in $\psi_t(i)$.

$$\delta_t(i) = \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ji}] b_i(o_t) \quad (3.12)$$

$$\psi_t(i) = \arg \max_{1 \leq j \leq N} [\delta_{t-1}(j) a_{ji}] \quad (3.13)$$

$$1 \leq i \leq N, 2 \leq t \leq T$$

3. *Termination:*

$$P^* = \max_{1 \leq j \leq N} [\delta_T(j)] \quad (3.14)$$

$$q_T^* = \arg \max_{1 \leq j \leq N} [\delta_T(j)] \quad (3.15)$$

4. *Path Backtracking:* As the last state of the optimum path was found in the previous step, all the states falling on this path are retrieved in descending order.

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad (3.16)$$

$$t = T - 1, T - 2, \dots, 1$$

An alternative implementation to the Viterbi algorithm is to take logarithms of the model parameters in a separate preprocessing step at the beginning. This eliminates multiplications and improves the accuracy in estimating small values. Below is the step sequence:

1. *Preprocessing:*

$$\tilde{\pi}_i = \log(\pi_i), \quad 1 \leq i \leq N \quad (3.17)$$

$$\tilde{a}_{ij} = \log(a_{ij}), \quad 1 \leq i, j \leq N \quad (3.18)$$

$$\tilde{b}_i(o_t) = \log[b_i(o_t)], \quad 1 \leq i \leq N, 1 \leq t \leq T \quad (3.19)$$

2. *Initialisation:*

$$\tilde{\delta}_1(i) = \log(\delta_1(i)) = \tilde{\pi}_i + \tilde{b}_i(o_1) \quad (3.20)$$

$$\psi_1(i) = 0 \quad (3.21)$$

$$1 \leq i \leq N$$

3. *Recursion:*

$$\tilde{\delta}_t(i) = \log(\delta_t(i)) = \max_{1 \leq j \leq N} [\tilde{\delta}_{t-1}(j) + \tilde{a}_{ji}] + \tilde{b}_i(o_t) \quad (3.22)$$

$$\psi_t(i) = \arg \max_{1 \leq j \leq N} [\tilde{\delta}_{t-1}(j) + \tilde{a}_{ji}] \quad (3.23)$$

$$1 \leq i \leq N, 2 \leq t \leq T$$

4. *Termination:*

$$\tilde{P}^* = \max_{1 \leq j \leq N} [\tilde{\delta}_T(j)] \quad (3.24)$$

$$q_T^* = \arg \max_{1 \leq j \leq N} [\tilde{\delta}_T(j)] \quad (3.25)$$

5. *Path Backtracking:*

$$q_t^* = \psi_{t+1}(q_{t+1}^*) \quad (3.26)$$

$$t = T-1, T-2, \dots, 1$$

3.3.2.3 Model parameter estimation

This is called the training stage where an attempt is made to optimise the model parameters, A , B and π , so as to best describe how the observed sequence occurs. The optimality criterion maximises the observation sequence probability, $P(O|\lambda)$, given the model, $\lambda(A, B, \pi)$. There is no analytical approach for this, however an iterative procedure such as the Baum-Welch method, known also as Expectation-Maximisation (EM), is applied to locally maximise the likelihood $P(O|\lambda)$ of the chosen $\lambda(A, B, \pi)$.

The reestimation procedure is highly dependent on the definition of each model parameter. Thus, a set of reasonable reestimation

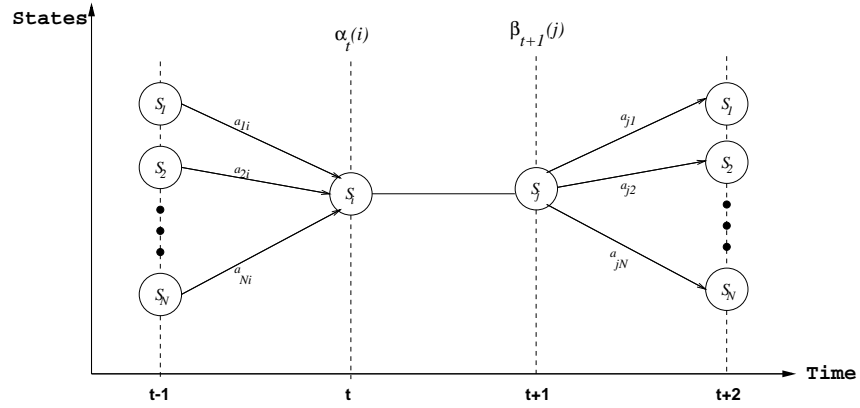


Figure 3.4: Sequence of operations required for the computation of the variables: $\xi_t(i, j)$ in equation 3.30, and $\gamma_t(i)$ in equation 3.32.

formulas for π , A and B are

$$\bar{\pi} = \text{Expected number of times in state } i \text{ at time } t = 1 \quad (3.27)$$

$$\bar{a}_{ij} = \frac{\text{Expected number of transitions from state } i \text{ to state } j}{\text{Expected number of transitions from state } i} \quad (3.28)$$

$$\bar{b}_j(k) = \frac{\text{Expected number of times in state } j \text{ and observing } v_k}{\text{Expected number of times in state } j} \quad (3.29)$$

To describe the previous training formulas mathematically, let $\xi_t(i, j)$ be the probability of being in state i at time t , and in state j at time $t + 1$ given the model and the observation sequence

$$\xi_t(i, j) = P(q_t = i, q_{t+1} = j | \mathbf{O}\lambda) \quad (3.30)$$

Looking to Fig 3.4 and the definition of the forward and backward variables, Eq. 3.30 may be written as

$$\begin{aligned} \xi_t(i, j) &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{P(\mathbf{O} | \lambda)} \\ &= \frac{\alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{i=1}^N \sum_{j=1}^N \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)} \end{aligned} \quad (3.31)$$

Summing $\xi_t(i, j)$ over $1 \leq t \leq T - 1$ results in the expected number of transitions from i to j .

For the purpose of training, define the probability of being in state i at time t , given the observation sequence and the model as $\gamma_t(i)$

$$\gamma_t(i) = P(q_t = i | \mathbf{O} \lambda), \quad (3.32)$$

again using the notation of the forward and backward variables, Eq. 3.32 may be written as

$$\gamma_t(i) = \frac{\alpha_t(i)\beta_t(i)}{\sum_{i=1}^N \alpha_t(i)\beta_t(i)}, \quad (3.33)$$

Accordingly, $\gamma_t(i)$ and $\xi_t(i, j)$ can be related by summing $\xi_t(i, j)$ over j

$$\gamma_t(i) = \sum_{j=1}^N \xi_t(i, j) \quad (3.34)$$

Summing $\gamma_t(i)$ over time gives the number of times state i is visited. If the time slot T is excluded, in other words summing over $1 \leq t \leq T - 1$, this gives the number of transitions from state i . Equations 3.28-3.29 may be rewritten now as

$$\bar{\pi}_i = \gamma_1(i) \quad (3.35)$$

$$\bar{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i, j)}{\sum_{t=1}^{T-1} \gamma_t(i)} \quad (3.36)$$

$$\bar{b}_j(k) = \frac{\sum_{t=1}^T \gamma_t(j)}{\sum_{t=1}^T \gamma_t(j)} \quad (3.37)$$

3.3.3 HMM Types

HMMs can be classified based on the structure of the transition matrix A . Since the beginning of this section HMM has been discussed as all states that are fully connected. This may be referred to as an *Ergodic* model where each state in the model may be reached from any state within one transition. Assume a model, in Fig 3.5-a, with $N = 5$, then all the elements in the transition matrix A are non-zero elements, $a_{ij} \neq 0$ where $1 \leq i, j \leq N$.

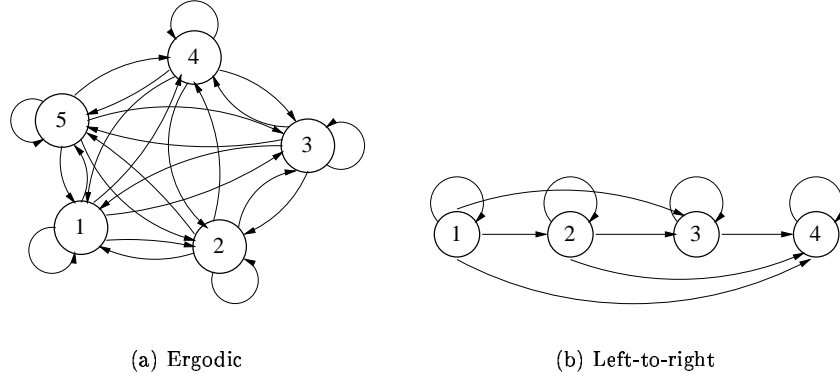


Figure 3.5: Two different types of HMMs.

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ a_{21} & a_{22} & a_{23} & a_{24} & a_{25} \\ a_{31} & a_{32} & a_{33} & a_{34} & a_{35} \\ a_{41} & a_{42} & a_{43} & a_{44} & a_{45} \\ a_{51} & a_{52} & a_{53} & a_{54} & a_{55} \end{bmatrix}$$

Some applications such as speech recognition and character recognition require certain constraints on the state transition matrix. This implies modifying the conventional ergodic design to what so called *Left-to-Right* HMM (LR-HMM), see Fig 3.5-b. This model inherently imposes a temporal order to the HMM since lower numbered states account for observations occurring prior to those for higher numbered states. Three main constraints are applied here:

1. The state sequence must start in state 1.

$$\pi_i = \begin{cases} 1 & i = 1 \\ 0 & i > 1 \end{cases} \quad (3.38)$$

2. No transitions are allowed to states whose indices are lower than the current state or at times lower or equal.

$$a_{ij} = 0 \quad i > j \quad (3.39)$$

The above transition matrix will be

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} & a_{15} \\ 0 & a_{22} & a_{23} & a_{24} & a_{25} \\ 0 & 0 & a_{33} & a_{34} & a_{35} \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & a_{55} \end{bmatrix}$$

3. Some applications restrict the large changes in state indices by imposing additional constraints on the permitted number of jumps

$$a_{ij} = 0 \quad j > i + \Delta \quad (3.40)$$

Assume $\Delta = 2$, then A will be

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} & 0 & 0 \\ 0 & a_{22} & a_{23} & a_{24} & 0 \\ 0 & 0 & a_{33} & a_{34} & a_{35} \\ 0 & 0 & 0 & a_{44} & a_{45} \\ 0 & 0 & 0 & 0 & a_{55} \end{bmatrix}$$

3.3.4 Application To Character Recognition

Hidden Markov Models have been extremely useful for a wide range of real-world applications [HK91, AE94, KC97, OAB97]. They were also successfully applied to connected, speaker-independent, automatic speech recognition with the advantage of modelling various patterns [LRS83, RLS83, RJLS85, JR85, AC93, DPH93, RJ93]. This success of automatic speech recognition systems based on the hidden Markov model has motivated recent attempts to apply similar methods to character/word recognition whether on-line [HBT96, KKKL97a, KKKL97b] or off-line [VK92, BK94, OHK95, KP96]. The HMM provides an explicit representation for time-varying patterns and probabilistic interpretations that can tolerate variations in these patterns.

In off-line recognition systems, the general idea is to transform the word image into a sequence of observations. The observations

produced by the training samples are used to tune the model parameters whereas those produced by the testing samples are used to investigate the system performance.

Some researchers made a critical assumption that the word image is already segmented and thus their research dealt with segmented characters [KHB89]. In this case, each state in the model represents a letter in the alphabet set, and each feature vector is equivalent to one observation. Others [CKZ94, CKS95] segmented the input word image into a sequence of segments in which an individual segment might be a complete character, a partial character, or joint characters. The HMM parameters were estimated from the lexicon and the training image segments. This segmentation may also be applied to the contour [EGSS99] to find segmentation points, then to extract the features from these segments and transfer the feature vectors into an observation sequence. Segmentation can be avoided if the skeleton of the word is decomposed into small strokes of which each is transformed into a feature vector and then an observation. Processing the original word image directly is another alternative [MG96]. This system combined segmentation-free and segmentation-based techniques. The segmentation-free technique constructed a continuous density HMM for each lexicon string. The segmentation-based technique used dynamic programming to match word image and strings.

Hidden Markov modelling is suitable for 1D time sequential signals such as speech. The image signal has two dimensions. The 2D nature of the OCR problem is a fundamental difference between text and speech recognition problems. This justifies the conclusion that extending a 1D-HMM to a 2D-HMM can achieve greater advantages. Fully connected 2D-HMMs would lead to a recognition algorithm of exponential complexity [AK93]. To remedy this, the connectivity of the network can be reduced. This results in pseudo 2D-HMMs which are successfully implemented to recognise type-written characters [AK93] and handwritten characters [PL98]. It is also used to spot keywords in poorly printed text [KA94].

Hidden Markov models have been also applied to Arabic character and text recognition. In [All95], the contour of the word image was segmented after baseline estimation. This resulted in a sequence of labels, the latter of which was classified by finding the HMM which gave the highest probability. To reduce the computation time and enhance the recognition rate, a segment was not compared to the whole set of models it was rather compared to a selected group according to the position of that segment within the word. Other ongoing research is by Makhoul et al [MLR⁺96, MSLB98, BSM99]. Their system depends on the estimation of character models, a lexicon, and grammar from training samples. The training phase takes scanned lines of text coupled with the ground truth, the text equivalent of the text image, as input. Then, each line is divided into narrow overlapping vertical windows from which feature vectors are extracted. The character modelling component takes the feature vectors and the corresponding ground truth and estimates the character models. The recognition phase follows the same step to extract the feature vectors which are used with different knowledge sources estimated in the training phase to find the character sequence with the highest likelihood $P(O|\lambda)$.

3.4 Summary

In this chapter I have discussed two important tools which are used throughout this dissertation: Vector Quantisation and Hidden Markov Models. I have given a hint of the advantages and disadvantages for applying VQ to word recognition. I have also presented the theory of HMMs. The focus was on physical explanations of the basic mathematics involved. I have also briefly reviewed the implementation of HMMs to character/word recognition of a number of languages including Arabic.

CHAPTER 4

ARABIC MANUSCRIPT SAMPLES

4.1 Introduction

To understand the importance of calligraphy for Arabs and Muslims we must first know that Arabic is the language of “الْقُرْآنُ الْكَرِيمُ” *alqur’ānu alkarīm* -the Holy Qur’an”. This required great care and energy in order to make the written word worthy of such honour. Consequently, a beautiful script to write the “Holy Qur’an” was developed. Calligraphy became a type of art that equally attracted the specialist and the ordinary person. Some of the decorative scripts are as attractive as the most beautiful paintings. Consider the script in Fig 4.1, it reads: *‘aliyyu ‘ibnu ‘abiy ṭālib raḍiya al-lāhu ta‘alā-anhu wakarrama waḡhahu*

عَلِيُّ بْنُ أَبِي طَالِبٍ رَضِيَ اللَّهُ تَعَالَى عَنْهُ وَكَرَّمَتْ وَجْهَهُ

Ali Ibn Abi Talib, may God Almighty be pleased with him and honour him. The script was structured into the shape of a lion. More of such samples may be retrieved from the of web site “[http :
//www.islamicart.com](http://www.islamicart.com)”.

This dissertation is focused on machine-printed and handwritten styles used for scribing books, and not on decorative style used for adornment purposes. This chapter illustrates a number of computer-



Figure 4.1: Script image constructed into the shape of a lion.

generated and handwritten fonts which are used throughout the dissertation.

4.2 Computer-Generated Fonts

As mentioned in Chapter 2, recognising Arabic typeset text is more challenging than recognising typewritten text because of ligatures and overlaps. This complexity is due to the difficulty in segmenting these glyphs. Recently, a number of fonts which are usually written by hand, like *Thuluth* and *Andalus*, or used as typeset, like *Naskh*, have been generated perfectly using a computer. In spite of this these fonts still retain their original specifications, they all share the regularity feature of computer-generated fonts. In the rest of this section, some of these fonts are presented.

4.2.1 Simplified Arabic

This is a typical machine-printed font. This font is a perfect candidate for the approach of segmenting a word into its characters, see Sec 2.4.3, since there is no ligature or overlap. Fig 4.2 shows a script written using this font.

4.2.2 Arabic Traditional

This is more complicated than the previous font due to the presence of overlap and ligature. It is similar to the font used by printing-houses to print books and newspapers. With this font it is not

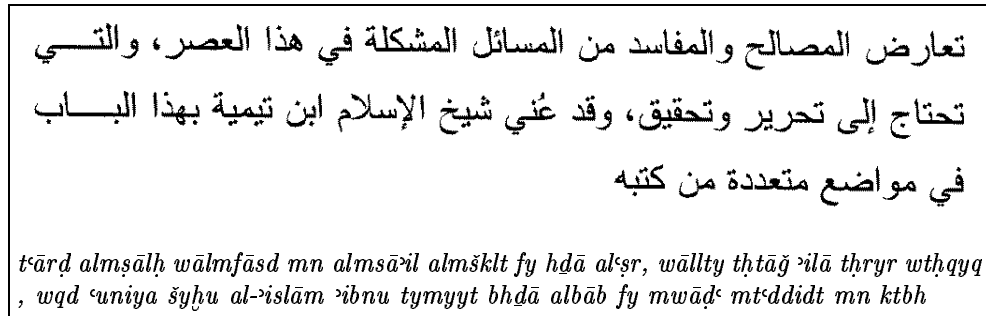


Figure 4.2: Sample text image written in Simplified Arabic computer-generated font, and the transliterated text.

always possible to segment a word into its characters. Fig 4.3 shows a script written using this font.

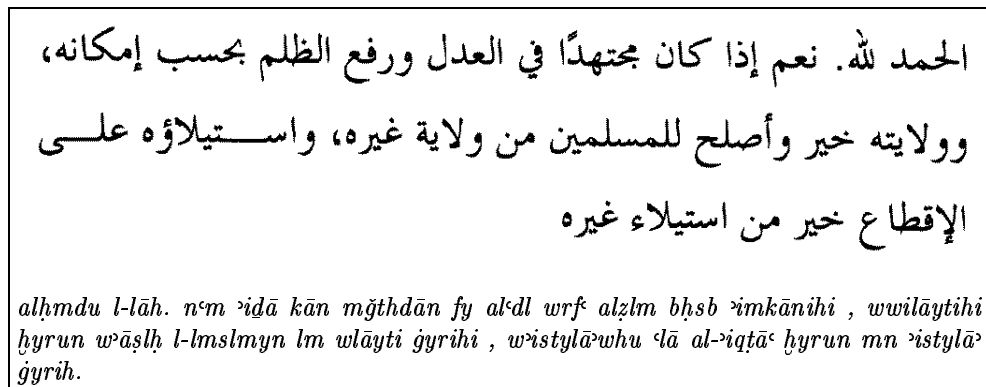


Figure 4.3: Sample text image written in Arabic Traditional computer-generated font, and the transliterated text.

4.2.3 Thuluth

This was first formulated during Umayyah Caliphate in the first century AH (c. the seventh century CE). The name means a ‘third’ and one explanation is that this is due to the proportion of straight lines to curves. Fig 4.4 shows a script written using this font.

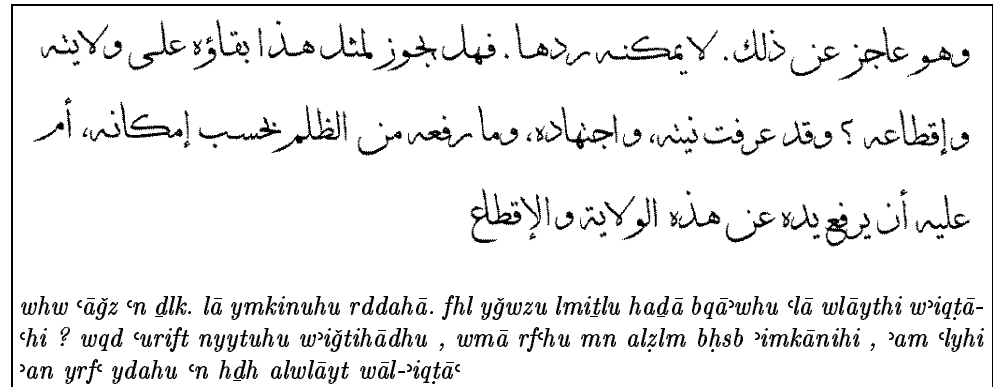


Figure 4.4: Sample text image written in Thuluth computer-generated font, and the transliterated text.

4.2.4 Andalus

This font is named after ‘*al-andalus* Al-Andalus’ a great period of Islamic civilisation in Spain starting from the end of the second century AH (*c.* the eighth century CE) and lasting for eight centuries. It uses serifs and it has a characteristic angular style. This was among the set of fonts imported into the computer-based environment. Fig 4.5 shows a script written using Andalus font.

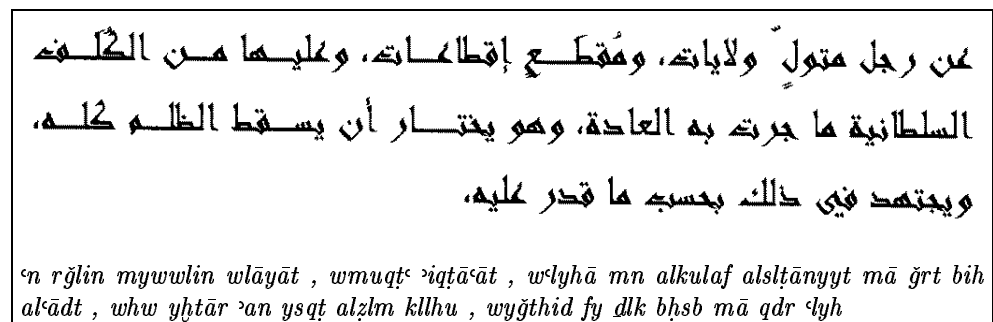


Figure 4.5: Sample text image written in Andalus computer-generated font, and the transliterated text.

4.3 Handwritten Manuscripts

This is a set of ancient Arabic manuscripts which is located in the Cambridge University Library (*UL*). Most of these manuscripts are without date or colophon. Some of them are more than 800 years old. Each manuscript is referred to using the class-mark index (*sigla*) of the library.

4.3.1 UL Moh194.a.4

This manuscript is entitled “جَمَهَرَةُ النَّسَبِ لِابْنِ الْكَلْبِيِّ” *ḡamhart al-nasab l-ʾibni alkalbī*” by هِشَامُ أَبُو الْمُنْدِرِ بْنِ مُحَمَّدٍ الْكَلْبِيِّ *hišām ʾabū almundir bin muḥammad alkalbī*, who died in 204 AH (c. 807 CE). It was first written during the Abbasid caliphate (c. 750-1543 CE) and includes all that has been recorded earlier by specialists on different tribes in Arabia. During this century, مُحَمَّدُ فَرْدُوسُ الْعَظَمِ *maḥmūd frdūs alʿaẓm* verified and rescribed the book. The manuscript includes two different styles: the style used to write the body of the book, see Fig 4.6, and the style used to write the footnotes, see Fig 4.7.

4.3.2 UL Or.172

This is a book in two parts:

1. كَشْفُ الصَّلَاةِ عَنْ وَصْفِ الزَّلْزَلَةِ *kšfu alṣalṣalat ʿn wṣfi alzalzalat*: This is followed by a short poem in praise of the prophet by عَمْرُو بْنُ الْوَرْدِيِّ *mruw ʾibni alwardiy*. The manuscript describes some of the earthquakes that hit different parts of the Islamic caliphate. This manuscript is written in legible Naskh with rubrications.
2. كِتَابُ مَرَوَاهُ الْوَأَعُونَ فِي أَخْبَارِ الطَّاعُونَ *kitāb mārwaḥu alwāʿawn fiy ʾaḥbāri alṭāʿawn*: This includes a history of the plague.

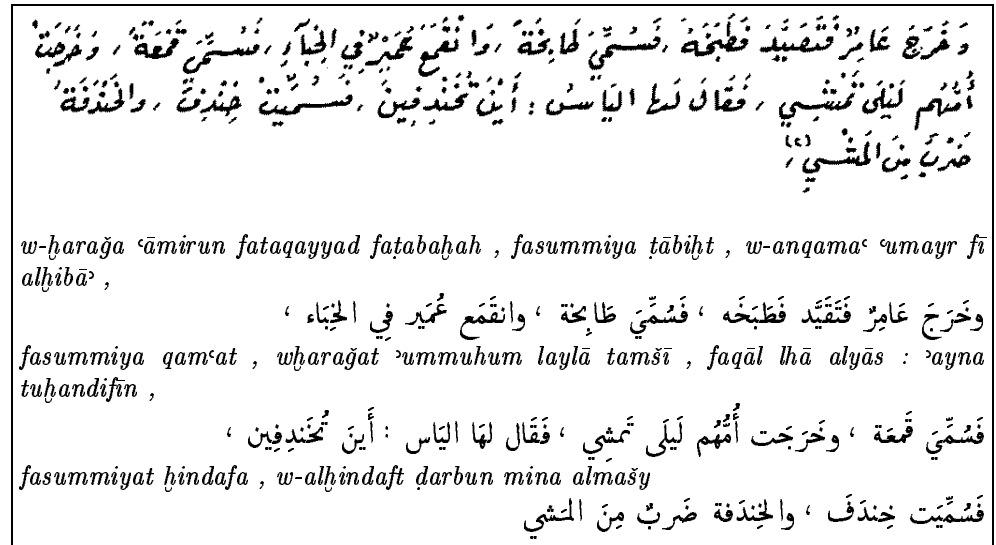


Figure 4.6: Sample text image extracted from the manuscript Moh194.a.4, and the transliterated text.

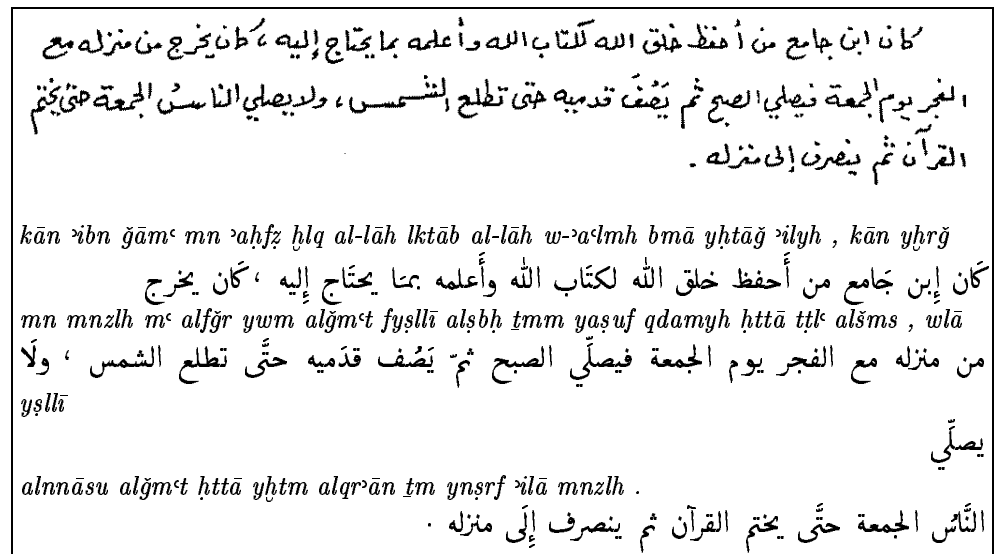


Figure 4.7: Sample text image extracted from the footnotes of the manuscript Moh194.a.4, and the transliterated text.

Both parts were written by the celebrated author: جَلَالُ الدِّينِ السُّيُوطِي *ġlālu alddīn alsuyūṭī*, who died in 911 AH (c. 1484 CE). Fig 4.8 shows a script extracted from this manuscript.

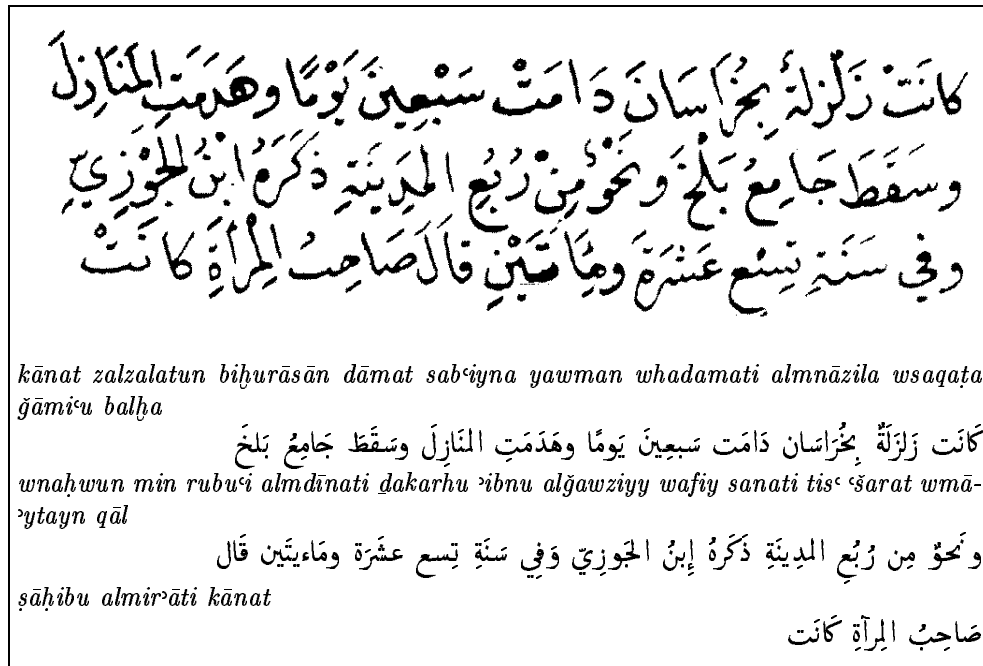


Figure 4.8: Sample text image extracted from the manuscript Or.172, and the transliterated text.

4.3.3 UL Or.1481

This manuscript is “رَوْضَاتُ الْمَنَاطِرِ فِي أَخْبَارِ الْأَوَّالِ وَالْآخِرِ” *rawḍātu almanāzir fy aḥbār al-ʾawāʾyl wāl-ʾawāḥir* by

زَيْنُ الدِّينِ أَبُو الْوَلِيدِ مُحَمَّدُ الْحَلَبِيُّ الْحَنْفِيُّ *ziyṇu aldiyn ʾabuw alwaliyd muḥammad alḥalabiy alḥanafiy*, who died in 815 AH (c. 1388 CE).

The manuscript presents a brief history during different ages starting from Adam, may peace be upon him. This is printed in clear Naskh. Updated the second time in the 17th century, and printed at Bulaq in 1290 AH (c. 1870 CE). Fig 4.9 shows a script extracted from this

manuscript.

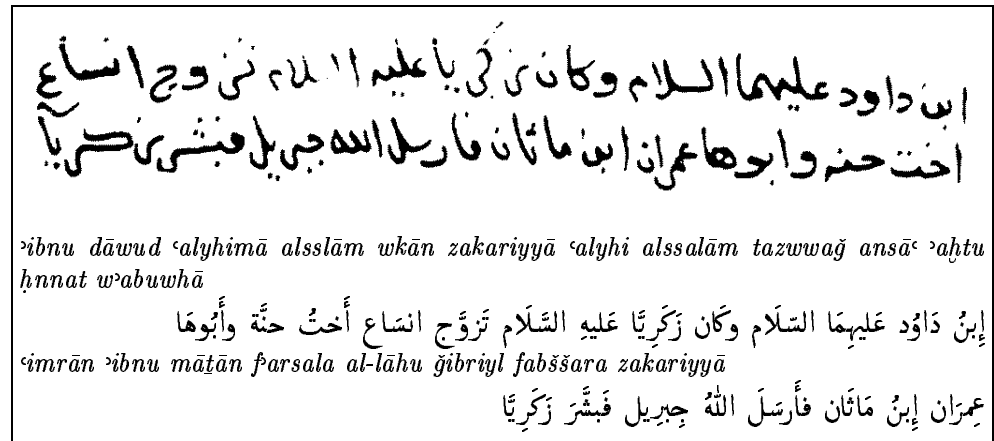


Figure 4.9: Sample text image extracted from the manuscript Or.1481, and the transliterated text.

4.3.4 UL Qq.65

This manuscript is entitled “تَقْوِيمُ الْبُلْدَانِ الْمَصْرِیَّةِ فِي الْأَعْمَالِ السُّلْطَانِيَّةِ” *taqwiym albuldān almasriyyat fy al-ʿamāl alsultāniyyat*. This is an account of the geography of Egypt and its territorial divisions in the reign of أَلْمَلِكُ الْأَشْرَفُ *almalik al-ʿašraf*. This was compiled by order of عَامِرُ شَعْبَانَ حَسَنَ *ʿāmir šaʿbān ḥasan* by an author whose name does not appear, in 777 AH (c. 1350 CE). This manuscript is written in good Naskh. Names and titles are written in gold, green, and red. Fig 4.10 shows a script extracted from this manuscript.

4.3.5 UL Qq.91

This is a book in two parts. Both volumes begin abruptly without a preface. The titles of the two parts are as follows:

1. فَضَائِلُ مِصْرَ *faḍāyil miṣr*
2. فَضَائِلُ بَيْتِ الْمَقْدِسِ وَالشَّامِ *faḍāyil bayti almqdis wālšām*

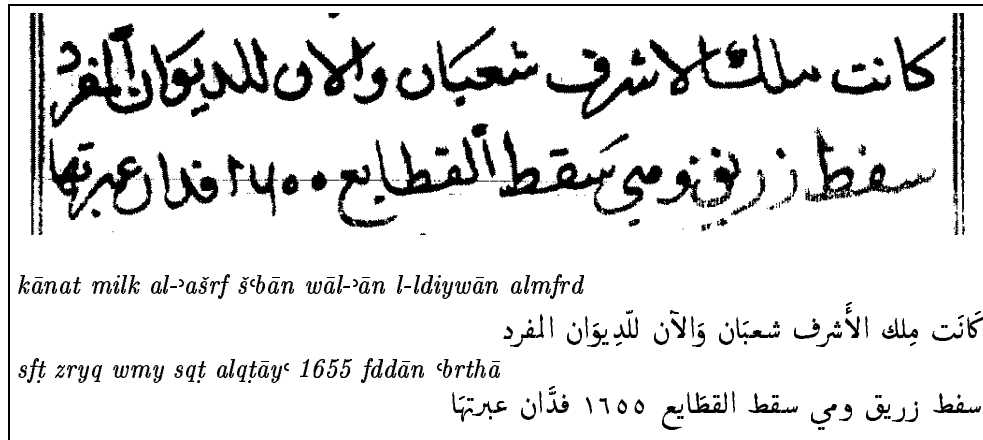


Figure 4.10: Sample text image extracted from the manuscript Qq.65, and the transliterated text.

The manuscript illustrates the features of Egypt, Jerusalem and old Syria. The first part is not pointed and not dated whereas the second part is written in large, bold, and good Naskh; pointed with rubrications, and is dated 'رجب' *raġab* - Rajab' 785 AH (c. 1358 CE). Fig 4.11 shows a script extracted from this manuscript.

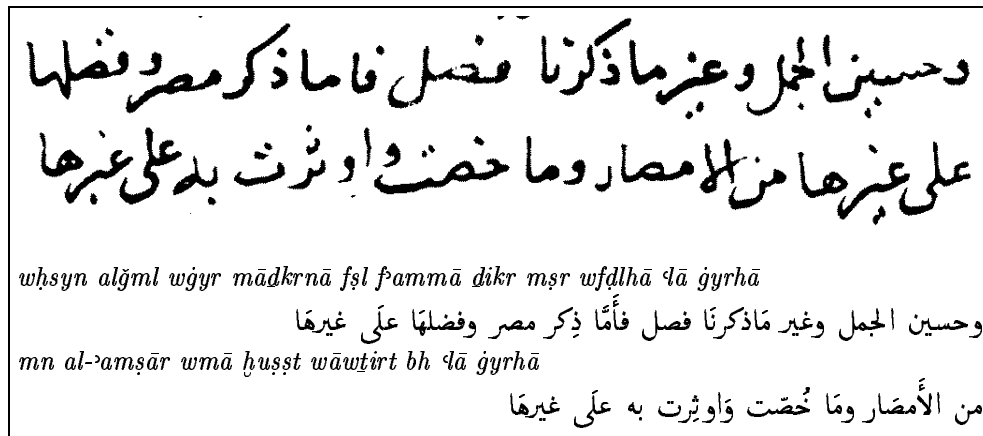


Figure 4.11: Sample text image extracted from the manuscript Qq.91, and the transliterated text.

4.3.6 UL Qq.74

This manuscript is entitled “جواهر السلوك في أخبار الخلفاء والملوك” *ġwā-hir alsulūk fy aḥbār alḥulfāʾi wālmulūk* by

شمس الدين محمد بن إياس الحنفي *šamsu alddīn muḥammad ʾibn ʾilyās alḥanafīy*. This is a general history of the Caliphs beginning with an account of the Prophet, may peace be upon him, and ending with the death of الله المتوكل على الله *almutawakkil ʿalā al-lāh* in 903 AH (c. 1476 CE). Fig 4.12 shows a script extracted from this manuscript.

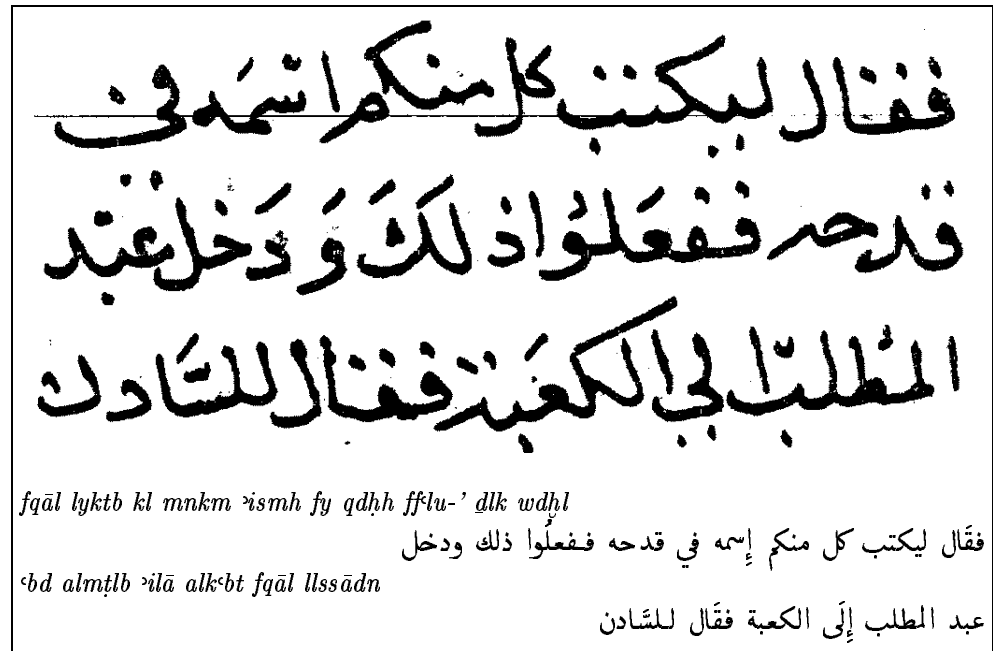


Figure 4.12: Sample text image extracted from the manuscript Qq.74, and the transliterated text.

4.3.7 UL Add.2923

This is one of the three volumes that form the book entitled “صَلَاحُ الدِّينِ مُحَمَّدِ بْنِ شَاكِرِ الْحَلَبِيِّ الْكُتُبِي” *ṣalāḥ alddīn muḥammad ʾibnu šākīr alḥalabī alkutbī* by صلاح الدين محمد بن شاكِر الحلبي الكُتبي *ṣalāḥ alddīn muḥammad ʾibnu šākīr alḥalabī alkutbī*. This volume

is a defective at the beginning. It includes the years 735 – 760 AH (c. 1308 – 1333 CE) and was written in fair Old Naskh. Fig 4.13 shows a script extracted from this manuscript.

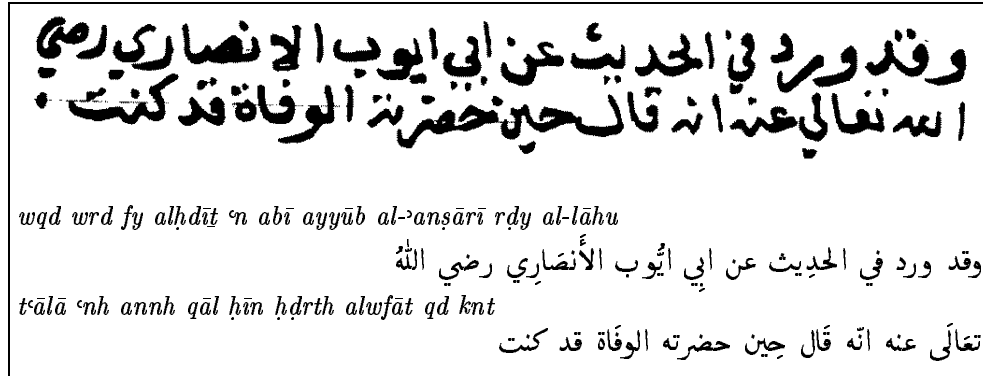


Figure 4.13: Sample text image extracted from the manuscript Add.2923, and the transliterated text.

4.3.8 UL Add.3257

This is a collection of tracts by جلال الدين السيوطي *ḡlālu alddīn alsuyūṭī*, who died in 911 AH (c. 1484 CE), to wit:

1. *كِتَابُ الْمُحَرَّرِ فِي قَوْلِهِ تَعَالَى لِيَغْفِرَ اللَّهُ لَكَ مَا تَقَدَّمَ مِنْ ذَنْبِكَ وَمَا تَأَخَّرَ* *kitābu almuḥṛra-r fī qwlihi tʿālā liyaḡfira al-lāhu laka mā tqad-dama min danbika wamā taḥḥar*. This presents an interpretation of a verse in Qua'an.
2. *نُحْفَةُ الْجُلَسَاءِ* *tuhfatu alḡulasā*.
3. *رِسَالَةُ الْكَاشِفِ عَنْ مُجَاوَزَاتِ هَذِهِ الْأُمَّةِ الْأَلْفِ* *risālatu alkāšif ʿan muḡāwzāt ḥḍihi al-ʾammat al-ʾalf*.
4. Answers of *إِبْنِ حَجَرٍ* *ibni ḥaḡar* to questions addressed to him regarding to the manner in which the turban should be worn.
5. A commentary by *شَيْخُ أَبُو الْعَبَّاسِ أَحْمَدُ شِهَابُ الدِّينِ الْأَنْصَارِيِّ* *šayḥ šayḥ ʾabū ʿabbās aḥmad šihāb al-dīn al-ʾanṣārī*.

ʿabū alʿbbās ʿaḥmad šihābu alddīn al-ʿanṣārī on the “سِتِّينَ مَسْأَلَةً” *sittīn masʿalat* - Sixty Questions” of *الزَّاهِدِ أَحْمَدُ الْعَبَّاسُ* *šīḥ ʿabū ʿishāq ʿibrāhīm bin muḥammad alšāfiʿī*. This includes some of the Prophet’s sayings.

6. A commentary on *سُورَةُ الْفَاتِحَةِ* *sūratu alfātiḥat*, the first chapter in the Qur’an.
7. *الْمُؤْمِنُ لَا فِي سُنَّةِ التَّلَقِينَ* *almuʿīn lā fī sunnatu altalqīn*
by *شَيْخُ أَبُو إِسْحَاقَ إِبْرَاهِيمَ بْنِ مُحَمَّدٍ الشَّافِعِيِّ* *šīḥ ʿabū ʿishāq ʿibrāhīm bin muḥammad alšāfiʿī*. This includes some of the Prophet’s sayings.
8. Answer of the *شَيْخُ الْإِسْلَامِ الْكَمَالِيُّ أَبُو الْمُتَعَالِيِّ مُحَمَّدُ بْنُ أَبِي شَرِيفٍ* *šīḥ al-islām alkamālī ʿabū almasʿālī muḥammad ʿibn ʿabī šarīf* to the question addressed to him in the year 881 AH (c. 1454 CE) as to whether it is lawful for the inhabitants of a city visited by the Plague to flee from it, followed by 20 questions touching a form of prayer prescribed by the Prophet, may peace be upon him, for use against the Plague.

Fig 4.14 shows a script extracted from this manuscript.

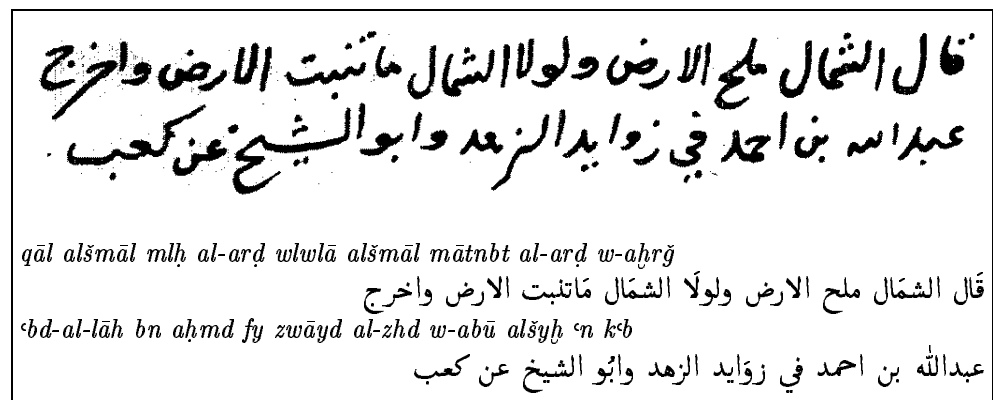


Figure 4.14: Sample text image extracted from the manuscript Add.3257, and the transliterated text.

Manuscript Description	Code
UL Moh194.a.4	A
UL Moh194.a.4 (footnotes)	B
UL Or.172	C
UL Or.1481	D
UL Qq.65	E
UL Qq.91	F
UL Qq.74	G
UL Add.2923	H
UL Add.3257	I
Simplified Arabic	J
Thuluth	K
Arabic Traditional	L
Andalus	M

Table 4.1: List of manuscript codes.

4.4 Summary

In this chapter we have seen a number of machine-printed text images and handwritten manuscripts. The extracted features from these illustrated styles are used for training and recognition purposes throughout the dissertation. Table 4.1 illustrates the code given to each manuscript which is later used to refer to that manuscript throughout the dissertation.

C H A P T E R 5

SPECTRAL FEATURES OF ARABIC WORDS

5.1 Introduction

There are two approaches to utilising word models in Arabic character recognition: the holistic approach and the analytic approach. The analytic approach is considered in Chapter 8. The next two chapters present two recognition systems based on the holistic approach. The approach does not require any kind of segmentation (refer to Sec 34 for different types of segmentations). Instead, it treats the word as a whole.

This chapter presents a method of extracting Fourier features from the Arabic word image. The method is a two step process: it first transforms the word image into a normalised polar image to compensate dilation and translation. It then applies a two-dimensional Fourier Transform to the polar image. The resulting spectrum magnitudes tolerate variations in size, displacement, and rotation.

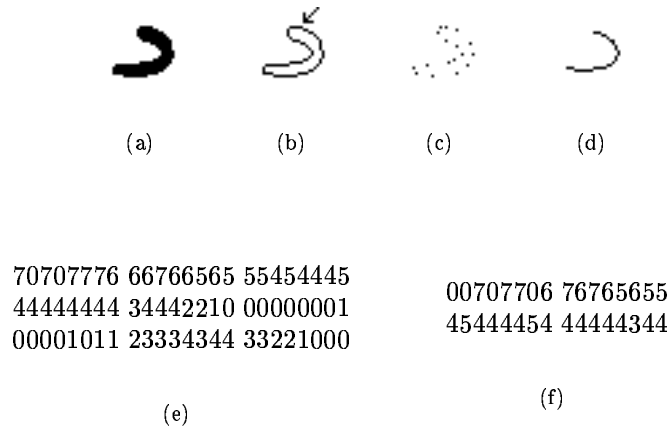


Figure 5.1: Two boundary descriptors: (a) original character image, (b) the character contour, (c) the skeleton, (d) the polygonal approximation, (e) the chain codes of (b) starting from the pixel marked with the arrow and moving in the clockwise direction, and (f) the chain codes of (d).

5.2 Two-dimensional Shape Description

The word image may be considered as a 2D shape, in this case the image pixels are represented and described in a form suitable for further computer processing. Two groups of algorithms are used for this purpose: the boundary descriptors, and the region descriptors.

5.2.1 Boundary Descriptors

These involve first detecting the edges of the character/word image. The edge is the transitional zone where a change from the foreground to the background, or vice versa, occurs. A closed curve boundary represents a contour, while an open curve boundary represents a skeleton. Following are some techniques that describe the boundary of the shape.

5.2.1.1 Chain codes

These describe the boundary pixels, the contour or the skeleton pixels, of a shape using Freeman codes [Fre61]. The representation is based on the 4- or 8-connectivity of the segments. The direction of each segment is coded by using a numbering scheme, see Fig 5.1. This compact representation saves a great amount of storage memory. This method has two drawbacks: it is sensitive to noise, and it is not invariant to rotation and dilation.

5.2.1.2 Polygonal approximation

This describes the contour or the skeleton pixels via a straight line sequence, see Fig 5.1. The method achieves a higher compression than the previous approach, however it still possesses the same drawbacks as the chain codes.

5.2.1.3 Signature

This reduces the dimensionality of the problem from a two-dimensional shape representation to a one-dimensional functional representation. The common approach is to calculate the centroid of a close boundary, then compute the distance from the centroid to the boundary pixels. Using this method for concave objects, like that found in typed script and handwritten script, is facing one of two obstacles: if the distance is calculated for each predefined value of the angle this will generate an ambiguous distance measure. Whereas if the distance is sampled along a constant curve length the distance measure is not 2π -periodic hence it is rotation variant.

5.2.1.4 Fourier descriptors

This method expresses each pixel (x, y) in the contour as a complex number $s = x + jy$. That is the x -axis is treated as the real axis and the y -axis as the imaginary axis. This reduces the problem dimensionality from 2D to 1D. Moreover applying the discrete fast Fourier transform to the resultant function produces the complex coefficients which are called the Fourier descriptors. The Fourier

Transform has an important feature which is that high-frequency components account for fine detail and low-frequency components determine the basic shape.

This method has been successfully applied to Arabic character recognition [EG88, Mah94]. However, it assumes that a reliable segmentation of the word is obtainable, which in practice is not the case. Furthermore, while FDs are suitable for the simple curves of characters, FDs cannot be applied directly to the entire word because of the complexity of the curve that connects all the characters.

5.2.1.5 UNL Transform

This method describes a 2D shape which is composed of parametric curves [RSG92, Rau94]. The method seeks a translation normalisation by shifting the centroid of the object to the origin of the polar coordinate system. However the UNL Transform requires that the objects are composed of a finite set of smooth parametric curves or small line segments. In character/word recognition these curves are obtained through applying a thinning or skeletonisation algorithm. This means that this method is sensitive to any noise that may affect the output of the thinning algorithm.

5.2.2 Region Descriptors

These represent the shape of its interior region. A simple approach to describing the interior region is to use the two descriptors: the area and the perimeter. The area is the number of pixels contained within the boundary of a region, and the perimeter is the length of the region boundary. This approach is applied to objects with invariant size. Two more sophisticated methods are: moments and log-polar transform.

5.2.2.1 Moments

These were first introduced by Hu [Hu62]. A representative application of moments to recognise Arabic characters can be found in [ERK90, EA90, AB94, EAIK94, San96].

For a 2D-shape image the moment of order $(p + q)$ is defined as

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y) \quad (5.1)$$

The central moments can be expressed as

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y) \quad (5.2)$$

where

$$\bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}}$$

The normalised central moments, denoted η_{pq} are defined as

$$\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^\gamma} \quad (5.3)$$

where

$$\gamma = \frac{p + q}{2} + 1 \quad p + q = 2, 3, \dots$$

In [GW92], a set of seven *invariant moments* were derived from the second and third moments. These moments tolerate changes in rotation, translation, and displacement.

5.2.2.2 Log-polar Transform

This approach is based on a previous work done by Schwartz [Sch77] in which he analysed the topology of the human visual system. Each pixel of the 2D shape is treated as a complex number $z = x + jy$. The approach is then based on the property of the logarithm of a complex number $\ln(z) = \ln(|z|) + j\theta + 2jk\pi$, where k is an integer number. Normally the principal part of the logarithm is used ($k = 0$)

$$z = |z|e^{j\theta} \quad (5.4)$$

$$|z| = \sqrt{x^2 + y^2} \quad (5.5)$$

$$\theta = \tan^{-1}\left(\frac{y}{x}\right) \quad (5.6)$$

The main feature of this transform is that both scaling and rotation of the 2D shape are transformed into a translation despite the

fact that the Log-polar Transform is not scale invariant. This is due to the stretching effect that can be observed when transforming the same object in different sizes.

5.3 Normalised Polar Image

The objective is to find a feature set for the word which is invariant to the Poincaré group of transformations: dilation, translation, and rotation. This set must compensate for all these transformations in order to be able to map all possible variations of a shape to the same prototype representation. At the same time this prototype should be capable of distinguishing between similar shapes. The first step is to construct a normalised polar image to compensate for dilation and translation.

5.3.1 The Centroid Of The 2D Shape

This is also called the center of gravity. The centroid is the balance point of the image $f(x, y)$ such that the mass of $f(x, y)$ left and right of O_x and above and below O_y is equal. The mathematical representation of the centroid is

$$O_x = \frac{m_{10}}{m_{00}} \quad O_y = \frac{m_{01}}{m_{00}} \quad (5.7)$$

where m_{00} is the sum of the pixel values of an image, for a binary image this equals to the number of black pixels in the image, m_{10} and m_{01} are defined as the first-order row and column moments respectively. m_{00} , m_{10} and m_{01} are calculated as in Eq 5.1.

The relative position of the centroid to the shape pixels does not change if the shape is translated within the image scene. Therefore the shape representation in the polar coordinates with the centroid as the origin of the shape pattern is insensitive to shape translation.



Figure 5.2: Two images of the character *Noon*. The centroid of each image is marked with a cross \times .

5.3.2 Maximum Distance Computation

A 2D shape may have a more robust representation using *relative* dimensions. These are calculated relative to the shape centroid. A pixel (x, y) is represented based on the centroid and the *maximum distance* of all shape pixels from that centroid.

D_{max} is the maximum Euclidean distance between the centroid, O , and all pixels

$$D_{max} = \max_{x,y} \left(\sqrt{(x - O_x)^2 + (y - O_y)^2} \right)$$

then a coordinate transform, \mathbf{U} , for each pixel exists which maps the 2D-shape from the Cartesian to the polar coordinate system with origin O .

$$\begin{aligned} \mathbf{U}(x, y) &= (r, \theta) \\ &= \left(\frac{\sqrt{(x - O_x)^2 + (y - O_y)^2}}{D_{max}}, \tan^{-1} \left(\frac{y - O_y}{x - O_x} \right) \right) \\ &\quad 0 \leq r \leq 1, -\pi \leq \theta \leq \pi \end{aligned}$$

The polar image is then normalised to the size $N \times N$. In this dissertation N is chosen to be 64, so the polar image has 64×64 pixels.

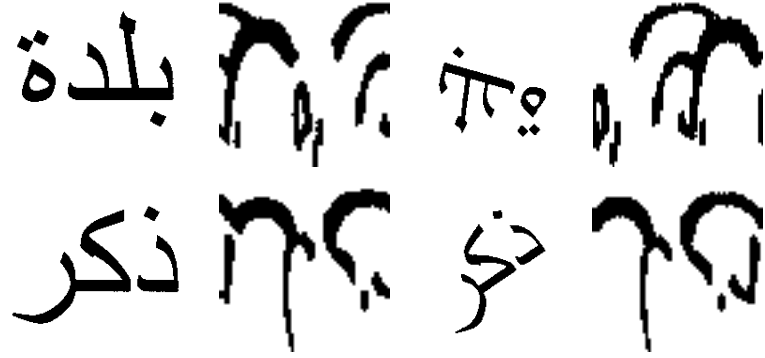


Figure 5.3: Normalised polar images for a number of Arabic words written in Simplified Arabic computer-generated font. The words are: 'بلدة *baladt* - Town', 'ذكر *dkr* - Male'

The resulting polar image of the 2D shape is invariant to changes in size and displacement. Moreover, the polar image is periodic; if the original shape is rotated by an angle κ in a clockwise direction this results in translating the polar image horizontally with the same angle to the right.

Figures 5.3- 5.6 show the result of applying this transformation to different font faces. They illustrate that the polar images are invariant to changes in size and displacement. In addition, they transform a rotation in the original word image with an angle κ into a horizontal displacement with the same angle. The illustrated word images are typewritten samples. They are printed using four different fonts: Simplified Arabic, Thuluth, Andalus, and Arabic Traditional. The samples are rendered at random angles ranging $0 \rightarrow 2\pi$, at random sizes ranging between 18 – 48pt, and at random translations. Handwritten samples are extracted from the manuscripts previously mentioned in Chapter 4.

Figures 5.7 and 5.8 show the word images of two words taken from different manuscripts and computer-generated font faces, beside each word image is the normalised polar image.

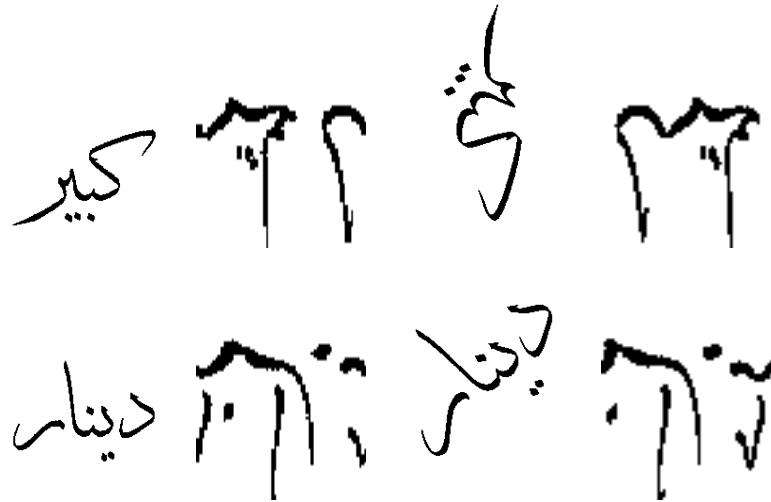


Figure 5.4: Normalised polar images for a number of Arabic words written in Thuluth computer-generated font. The words are: 'كبير' *kbīr* - Big', 'دينار' *dīnār* - Dinar'

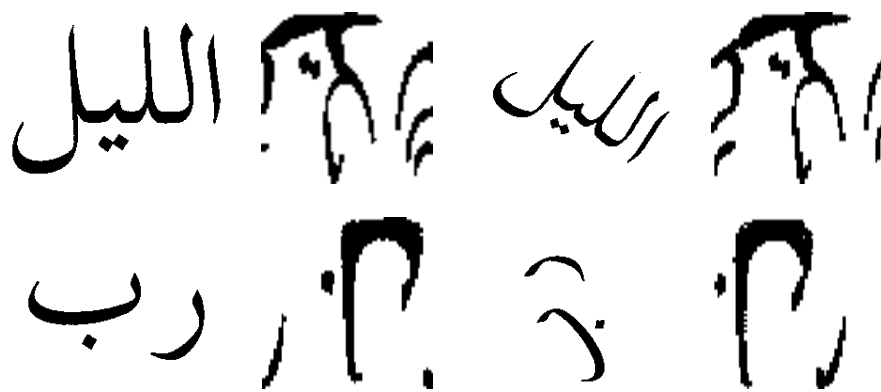


Figure 5.5: Normalised polar images for a number of Arabic words written in Arabic Traditional computer-generated font. The words are: 'الليل' *al-layl* - The night', 'رب' *rb* - Lord'



Figure 5.6: Normalised polar images for a number of Arabic words written in Andalus computer-generated font. The words are: 'رسول' *rsūl* - Prophet', 'مصر' *mṣr* - Egypt'

5.4 Spectral Features

In this section the Fourier transform is applied to the normalised polar image which resulted in the previous section. The objective is to produce a Fourier coefficient set which is invariant to changes in size, displacement, and rotation.

The Fourier Transform [Par97] is one of the most important tools for signal processing. The application of the Fourier Transform to an image $g(x, y)$ can be represented as

$$\mathbf{G}(w_x, w_y) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} g(x, y) e^{-j \frac{2\pi}{N} (xw_x + yw_y)} \quad (5.8)$$

$$0 \leq w_x, w_y < N$$

The Fast Fourier Transform is a practical way to compute the Fourier Transform of a function [GW92]. It represents a significant saving in computation complexity by reducing the required operations from N^2 down to $N \log_2 N$.

In the previous section, it was shown that any rotation in the word image is translated in the normalised polar image. Consider

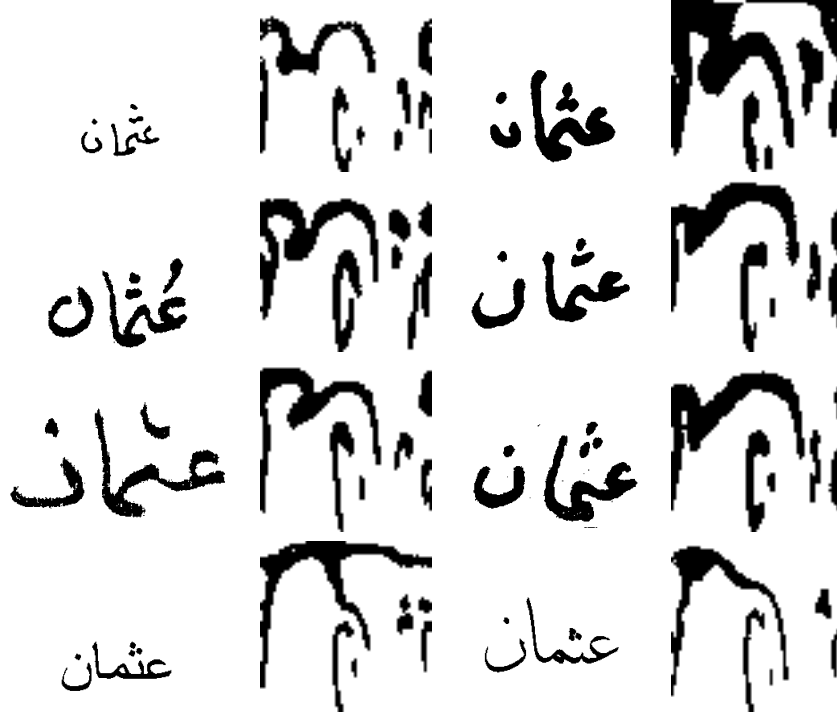


Figure 5.7: Normalised polar images of the Arabic word 'عثمان' *ʿṭmān* - OTHMAN' written in different handwritten and computer-generated fonts.

the translation property of the Fourier Transform which is represented as

$$g(x - x_0, y - y_0) \xrightarrow{\mathcal{F}} G(w_x, w_y) e^{-j \frac{2\pi}{N} (x_0 w_x + y_0 w_y)} \quad (5.9)$$

This shows that a shift in the origin of the function $g(x, y)$ results in multiplying $G(w_x, w_y)$ by the indicated exponential term. Take the magnitude

$$\begin{aligned} |G(w_x, w_y)| |e^{-j\vartheta}| &= |G(w_x, w_y)| |\cos \vartheta - j \sin \vartheta| \\ &= |G(w_x, w_y)| [\cos^2 \vartheta + \sin^2 \vartheta]^{\frac{1}{2}} \\ &= |G(w_x, w_y)| \end{aligned} \quad (5.10)$$

therefore a shift in $g(x, y)$ does not affect the magnitude of its Fourier Transform. The resultant Fourier spectrum of the normalised polar



Figure 5.8: Normalised polar images of the Arabic word 'النبي' *al-nabī* - The Prophet' written in different handwritten and computer-generated fonts.

image is invariant to any changes in size, displacement, and rotation in the original word image.

Since the normalised polar image is a real signal the Fourier Transform of this image exhibits *conjugate symmetry* [GW92]

$$\mathbf{G}(w_x, w_y) = \mathbf{G}^*(-w_x, -w_y) \quad (5.11)$$

where $\mathbf{G}^*(w_x, w_y)$ is the complex conjugate of $\mathbf{G}(w_x, w_y)$. This leads to a more interesting note

$$|\mathbf{G}(w_x, w_y)| = |\mathbf{G}(-w_x, -w_y)| \quad (5.12)$$

This means that half the Fourier spectrum is sufficient to reconstruct the image completely.

Figures 5.9, 5.10, and 5.11 show that the most discriminant Fourier coefficients are located near the origin. The origin itself is represented by the basic signal energy of the pattern, $G(0, 0)$. This magnitude is referred to as the DC coefficient because no imaginary part is included. $G(0, 0)$ is proportional to the average value of the image $g(x, y)$. A normalisation process to the Fourier spectrum is performed by dividing all Fourier spectrum magnitude values by the DC coefficient $G(0, 0)$. Since $G(0, 0)$ has the highest value, the normalisation process scales all Fourier spectrum magnitudes to the interval $[0, 1]$.

5.5 Summary

A method of extracting Fourier features from a word image has been presented. The method transfers the word image into a normalised polar image in three steps. Firstly, it computes the image centroid. Then, it calculates the maximum distance between all pixels and the centroid. Finally, it maps each image pixel into the polar coordinates using the centroid and the maximum distance values.

To obtain the Fourier features, the method applies the two-dimensional Fourier Transform to the polar image. This results in Fourier spectrum magnitudes which are invariant to changes to displacement, size, and rotation.

The following two chapters use these features in order to design two systems used to recognise Arabic words. These systems are based on template matching and hidden Markov models.

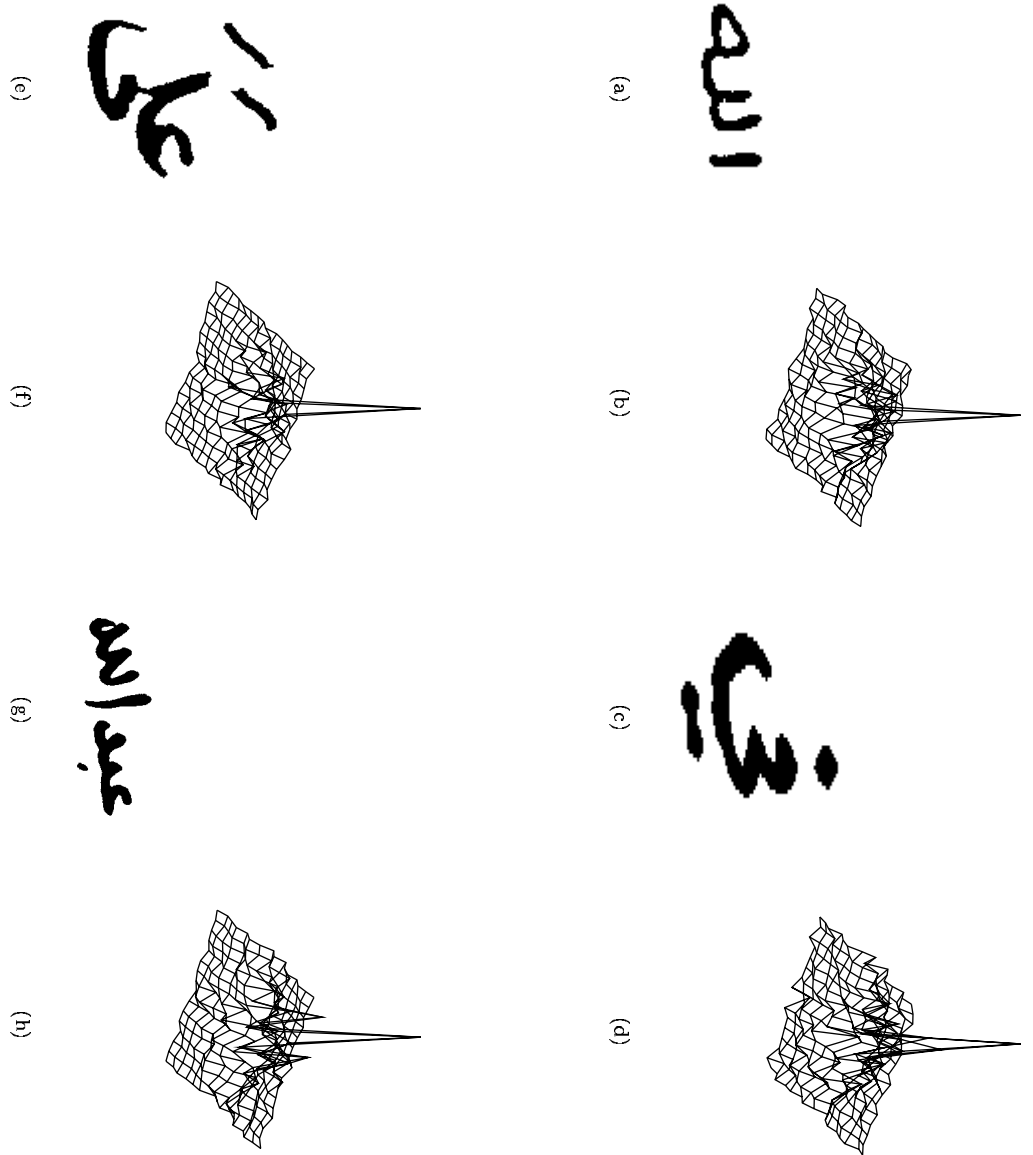


Figure 5.9: Word samples represent the words: (a) 'الله' *al-lh - God*, (c) 'في' *fy - In*, (e) 'علي' *'a - On*, (g) 'عبدالله' *'bdal-lh - Abdullah*, (b),(d),(f), and (h) are the normalised Fourier spectrums of (a),(c),(e), and (g), respectively.

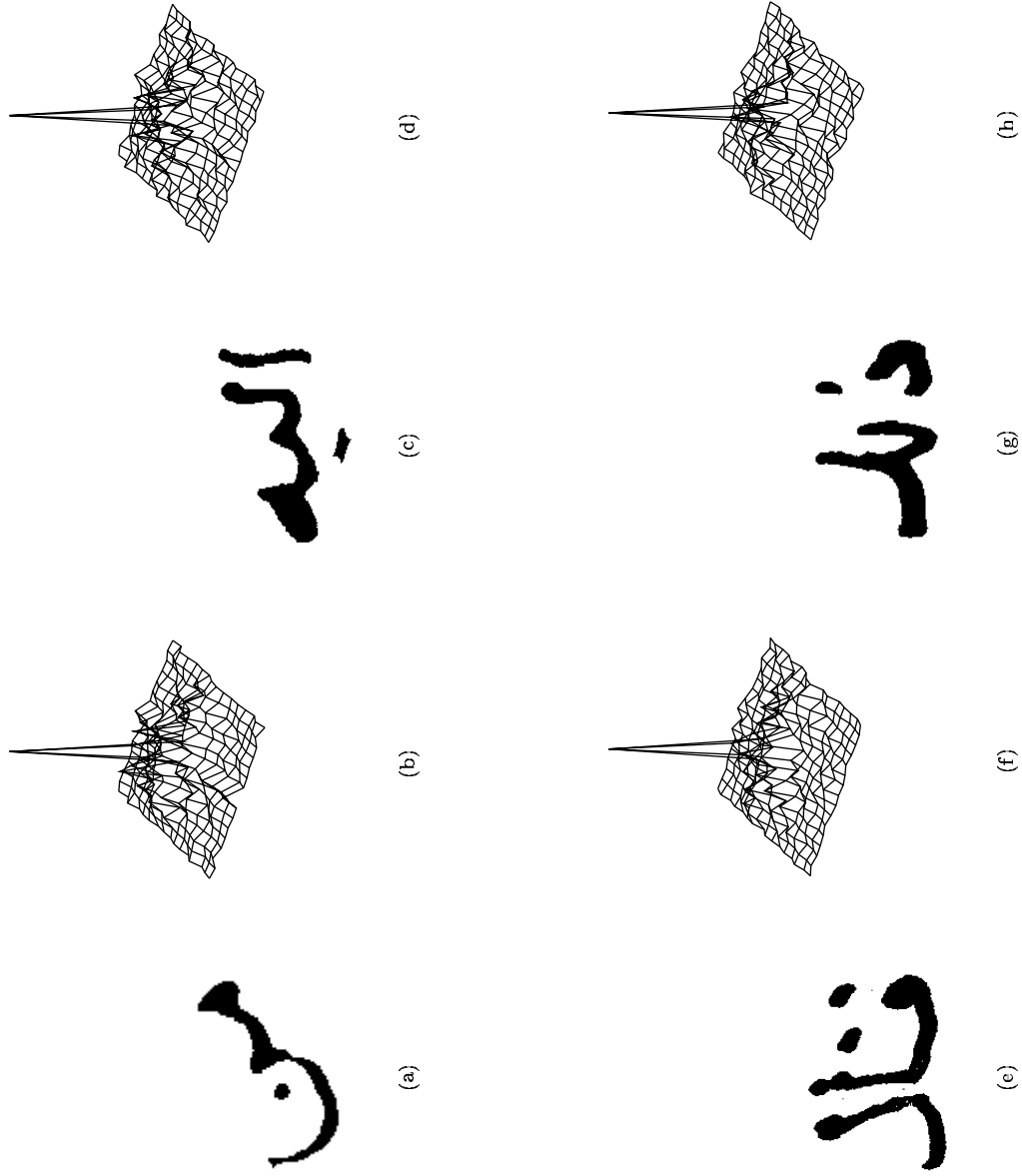


Figure 5.10: Word samples represent the words: (a) 'min - From', (c) 'ilayh - To/for him', (e) 'qal - Said', (g) 'dal - That', (b),(d),(f), and (h) are the normalised Fourier spectra of (a),(c),(e), and (g), respectively.

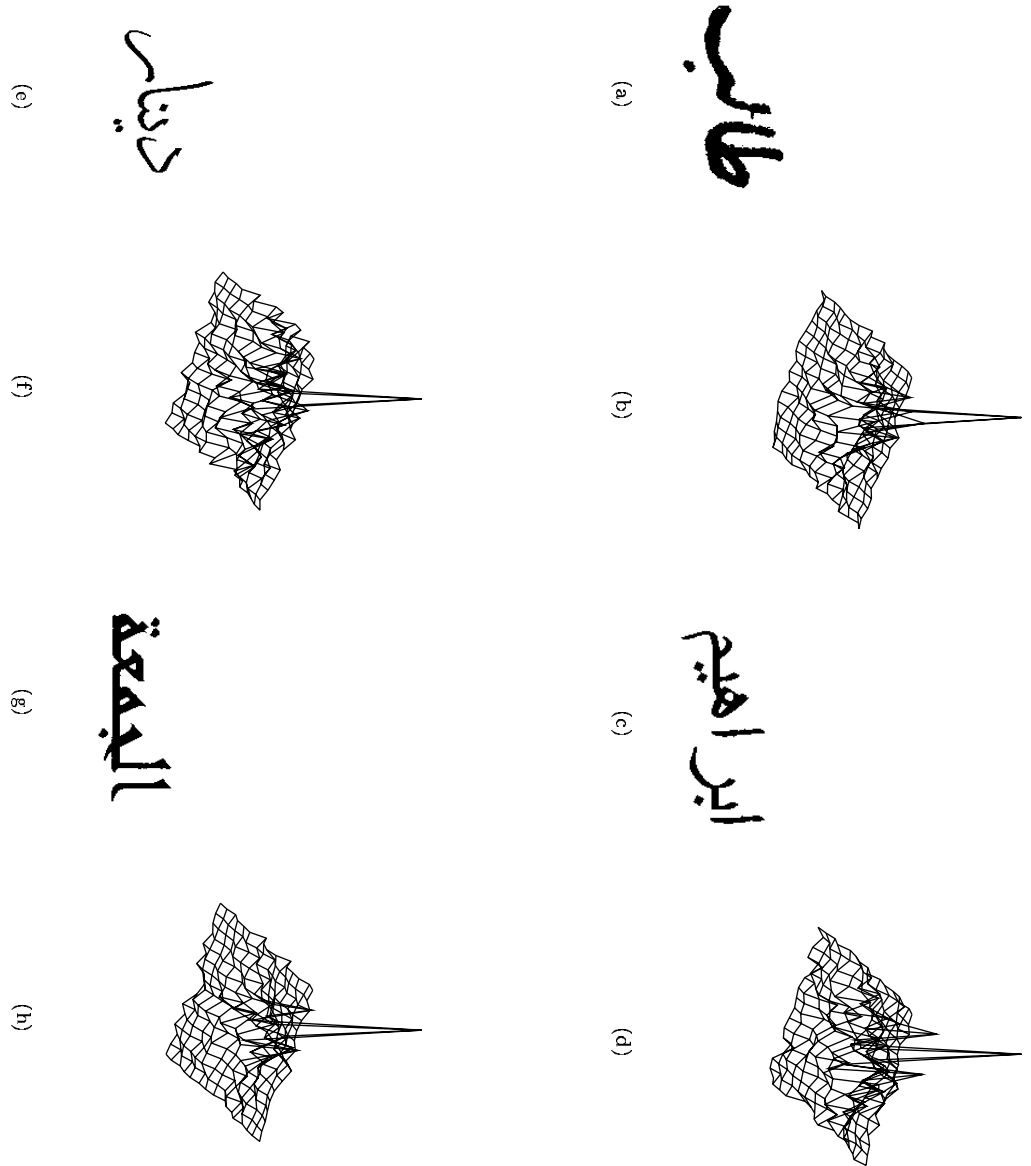


Figure 5.11: Word samples represent the words: (a) 'طالب' *talb* - Student', (c) 'إبراهيم' *ibrahim* - Abraham', (e) 'دینار' *dinar* - Dinar', (g) 'الجمعة' *al-jum'at* - Friday', (b),(d),(f), and (h) are the normalised Fourier spectrums of (a),(c),(e), and (g), respectively

CHAPTER 6

TEMPLATE-BASED RECOGNITION SYSTEM

6.1 Introduction

One of the most common modules in the pattern matching approach is the method for constructing reference patterns. Each reference pattern may be represented by a *prototype* or *template* vector. The recognition process is then based upon a certain measure which is computed relative to these reference patterns.

In this chapter, a template-based word recognition system is presented. The system is designed for a finite lexicon and each word in this lexicon is represented by a single prototype that only includes the lower frequencies of the Fourier spectrum. Recognition is based on two distance measures from these prototypes. The applied distance measures are the Euclidean distance and the Standard distance. The method is applied to modern multi-font manuscripts as well as historical handwritten manuscripts.

6.2 Pattern-Comparison Techniques

An essential consideration in word recognition is how different word patterns are compared in order to determine their similarity. De-

pending on the recognition system specifications, pattern comparison can be completed in a wide variety of ways. The common factor between most pattern comparison techniques is a prescribed measurement of dissimilarity between two feature sets. This measurement of dissimilarity can be handled mathematically if the patterns are visualised in a feature space.

The most commonly used technique is that of distance measures which calculates the distance between the input feature set and each of the reference feature sets in the database in the n -dimensional feature space.

6.2.1 Euclidean distance measure

This is the simplest distance measure. This assigns the input feature set to its closest reference feature set. Using Euclidean distance to determine closeness reduces the problem of recognition in computing the distance measures.

The Euclidean distance between two points i and j in n -dimensional space is

$$\begin{aligned} \mathbf{ED}_{ij} &= \|\mathbf{X}_i - \mathbf{X}_j\| \\ &= \sqrt{\sum_{k=1}^n (x_{ik} - x_{jk})^2} \end{aligned} \quad (6.1)$$

The Euclidean distance performs well when the variance in each feature is approximately the same since it assigns equal weight to each dimension in the feature space.

6.2.2 Statistical distance measure

This is also called the *standard distance*. This may be regarded as an adjusted Euclidean distance measure. The distance is obtained by normalising the sample variance. Consider a one-dimensional feature vector where the standard distance measure between two

points i and j is given by

$$\mathbf{SD}_{ij}^2 = \frac{|x_i - x_j|}{\sigma} \quad (6.2)$$

This may be extended to the n -dimensional feature space

$$\begin{aligned} \mathbf{SD}_{ij}^2 &= \frac{|\mathbf{X}_i - \mathbf{X}_j|}{\sigma} \\ &= \sum_{k=1}^n \frac{|x_{ik} - x_{jk}|}{\sigma_k} \end{aligned} \quad (6.3)$$

6.3 Template Forming

The presence of the discriminant coefficients around the origin allows a priori reduction in the Fourier coefficients by bounding the range to the low frequencies. These coefficients have low values for w_x and w_y , and they fall in a small *frequency band*, of an annular-shape, within the Fourier spectrum. In this chapter, two sampling densities are considered, see Fig 6.1: the $Band_8$ and the $Band_{16}$. The $Band_8$ includes half the Fourier spectrum coefficients which fall within a radial frequency $\frac{N}{8}$. Since N was chosen to be 64 there are 106 coefficients in this band. The $Band_{16}$ includes half the Fourier spectrum coefficients which fall within a radial frequency $\frac{N}{4}$. There are 402 coefficients in this band. In both cases the same density is used for all orientations. As shown in the figure only half the band is considered. This is due to the conjugate symmetry mentioned in Eq. 5.12.

In this section, we discuss how reference patterns are created so that the pattern matching techniques can be properly applied to achieve the best results.

6.3.1 A Single-Font Case

Only one font, whether typewritten or handwritten, is used to form the prototype of the word. This is done by selecting one of the normalised Fourier spectrum magnitude samples representing a word

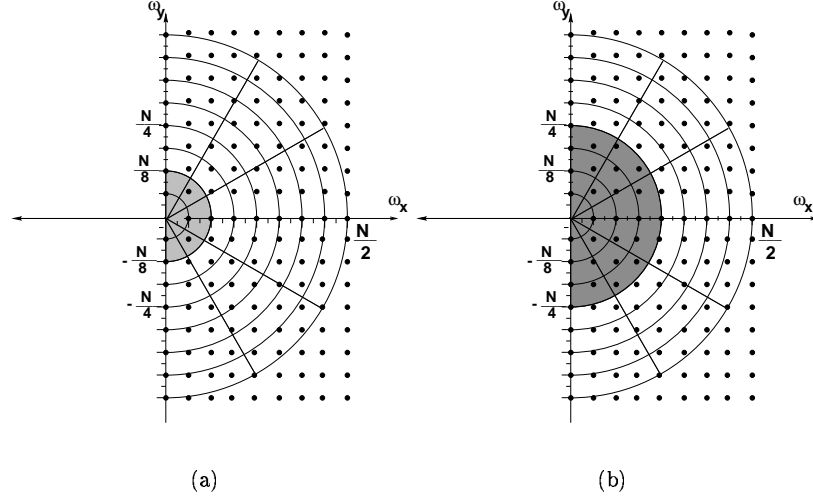


Figure 6.1: The Fourier spectrum decomposition into smaller frequency bands, ($N = 64$): (a) $Band_8 \rightarrow \frac{N}{8} = 8$, (b) $Band_{16} \rightarrow \frac{N}{4} = 16$. These are sub-samples of a factor of four of the real illustrations.

w_i , $i = 2, \dots, W$, where W is the lexicon size. In the experiments described below, $W = 145$.

6.3.2 Multiple Font Case

Each word in the lexicon is represented by a prototype formed from the average coefficient values of a sample of training words.

$$\mu_i = \frac{1}{v_i} \sum_{\mathbf{x} \in \mathbf{w}_i} \mathbf{x} \quad i = 1, 2, \dots, W \quad (6.4)$$

where v_i is the number of word images representing word \mathbf{w}_i and is used to calculate this template. This is sufficient for Euclidean distance measure. For the Standard distance measure, the *standard deviation* is calculated to normalise the distance.

$$\sigma_i = \frac{1}{v_i} \left[\sum_{\substack{i=1 \\ \mathbf{x} \in \mathbf{w}_i}}^{v_i} (\mathbf{x} - \mu_i)^2 \right]^{\frac{1}{2}} \quad i = 1, 2, \dots, W \quad (6.5)$$

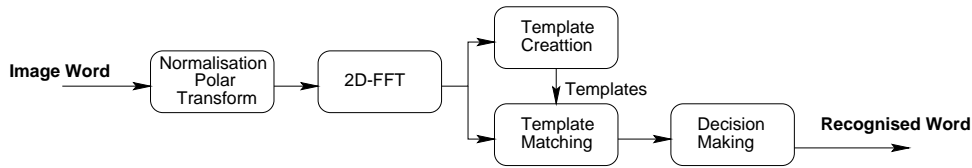


Figure 6.2: A block diagram of a conventional template-based pattern recognition system.

6.4 Evaluation Experiments and Results

We have discussed a method of creating templates for words in the vocabulary and two techniques to compare word patterns. Fig 6.2 shows the block diagram of the word recognition system. However the performance of the system depends on the distance measures, and the manner in which the word templates are created.

6.4.1 Typewritten Sample Results

More than 1700 samples representing 145 words were used to assess the performance of the proposed method. The samples were printed using four different computer-generated fonts: Simplified Arabic, Thuluth, Andalus, and Arabic Traditional referred to in Chapter 4 as fonts *J*, *K*, *L*, and *M*, respectively. The samples were rendered at random angles ranging from 0 to 2π , at random sizes ranging between 18pt and 48pt, and at random translations up to twice the size of the sampled word.

A number of experiments were performed. In each experiment, a template for each word in the lexicon was selected randomly from those images representing that word in one font. The templates were then used to recognise word images from all fonts based on the Euclidean distance measure. Table 6.1 shows the results of these experiments. While each template could successfully recognise word images of the same font invariant to dilation, rotation, and translation, it performed poorly with other fonts. This is not surprising

Testing Fonts		Training Fonts			
		Simplified	Thuluth	Traditional	Andalus
Simplified	<i>Band₈</i>	95.86	10.80	14.71	22.30
	<i>Band₁₆</i>	96.09	10.57	13.56	23.45
Thuluth	<i>Band₈</i>	12.64	99.31	21.38	05.29
	<i>Band₁₆</i>	17.24	99.08	24.60	05.06
Traditional	<i>Band₈</i>	28.28	19.54	94.25	12.41
	<i>Band₁₆</i>	28.28	19.08	94.25	11.95
Andalus	<i>Band₈</i>	12.64	02.99	05.06	98.85
	<i>Band₁₆</i>	10.80	03.91	04.60	98.62

Table 6.1: Recognition rates (%) of the first set of experiments. Each experiment used a single font to build the template database.

No. of words	Template size	Top 1		Top 5		Top 10	
		ED	SD	ED	SD	ED	SD
50	<i>Band₈</i>	86.46	84.38	96.04	92.29	97.50	95.00
	<i>Band₁₆</i>	90.00	67.50	96.25	80.63	96.88	88.33
100	<i>Band₈</i>	83.25	78.08	94.00	90.25	96.42	92.33
	<i>Band₁₆</i>	85.50	62.67	94.33	76.08	96.42	81.75
145	<i>Band₈</i>	82.73	77.14	92.92	89.94	95.65	92.44
	<i>Band₁₆</i>	85.24	62.26	93.21	74.58	95.77	79.40

Table 6.2: Recognition rates (%) of typewritten word samples.

since the template did not extract any features from other fonts.

A second set of experiments was performed with word templates calculated by averaging four normalised word Fourier spectrum magnitude values. The recognition of an input word was based on two distance measures: the Euclidean (ED) and the Standard (SD) distance measures. The normalised Fourier spectrum magnitudes of the input word were compared against all templates in the database, and the word was then assigned to the word template with the minimum distance. Table 6.2 shows the recognition rate results of more than 1700 typewritten word samples with templates calculated by averaging *Band₈* and *Band₁₆* frequency bands extracted from the normalised Fourier spectrum magnitudes. The recognition decision is based on the minimum Euclidean distance measure.

Fig 6.3 summarises the error rates when the Euclidean and the Standard distance measures are used to compare an input word sample with every template. Four curves are shown: two in which $Band_8$ was used to construct the templates, and the other two where $Band_{16}$ was used. The word was assumed to be unrecognised when it does not appear among the top ten possible alternatives of the word. The error rate, hence, is the number of unrecognised words divided by the total number of input words.

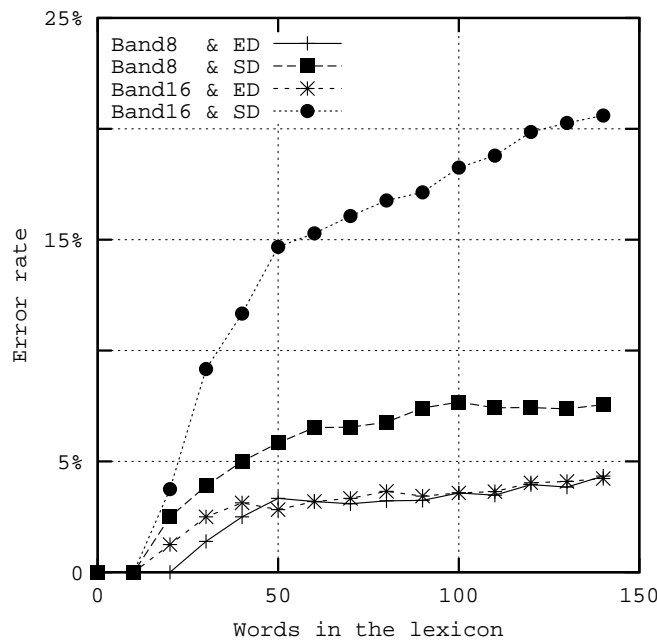


Figure 6.3: Error rate against lexicon size for typewritten fonts.

Fig 6.3 shows that based on the Standard distance measure the templates formed using $Band_{16}$ coefficients showed lower performance than those formed using $Band_8$ coefficients. This observation concurs with general experience that some classification functions seem much more sensitive to the feature dimensionality than others. It should also be noted that because templates are formed from samples written in multiple fonts, the style variations and consequently σ are large. This has a dominant effect on the SD measure.

Testing Fonts		Top 1	Top 5	Top 10
Font <i>J</i>	<i>ED</i>	90.24	97.38	98.81
	<i>SD</i>	61.67	74.76	80.00
Font <i>L</i>	<i>ED</i>	95.71	100.0	100.0
	<i>SD</i>	56.43	69.29	74.76
Font <i>K</i>	<i>ED</i>	92.38	97.38	98.57
	<i>SD</i>	58.57	67.14	72.62
Font <i>M</i>	<i>ED</i>	62.62	97.38	98.81
	<i>SD</i>	72.38	87.14	90.24

Table 6.3: Recognition rates (%) of the four computer-generated fonts of a 145-word lexicon. Templates are $Band_{16}$ size.

Vocabulary size in words	Top 1		Top 5		Top 10	
	ED	SD	ED	SD	ED	SD
50	93.89	59.44	98.61	75.00	99.44	83.06
100	93.22	56.89	98.11	68.56	98.89	74.78
145	92.22	53.97	97.62	65.87	98.17	71.35

Table 6.4: Recognition rates (%) of three computer-generated fonts: *J*, *K*, and *L*. Templates are $Band_{16}$ size.

To understand the difference in performance when using Euclidean or Standard distance measures, each computer-generated font was investigated separately in Table 6.3 using the same word templates used to report the results of Table 6.2. Table 6.3 shows that the first three fonts *J*, *K*, and *L* perform better when using the Euclidean distance measure where font *M* has a higher recognition rate than when using the Standard distance measure which includes the standard deviation σ in the calculations. This indicates that the first three fonts have a similar shape structure. To check the validity of this claim, another experiment was performed where each word template is formed by averaging one Fourier spectrum magnitude set from each of the three fonts *J*, *K*, and *L*. Table 6.4 shows the recognition rates for different vocabulary sizes. For a 145-word lexicon an increase of 7% in recognising the exact word is achieved when discarding font *M*.

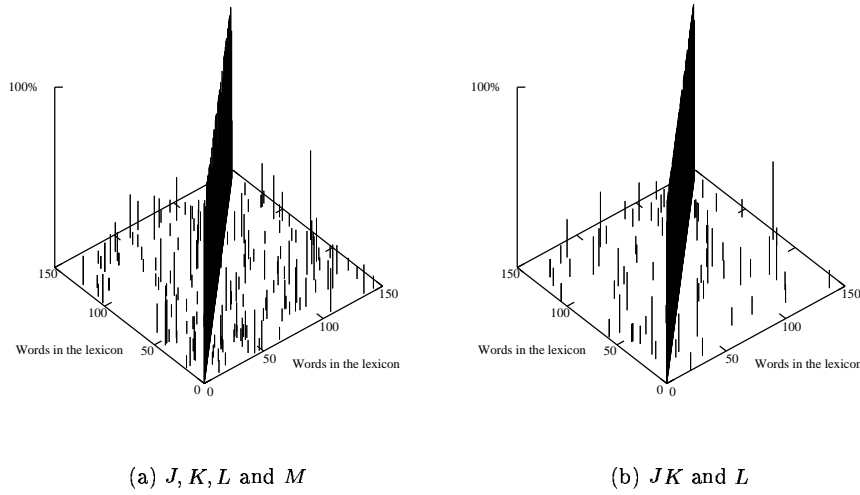


Figure 6.4: Three-dimensional depictions of confusion matrices for two different font sets, subjected to: (a) 252300 recognition tests, and (b) 189225 recognition tests.

Fig 6.4 shows the confusion matrix depictions of these two situations. It illustrates that the first situation which covers the fonts *J, K, L* and *M* has a lower recognition rate than the second situation which only covers the fonts *J, K* and *L*. $Band_{16}$ template size was used in both situations to build word templates, and the recognition was based on the minimum ED measure.

6.4.2 Single Handwritten Font Results

In Chapter 4, a number of manuscripts were illustrated which are stored in the Cambridge University Library (UL). Sample words extracted from those manuscripts were used to build the template database. At the beginning each handwritten font was tested separately to have an indication of how consistent the writing style of that font is. Nine experiments were performed on samples representing 145 words and extracted from the nine manuscripts. In

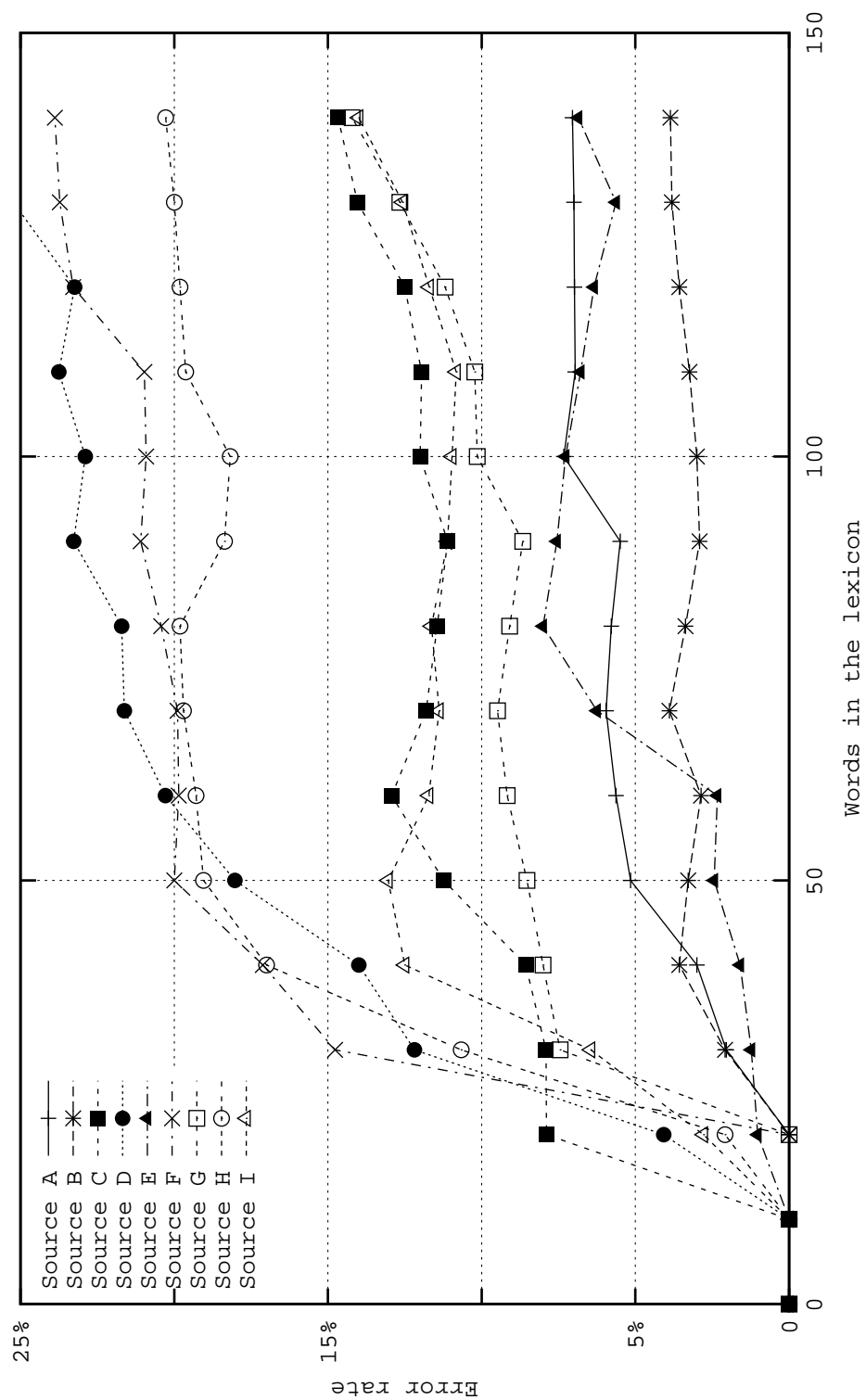


Figure 6.5: Error rate against lexicon size of handwritten fonts. Templates are $Band_8$ size.

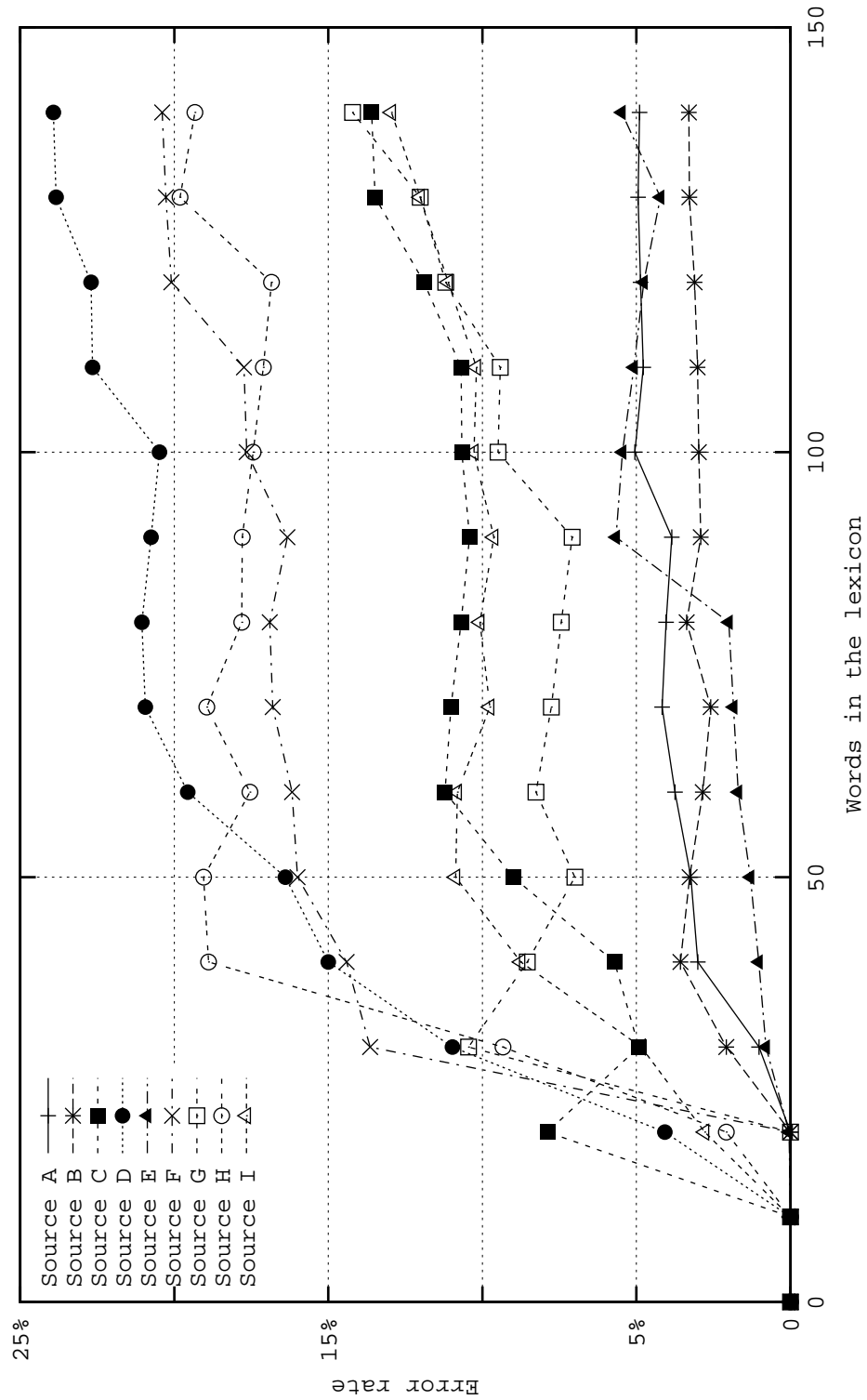


Figure 6.6: Error rate against lexicon size of handwritten fonts. Templates are $Band_{16}$ size.

Template size	Top 1	Top 5	Top 10
$Band_8$	84.06	93.52	95.99
$Band_{16}$	85.51	93.61	96.16

Table 6.5: Recognition rates (%) of the four selected fonts: A , B , J , and L of a 145-word lexicon. Euclidean distance is used as a dissimilarity measure.

each experiment a template for each word in the lexicon was selected randomly from word images in one handwritten manuscript. The templates were then used to recognise word images from the same font.

According to figures 6.5 and 6.6 the nine fonts can be divided into five groups relative to the recognition error rate percentage. Font B shows the best performance among the nine which indicates that it has the most consistent writing style amongst all the handwritten fonts. Fonts A and E follow, then fonts C , G , and I come third, and in fourth place come fonts F and H . Finally, in fifth place and with an error rate of more than 25%, font D represents the most inconsistent writing style among all the manuscripts.

6.4.3 Four-Font Experiments

Two experiments were performed on four selected fonts: two fonts were handwritten A and B , and the other two were computer-generated fonts J and L . Fonts A and B were selected because they have the best consistent writing style amongst the nine handwritten fonts as previously illustrated.

The word templates were formed by calculating the average values of the Fourier spectrum magnitudes of one randomly selected word sample from each selected font. Table 6.5 illustrates the recognition rates for these experiments. The table shows an increase of less than 2% in the recognition rate when using $Band_{16}$ instead of $Band_8$. Table 6.6 shows the detailed recognition rate of each font separately.

Testing Fonts		Top 1	Top 5	Top 10
Font <i>A</i>	<i>Band</i> ₈	65.22	81.30	87.83
	<i>Band</i> ₁₆	66.09	81.74	88.70
Font <i>B</i>	<i>Band</i> ₈	79.61	87.38	92.20
	<i>Band</i> ₁₆	79.61	89.32	92.23
Font <i>J</i>	<i>Band</i> ₈	86.67	96.43	98.33
	<i>Band</i> ₁₆	88.33	96.43	98.10
Font <i>L</i>	<i>Band</i> ₈	92.86	98.81	99.05
	<i>Band</i> ₁₆	94.76	99.05	99.29

Table 6.6: Recognition rates (%) of each of the four selected fonts: *A*, *B*, *J*, and *L* of a 145-word lexicon.

6.4.4 All-Font Experiments

Two experiments were performed using word templates calculated by averaging the Fourier spectrum magnitudes of one randomly selected word sample from each font, handwritten or computer-generated, as in Eq. 6.4. Thirteen word samples were used to calculate a word template. The inconsistent writing styles of most of these fonts produced distorted word templates which were not similar to any of the fonts. Moreover, these word templates performed poorly for both handwritten and computer-generated fonts.

Fig 6.7 summarises the recognition error rate of *Band*₈ and *Band*₁₆ templates. Four curves are shown for the Euclidean and the Standard distance measures. This figure proves that although the spectral features are well defined between different words they are not suitable to be used with a template-based recogniser, especially if the input word samples are handwritten. This requires a kind of learning method to be able to adopt to variant writing styles.

6.4.5 Single-Font Performance

The performance of two different fonts were investigated: font *A* and font *L*. Font *A* is a handwritten font with a good degree of consistency in writing style. Font *L* is an Arabic Traditional computer-generated font. The comparison took place when word templates

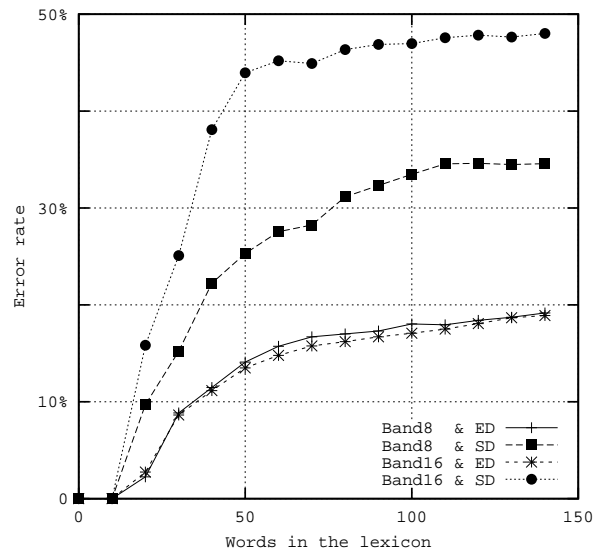


Figure 6.7: Error rate against lexicon size. Word templates are formed from all fonts mentioned in Chapter 4.

were formed from the font itself, with a limited number of fonts, and with all the available fonts.

6.4.5.1 Font A performance

This considers three different cases. In case one each word template was formed by randomly selecting one sample from those samples representing that word. This gives us an indication of the consistency of the writing style, as previously discussed. In case two four fonts were included in forming word templates, two were handwritten and two were computer-generated. Each word template was calculated by averaging four word samples that were randomly selected from those samples representing that word, one sample from each font. In case three all fonts previously mentioned in Chapter 4, were included in calculating word templates.

Fig 6.8 illustrates the recognition rate of word samples from font A in the three cases. It is obvious that case one represents the

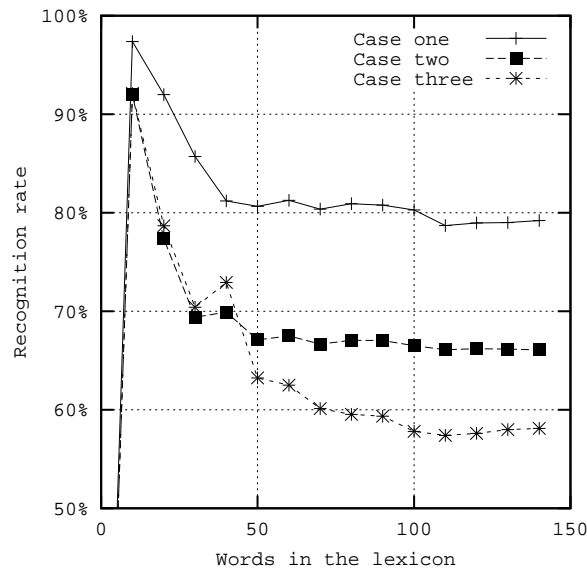


Figure 6.8: Font *A* recognition rate against lexicon size.

highest rate among the three since the word templates were not distorted by any other font which might have a different word shape. Case two and case three represent the same concept in which the word templates do not represent any of the participating fonts. They otherwise represent a distorted version, the average, of all of them. The effect of this is clearer in case three since more fonts participate in calculating the average values.

6.4.5.2 Font *L* performance

This considers four different cases. In case one each word template was formed by randomly selecting one sample from those samples representing that word. This gives us an indication of the consistency of the writing style, as previously discussed. In case two three computer-generated fonts were included in forming word templates: *J*, *K*, and *L*. Each word template was calculated by averaging three word samples that were randomly selected from those samples representing that word, one sample from each font. Case three was

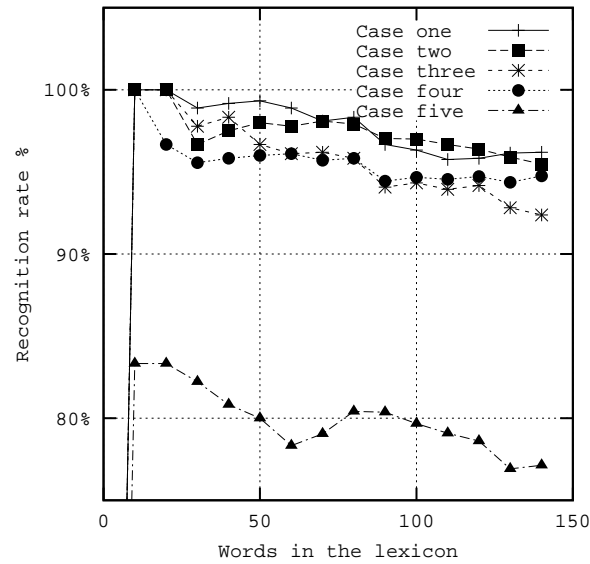


Figure 6.9: Font *L* recognition rate against lexicon size.

similar to case two except that one more computer-generated font was added, font *M* Andalus. This font had a negative effect on the entire performance of the recogniser as shown in Fig 6.9. Case four grouped four fonts to form each word template, two were handwritten and two were computer-generated. Each word template was calculated by averaging four word samples that were randomly selected from those samples representing that word, one sample from each font. In case five all fonts, previously mentioned in Chapter 4, were included in calculating word templates.

Fig 6.9 illustrates the recognition rate of word samples from font *L* in the five cases. As previously mentioned, when considering font *A*, case one, here, represents the highest rate among the five cases as the word templates were not distorted by any other font which might have a different word shape. Case two is a special situation of case three. In case three all the computer-generated fonts mentioned in Chapter 4 were included to form the word templates. In case two, font *M* was discarded which had a unique word shape that did not

harmonise with the other three. This enhanced the recognition rate of the word samples from font *L*. Case four and case five represent the same concept in which a mixture of handwritten and computer-generated fonts were included to form the word templates. This resulted in a distorted version of all word samples. The effect of this is clearer in case five since more fonts were included in calculating the average values.

6.4.6 Recognised Word Samples

Some of the word samples that were successfully recognised using the proposed template-based recogniser, are shown here. The figures illustrate various aspects of the system's performance on the handwritten manuscript samples.

Fig 6.10 shows eight case of typical recognition of a word with a similar but not identical word defining the template.

Fig 6.11 shows a situation where a word was correctly recognised despite the difference in diacritics which is a common orthographic variation.

Fig 6.12 shows that recognition is likely in the presence of extraneous data, in this case, parts of characters from neighbouring words. The method used for isolating words within the manuscript relies on white space to separate words. If parts of neighbouring characters overlap in the white space, it is possible for the method to include these parts as extraneous data.

Fig 6.13 illustrates a situation where a considerable amount of noise has distorted the image. This situation is interesting because it shows a word from which it is difficult to extract a reliable contour or stroke path using standard methods for skeletonisation or thinning.

Fig 6.14 shows a situation where parts of certain characters are missing either from the word template or from the input word sample. As in Fig 6.13, using a conventional feature extraction method is problematic because the stroke can be broken into many small pieces, giving no useful features. By contrast, the template-based system uses pixels as 'evidence' for a particular interpretation.

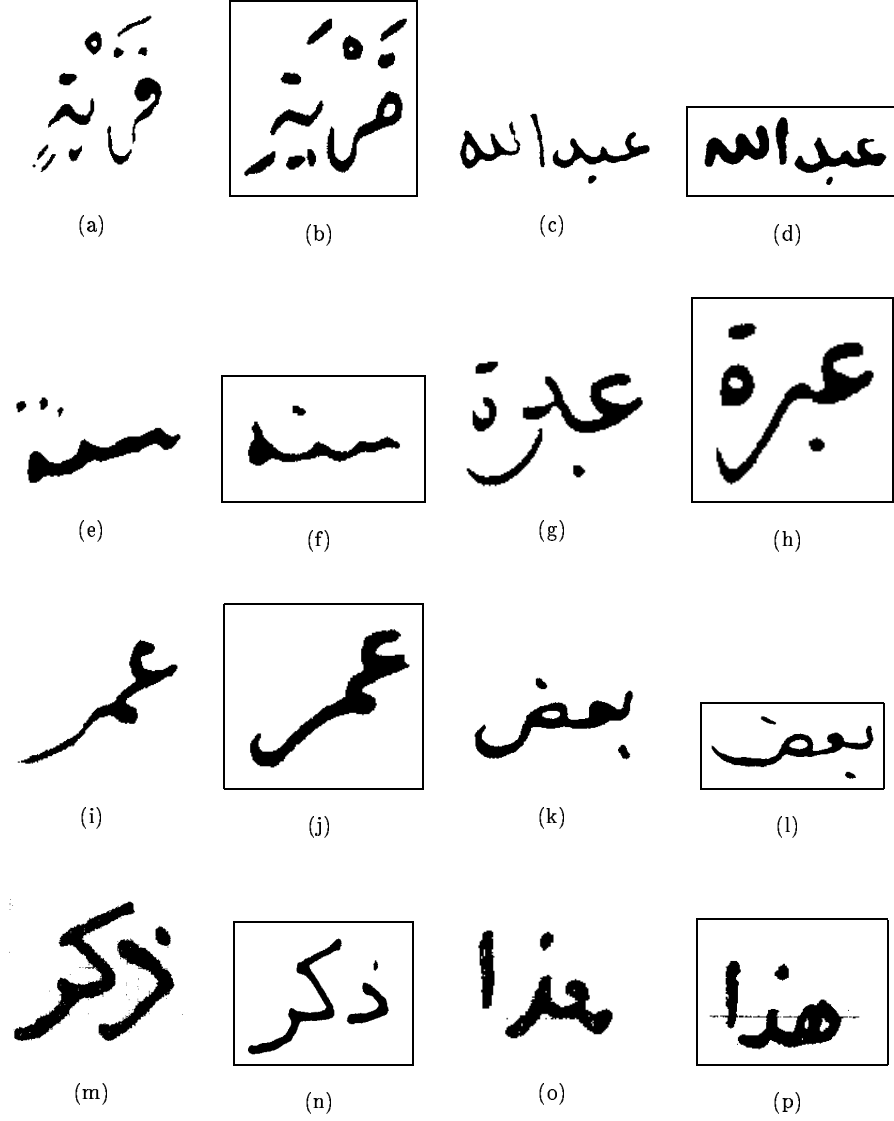


Figure 6.10: Word templates and samples: (a,b) 'قَرْيَةٌ - Village', (c,d) 'عَبْدُ اللَّهِ - Abdullah', (e,f) 'سَنَةٌ - A year', (g,h) 'عِبْرَةٌ - A lesson', (i,j) 'عُمَرُ - Omar', (k,l) 'بَعْضُ - Some', (m,n) 'ذَكَرَ - Male', and (o,p) 'هَذَا - This'. Word templates are framed.

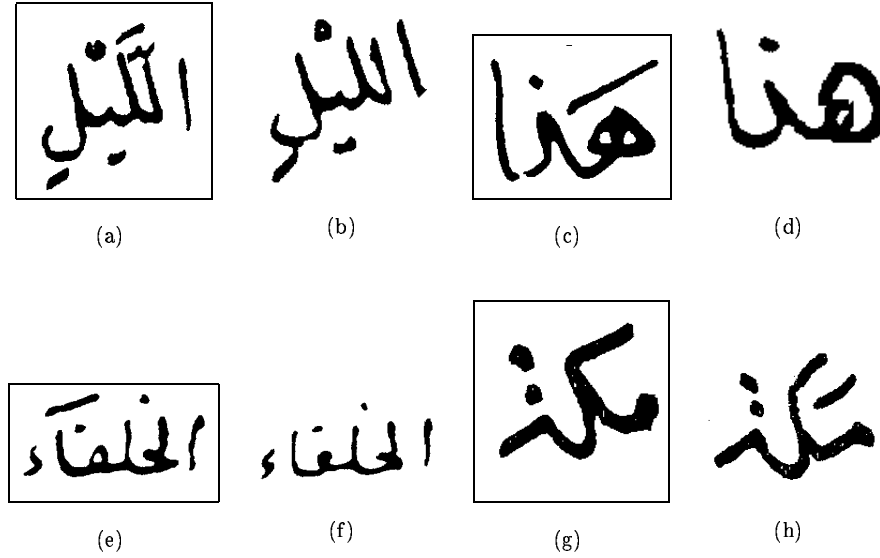


Figure 6.11: Word templates and samples: (a,b) 'الليل - The night', (c,d) 'هَذَا - This', (e,f) 'الْخُلَفَاءُ - The Caliphs', and (g,h) 'مَكَّة - Mecca'. Word templates are framed.

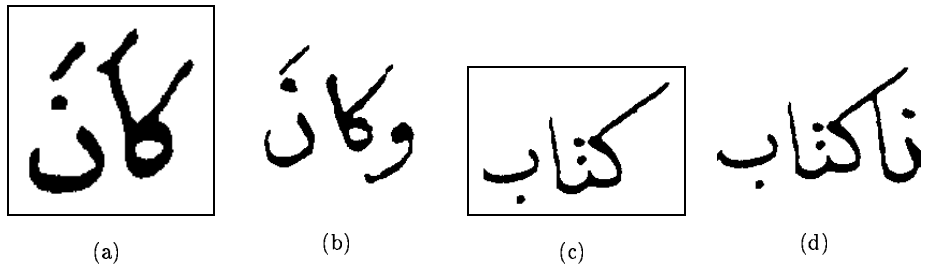


Figure 6.12: Word templates and samples: (a,b) 'كَانَ - Was', and (c,d) 'كِتَاب - A book'. Word templates are framed.

6.5 Summary

I have presented a template-based system for recognising cursive Arabic words. Each word in the lexicon was represented by a sin-

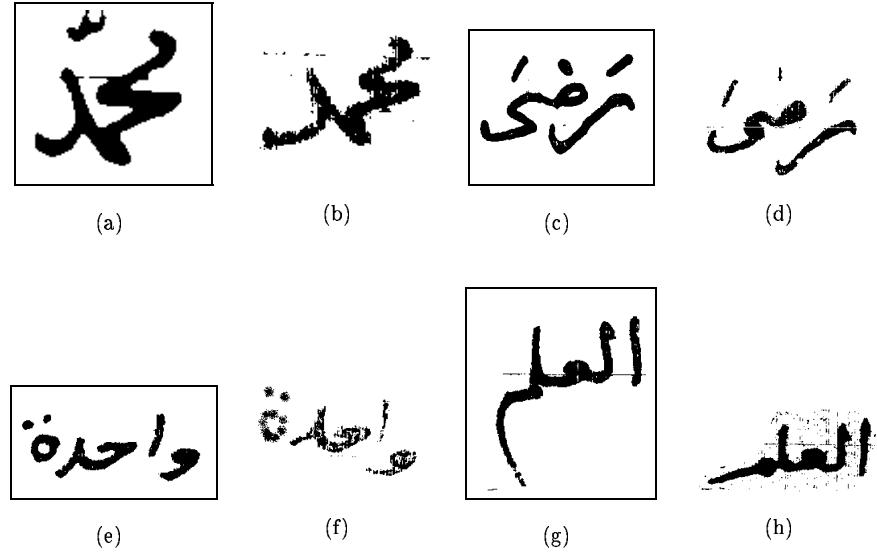


Figure 6.13: Word templates and samples: (a,b) 'مُحَمَّد' - Mohammad', (c,d) 'رَضِيَ' - Be pleased', (e,f) 'وَاحِدَةً' - Single', and (g,h) 'الْعِلْم' - The science'. Word templates are framed.

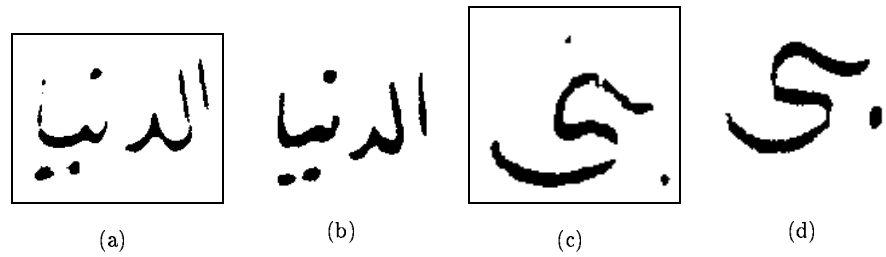


Figure 6.14: Word templates and samples: (a,b) 'الدُّنْيَا' - This life', and (c,d) 'بَنِي' - Sons of'. Word templates are framed.

gle template. This template was formed by binding the range of the Fourier coefficients, described in the previous chapter, to low

frequencies.

The template-based recogniser performed well in recognising type-written fonts, and individual handwritten fonts with a consistent writing style. In contrast, it performed poorly when a template was formed from multiple handwritten fonts. This was due to the averaging process of different word samples. The process produced a distorted Fourier spectrum magnitudes as far as each individual font is concerned. To remedy this a learning method for extracting Fourier features from each single font should be implemented. In the following chapter hidden Markov models play this role.

CHAPTER 7

MULTI-HMM RECOGNITION SYSTEM

7.1 Introduction

Chapter 5 discusses the usage of spectral features for recognising Arabic words. These features are discriminating word samples to the extent that a template-based recogniser with a simple dissimilarity measure can achieve up to 99.52% recognition rate for a typewritten font and 93.2% for a handwritten font. A drawback of this approach is that when word samples are taken from different fonts, typewritten and handwritten, to form the word templates, these templates are not robust enough to extract features from participating fonts. This is because the templates are formed by averaging a number of Fourier spectrum magnitudes. The averaging process produces a very distorted Fourier spectrum magnitude set, as far as each separate font is concerned. This motivates using a learning method to train the model with spectral features extracted from each font. Doing this enables the model to efficiently recognise word samples from all fonts that the model is trained for.

This chapter presents a method to recognise cursive Arabic words. This method combines the techniques of VQ and HMM (see Chapter 3) together with the spectral features of word samples. Both techniques are trained for the 145-word vocabulary used in Chapter

6. Each word in the vocabulary is represented by a distinct HMM. The normalised polar transform described in Chapter 5, is first applied to the word image. The two-dimensional Fourier Transform is then applied to the polar image. The resulting Fourier spectrum is divided into a sequence of wedge-shaped sectors. These sectors are vector quantised into a finite codebook, where each sector represents one *observation* or *symbol*. The recognition process consists of computing the likelihood probability of an input word against each word model and assigning the word model with the highest probability.

7.2 VQ Implementation

In Chapter 3 we learned that the inputs to the HMM are sequences of discrete symbols selected from a finite codebook. This codebook is formed using the method of VQ, Sec 3.2, of the spectral feature sets measured as described in Chapter 5. In this section I discuss the implementation of VQ for word recognition.

7.2.1 Dimensionality Reduction

In speech processing the input speech signal is a one-dimensional time varying signal [DPH93]. The pre-emphasised signal is divided into a frame sequence. Then using *Linear Predictive Coefficient* (LPC) analysis each frame is transformed into a p -dimensional spectral vector. A practical concept of building a codebook of distinct vectors attracts the work done in implementing a VQ technique [RLS83, LRS83, RJLS85].

Unlike the situation for speech signals, applying the Fourier transform to images results in a two-dimensional matrix. This is prohibitive for subsequent VQ and feature extraction operations. One alternative is to reduce the dimensionality of the problem from the entire space bandwidth of the input to a 64-dimensional vector such as a *Wedge-Ring Detector* (WRD) [LS70, CS83, CF84]. WRD samples the Fourier transform plane by detecting 32 wedge-shaped ele-

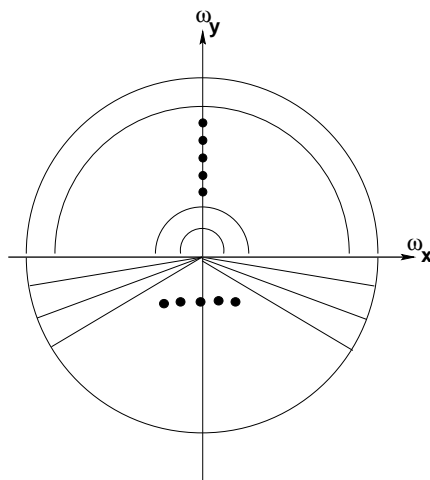


Figure 7.1: The Wedge Ring Detector (WRD) of the Fourier spectrum.

ments in one-half of a circular detector and 32 annular-shaped detector elements in the other half, see Fig 7.1. The ring detector elements provide rotation-invariance, whereas the wedge detector elements provide scale-invariance.

In this case, the resulting Fourier spectrum magnitudes are scale, translation, and rotation invariance. Moreover, half the Fourier spectrum is sufficient to handle the recognition successfully, as illustrated in Chapter 5. However, the value set is still considered as a two-dimensional signal. Hence, to reduce the dimensionality of the problem, half of the Fourier spectrum magnitudes are divided into six sectors, as shown in Fig 7.2. Each *sector* is a 30° wedge-shaped sampling window. This obtains the contributions of adjacent directions and has two advantages: reducing the number of samples required and reducing the effect of small variations.

7.2.2 Segmental VQ

Each *sector* of the six sectors previously described has a different number of Fourier coefficients and more critically represents the contributions of a separate direction group within the Fourier spec-

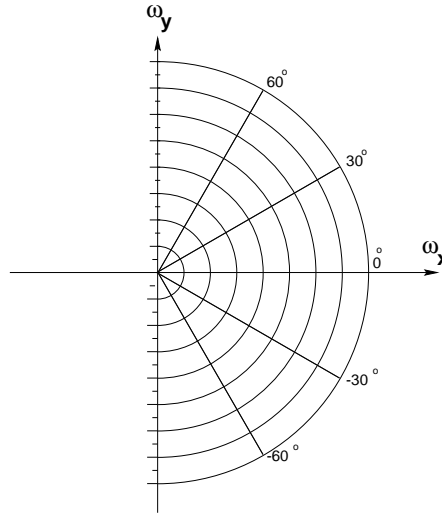


Figure 7.2: The Fourier spectrum decomposition into six adjacent sectors.

trum. Using a single vector quantiser for all sectors of the Fourier spectrum does not preserve the sequential characteristics of these sectors. This can be remedied by treating the Fourier spectrum as a concatenation of these sectors each of which is represented by a separate VQ codebook. This approach is referred to as a *segmental VQ*, and it is widely implemented in speech recognition [RJ93].

As in a single vector quantiser each set of S successive codebook symbols represents one class, $S = 6$. The discriminant score of a class equals the average value of the distance measures obtained from the successive vector quantisers. Assume we have I word samples as a training set, each sample has S Fourier spectrum sectors, a^s , $s = 1, 2, \dots, S$. There are S codebooks, each one of them has M vectors, \hat{a}_m^s , $m = 1, 2, \dots, M$ and $s = 1, 2, \dots, S$. The average distance measure of assigning a word sample to one class of S successive

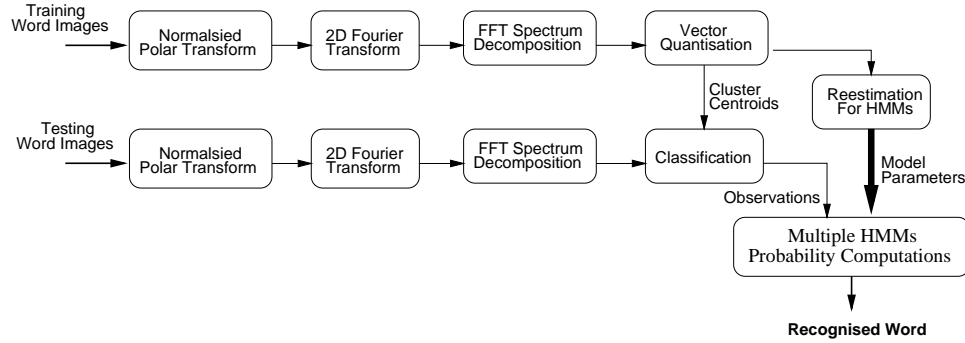


Figure 7.3: The block diagram of vector quantiser and HMM word recogniser.

symbols is

$$\|D_M\| = \frac{1}{S} \sum_{s=1}^S \|D_M^s\| \quad (7.1)$$

$$= \frac{1}{S} \sum_{s=1}^S \left[\min_{\hat{a}_m^s} \left\{ \frac{1}{I} \sum_{i=1}^I \min_{1 \leq m \leq M} [d(\hat{a}_m^s, a_i^s)] \right\} \right] \quad (7.2)$$

where $d(\hat{a}_m^s, a_i^s)$ is the Euclidean distance measure between the codebook vector \hat{a}_m^s and an input vector a_i^s .

This increases the complexity of the codebook storage by a factor of S . While it preserves the same computational complexity as a single codebook based vector quantiser.

7.2.3 VQ Results

In this chapter, the VQ algorithm is applied to three different cases. In each case, the vector quantiser is trained using a set of Fourier spectrum magnitude vectors.

1. Case-I

The training vectors here are obtained using all word samples representing different computer-generated fonts. The training set contains 10,440 vectors.

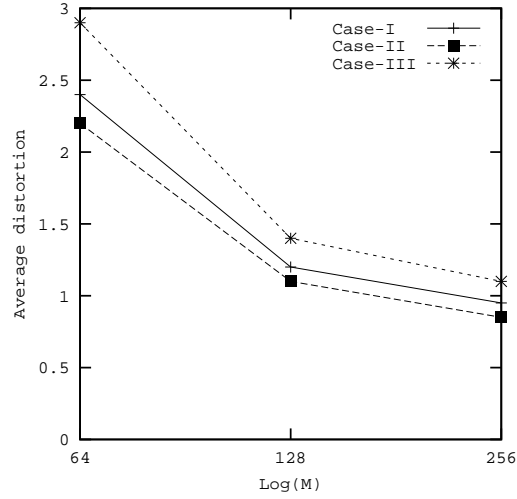


Figure 7.4: Plots of the average distortion $\|D_M\|$ versus size of codebooks.

2. Case-II

The training vectors here are extracted from all word samples representing fonts *A*, *B*, *J*, and *L*. The training set contains 7,038 vectors.

3. Case-III

The training vectors here are obtained using all word samples representing all fonts listed in Chapter 4. The training set contains 20,220 vectors.

Applying the K-means clustering algorithm, as described on page 53, six successive vector quantisers were generated each with a separate codebook. During the course of running the algorithm the average distortion performance criterion, referred to as $\|D_M\|$ in Eq. 7.2, was monitored. Fig 7.4 shows the plots of $\|D_M\|$ versus M , on a log scale, for $M = 64, 128$, and 256 . One can notice a large decrease in the average distortion when the number of clusters in each codebook increased from $M = 64$ to $M = 128$. This significant difference justifies the increased computation owing to the larger codebook. Where there is a small difference in the average

distortion between $M = 128$ and $M = 256$ this justification does not apply. This can also be related to the HMM-based recognition system performance as we will see later in this chapter.

7.3 HMMs For Word Recognition

The word sample is now transferred into a finite sequence of observations. The system is designed using multiple HMMs where each distinct word in the lexicon is represented by a separate model. The system can be described in two phases. In the training phase, the HMMs are built using a labelled training set of data. In the recognition phase an input word observation sequence is used to compute the probability scores of the word models. The word model with the highest probability score is selected as the recognised word. In this section, these two phases are discussed.

7.3.1 The Training Phase

The implementations of normalised polar transform and Fourier transform to the word image result in a two-dimensional Fourier spectrum. Using the VQ, as described in the previous section, transfers this spectrum into a sequence of observations. Here each word model is trained with observation sequences representing that word from different fonts. The process includes adjusting the model parameters in order to maximise the model probability given the observation sequences.

Three factors affect the determination of the optimum HMMs for each lexicon word: the model structure, the estimate of the model parameters, and the observation sequences used for training.

7.3.1.1 The model structure

Different HMM types are reviewed in Sec 3.3.3. Among those types is the *serial* or the *left-to-right* model. This model proceeds sequentially through the states starting from state number one and it can

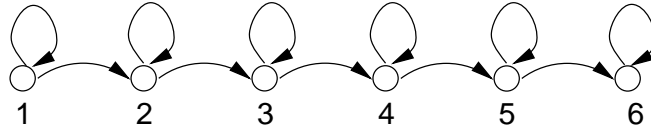


Figure 7.5: The HMM with serial constraints, only one skip is allowed.

be generalised to include any number of states, though an accurate estimation of the model parameters can be difficult if the number of states per model becomes too large. In this chapter, the *left-to-right* type is selected for the word models where the number of states N equals 6, see Fig 7.5. This constrained serial model allows only one transition from a state to its successor. It can be mathematically represented as

$$a_{ij} = 0 \quad \text{if } i > j \quad \text{or} \quad i < j - 1$$

7.3.1.2 The estimation of the model parameters

The left-to-right HMM structure requires tuning only A and B parameters. The initial probability parameter π has a binary value

$$\pi_i = \begin{cases} 1 & i = 1 \\ 0 & i = 2, 3, \dots, N \end{cases}$$

The Baum-Welch training procedure mentioned in Sec 3.3.2.3 is guaranteed to reach a local maximum. However, an alternative start of model parameter values could yield models with higher or lower values of P . For the current experiments, the elements of A and B were assigned different randomly selected values, followed by a normalisation process to satisfy the constrain

$$\sum_{j=1}^N a_{ij} = 1 \quad i = 1, 2, \dots, N \quad (7.3)$$

$$\sum_{k=1}^M b_j(k) = 1 \quad j = 1, 2, \dots, N \quad (7.4)$$

Equations 3.37 and 3.37 are used to reestimate A and B matrices. However for a finite training sequence of length T , $b_j(k)$ may equal zero. This concludes that $\alpha_{t-1}(i)$ is not zero for only one value j . This is known as an inadequate training data problem. To remedy this, post-estimation constraints on $b_j(k)$'s are used as follows

$$\begin{aligned} b_j(k) &\geq \epsilon & j &= 1, 2, \dots, N \\ k &= 1, 2, \dots, M \end{aligned} \quad (7.5)$$

where ϵ is a predefined threshold value. After the replacement $b_j(k)$'s are normalised again.

7.3.1.3 Multiple observation sequences

Word samples are taken from different fonts, typewritten and handwritten. The objective of the recognition system is to obtain font-independent models. The observation sequence, \mathbf{O} , actually consists of several independent sequences $\mathbf{O}^{(k)}$, $k = 1, 2, \dots, K$, where $\mathbf{O}^{(k)}$ is the training sequence for font k , and K is the total number of fonts used for training the word models. A complete list of different fonts used in this work is shown in Chapter 4. The likelihood probability of multiple observation sequences is handled in a two step process. Using Eq. 3.7, the first step calculates $P(\mathbf{O}^{(k)}|\lambda)$ for each sequence, taking into consideration that the process starts from state 1. The second step maximises the product of the probabilities

$$P = \prod_{k=1}^K P(\mathbf{O}^{(k)}|\lambda) \quad (7.6)$$

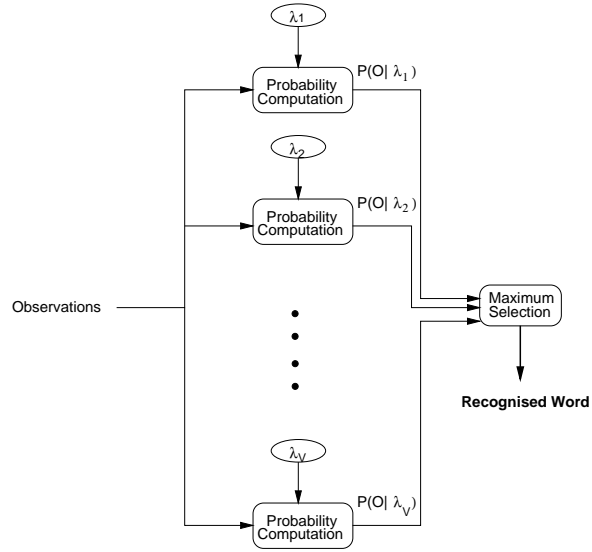


Figure 7.6: A block diagram illustrating HMMs competition in assigning the input observation sequence to the winning model.

The A and B reestimation formulas 3.37 and 3.37, found on page 65, are adjusted accordingly

$$\begin{aligned}
 \bar{a}_{ij} &= \frac{\sum_{k=1}^K \frac{1}{P(\mathbf{O}^k|\lambda)} \sum_{t=1}^{T-1} \xi_t^k(i, j)}{\sum_{k=1}^K \frac{1}{P(\mathbf{O}^k|\lambda)} \sum_{t=1}^{T-1} \gamma_t^k(i)} \\
 &= \frac{\sum_{k=1}^K \frac{1}{P(\mathbf{O}^k|\lambda)} \sum_{t=1}^{T-1} \alpha_t^k(i) a_{ij} b_j(o_{t+1}^k) \beta_{t+1}^k(j)}{\sum_{k=1}^K \frac{1}{P(\mathbf{O}^k|\lambda)} \sum_{t=1}^{T-1} \alpha_t^k(i) \beta_t^k(i)} \quad (7.7)
 \end{aligned}$$

$$\begin{aligned}
 \bar{b}_j(l) &= \frac{\sum_{k=1}^K \frac{1}{P(\mathbf{O}^k|\lambda)} \sum_{\substack{t=1 \\ o_t=v_l}}^T \gamma_t^k(j)}{\sum_{k=1}^K \frac{1}{P(\mathbf{O}^k|\lambda)} \sum_{t=1}^T \gamma_t^k(j)} \\
 &= \frac{\sum_{k=1}^K \frac{1}{P(\mathbf{O}^k|\lambda)} \sum_{\substack{t=1 \\ o_t=v_l}}^T \alpha_t^k(j) \beta_t^k(j)}{\sum_{k=1}^K \frac{1}{P(\mathbf{O}^k|\lambda)} \sum_{t=1}^T \alpha_t^k(j) \beta_t^k(j)} \quad (7.8)
 \end{aligned}$$

7.3.2 The Recognition Phase

Like the image training phase the procedures of the normalised polar transform, the two-dimensional Fourier transform and the Fourier spectrum decomposition are first applied to the word image to be recognised. Each sector is then assigned to a symbol using the classification algorithm, K -means clustering. Finally, with model parameters from the training phase a scoring procedure, which is based on the Viterbi algorithm, is applied to compute the likelihood probability of the input observation sequence \mathbf{O} given the model λ , $P(\mathbf{O}|\lambda_v)$, $v = 1, 2, \dots, V$. Where V is the number of words in the lexicon. The scoring procedure is as follows:

1. For each word model in the lexicon perform steps 2 through 5.

2. *Preprocessing*

$$\tilde{a}_{ij} = \begin{cases} \log(a_{ij}) & j \leq i \leq j+1, j = 1, 2, \dots, N \\ \log(\zeta) & \text{Otherwise} \end{cases} \quad (7.9)$$

$$\tilde{b}_i(o_t) = \log[b_i(o_t)], \quad 1 \leq i \leq N, 1 \leq t \leq T \quad (7.10)$$

where $0 < \zeta < 1$ to avoid undefined value for $\log(0)$

3. *Initialisation*

$$\tilde{\delta}_1(i) = \begin{cases} \tilde{b}_1(o_1) & \text{if } i = 1 \\ \log(\zeta) & \text{Otherwise, } i = 2, 3, \dots, N \end{cases} \quad (7.11)$$

4. *Recursion*

$$\tilde{\delta}_t(j) = \max_{1 \leq i \leq N} [\tilde{\delta}_{t-1}(i) + \tilde{a}_{ij} + \tilde{b}_j(o_t)] \quad (7.12)$$

$$1 \leq j \leq N, 2 \leq t \leq T$$

5. *Termination*

$$P(\mathbf{O}|\lambda_v) = \tilde{\delta}_T(N) \quad (7.13)$$

Now select the word whose model likelihood is highest, i.e.

$$v^* = \arg \max_{1 \leq v \leq V} [P(\mathbf{O}|\lambda_v)] \quad (7.14)$$

7.4 Recognition and Evaluation

Each word image was transformed into a normalised polar image, as described in Chapter 5. Then the Fast Fourier Transform was applied to the polar image. This resulted in Fourier spectrum magnitudes that were invariant to translation, dilation, and rotation changes. This two-dimensional signal was reduced into a one-dimensional signal by dividing half of the Fourier spectrum into six consecutive 30° wedge-shaped sectors. Using six different codebooks each sector was assigned to one symbol based on the minimum Euclidean distance. Each word image was then represented by six consecutive symbols/observations. The observation sequences were used to train word models and each model represents one word in the lexicon.

Several evaluation tests were performed for each one of the three cases mentioned in Sec 7.2.3. In this section, the performance of each case is discussed separately. Beside this, the performance of the HMM-based and template-based recognisers are compared.

7.4.1 Case-I

More than 1700 samples representing 145 words were used to assess the performance of the HMM/VQ word recogniser. The samples were printed using four different computer-generated fonts: Simplified Arabic, Thuluth, Andalus, and Arabic Traditional referred to in Chapter 4 as fonts *J*, *K*, *L*, and *M*, respectively. The samples were rendered at random angles ranging from 0 to 2π , at random sizes ranging between 18pt and 48pt, and at random translations up to twice the size of the sampled word.

Four experiments were performed. Using six 128-symbol codebooks, the first experiment assigned each one of the six Fourier spectrum sectors to one symbol. The word image was transferred into a six-observation sequence. Each word model was then trained using four observation sequences, one from each font. This experiment used 33% of the data set to train the word models and is referred to as *HMM - I1/128*. The second experiment used the same number

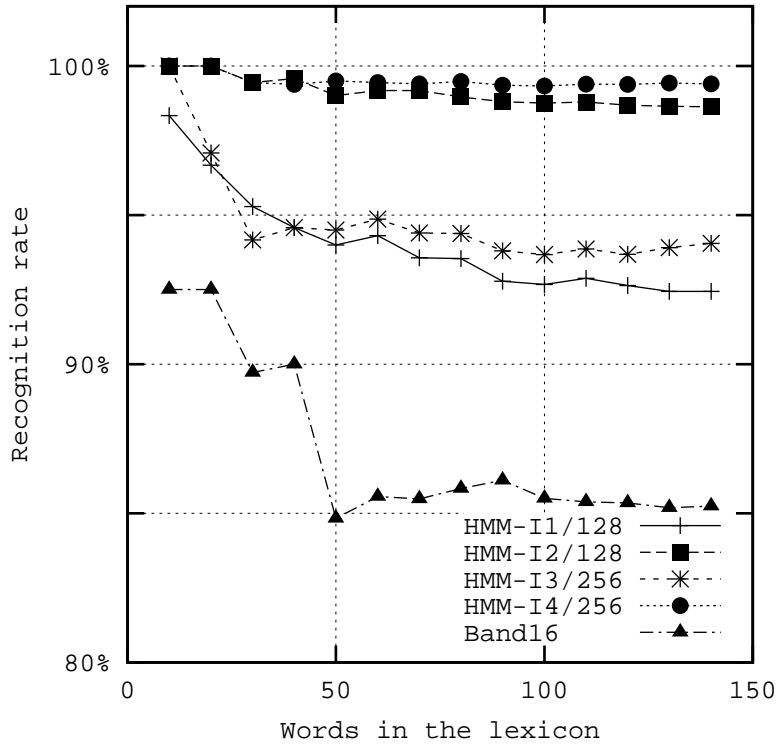


Figure 7.7: Recognition rates of typewritten word samples.

of symbols per codebook, and 66% of the data set to train the word models. This experiment is referred to as *HMM – I2/128*. In the third experiment each codebook was partitioned into 256 symbols. Like the first experiment 33% of the data set was used to train the word models. This experiment is referred to as *HMM – I1/256*. The fourth experiment is similar to the previous experiment except that 66% of the data set was used to train the word models. This experiment is referred to as *HMM – I2/256*.

Fig 7.7 illustrates the recognition results of the four experiments together with the performance of the template-based recogniser described in the previous chapter. The template-based recogniser used *Band₁₆*-size word templates and the Euclidean distance measure. Three remarks can be concluded from this figure: first, a 7% in-

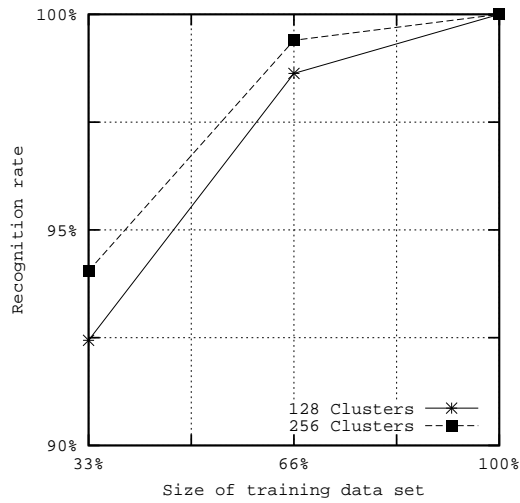


Figure 7.8: The effects of the codebook and the training data set sizes on the recognition rate.

crease in the recognition rate was achieved when using the HMM. This is due to the fact that unlike the word template, which was distorted by the averaging process of different writing styles, word models adjust their parameters to learn the features extracted from each font. The second remark is regarding the amount of observation sequences used to train the word models and how they are proportional to the recognition rate. An increase of 9% was achieved when the size of the training set was doubled from 33% to 66%, see Fig 7.8. The third remark is concerned with the number of symbols in each codebook. Sec 7.2.3 shows that the minor difference in the average distortion when $M = 128$ and $M = 256$ does not justify the increased computation owing to the larger codebook. This can also be noticed when considering $HMM - I1/128$ and $HMM - I1/256$, or $HMM - I2/128$ and $HMM - I2/256$. An increase of only 1.5% was achieved when the codebook size was doubled.

Table 7.1 shows the recognition rates of each computer-generated font separately. The table compares the performance of the four experiences together with the template-based recogniser. For fonts

Recognition System	Font <i>J</i>			Font <i>K</i>		
	Top-1	Top-5	Top-10	Top-1	Top-5	Top-10
Template <i>Band</i> ₁₆	90.24	97.38	98.81	95.71	100.0	100.0
<i>HMM</i> – <i>I1</i> /128	89.76	95.24	97.14	95.48	99.05	99.29
<i>HMM</i> – <i>I3</i> /256	91.43	97.86	98.57	96.67	99.05	99.52
<i>HMM</i> – <i>I2</i> /128	98.81	100.0	100.0	98.57	99.76	100.0
<i>HMM</i> – <i>I4</i> /256	99.29	100.0	100.0	98.81	99.76	100.0
	Font <i>L</i>			Font <i>M</i>		
	Top-1	Top-5	Top-10	Top-1	Top-5	Top-10
Template <i>Band</i> ₁₆	92.38	97.38	98.57	62.62	78.10	85.71
<i>HMM</i> – <i>I1</i> /128	86.90	94.29	96.67	97.62	99.05	99.76
<i>HMM</i> – <i>I3</i> /256	90.71	95.71	97.38	97.38	99.29	99.76
<i>HMM</i> – <i>I2</i> /128	98.10	99.76	99.76	99.05	100.0	100.0
<i>HMM</i> – <i>I4</i> /256	99.76	100.0	100.0	99.76	100.0	100.0

Table 7.1: Recognition rates (%) of typewritten word samples representing 145-word lexicon.

J, *K*, and *L* the templates-based recogniser showed competitive results relative to the HMM-based recogniser because of the similarity between these fonts. In contrast the HMM-based recogniser surpasses the template-based recogniser when recognising word samples from font *M*. This means that the HMM-based recogniser outperforms the template-based recogniser with regard to overall typewritten word samples, see Fig 7.7.

Fig 7.9 shows six different confusion matrices for the computer-generated fonts of the 145-word lexicon. The sub-figures represent a 128-symbol codebook with 33% and 66% of the data set used for training purposes, and a 256-symbol codebook with 33% and 66% of the data set used for training purposes. They also include the performance of the template-based recogniser mentioned in the previous chapter.

7.4.2 Case-II

In this case, four fonts, from the available thirteen fonts listed in Chapter 4, were selected. Two of these fonts were handwritten *A* and *B*, and the other two were typewritten *J* and *L*. There are

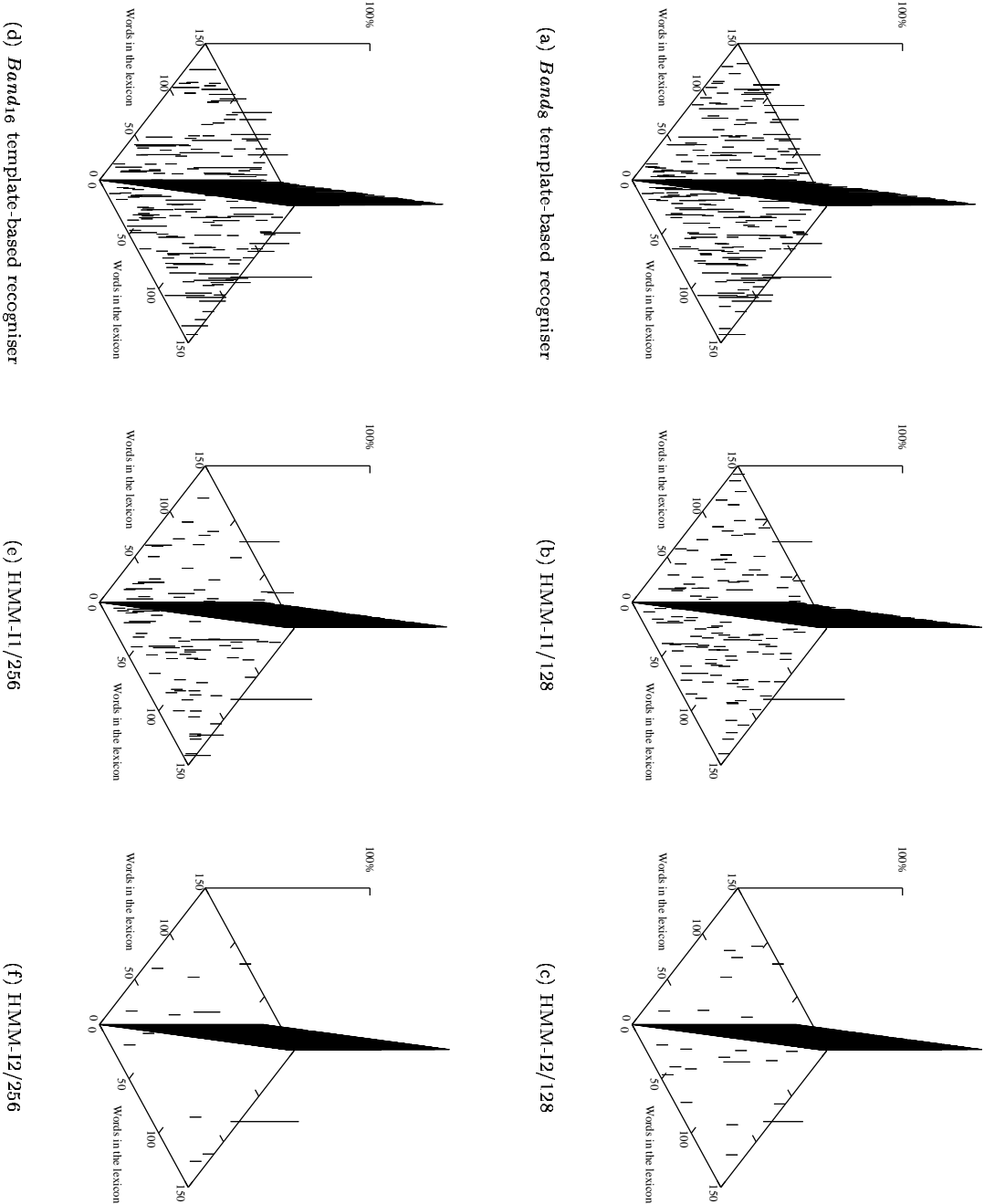


Figure 7.5: Three-dimensional depictions of confusion matrices for the computer-generated fonts, subjected to 25/300 recognition tests. In (b) and (e) 33% of the data set was used for training, while in (c) and (f) 66% of the data set was used for training.

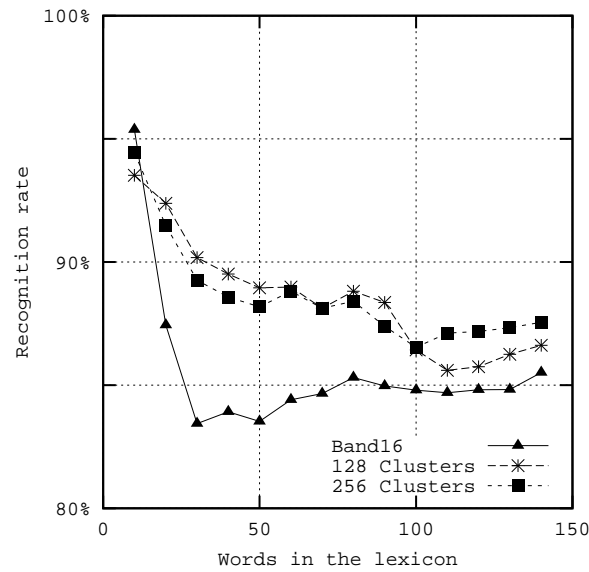


Figure 7.10: Recognition rates of word sample from fonts: *A*, *B*, *J*, and *L*.

some word shape similarities between the four fonts, and this was the reason behind choosing them. Furthermore, fonts *A* and *B* have the best consistent writing style among the nine handwritten fonts, see figure 6.5 and 6.6. Two experiments were performed. The first experiment used 128 symbols per codebook, whereas the second experiment used 256 symbols. Both experiments used 44% of the data set for training the word models. Fig 7.10 illustrates the recognition results of the two experiments together with the performance of the template-based recogniser, described in the previous chapter. The template-based recogniser used $Band_{16}$ -size word templates and the Euclidean distance measure.

Table 7.2 shows the recognition rates of each font separately. The table compares the performance of the two experiments together with the template-based recogniser.

Fig 7.11 shows different confusion matrices of the 145-word lexicon for the four fonts. The figure compares the performance of four recognition schemes: the 128-symbol codebook, the 256-symbol

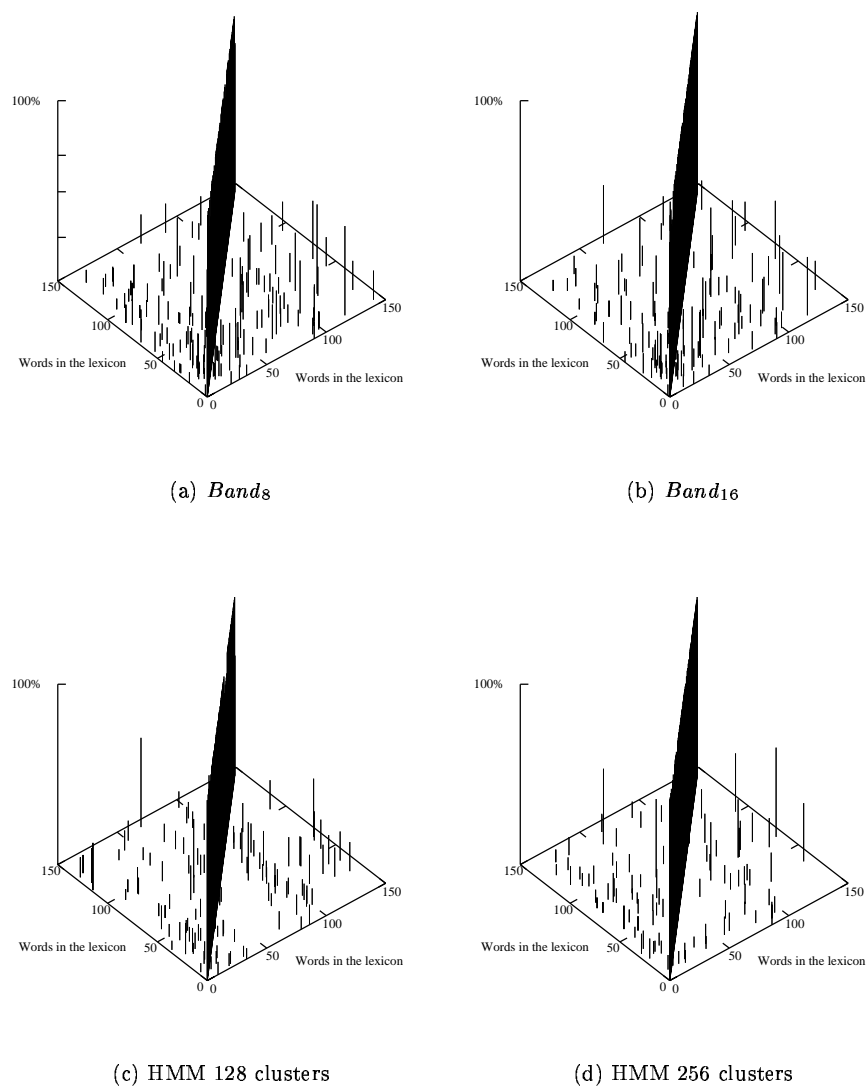


Figure 7.11: Three-dimensional depictions of confusion matrices for two recognition systems, subjected to 174435 recognition tests. Included fonts are: *A*, *B*, *J*, and *L*. In (c) and (d) 40% of the data set was used for training the word models.

Fonts	Recognition System	Recognised word		
		Top-1	Top-5	Top-10
<i>A</i>	Template <i>Band</i> ₁₆	66.09	81.74	88.70
	HMM 128 Clusters	77.39	87.39	90.00
	HMM 256 Clusters	80.00	86.09	89.13
<i>B</i>	Template <i>Band</i> ₁₆	79.61	86.41	92.23
	HMM 128 Clusters	87.38	93.20	96.12
	HMM 256 Clusters	90.29	94.17	96.12
<i>J</i>	Template <i>Band</i> ₁₆	88.33	96.43	98.10
	HMM 128 Clusters	89.76	95.00	95.71
	HMM 256 Clusters	89.05	95.00	97.62
<i>L</i>	Template <i>Band</i> ₁₆	94.76	99.05	99.29
	HMM 128 Clusters	88.33	95.71	97.62
	HMM 256 Clusters	89.52	95.00	96.19

Table 7.2: Recognition rates (%) of four font word samples representing 145-word lexicon.

codebook, the *Band*₈ template-based, and the *Band*₁₆ template-based.

7.4.3 Case-III

More than 3,300 word samples extracted from the thirteen different fonts, and representing 145 words were used to assess the performance of the HMM/VQ word recogniser in this case. Four experiments were performed. Using six 128-symbol codebooks, the first experiment assigned each one of the six Fourier spectrum sectors to one symbol. The word images were then transferred into six-observation sequences. Each word model was trained using observation sequences representing its equivalent word. This experiment used 47% of the data set to train the word models and is referred to as *III1/128*. The second experiment used the same number of symbols per codebook and more training sequences; 66% of the data set was used for training the word models. This experiment is referred to as *III2/128*. In the third experiment each codebook was partitioned into 256 symbols. Like the first experiment 47% of the data set was used for training the word models. This experi-

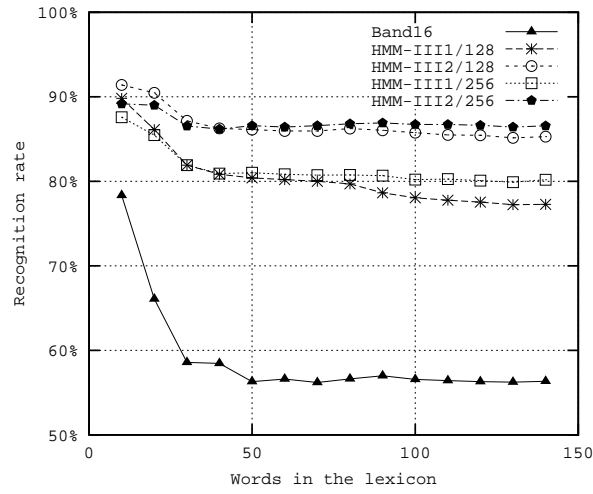


Figure 7.12: Recognition rates of word samples from all available fonts: *A, B, C, D, E, F, G, H, I, J, K, L*, and *M*. The recognised word appeared as the first option.

ment is referred to as *III1/256*. The fourth experiment is similar to the previous experiment except that 66% of the data set was used for training the word models. This experiment is referred to as *III2/256*. Fig 7.12 and Table 7.3 illustrate the recognition results of the four experiments together with the performance of the template-based recogniser, described in the previous chapter. The template-based recogniser used $Band_{16}$ -size word templates and the Euclidean distance measure. Table 7.4 shows detailed recognition

	Recognised word		
	Top 1	Top 5	Top 10
$Band_{16}$	56.34	74.72	81.10
III1/128	77.27	87.51	90.40
III1/256	80.19	87.85	90.88
III2/128	85.29	91.31	93.10
III2/256	86.57	90.64	93.13

Table 7.3: Recognition rates (%) of the four experiments and the template-based recogniser.

Testing Fonts	Recognised word				Testing Fonts	Recognised word				Testing Fonts	Recognised word			
	Top-1	Top-5	Top-10			Top-1	Top-5	Top-10			Top-1	Top-5	Top-10	
<i>A</i>	<i>Band</i> ₁₆	56.52	76.52	81.30	<i>B</i>	<i>Band</i> ₁₆	61.17	83.50	87.38	<i>C</i>	<i>Band</i> ₁₆	40.22	60.33	68.48
	III1/128	70.00	83.04	86.52		III1/128	82.52	90.29	92.23		III1/128	72.83	80.98	83.70
	III1/256	71.30	81.74	87.83		III1/256	85.44	91.26	95.15		III1/256	73.37	76.63	79.89
	III2/128	69.13	81.30	87.39		III2/128	81.55	90.29	92.23		III2/128	74.46	81.52	83.70
	III2/256	69.13	77.83	86.83		III2/256	85.44	88.35	93.20		III2/256	75.54	79.35	81.52
<i>D</i>	<i>Band</i> ₁₆	37.38	54.21	63.08	<i>E</i>	<i>Band</i> ₁₆	53.42	69.86	79.45	<i>F</i>	<i>Band</i> ₁₆	47.95	69.86	78.54
	III1/128	68.22	78.50	82.24		III1/128	76.71	79.45	82.19		III1/128	68.04	81.74	84.47
	III1/256	68.69	78.97	84.11		III1/256	75.34	79.45	84.93		III1/256	68.95	79.91	86.30
	III2/128	70.09	79.91	84.11		III2/128	80.82	86.30	87.67		III2/128	71.23	81.74	84.93
	III2/256	71.50	80.37	85.05		III2/256	83.56	86.30	90.41		III2/256	71.69	80.37	87.67
<i>G</i>	<i>Band</i> ₁₆	38.25	65.57	76.50	<i>H</i>	<i>Band</i> ₁₆	30.19	52.36	61.79	<i>I</i>	<i>Band</i> ₁₆	45.60	65.28	70.98
	III1/128	69.40	77.60	83.61		III1/128	74.06	80.66	82.08		III1/128	76.68	85.49	89.12
	III1/256	68.31	76.50	81.42		III1/256	76.42	81.13	84.43		III1/256	78.76	83.94	87.56
	III2/128	73.22	81.42	87.43		III2/128	75.00	82.08	83.02		III2/128	77.72	85.49	88.60
	III2/256	72.68	78.14	83.61		III2/256	75.94	80.19	85.38		III2/256	79.79	85.49	88.08
<i>J</i>	<i>Band</i> ₁₆	76.43	89.76	93.57	<i>K</i>	<i>Band</i> ₁₆	70.95	86.67	90.71	<i>L</i>	<i>Band</i> ₁₆	77.14	92.14	95.95
	III1/128	79.05	92.14	95.00		III1/128	83.10	94.29	96.19		III1/128	74.29	87.86	92.86
	III1/256	82.14	92.38	95.24		III1/256	91.19	96.43	96.90		III1/256	79.76	91.19	93.57
	III2/128	97.14	99.52	100.0		III2/128	95.00	99.05	99.52		III2/128	95.24	100.0	100.0
	III2/256	97.86	99.76	99.76		III2/256	96.43	99.05	99.29		III2/256	98.33	100.0	100.0
<i>M</i>	<i>Band</i> ₁₆	47.14	66.90	75.24	<i>M</i>	III1/128	92.14	98.10	98.57	<i>M</i>	III2/128	96.90	99.76	99.76
						III1/256	94.52	99.05	99.05		III2/256	98.52	99.52	99.76

Table 7.4: Recognition rates (%) of word samples extracted from all available fonts, and representing 145-word lexicon.

results for each font of the thirteen fonts. The table compares the performance of the four experiments and the template-based recogniser with $Band_{16}$ -size word templates.

Fig 7.13 shows the confusion matrices of the 145-word lexicon for all the thirteen fonts presented in Chapter 4. Beside the performance of the template-based recogniser, $Band_8$ and $Band_{16}$, the performance of HMM-based recogniser is also included.

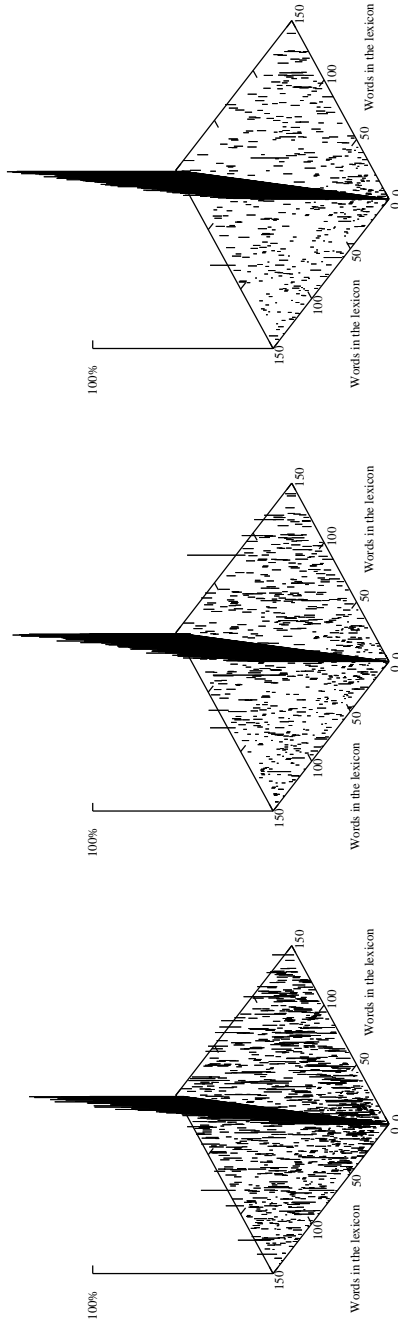
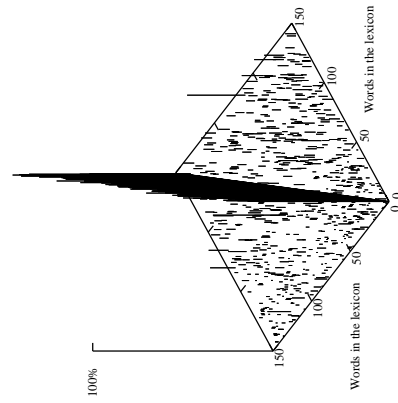
7.4.4 Recognised Word Samples

Some of the word samples were not recognised by the template-based recogniser whereas they were successfully recognised using the HMM-based recogniser are shown here. Fig 7.14 illustrates some of the input word samples. It also shows the word samples the template-based recogniser used to calculate the averaged templates representing those words. There is a difference between the template and the input sample word shapes. This made the recognition very difficult using a simple template-based recogniser. HMMs managed to recognise those words, which improved the recognition rate to 85% for all font samples. However, HMMs were not capable of recognising all input word samples, even with 66% of the data set used for training the word models. Fig 7.15 shows some of the input word samples which could not be recognised using HMMs.

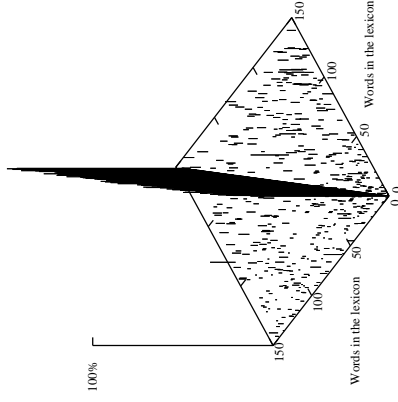
7.5 Summary

A method of recognising cursive words in Arabic manuscripts has been presented. The word-model method was designed using multiple HMMs, where each word in the lexicon was represented by a separate HMM. The word models were trained using the Fourier spectrum magnitudes, described in Chapter 5.

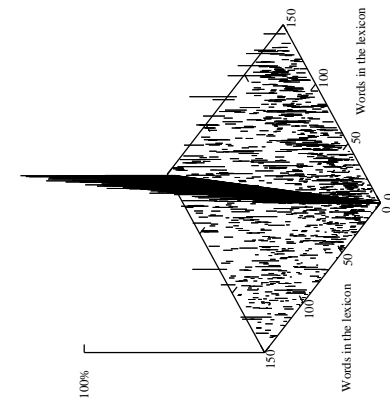
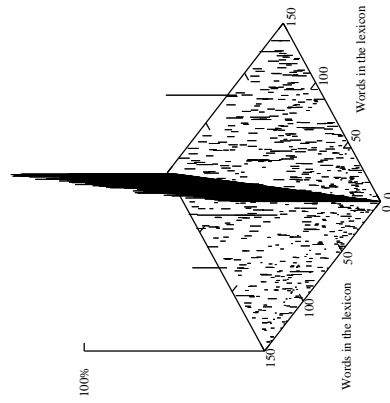
The Fourier spectrum of a word sample was divided into six wedge-shaped sectors. Each sector was assigned to one codebook symbol. This transferred the word image into a six-observation se-

(a) *Bands* template-based recogniser

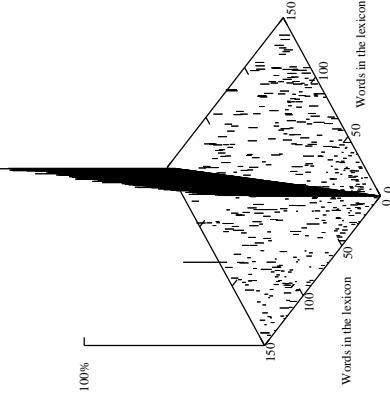
(b) HMM-III1/128



(c) HMM-III2/128

(d) *Band*₁₆ template-based recogniser

(e) HMM-III1/256



(f) HMM-III2/256

Figure 7.13: Three-dimensional depictions of confusion matrices for HMM-based and template-based recognition systems, subjected to 488650 recognition tests. The data set includes all the thirteen fonts. In (b) and (e) 33% of the data set was used for training the word models, while in (c) and (f) 66% was used for this purpose.

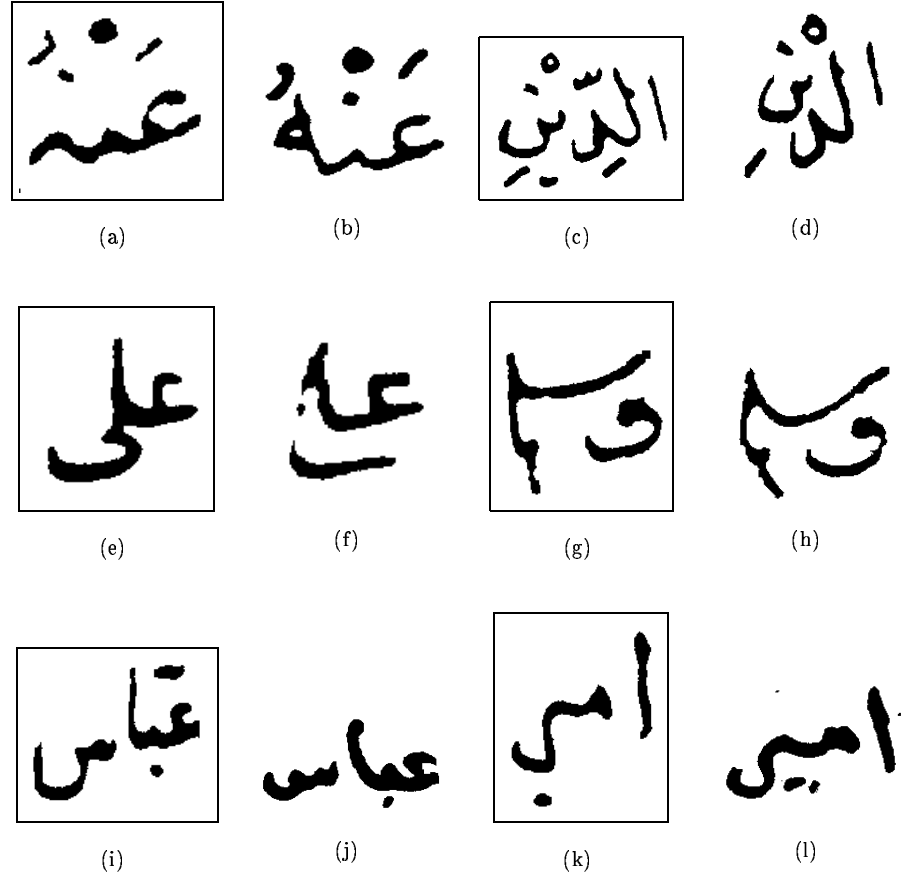


Figure 7.14: Word templates and samples: (a,b) 'عنه' *ʿnḥu* - About him', (c,d) 'الدين' *ad-dīn* - The religion', (e,f) 'على' *ʿlā* - On', (g,h) 'واسلم' *wasllam* - May peace be upon', (i,j) 'عباس' *abbās* - Abbas', and (k,l) 'امير' *amīr* - Prince'. Word templates are framed.

quence. To overcome the variation of the sector sizes, the sequential VQ was implemented. This required the use of six codebooks, one for each sector group.

Part of the data set was used for training the word models, while the rest was used for assessing the recognition system per-



Figure 7.15: Word samples that HMMs could not recognise correctly: (a) 'خير' *hayr* - 'Goodness', and (b) 'الشَّام' *aš-šām* - 'Al-Sham'.

formance. The system achieved a higher recognition rate compared to the template-based recogniser, described in Chapter 6.

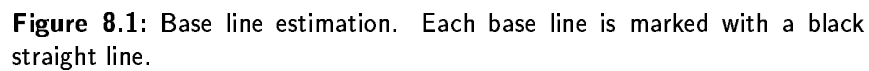
CHAPTER 8

STRUCTURAL FEATURES OF ARABIC WORDS

8.1 Introduction

Chapter 5 presents a method to extract spectral features from a word image. A single feature vector represents the entire word. The method is sensitive to the centroid position. A more robust algorithm is required in order to extract a feature set that is capable of tolerating variations in handwritten script.

In this chapter I present two techniques for extracting structural features from cursive Arabic words. After preprocessing, the skeleton of the binary word image is decomposed into a number of links in a certain order. In the first technique, each link is transformed into a feature vector. The target features are the curvature of the segment, its length relative to other segment lengths of the same word, the position of the segment relative to the baseline of the skeleton, and a detailed description of curved segments. In the second technique, each link is resolved into small line segments using edge linking. By measuring the Euclidean distance and the orientation angle for these line segments, each link is transformed into a sequence of feature vectors.



The image of the word to be recognised is introduced to the system as a matrix of black pixels (the foreground) and white pixels (the background). In this section, a number of steps used to prepare the word image for feature extraction are discussed. These steps include base line estimation, thinning, and skeleton modification. But first, to ensure better processing for the word image a smoothing process, refer to Sec 2.4.2.1, is applied which reduces noise in that image.

This is a horizontal line that runs through the connected primitives of a text. In a binary image test, the baseline will have the maximum number of black pixels. For handwritten scripts, the situation holds this simplicity in deciding the base line with precision. This is due to the movement of the pen above and below this line. However, this crude estimation of the base line is sufficient for the method so as to fix a reference point used later to decide the relative position of diacritics, especially dots, above or below this line.

8.2.2 Thinning

Reducing the strokes in the writing to a width of one pixel will definitely simplify traversing them later. Sec 2.4.2.5 reviewed a number of thinning algorithms. Here I am implementing the Zhang-Suen algorithm [ZS84]. This is a parallel method where the pixel's new value depends only on the values known from the previous iteration. There are two sub-iterations within each iteration. In the first sub-iteration, a pixel is marked for deletion if the following four conditions are fulfilled:

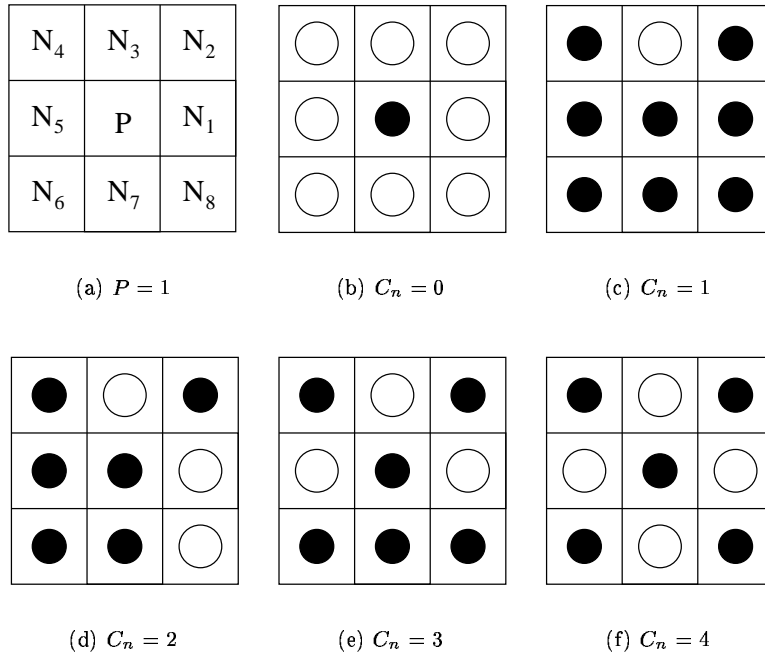


Figure 8.2: An illustration of the connectivity number. N_i is one if the pixel is white

1. Its connectivity number is one. The connectivity number is a measure of how many objects a particular pixel might connect.

One measure [Par97] to calculate this number is shown below

$$C_n = \sum_{k \in S} N_k - (N_k \times N_{k+1} \times N_{k+2}) \quad (8.1)$$

where N_k is the colour value of one of the eight neighbours of the pixel under processing, and $S = 1, 3, 5, 7$. The connectivity number can also be the number of transitions from the foreground (black) to the background (white) within the pixel 8-neighbours, see Fig 8.2.

$$C_n = \sum_{k=1}^8 N_k - (N_k \times N_{k+1}) \quad (8.2)$$

if $k > 8 \Rightarrow k = k - 8$

2. It has between two and six black neighbours.
3. At least one of N_1, N_5 , and N_7 are white.
4. At least one of N_1, N_3 , and N_7 are white.

All pixels marked for deletion are deleted. The second sub-iteration is the same except for conditions three and four

1. At least one of N_3, N_5 , and N_7 are white.
2. At least one of N_1, N_3 , and N_5 are white.

At the end of the sub-iteration all pixels marked for deletion are deleted. If at the end of either sub-iterations there are no pixels for deletion, then the algorithm is completed.

8.2.3 Skeleton Modification

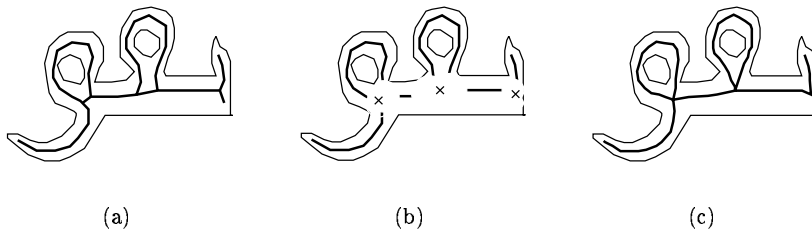
Thinning algorithms produce, in general, distorted skeletons [GH92, JC92, LLS92, FCW98], as shown in Fig 8.3. To remedy this the skeleton needs to be modified in order to remove spurious branches and merge false feature points. A feature point is a black pixel in the skeleton which has a connectivity number which does not equal *two*. The connectivity number of that pixel may equal one, three,

مكة المسجد مفهوم الحاكم

(a) Original images

مكة المسجد مفهوم الحاكم

(b) Thinned images

Figure 8.3: Results of thinning algorithms.**Figure 8.4:** Skeleton modification: (a) the word skeleton imposed on the word contour, (b) merging adjacent feature points and calculating new group centres, and (c) reconstructing the word skeleton.

or four, and that pixel is referred to as an *End point*, *Branch point*, or *Cross point*, respectively.

The local features of the skeleton are affected by the thinning algorithm. For example, the cross point is resolved into two neighbouring branch points. Thus, it is important to use both the original and the thinned word images to correct the resultant skeleton. The applied technique here is an adaptation of that used in [LH90] where the maximum circle technique was used to modify the thinning result. The algorithm is stated as follows:

1. Allocate all feature points in the thinned image T .
2. For each feature point fp , measure the radius R_{fp} of the largest circle of black pixels within the original image S that is centred at fp . The radius has a minimum value of one.
3. For each pair of feature points fp_1 and fp_2 calculate the Euclidean distance $d_{fp_1 fp_2}$. If this distance is less or equal to $R_{fp_1} + R_{fp_2}$ then mark these two feature points for merging into one group.
4. If fp_1 and fp_2 are marked to be merged and fp_2 and fp_3 are also marked to be merged, then fp_1 , fp_2 and fp_3 are all marked for mergence in a single group.
5. For each group find the value of the new center by averaging its feature point positions. A group of one member has a center identical to its single feature point.
6. For each group that has more than one member delete all black pixels in T fallen within each circle of its feature point members.
7. Reconnect each stroke with the appropriate group center.
8. Compute the connectivity number of each group center as follows:

$$\begin{aligned}
 &C_g = 2 ; \\
 &\textbf{for every } fp \textbf{ in this group} \\
 &\quad C_g = C_g + \textbf{Connectivity}(fp) - 2
 \end{aligned}$$

9. Remove any group with a connectivity number which equals *two*. This indicates that there was a spurious branch which caused the existence of a branch point and an end point. When these two feature points were merged, the end point disappeared and the branch point became a normal pixel.

8.3 Feature Extraction

The skeleton graph of a word image consists of a number of links where each link commences and terminates with a feature point. This stage extracts these links, it then transforms the word skeleton into a sequence of feature vectors. There are two different structural feature sets used in this dissertation. Both sets extract loops from the skeleton graph before extracting the remaining features.

8.3.1 Link Extraction

This traces the skeleton graph and extracts its links. This is done by traversing each link starting from one feature point and then proceeding from one pixel to its adjacent until reaching a feature point. During this process the procedure marks any pixel that has been visited to avoid traversing it again. The word images used in this dissertation are *Portable Bit Map* (PBM) files; a black pixel equals one, a white pixel equals zero, and a visited black pixel equals two.

An important requirement here is to ensure that links are assigned a canonical order so that later the observation sequence for the HMM is well defined. The link extraction procedure used here is guided by the following rule: if fp_1 and fp_2 are two feature points in the same sub-word and fp_1 is located to the right of fp_2 , then all links which branch from fp_1 should be extracted before any of the links branching from fp_2 .



Figure 8.5: Simple and complex loops that can be seen in some handwritten letters.

8.3.2 Loop Extraction

During skeleton tracing, the skeleton is checked for loops which are represented by a closed path of pixels. A number of Arabic letters have a loop-like shape, refer to Table 2.1. These letters can be divided into three categories: simple loop, complex loop, and double loop.


8.3.2.1 Simple loop

This consists of a single link that begins from a feature point and returns to the same point again. Examples of this category can be seen in the letters *ة* -*t*, *ه* -*h*, *م* -*m*, *ق* *ق* -*q*, *ف* *ف* -*f*, *ص* *ص* -*s*, *د* *د* -*d*, *و* *و* -*w*, and *و* -*w*. Also letters *ح* *ح* -*h*, *ج* *ج* -*ġ*, and *خ* *خ* -*h* may form a simple loop when they are handwritten as in word ‘خرج’ *ħrġ* - ‘Went’ shown in Fig 8.5.

8.3.2.2 Complex loop

This consists of either two or more links which connect two or three feature points. Examples of a two-link loop can be seen in letters *م* -*m*, *ص* -*s*, *د* -*d*, *ط* *ط* -*t*, and *ظ* *ظ* -*z*. Examples of a three-link loop can be seen in letters *ط* -*t* and *ظ* -*z*. Fig 8.5 shows two two-link loops: letter *ح* *ح* -*h* in word ‘حتى’ *ḥtā* - ‘Though’ and letter *ج* *ج* -*ġ* in word ‘جامع’ *ġāmʿ* - ‘Mosque’.

8.3.2.3 Double loop

This consists of two connected simple and/or complex loops. Examples of this category can be seen in the letter ( *h* - *h*).

8.3.3 Feature Set FS_1

Having extracted links and loops from the skeleton graph of the word image, each link is now transformed into an 8-dimensional feature vector. Each feature has the following description:

1. Normalised length feature (f_1): This feature gives the length of a link relative to other link lengths in the same word. It is calculated as follows:

$$f_1 = \frac{ActualLength - EL_{min}}{EL_{max} - EL_{min}} \quad (8.3)$$

where EL_{min} and EL_{max} are the minimum and the maximum link lengths for that word, respectively. This scales all f_1 values to the interval $[0, 1]$. This feature tolerates font size and rotation.

2. Curvature feature (f_2): This feature measures the curvature of a link. Simply divide the Euclidean distance between the two feature points of that link by its actual length. This feature equals zero when the link is a simple loop, and 1 when the link is a straight line. By definition f_2 equals two when there is a complex loop, and three when there is a double loop.

Although this feature does not measure the curvature precisely, these two curves \subset and \supset have the same f_2 value, they are still useful when combined with other features.

3. Endpoint feature (f_3): This feature defines the two endpoints of the link. It has one of the following values:

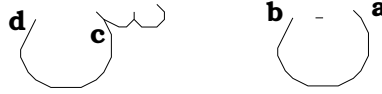


Figure 8.6: Feature f_3 calculation: link \overline{ab} has feature f_3 whose value equals 0, while link \overline{cd} has feature f_3 whose value equals 2.

Value	Description
0	<i>end point \rightarrow end point</i>
1	<i>end point \rightarrow branch point</i>
2	<i>end point \rightarrow cross point</i>
3	<i>branch point \rightarrow end point</i>
4	<i>branch point \rightarrow branch point</i>
5	<i>branch point \rightarrow cross point</i>
6	<i>cross point \rightarrow end point</i>
7	<i>cross point \rightarrow branch point</i>
8	<i>cross point \rightarrow cross point</i>

This feature can distinguish between similar links belonging to different characters. Fig 8.6 illustrates two links \overline{ab} in ‘ن’ n ’ and \overline{cd} in ‘س’ s ’. These two links have almost identical feature values except for f_3 which equals zero and three, respectively.

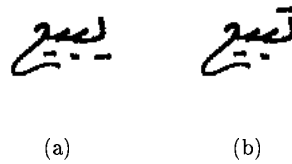


Figure 8.7: The importance of deciding the dot positions relative to the base line. The two dots of the first letter are: (a) below the base line, this makes the word ‘يبيع’ ybi' - He buys’, and (b) above the base line, this makes the word ‘تبيع’ tbi' - She buys’.

4. Relative location feature (f_4): This binary feature indicates

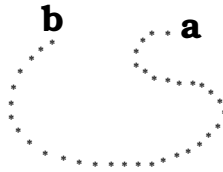


Figure 8.8: Calculating features $f5 \rightarrow f8$ to the link \overline{ab} which represents the main stroke of the letter 'ي' y .

whether the starting feature point of a link falls above/below the base line and it shows this by one/zero. This feature helps to decide whether a dot is above or below the character. Fig 8.7 shows the importance of deciding the dot positions.

5. Curved features ($f5 - f8$): These features calculate the percentage of pixels above the top feature point, below the bottom feature point, left of the left-most feature point and right of the right-most feature point of that link, respectively. The importance of these features are noticed when considering characters such as ي y and ك $-k-$. Consider the curve shown in Fig 8.8 where the total number of pixels is 37 pixels. The curved features have the following values: $f5 = 0$, $f6 = 33/37$, $f7 = 9/37$, and $f8 = 11/37$.

8.3.4 Feature Set FS_2

The word skeleton has now been transformed into a number of feature points that are connected using links and loops. Each link is represented by a list of pixels that may form any shape since the goal of thinning is to have a smaller pixel set that represents the same word image. For further reduction the pixels of the skeleton need to be collected into smaller line segments. This process is called *edge linking* or *vectorisation* [Par97].

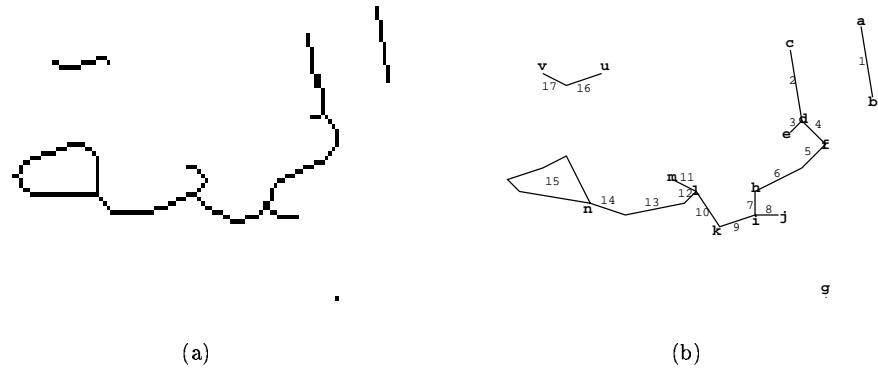


Figure 8.9: The edge linking process: (a) the original skeleton graph of the word 'الجمعة' *alġmʿt* - Friday', and (b) the skeleton after being transformed into a sequence of line segments each marked with a number. Feature points are marked with small letters.

8.3.4.1 Edge Linking

Starting from a feature point, each pixel in that link is presumed to belong to a straight line segment, and so adjacent pixels are added to a set until some linearity constraint is violated. Each set is then represented by its two endpoints, and the rest of the set's pixels are deleted. In the same way, the following lines from that link are extracted until the next feature point is reached. Fig 8.9 illustrates an example of this process which is listed below:

1. For each list of pixels which represents a link between two feature points.
 - (a) Start from the first pixel in the list.
 - (b) Add the next pixel on the list to a set that will be the set of pixels belonging to the next line in the link.
 - (c) Check the perpendicular distance between all pixels in the set and the straight line between the first and the last pixel
2. Transfer the pixel list into a sequence of line segments as follows:
 - (a) Start from the first pixel in the list.
 - (b) Add the next pixel on the list to a set that will be the set of pixels belonging to the next line in the link.
 - (c) Check the perpendicular distance between all pixels in the set and the straight line between the first and the last pixel

in the set.

- (d) If this distance is less than a predefined threshold D goto (2b), else proceed to the next step.
- (e) Stop adding pixels to the current line, and discard the last pixel in the set.
- (f) Assign the first and the new last pixels as the line segment endpoints.
- (g) If there are no more pixels on the list goto (3), else proceed.
- (h) Let the next pixel on the list be the new starting point. Goto (2a).

3. STOP.

8.3.4.2 Length and orientation

Each link is now transformed into a sequence of small line segments. Each line segment has two endpoints $P1$ and $P2$, which have the coordinates (x_1, y_1) and (x_2, y_2) , respectively. To characterise this line segment two features are extracted the line segment length and the orientation angle. The line segment length, l , is simply the Euclidean distance between the two endpoints

$$l = \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \quad (8.4)$$

The orientation angle, θ , is the angle the line segment makes with the x -axis.

$$\theta = \tan^{-1} \left(\frac{y_2 - y_1}{x_2 - x_1} \right) \quad (8.5)$$

This produces a two-dimensional feature vector for each line segment belonging to a link.

8.3.4.3 Turning points

These are not feature points, they are rather normal black pixels where a skeleton changes dramatically from one direction to another. These points give valuable information about the behaviour

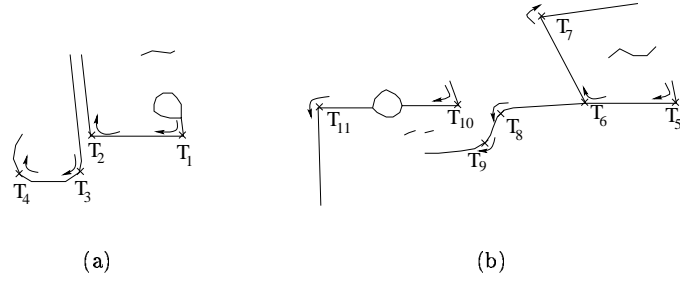


Figure 8.10: Turning points imposed on two word skeletons: (a) 'قال *qāl* - Said' and (b) 'تكرّم *tkrim* - '.

of curves. Fig 8.10 shows a number of turning points. T_1, T_3, T_5, T_9 and T_{10} are down-left turning points. T_2, T_4 and T_6 are left-up turning points. T_7 is an up-right turning point. T_8 and T_{11} are left-down turning points.

8.4 Feature Vector Encoding

A feature vector needs to be encoded into one of the discrete symbols of the codebook in order to reduce the computation required in the HMM-based recognition system. VQ, page 50, is used to form this codebook. This is done by partitioning the training samples into several classes. Each class is then represented by its centroid, and each codebook symbol represents one class. Links that represent loops are not included among the training feature vectors during the clustering algorithm, they are instead assigned to predefined symbols.

Two separate codebooks are formed each for one feature set. This section presents these two codebooks.

Class #	Description
0	Dot
1	Simple loop
2	2-link complex loop
3	3-link complex loop
4	Double loop
5-75	Classes found using k-means clustering

Table 8.1: Codebook classes for FS_1 feature set.

Class #	Description
0	Dot
1	End point
2	Branch point
3	Cross point
4	Simple loop
5	2-link complex loop
6	3-link complex loop
7	Double loop
8	Left-up turning point
9	Left-down turning point
10	Top-right turning point
11	Down-left turning point
12-59	Classes found using k-means clustering

Table 8.2: Codebook classes for FS_2 feature set.

8.4.1 FS_1 Codebook

This codebook includes 76 symbols. These symbols are listed in Table 8.1. The first five symbols are reserved for dots, black pixels with zero connectivity value, or loops which were formed while extracting links. The rest of the symbols are represented by the class centroids which are 8-dimensional vectors.

8.4.2 FS_2 Codebook

The codebook includes 60 symbols. These symbols are listed in Table 8.2. The first twelve symbols are reserved for dots, feature points, loops, and turning points. The rest of the symbols are represented by the class centroids which are two-dimensional vectors.

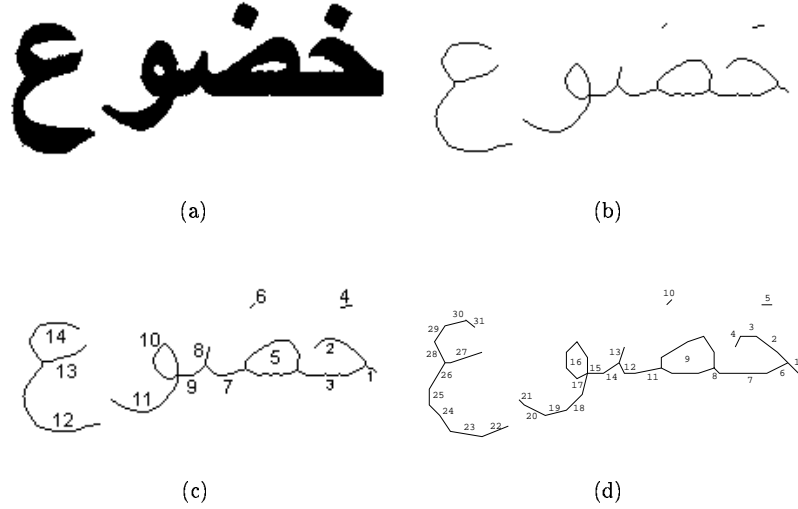


Figure 8.11: An illustration of transferring a typewritten word image into a sequence of feature vectors

8.4.3 Encoding Examples

This section illustrates the process to extract FS_1 and FS_2 from a given word image, and then transfer this image into a sequence of symbols.

8.4.3.1 Extracting FS_1

Starting from an original word image, the modified skeleton graph is first observed, the word feature vectors are then extracted, and finally the codebook data is used to form the observation sequence. Following are the steps to perform this task:

1. Apply the thinning algorithm, as in Sec 8.2.2, followed by a skeleton modification, as in Sec 8.2.3. This results in a modified skeleton, as shown in Figures 8.11-b and 8.12-b.

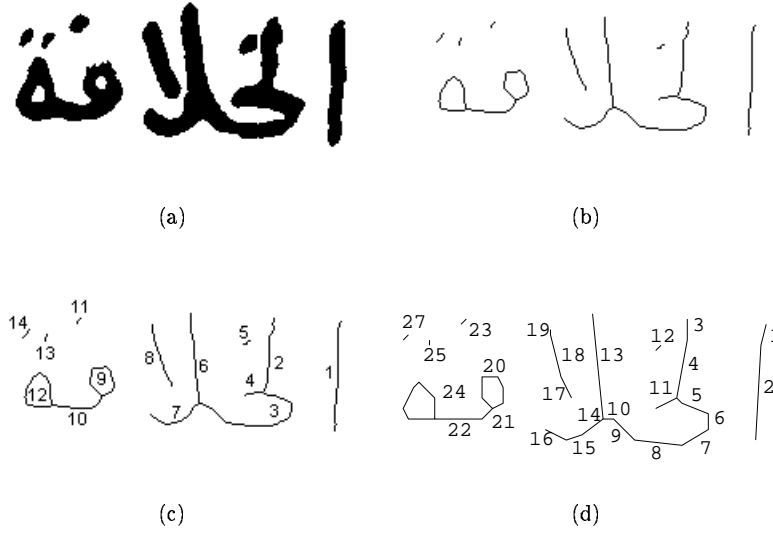


Figure 8.12: An illustration of transferring a handwritten word image into a sequence of feature vectors

2. Extract links and loops from the modified skeleton as explained in Sections 8.3.1 and 8.3.2. The word image shown in Fig 8.11-c produces 12 feature vectors, a 2-link complex loop, and a simple loop. While the word image shown in Fig 8.12-c produces 12 feature vectors and two simple loops.
3. Assign each loop to the proper codebook symbol.
4. For each non-loop link:
 - find FS_1 values mentioned in Sec 8.3.3.
 - assign each feature vector to one of the last seventy codebook symbols based on the minimum Euclidean distance measure, as in Table 8.1.

8.4.3.2 Extracting FS_2

Like in the previous section, the process starts with an original word image to produce a modified skeleton, then it extracts word feature

vectors, and finally it uses codebook data to calculate the equivalent sequence of symbols. The steps of this process are as follows:

1. Apply step (1) to (3) as in Sec 8.4.3.1.
2. For each non-loop link:
 - find FS_2 values mentioned in Sec 8.3.4.
 - assign each feature vector to one of the last 48 codebook symbols based on the minimum Euclidean distance measure, as in Table 8.2.
3. Extract turning points from the modified skeleton.
4. Assign each feature or turning point to the proper codebook symbol as in Table 8.1.

Applying this process to the word image shown in Fig 8.11-d results in 53 symbols that represent: 31 line segments, 17 feature points, and 5 turning points, while applying this process to the word image shown in Fig 8.12-d results in 50 symbols that represent: 27 line segments, 20 feature points, and 3 turning points.

8.5 Summary

In this chapter I have presented two techniques for extracting structural features from cursive Arabic words. The skeleton graph of the word image was first modified to merge adjacent feature points and remove spurious branches. The word skeleton was then decomposed into a sequence of links. These links were placed in descending order relative to the first feature point of the link. Each link was processed separately.

The first technique transformed each link into an 8-dimensional feature vector, where the second technique transferred each link into a sequence of small line segments. Each technique used a separate codebook to encode the feature vectors to discrete symbols suitable for training and recognition purposes using HMMs.

CHAPTER 9

SINGLE HMM RECOGNITION SYSTEM

9.1 Introduction

Chapters 6 and 7 presented a method of recognising cursive words in Arabic manuscripts based on the segmentation-free technique. The method showed a good performance in recognising odd word samples that were deformed due to noise or crude page/line decomposition. There are two drawbacks to this method: it is a lexicon-based system, this means that the system can only recognise the word samples that have equivalents in the lexicon. Secondly, each word in the lexicon is represented by a separate template or model.

In this chapter two single HMM recognition schemes are presented. The first scheme is a lexicon-driven system in which the vocabulary probabilities are incorporated with the low-level knowledge from the training samples to tune the HMM parameters. The second scheme is a lexicon-free system which is composed of smaller character-models. Each one of these models is trained using the Baum-Welch method.

At the recognition stage the input word observation sequence is first introduced to the HMM. The modified Viterbi algorithm is then applied to provide an order list of the best paths through the HMM.

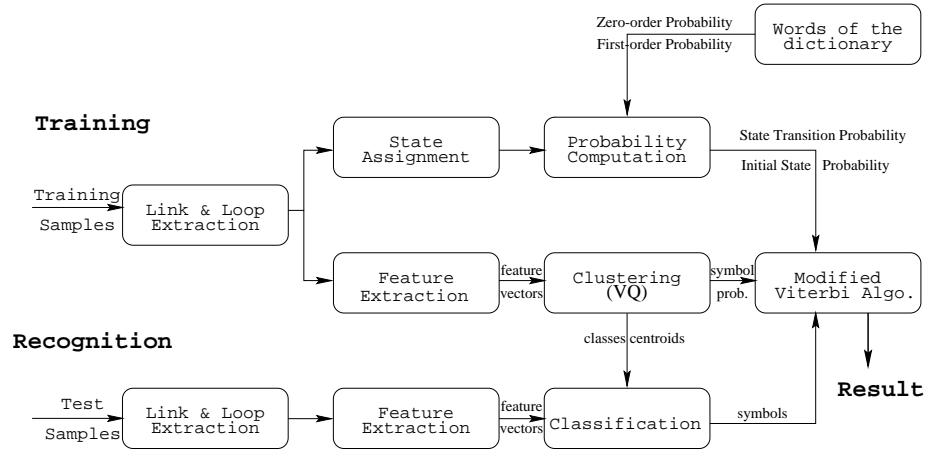


Figure 9.1: A block diagram of the HMM-based lexicon-driven recognition system.

9.2 Lexicon-Driven Scheme

This section attempts to solve the second drawback of the word-model recognition system. It presents a single HMM lexicon-driven recogniser and discusses the model structure. Finally this section provides evaluation results of applying the lexicon-driven scheme to recognise typewritten Arabic script.

9.2.1 System Overview

In this scheme the whole lexicon is represented by a single HMM where each word is represented by one path through the HMM. The lexicon-driven scheme divides the HMM states into small groups, elementary units. Each unit represents a character and consists of a varying number of states. Each state may signify only one link, and this link represents a complete character ‘*d*’, a fraction of a complete character, or touching characters ‘*kā*’. After extracting links and loops, Sections 8.3.1 and 8.3.2, from the skeleton graph, each non-loop link is transformed into an 8-dimensional feature vec-

tor, Sec 8.3.3. It is then assigned to the appropriate symbol using VQ.

The system shown in Fig 9.1 may be described in two stages: the training stage and the recognition stage. In the training stage, each symbol is assigned to one of the states within an elementary unit. The stage also incorporates a high-level knowledge from the given vocabulary and a low-level knowledge from the training samples to estimate the HMM parameters. The recognition stage simply retrieves an ordered list of state sequences associated with a given sequence of observations/symbols.

9.2.1.1 Training stage

In Chapter 8, a number of preliminary preprocessing operations followed by link and loop extraction precede the transformation of the skeleton graph into a sequence of feature vectors. Parallel to this, the link sequence is also passed to the state assignment. In the state assignment, each link is labelled with its semantic meaning, e.g. this edge is the beginning of *Seen* ‘س s’, that edge is a complete *Dal* ‘د d’, that edge is a 2-link complex loop of *Sad* ‘س-د -s-’ and so on. By calculating the number of entries and their labelled meanings, the symbol probabilities are computed. The other valuable information source is the given vocabulary from which the letter-based probabilities are observed. This high-level knowledge is incorporated with the symbol probability previously obtained.

9.2.1.2 Recognition stage

Like the training stage, the recognition stage begins by applying the procedures of link and loop extraction and feature extraction to the input word skeleton. A sequence of observations is obtained by assigning each link to the closest codebook symbol using the k-means clustering algorithm. Finally, the modified Viterbi algorithm is applied, with the model parameters from the training stage, to find the optimal paths through the HMM for the given observation sequence.

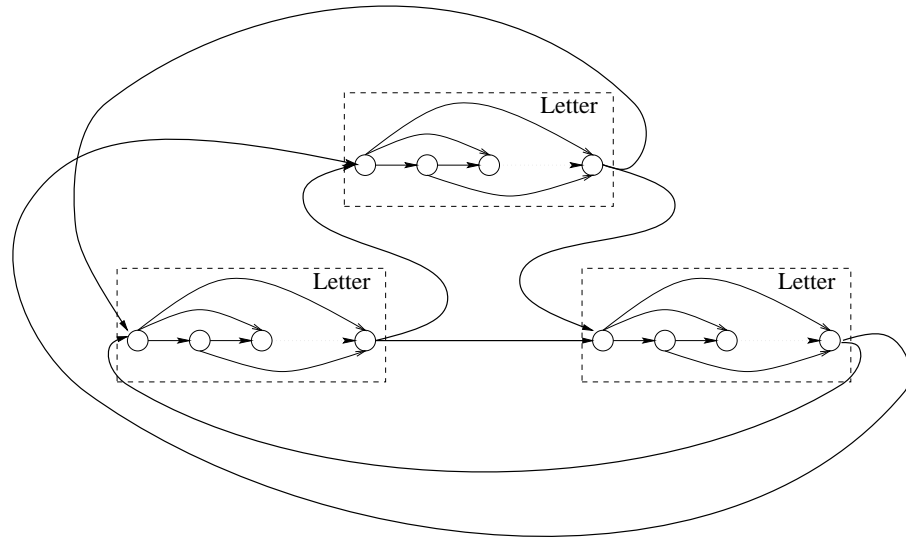


Figure 9.2: Single HMM recognition system.

9.2.2 HMM Implementation

The lexicon-driven scheme builds only one model for all the words in the lexicon and uses different paths (state sequences) through the model to distinguish one pattern from the others. A pattern is classified to the word which has the maximum path probability over all possible paths. This is called a path discriminant HMM in which states are transparent. This means that states are semantically meaningful. The training method here does not comply with the Baum-Welch method or any optimisation criterion, such as the maximum likelihood (ML) [RJ86]. The reason for this is that any optimisation criteria produces a better model but does not preserve the correspondence of the states to individual characters which yields a lower recognition rate [CKZ94].

The HMM is formed from ergodic elementary units, as shown in Fig.9.2. The model contains 166 states divided between 51 elementary units. These units include the 28 basic letters mentioned in Table 2.1, ‘ δ t ’, ‘ ζ \bar{a} ’, ‘ ϵ ν ’, ‘ λ $\bar{l}\bar{a}$ ’, and 19 two-letter combinations. These combinations consist of two letters which are not separated

by any feature point thus it is impossible to decompose the main stroke into more than one link. The nineteen combinations comprise ‘جا’ $gā$, ‘حا’ $hā$, ‘خا’ $hā$, ‘جى’ $gā$, ‘حى’ $hā$, ‘خى’ $hā$, ‘با’ $bā$, ‘تا’ $tā$, ‘ثا’ $tā$, ‘نا’ $nā$, ‘يا’ $yā$, ‘بى’ $bā$, ‘تى’ $tā$, ‘ثى’ $tā$, ‘نى’ $nā$, ‘يى’ $yā$, ‘لى’ $lā$, ‘كا’ $kā$, and ‘كى’ $kā$.

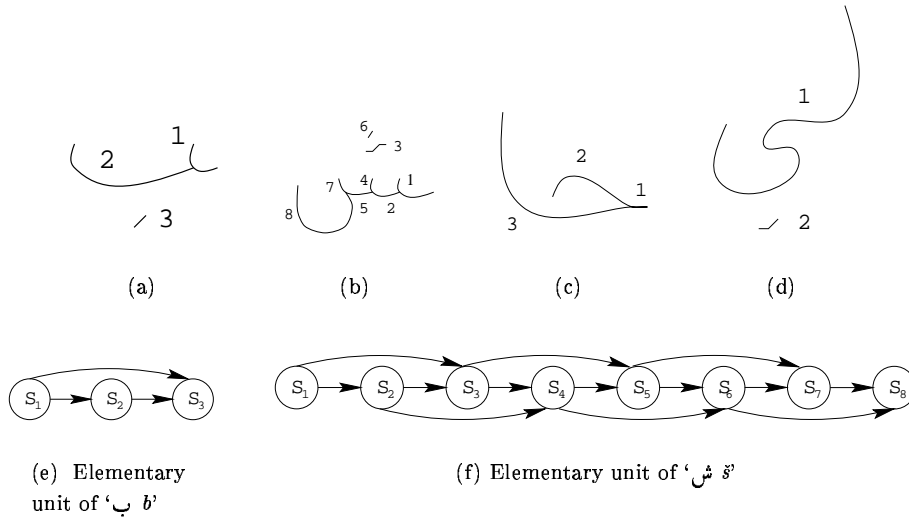


Figure 9.3: The end forms of letters: (a) ‘ب’ b and (b) ‘ش’ sh . The two-letter combination of: (c) ‘حا’ $hā$ and (d) ‘لى’ ly Elementary units representing letters: (c) ‘ب’ b and (d) ‘ش’ sh .

Each elementary unit represents at least one letter and is structured as a left-to-right HMM. The number of states in that model is relative to the number of links the letter or the two-letter combination has. For example the letter ‘ب’ b may be decomposed into: two links when it is in the isolated, initial, or middle forms, or three links when it is in the end form. This implies that the elementary unit representing letter ‘ب’ b has as little as three states which can sequentially preserve the maximum number of links for letter ‘ب’ b . The letter ‘ش’ sh may be decomposed into eight links or fewer. This

implies that the elementary unit representing letter ‘ش s ’ comprises eight states. Fig 9.3 shows the end form of letters ‘ب b ’ and ‘ش s ’. It also shows the combinations ‘ح h ’ and ‘ل l ’.

9.2.2.1 Symbol probabilities

The VQ technique is most often used for assigning symbols in the discrete HMM approach. The K-means clustering algorithm, Sec 3.2.4.1, is applied to partition the training samples in the Euclidean space into a number of symbols/classes. This step was discussed in Sec 8.4.3.1. The symbol probabilities can be calculated as follows

$$b_i(k) = \frac{\text{No. of times in state } i \text{ and observing symbol } v_k}{\text{Total number of times in state } i} \quad (9.1)$$

The symbol probabilities computation is completed in the training stage. In the recognition stage each feature vector is assigned to the nearest codebook symbol centroid based on the Euclidean distance measure.

9.2.2.2 State probabilities

An important step to putting the HMM into practice is to estimate the model’s parameters. The first step is to derive the dictionary statistics from the given vocabulary which includes the zero-order and the first-order letter transition probabilities.

$$P_0(\alpha) = \frac{\text{No. of words starting with letter } \alpha}{\text{Total number of words in the lexicon}} \quad (9.2)$$

$$P_1(\alpha \rightarrow \beta) = \frac{\text{No. of transitions from letter } \alpha \text{ to letter } \beta}{\text{Total number of transitions from letter } \alpha} \quad (9.3)$$

Where $P_0(\alpha)$ is the probability that the initial letter in a given word is α , and $P_1(\alpha \rightarrow \beta)$ is the probability that the letter β follows the letter α in a given word.

The initial state probability can be computed based on the location of the state within its elementary unit and the type of elementary unit whether it represents a single letter or touching letters. If the state is the first within a unit which represents a single letter

then the initial probability is equal to P_0 of that letter. If the elementary unit represents two touching letters then the initial probability of the first state equals the product of P_0 of the first letter and the letter transition probability from the first letter to the second letter $P_1(\alpha \rightarrow \beta)$. The initial state probability equals zero otherwise.

$$\pi_i = \begin{cases} P_0(\alpha) & i \text{ is the } 1^{st} \text{ state in the elementary unit } \alpha \\ P_0(\alpha) \times P_1(\alpha \rightarrow \beta) & i \text{ is the } 1^{st} \text{ state in the elementary unit } \alpha\beta \\ 0 & \text{Otherwise} \end{cases}$$

The state transition probability within the same elementary unit is calculated as follows

$$\begin{aligned} P_{s_i \rightarrow s_j}(\alpha) &= \frac{\text{Transition from state } s_i \text{ to } s_j}{\text{Total number of training samples of letter } \alpha} \\ &= P(q_j \text{ at } t+1 | q_i \text{ at } t) \end{aligned} \quad (9.4)$$

The state transition probability between any two states in the model is calculated accordingly

$$a_{ij} = \begin{cases} 0 & i \text{ \& } j \text{ are middle states in different letters} \\ 0 & i \text{ \& } j \text{ are in the same letter, } i \geq j \\ P_{s_i \rightarrow s_j}(\alpha) & i \text{ \& } j \text{ are in the same letter, } i < j \\ P_1(\alpha \rightarrow \beta) & i \text{ is the last state in } \alpha \text{ \& } j \text{ is the } 1^{st} \text{ state in } \beta \end{cases}$$

9.2.2.3 Modified Viterbi Algorithm

Each word in the lexicon is represented by at least one path through the HMM. There are several possible ways of locating the optimal state sequence associated with the given observation sequence. I use a modified form of the Viterbi algorithm, Sec 3.3.2.2, in order to find the optimal path and some near-optimal paths.

For the sake of word recognition application, a considerable improvement can be made if more than the first optimal path is recovered. The approach is to extend the δ and ψ to another dimension which represents the choice W . Assuming the model in state j at instance t then all the possible $\delta_{t-1}(i, w)$ are considered and the W best paths are recorded in $\psi_t(j, w)$, with their probabilities in $\delta_t(j, w)$ where $w = 1, 2, \dots, W$.

9.2.3 Evaluation Results

The word images belonging to three out of the thirteen fonts presented in Chapter 4 were: Simplified Arabic, Thuluth, and Arabic Traditional fonts. Each image first passed the five step processing sequence mentioned in Chapter 8: thinning, skeleton modification, link and loop extraction, feature extraction, and VQ. This resulted in a sequence of observations that were introduced to the HMM. Table 9.1 shows the recognition rate of the implemented system.

Font type	Recognition rate (%)	
	Top-1	Top-5
<i>Simplified Arabic</i>	70.3%	94.2%
<i>Thuluth</i>	69.8%	93.1%
<i>Arabic Traditional</i>	65.7%	90.6%

Table 9.1: The recognition rates of the single HMM lexicon-driven system.


		
System Output		$P(\lambda O)$
	قيل $q\bar{il}$	8.12×10^{-11}
†	فيل $f\bar{il}$	3.72×10^{-11}
	قبل qbl	6.51×10^{-12}
†‡	فبل fbl	4.31×10^{-12}
	قيل $q\bar{il}$	1.84×10^{-12}

Table 9.2: The system output of the word sample 'قيل $q\bar{il}$ - said' from J font. † means not in the lexicon, and ‡ means not a valid Arabic word.

To assess the performance of this scheme, the HMM was trained using 800-word lexicon. Part of the corpus was used for training purposes. Tables 9.2 , 9.3 , and 9.4 show the system output of three word samples one from each font, J , K and L . Where the system output shows the same word more than once, this means the same word was recognised by a different path through the HMM. At

علي	
System Output	$P(\lambda O)$
علي $\text{'}l\bar{i}$	7.43×10^{-09}
على $\text{'}l\bar{a}$	3.89×10^{-09}
علي $\text{'}l\bar{i}$	9.57×10^{-10}
† حلي $hl\bar{i}$	4.65×10^{-11}
† غلي $g\bar{l}i$	1.54×10^{-12}

Table 9.3: The system output of the word sample 'علي $\text{'}l\bar{i}$ - ALI' from *K* font. † means not in the lexicon, and ‡ means not a valid Arabic word.

العرب	
System Output	$P(\lambda O)$
† الحرب $alh\bar{r}b$	5.13×10^{-16}
الغرب $algr\bar{b}$	3.94×10^{-16}
‡‡ العزب $al\bar{z}b$	9.57×10^{-17}
العرب $al\bar{r}b$	6.03×10^{-17}
‡‡ العذب $al\bar{d}b$	5.54×10^{-17}

Table 9.4: The system output of the word sample 'العرب $al\bar{r}b$ - The Arabs' from *L* font. † means not in the lexicon, and ‡ means not a valid Arabic word.

times, the HMM throws up a sequence which was not included in the lexicon, as shown in the Tables. Eliminating these sequences from the solution list and otherwise considering their successors could enhance the overall recognition rate of the system.

The last result concerns generalisation. Although it is difficult to predict in advance which untrained words the HMM will recognise, I found a number of words recognised by the HMM which were not among the training set nor the lexicon. An example word is shown in

ثقوب	
System Output	$P(\lambda O)$
ثقوب $tqūb$	9.34×10^{-11}
†† ثقوت $tqūt$	1.56×10^{-11}
†† ثقوت $tqūt$	1.07×10^{-12}
†† ثقوز $tqūz$	1.26×10^{-13}
†† ثقوذ $tqūd$	7.17×10^{-14}

Table 9.5: The system output of the word sample 'ثقوب $tqūb$ - holes' from L font. † means not in the lexicon, and ‡ means not a valid Arabic word.

Table 9.5, and other words include 'إستعراض $ʾistrāḍ$, إقتراح $ʾiqtrāḥ$, بصر $bṣr$, بهاء $bhā$, and عامل $ʿāml$.

تماما	
System Output	$P(\lambda O)$
†† تمامما $nmāmā$	1.99×10^{-10}
†† تمامما $bmāmā$	1.25×10^{-10}
†† تمامما $nmāmā$	3.12×10^{-11}
†† تمامما $nmāmā$	1.96×10^{-11}
†† تمامم $nmām$	4.34×10^{-13}

Table 9.6: The system output of the word sample 'تماما $tmāmā$ - completely' from L font. † means not in the lexicon, and ‡ means not a valid Arabic word.

The HMM was not always able to list the correct word among the best five paths. An example of such a case may be seen in Table 9.6, in which a dot is missing owing to a problem with thinning.

9.3 Lexicon-Free Scheme

Although the last section solves the memory limitation caused by using a separate model for each word in the lexicon, the model

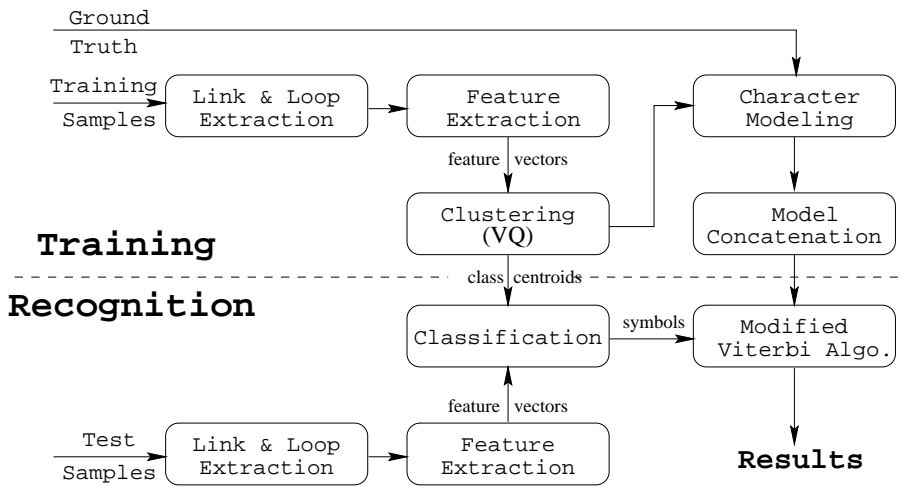


Figure 9.4: A block diagram of the HMM-based lexicon-free recognition system.

parameters are still lexicon sensitive. This implies that any change in the lexicon size yields, reestimating the model parameters using the new lexicon.

This section attempts to solve both drawbacks mentioned at the beginning of this chapter and resulting by using the word-model recognition system. It presents a single HMM lexicon-free recogniser. In this scheme, each letter in the alphabet is represented by a separate HMM. A word is a concatenation of a number of character-models. The training of each model is performed using the Baum-Welch method. The modified VA, as stated in Sec 9.2.2.3, is applied to produce an order list of the best state sequences. This section also provides evaluation results of applying the lexicon-free scheme to recognise handwritten Arabic words.

9.3.1 System Overview

This section gives a brief overview of the implemented scheme. Fig 9.4 shows the block diagram of the system. The blocks of line and loop extraction, feature extraction, VQ, and classification have all been

discussed in Chapter 8. The system is a mixture of path and model discriminant HMMs. This may be clarified as follows: each word is represented by one or more paths through the HMM while at the same time the states that form the path are not transparent. In other words are not semantically meaningful. This is because each word is a concatenation of different models representing characters that form this word.

The system consists of multiple character-models where each model represents only one letter in the alphabet. The character-models are fully interconnected to form the global model.

The system shown in Fig 9.4 may be described in two stages: the training stage and the recognition stage. In the training stage, the character-model parameters are estimated based solely on the training samples associated with their ground truth. The recognition stage simply retrieves an order list of state sequences associated with a given sequence of observations.

9.3.1.1 Training stage

This stage starts with the preliminary preprocessing operations, Sec 8.2, followed by the link and loop extraction then the feature extraction, Sec 8.3, before transforming the skeleton graph into a sequence of symbols, Sec 8.4.

The training stage can be divided into two phases. In the first phase, all the models' parameters are estimated using the Baum-Welch method. This is done through introducing the letter samples to the equivalent model. The training samples for each letter represent different forms of that letter: initial, middle, end, and isolated. In the second phase, all the character-models are fully interconnected to form a single HMM. This is similar to what has been done with the lexicon-driven scheme with an exception of the procedure used to estimate the models'/elementary units' parameters. While the estimation procedure in the lexicon-driven scheme is based on the state assignment, the estimation procedure here is based on the Baum-Welch method.

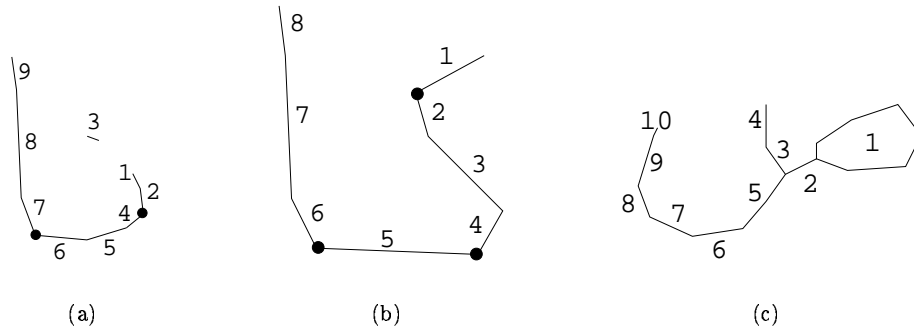


Figure 9.5: Line segments of: (a) a two-letter combination 'نَا *nā*', (b) a two-letter combination 'كَا *kā*', and (c) letter 'ص *ṣ*'. The black mark denotes a turning point.

The training stage ensures that only the observation sequence, and the order of those observations within that sequence, determine the recognised pattern of the input sample.

9.3.1.2 Recognition stage

This is similar to the training stage in obtaining loops, line segments, and other features. The feature vector sequence is assigned to the closest codebook symbols based on the minimum Euclidean distance measure. The observation sequence is then thrown to the HMM and the VA is applied to issue an order list of the best paths through the model. The various state sequences are sorted in descending order relative to the path likelihood probability.

9.3.2 HMM Implementation

Like the lexicon-driven scheme, only one model is built and each path through that model represents a sequence of letters. The system consists of 32 character-models representing the 28 basic letters in the alphabet, Table 2.1, and letters: 'ت *t*', 'ا *ā*', 'ة *ʾ*', and 'ل *lā*'. The character-models are fully interconnected to form the

global model. The number of character-models is less than the number of elementary units in the previous section, because this scheme utilises FS_2 feature set which decomposes the skeleton graph into small line segments. This makes it possible to divide two touching letters even if they share a single link. Consider the two-letter combinations 'न $nā$ ' and 'क $kā$ ' in Fig 9.5. Although each combination forms only one link it is still feasible to decompose this link into separate letters. 'न $nā$ ' is decomposed into two letters: 'न n -' includes 6 line segments, 3 end points, and a down-left turning point, and 'न $-ā$ ' includes three line segments, a left-up turning point, and an end point. 'क $kā$ ' is decomposed into two letters: 'क k -' includes 5 line segments, left-down and down-left turning points, and an end point, and 'क $-ā$ ' includes three line segments, a left-up turning point, and an end point.

One constraint applied to ensure the influence of processing. that is connections to/from a model from/to other character-models, are only permitted through the first and the last states, respectively. This means that the last state in each character-model is connected to the first state in each character-model including the source character-model. These connections are referred to as *interconnections*. Confining the interconnections between various character-model only through two states, the first and the last, prevents the deluded repetitions of a letter within the same word.

The state transition probability of the *interconnections* equals one. This ensures an unbiased transition between letters. On the same level, all other connections between the models are set to zero.

The initial state probability of the first state in all the character-models equals one. The rest of the states have zero initial state probabilities.

An important issue that must be resolved before putting this scheme into use is to decide how many states each character-model should be comprised of. Two different configurations are implemented: the four-state character-models and the various-state character-models. Following are the structural details of each configuration.

9.3.2.1 Four-state models

All the character-models here have an equal number of states, four states. This number was chosen as a compromise between letters with large number of line segments such as ‘س’ and others with a smaller number of line segments as ‘ا’. Fig 9.5 shows that while ‘س’ in isolated form may be decomposed into a simple loop, nine line segments, two branch points, and two end points, ‘ا-ā’ in end form may be decomposed into three line segments, a left-up turning point, and an end point.

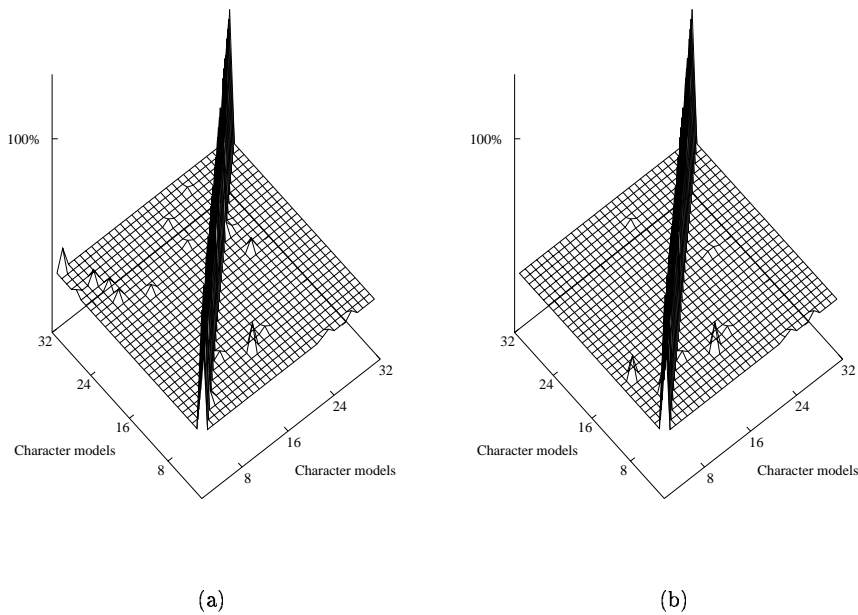


Figure 9.6: Three dimensional depictions showing the confusion matrices of: (a) Four-state character-models, (b) Varying-size character-models. Total number of recognition tests is 12960 associated with 405 character samples representing 32 character-models.

The system contains 128 states equally distributed on the 32 character-models. Each model is a left-to-right HMM with no constraints on the jump margin within. This means that it is valid to transit from state s_x to state s_y as far as $x \leq y$, so it is possible to transit from s_1 to s_4 . Fig 9.6-a shows the confusion matrix of this configuration. The word samples were extracted from font *B*, Sec 4.3.1.

9.3.2.2 Various-state models

The system here comprises 296 states unequally distributed on the 32 character-models. Each character-model has different number of states relative to the number of line segments that the letter has. For example while letter ‘ض’ *d* has 17 states in its character-model, the character-model of letter ‘ة’ *t* has only 5 states.

Fig 9.6 shows that this configuration is more capable of recognising separate letters than the previous configuration. However, what matters most is recognising cursive words rather than separate letters. Following this, the evaluation results of both configurations recognising cursive handwritten words are shown.

9.3.3 Evaluation Results

Word samples belong to the handwritten font *B*, Sec 4.3.1. Each word was transferred into a sequence of discrete symbols as described in Sec 8.4.3.1.

Consider the two words ‘محمد’ *mḥmd* and ‘حمد’ *ḥmd* shown in Fig 9.7. Both words consist of the same letters ‘م’ *m*’, ‘ح’ *ḥ*’, and ‘د’ *d*’. The word ‘محمد’ *mḥmd* is broken into ‘م’ *m*’, ‘ح’ *ḥ*’, ‘م’ *m*’ and ‘د’ *d*’. This means that its path starts with ‘م’ *m*’, then transits to ‘ح’ *ḥ*’, transits back to ‘م’ *m*’, and finally ends up at ‘د’ *d*’. In contrast, the word ‘حمد’ *ḥmd* is broken into ‘ح’ *ḥ*’, ‘م’ *m*’, and ‘د’ *d*’. Therefore the equivalent path starts from ‘ح’ *ḥ*’, transits then to ‘م’ *m*’, and finally ends up at ‘د’ *d*’.

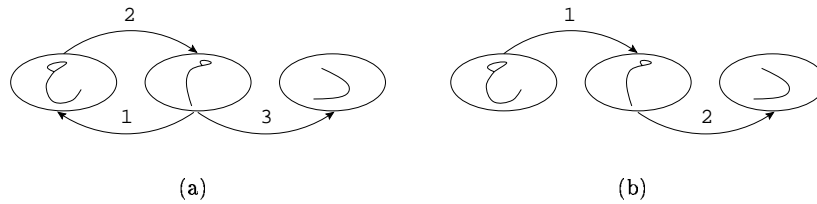


Figure 9.7: Two block illustrations of the word path building through the HMM for: (a) 'محمد *muḥammad* - MOHAMMAD', (b) 'حمد *ḥamad* - HAMAD'. The two words share the same character-models but in a different order and with different frequencies.

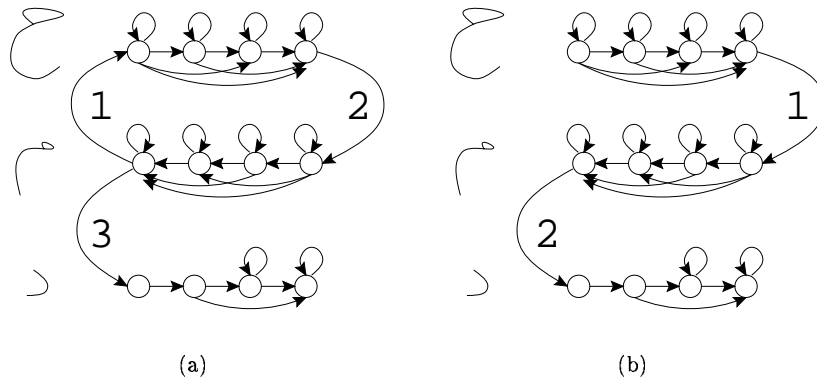


Figure 9.8: The word paths through the HMM for: (a) 'محمد *muḥammad* - MOHAMMAD', (b) 'حمد *ḥamad* - HAMAD'. There are four states in each character-model.

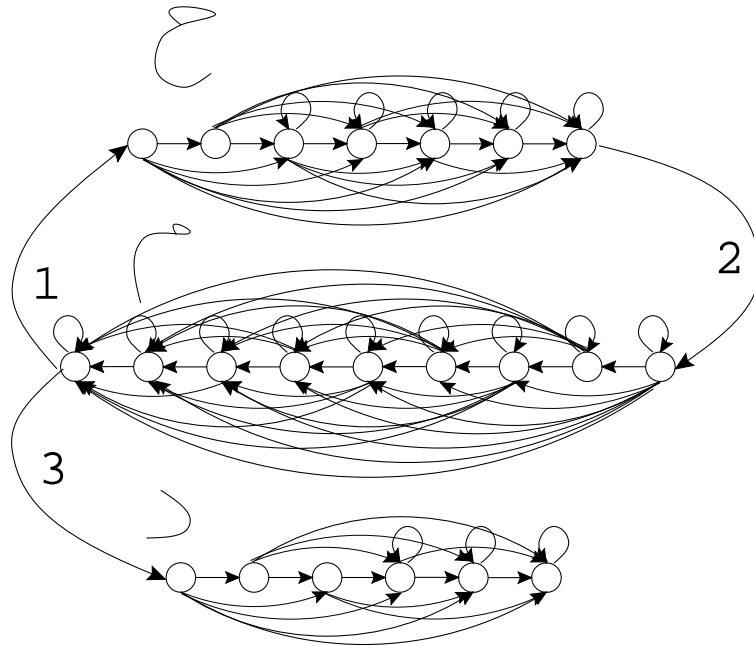
Fig 9.7 illustrates the block diagrams of the two word paths where each block represents a separate character-model. Fig 9.8 shows these two paths after replacing each block in Fig 9.7 with the equivalent four-state character-model. Although all the character-models have the same number of states, it is not necessary that they have the same state-transition matrix form. This is clear when comparing the character-models of ‘م’ *m* and ‘ح’ *h* on one side and the character-model of ‘د’ *d* on the other side. In more detail, when the system is in the first state of the ‘د’ *d* character-model, it is only allowed to transit to the second state. This is not a configuration constraint, but a result of the training samples used to tune this character-model.

Using various-state model configuration illustrates more clearly these differences in model structures. Fig 9.9 shows that none of the three character-models share the same state-transition matrix form. This is not surprising since each letter is decomposed into different line segments, feature points, and turning points.

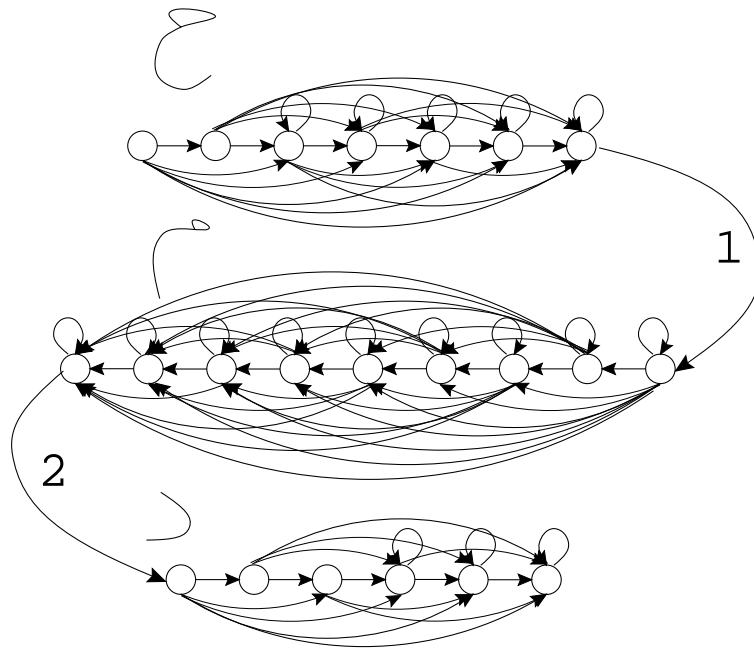
<div style="text-align: center; font-size: 1.5em; font-family: cursive;">مكة</div>			
four-state configuration		various-state configuration	
System Output	$P(\lambda O)$	System Output	$P(\lambda O)$
مرلة <i>mrlt</i>	1.58×10^{-17}	مكة <i>mkt</i>	4.10×10^{-16}
كمكة <i>kmklt</i>	5.96×10^{-18}	بكة <i>bkt</i>	2.50×10^{-16}
براهة <i>bräht</i>	5.51×10^{-18}	مكة <i>mkt</i>	2.19×10^{-16}
بكاهة <i>bkäht</i>	5.32×10^{-18}	كمكة <i>kmkt</i>	8.57×10^{-17}
مرلة <i>mrlt</i>	4.15×10^{-18}	بكة <i>bkt</i>	8.24×10^{-17}

Table 9.7: The system output of the word sample ‘مكة’ *mkt* - MECCA’.

Tables 9.7 and 9.8 show the lexicon-free system output of two word samples, both from the handwritten font *B*. The first sample comprises less symbols in the observation sequence than the second one. This directly affects the likelihood probability of the solution.



(a)



(b)

Figure 9.9: The word paths through the HMM for: (a) 'محمد' *muḥammad* - MOHAMMAD', (b) 'حمد' *ḥamad* - HAMAD'. Each character-model has a varying number of states relative to the number of line segments the letter has.

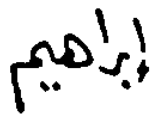
			
four-state configuration		various-state configuration	
System Output	$P(\lambda O)$	System Output	$P(\lambda O)$
ابراجم <i>abrāġm</i>	8.05×10^{-28}	ابراهيم <i>abrāhym</i>	1.33×10^{-29}
كبراجمل <i>kbrāġml</i>	8.04×10^{-28}	كابراهيم <i>kābrāhym</i>	1.30×10^{-29}
ابراهيم <i>abrāhym</i>	8.04×10^{-28}	ابراهيمد <i>abrāhymd</i>	1.30×10^{-29}
ابراهيمد <i>abrāhymd</i>	8.03×10^{-28}	ابراهيمس <i>abrāhymms</i>	1.29×10^{-29}
لكبراجم <i>lkbrāġm</i>	8.01×10^{-28}	كابراهيم <i>kābrāhym</i>	1.25×10^{-29}

Table 9.8: The system output of the word sample 'ابراهيم' *abrāhym* - ABRAHAM'.

	Four-state model		Various-state model	
	Top-1	Top-5	Top-1	Top-5
Without spelling-check	69%	77%	72%	81%
With spelling-check	84%	93%	87%	97%

Table 9.9: The recognition rates of the single HMM lexicon-free system.

These tables illustrate the output of both configurations: the various-state and the four-state. The various-state configuration outperforms the four-state configuration. This result seems logical since each letter comprises a different number of symbols. Moreover, the various forms of the same letter are decomposed into different observation sequences.

Table 9.9 shows the recognition rate of each configuration. When using a simple Arabic spell-check, like the one found in Microsoft Word, the system performance was enhanced by a rate of 15%-16%. These recognition rates are expected to decrease when including more handwritten fonts, regardless of the implemented configuration.

9.4 Summary

This chapter presented two approaches of recognising cursive Arabic words. Both approaches transferred the skeleton graph of the word image into a sequence of discrete symbols which were then processed using a single HMM.

The first approach divided the model into smaller units where each unit represented a letter. The training in this approach was based on the state assignment method. The dictionary statistics were used to calculate the state probabilities, initial state and state transition probabilities. This lexicon-driven recognition system was used to recognise cursive words extracted from typewritten fonts, previously mentioned in Chapter 4.

The second approach was lexicon-free. The system was a collection of character-models and with each model was a left-to-right HMM. Two configurations were implemented to construct the character-models. The first configuration used the same number of states for all models whereas the second assigned a different number of states to different models, based on the number of line segments that letter had. The training of each character-model was performed separately using the Baum-Welch method. The resulting global HMM was used to recognise cursive handwritten Arabic words.

CHAPTER 10

CONCLUSION

Cursiveness represents the main challenge when recognising Arabic words. Previous research has proved the failure of segmenting the word into its characters. The implemented algorithms for feature extraction and classification determine the operational characteristics of the recognition system. The research effort presented in this dissertation to overcome the obstacle of cursiveness can be presented using two techniques: the segmentation-free technique and the segmentation-based technique.

The segmentation-free technique, which implements the word-model scheme, processes the word as a single unit. The feature extraction here applies a two-step process, a normalised polar transformation followed by a two-dimensional Fourier Transform, to represent the entire word in a global feature vector. This process results in a two-dimensional Fourier spectrum that tolerates affine changes in size, displacement, and rotation.

The resulting Fourier spectrum can be used as a prototype in a template-based system to recognise samples of words from a predefined lexicon. Each word in that lexicon is represented by a separate template which is formed by bounding Fourier coefficients falling within the lower frequency bands. These coefficients simultaneously contain the global features of the same word written in different fonts and the discriminant features of similar words in the same font. Beside the typical recognition of similar, but not identical,

samples of the same word, the template-based recognition system can correctly recognise words despite differences in diacritics, which is a common orthographic variation, and word samples that are deformed due to noise or crude segmentation. A drawback of this system is that it performs poorly when a template is formed from multiple handwritten fonts. This is due to the averaging process that produces the template and weakens the discriminant features between various words within that template.

To prevent the discriminant features from being weakened, these features must be utilised and exploited by a learning method that can observe the differences and similarities among various samples. However firstly, the problem of dimensionality needs to be solved. This is achieved via dividing half of the Fourier spectrum into six wedge-shaped sectors and then implementing the segmental VQ. This transfers the two-dimensional feature vector into a one-dimensional observation sequence consisting of six discrete symbols. At this stage HMMs may be used for training and recognition purposes. Each word in the lexicon is now represented by a separate HMM. This is referred to as the model discriminant HMM. The training of each model is performed using the Baum-Welch method. The system achieves a higher recognition rate compared to the template-based recognition system.

The word-model scheme has two drawbacks: the limited recognition capability and the excessive use of memory. The first drawback is due to using a predefined lexicon, and that limits the outputs to the words in that lexicon. The second drawback occurs when the lexicon size exceeds 10,000 words. This also causes a delay in producing the equivalent outputs for all words in the lexicon.

The segmentation-based technique attempts to solve the previous drawbacks by presenting two single HMM recognition systems. Both systems implement the analytical approach in extracting feature vectors from word samples. This implies that the systems first apply a procedure to divide the word into smaller units and then extract features from each unit separately. This process results in a

sequence of feature vectors. VQ is then applied to cluster the feature space and transform these n -dimensional feature vectors into the discrete symbols making up the observation sequence. Observation sequences are used to train a single HMM. At the recognition stage, the input word observation sequence is presented to the HMM and the Viterbi algorithm issues an order list of all possible state sequences through the model that are associated with the input sequence. This is done by calculating the likelihood probability of each path, then sorting them in descending order.

Each of the two single HMM recognisers has a different scheme. Each scheme extracts a different feature set, and implements a different training method. On the other hand, both schemes decompose the word skeleton into various links. The first scheme treats each link as a single unit where the second scheme breaks this link into multiple feature vectors using line approximation. This means that while the first scheme transfers the word skeleton into a number of discrete symbols equal to the number of links, the second scheme has a far greater number of symbols. Moreover, the second scheme includes the feature points which defines the beginning and the end of a link and the turning points, which define where the skeleton has a dramatic change, as separate features. This makes the second feature set more capable of describing the word skeleton than the first feature set.

The training methodology in the two schemes is different. While the first scheme bases its training procedure on the state assignment method, the second scheme applies the Baum-Welch method for training. The state assignment method requires a massive training data set so that the resulting parameters actually reflect the true image. Since the first scheme is only implemented in typewritten fonts the problem of style variation does not affect its performance.

10.1 What can we learn?

Each one of the four implemented recognition systems in this dissertation has its own advantages and drawbacks. First, let us consider the template-based recognition system. It is very simple to extract a certain set of coefficients from the Fourier spectrum and have this set to represent a word. Although only two different distance measures are implemented that are the Euclidean distance and the Standard distance measure, it is possible to use a statistically related metric. One of these is the Mahalanobis distance [NS93]. This takes into accounts the correlation among the features. One problem to implement the Mahalanobis distance is that the feature vector extracted from the Fourier spectrum is at least 106-dimensional vector, which makes the dimensions of the variance-covariance matrix Σ equal 106×106 . Such a matrix is difficult to find its inverse with the limited data set used in this dissertation. Moreover, implementing the template-based recogniser is not the main objective of this dissertation, it rather a way to prove the discriminating power of the Fourier feature set.

The template-based system deforms the word templates when recognising multiple fonts, particularly handwritten fonts. This is due to the averaging process when calculating a single template from word samples belonging to different fonts. Therefore this type of system is more appropriate for recognising one source of data or different sources with a minimum style variation to ensure a high recognition rate.

The multi-HMM recognition system shares the basic concept with the template-based system; both have a separate model/template for each word in the lexicon. However, due to its learning ability the multi-HMM recognition system is more capable of discerning the style variations. The implementation of this system requires the reduction of the dimensionality of the Fourier spectrum into one dimension. Half of the spectrum is divided into six wedge-shaped sectors. Each sector reduces the effect of the variations within a 30°

wedge-shaped sampling window. This window size is acceptable for typewritten and some handwritten fonts. However, increasing the number of these sectors through decreasing the sampling window size will provide a more specific description of the word, and improve the recognition rate.

The data set used for training and recognition purposes by the template-based and the multi-HMM recognition systems has two main features: the limited lexicon size and the data source diversification. The limited lexicon size is due to that this data set is not taken from a ready-made database, but it is scanned from Arabic manuscripts. This process consumes a considerable time. The data source diversification is essential to test the Fourier features on a wide spectrum of typewritten and handwritten fonts.

In the two single HMM recognition systems, each word is represented by a path through the model. The lexicon-driven scheme has the feature of limiting the output to a predefined lexicon. The data set used here is different from that used by the template-based and the multi-HMM recognition systems. This is because the objective here is to see how incorporating the high-level knowledge from the dictionary can improve the recognition rate. This would not be possible with the limited lexicon size used previously. Thus, I can import Arabic text from the Internet and print it using three different typefaces using Arabic Windows, then scan these documents. Accordingly, the comparison between the multi-HMM recogniser and the single HMM lexicon-driven recogniser is not feasible since each recogniser is using a different test bed. However, considering Tables 7.2 and 9.1 we notice that where the multi-HMM system recognition rate ranges between 90% and 99%, for a 145-word lexicon, the single HMM lexicon-driven system can hardly reach 70%, for an 800-word lexicon. The cause of this, besides the lexicon size difference, is the feature set. The Fourier features seem to be more capable for describing word samples than the FS_1 feature set.

The single HMM lexicon-driven system has one drawback, that is when a word, which is not included in the lexicon, appears in

the system output list, it cannot be accepted as a solution. This commonly occurs when recognising an input word from outside the lexicon. By contrast, this could be the most significant advantage of the single HMM lexicon-free system. Here, the training is performed on the character level and the word is a concatenation of different character-models. Moreover, there is no lexicon to restrict the recognition results. Using a word-processor spell-check can eliminate some of the letter sequences that the HMM produces. This scenario is not always applicable: to illustrate this consider the system output ‘*كإبراهيم* *k-ibrāhīm* - as Abraham’ resulting after introducing the input image of the word ‘*إبراهيم* *ibrāhīm* - Abraham’. In English the additional letter is removed by the spell-check whereas in Arabic ‘*كإبراهيم* *k-ibrāhīm*’ is accepted as a valid word derived from the source ‘*إبراهيم* *ibrāhīm*’

Due to time limitation, only a single handwritten font *B* is tested using the single HMM lexicon-free system. Two objectives are behind this implementation: verifying the recognition ability of this scheme and testing the discriminant power of the FS_2 feature set. The data set, which is used for training and recognition purposes by the single HMM lexicon-free system, is different from that used by the template-based and the multi-HMM systems. However, the single HMM lexicon-free system has one advantage over the latter systems, that it implements a structural feature set that is invariant to centroid location, which could alter the resulting polar image implemented by the template-based and the multi-HMM systems.

10.2 Contributions

The following are the significant technical contributions of research presented in this dissertation:

- *Arabic handwriting:* This dissertation is the first to attempt to recognise words taken from ancient Arabic manuscripts, which represent a rich and valuable collection around the world.

- *Word templates/models:* This dissertation is the first to use the 2D-FT together with the polar map to recognise Arabic script. The Fourier spectrum provides a large set of features. It is possible to reduce this set to a subset that is capable of discriminating between samples of similar words and different samples of the same word.
Reducing the Fourier spectrum into a one-dimensional discrete symbol sequence facilitates building a separate model for each word in the lexicon. This has the advantage of learning various handwriting styles.
- *Dictionary statistics:* Incorporating the high-level knowledge of the dictionary with the low-level knowledge of the training samples provides the HMM with the necessary parameters to recognise different samples for words in the predefined lexicon.
- *Single model recogniser:* The two HMM schemes used for recognising Arabic script in Chapter 9 are the first to use structural features to describe the character/word images. Makhoul et al [MLR⁺96, MSLB98, BSM99, NBL⁺99] implemented a single HMM-based system to recognise typewritten Arabic text using statistical features. They used the feature vector extracted separately from each image column for training and recognition purposes. In my implementation, the features are structural and extracted from the word skeleton.

10.3 Future Work

Some achievements have been made over the course of this research and further work in the areas presented here is promising. Future work could be presented in five areas:

- *Expanding the lexicon:* Since the main theme of this dissertation is to recognise words from Arabic manuscripts, it was not possible to use the available corpus which is mainly for recent typewritten script. The limited time frame of this research re-

flects the limited size of the lexicon. Hence, the first objective in order to continue is to increase the lexicon capacity. This process is time consuming however it is essential for proving the system's efficiency.

- *Smaller sampling windows:* Dividing the Fourier spectrum into more wedge-shaped sectors can enhance the overall recognition rate. A sampling window with a size of $5^\circ - 10^\circ$ can take into consideration more detailed variations than the current system.
- *The Multi-HMM system using structural features:* Instead of using the Fourier features, structural features can be extracted from the word image and transferred into a sequence of discrete symbols which are then used for training and recognition purposes. This may have a higher recognition rate due to the varying size of each word sequence and the invariance of these features to the image centroid location.
- *Using a different feature set with the lexicon-driven system:* The feature set which is based on decoding the line segments belonging to each link will provide a better description of a word image especially if this is incorporated in the dictionary statistics.
- *Including more fonts to the lexicon-free system:* The simplicity of training various character-models makes this approach more favourable than the state assignment training methodology. However, segmenting words into their characters for training purposes could be a tedious and time consuming task for highly varying writing styles.

Bibliography

- [AA91] A. Amin and S. Alfedaghi. Machine recognition of printed arabic text utilizing natural language morphology. *Int. Journal Man-Machine Studies*, 35:769–788, 1991.
- [AA93a] I. Abuhaiba and P. Ahmed. Restoration of temporal information in off-line arabic handwriting. *Pattern Recognition*, 26(7):1009–1017, 1993.
- [AA93b] A. Amin and H. Alsadoun. Issues on arabic character recognition. *Arabian Journal for Science and Engineering*, 18(3):320–341, 1993.
- [AA94] A. Amin and H. Alsadoun. Hand printed arabic character recognition system. In *Proceeding of The 12th International Conference on Pattern Recognition*, pages 536–539. IAPR, 1994.
- [AA96] Y. Alohalı and P. Ahmed. A software environment for the development and evaluation of arabic character recognition systems. In *The 5th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1996. Cambridge University Press.
- [AAF96] A. Amin, H. Alsadoun, and S. Fischer. Hand-printed arabic character recognition system using an artificial network. *Pattern Recognition*, 29(4):663–675, 1996.

- [AB94] M. Altuwaijri and M. Bayoumi. Arabic text recognition using neural networks. In *Proceedings of IEEE International Symposium on Circuits and Systems*, pages 415–418, London-UK, 1994. IEEE.
- [AC93] Y. Arriola and R. Carrasco. Real-time parallel processing implementation for word recognition using mlp and hmm. *Microprocessor and Microsystems*, 17(10):611–626, 1993.
- [AE94] G. Adams and R. Evans. Neural networks for frequency line tracking. *IEEE Transactions on Signal Processing*, 42(4):936–941, 1994.
- [AFCB94] M. Arbabi, S. Fischthal, V. Cheng, and E. Bart. Algorithms for arabic name transliteration. *IBM Journal of Research and Development*, 38(2):183–193, 1994.
- [AH88] H. Abdelazim and M. Hashish. Arabic reading machine. In *The 10th National Computer Conference*, pages 733–744. King Abdul Aziz University, 1988.
- [AH89] H. Abdelazim and M. Hashish. Automatic recognition of handwritten hindi numerals. In *Proceedings of The 11th National Computer Conference*, pages 287–298, Dhahran, Saudi Arabia, 1989. King Fahad University Of Petroleum and Minerals.
- [AH94] B. Albadr and R. Haralick. Symbol recognition without prior segmentation. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 2181, pages 303–314. SPIE, 1994.
- [AH96] B. Albadr and R. Haralick. Segmentation-free recognition of arabic text. In *The 5th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1996. Cambridge University Press.

- [AHD98] I. Abuhaiba, M. Holt, and S. Datta. Recognition of off-line cursive handwriting. *Computer Vision and Image Understanding*, 64(1):19–38, 1998.
- [AJAA96] R. Amer, K. Jambi, I. Albideiwi, and A. Alaidarous. Arabic character recognition. Technical report, Computer Science Department, Faculty of Science, King Abdul Aziz University, 1996.
- [AK93] O. Agazzi and S. Kuo. Hidden markov model based optical character recognition in the presence of deterministic transformations. *Pattern Recognition*, 26(12):1813–1826, 1993.
- [AK94] P. Ahmed and M. Khan. Computer recognition of arabic script based text : The state of art. In *The 4th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1994. Cambridge University Press.
- [AKHM80] A. Amin, A. Kaced, J. Haton, and R. Mohr. Hand-written arabic character recognition by the irac system. In *Proceedings of The 5th International Conference on Pattern Recognition*, pages 729–731, Floarida, US, 1980. IAPR.
- [Ala96] H. Abd Alazim. A hybrid fuzzy-neural approach to the recognition of arabic script. In *The 5th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1996. Cambridge University Press.
- [Ali97] A. Alimi. An evolutionary neuro-fuzzy approach to recognise on-line arabic handwriting. In *Proceeding of The Fourth International Conference on Document Analysis and Recognition, ICDAR'97*, pages 382–386, Ulm, Germany, August 1997. IAPR.

- [All94] M. Allam. Arabic character recognition. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 2181, pages 351–359. SPIE, 1994.
- [All95] M. Allam. Segmentation vs. segmentation-free for recognising arabic text. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 2422, pages 228–235. SPIE, 1995.
- [AM82] A. Amin and G. Masini. Machine recognition of cursive arabic words. In *Proceeding of The Applications Of Digital Image Processing IV*, volume 359, pages 286–292. SPIE, 1982.
- [AM86] A. Amin and G. Masini. Machine recognition of multi font printed arabic texts. In *Proceedings of The 8th International Joint Conference on Pattern Recognition*, pages 392–395, Paris, France, 1986. IAPR.
- [AM89] A. Amin and J. Mari. Machine recognition and correction of printed arabic text. *IEEE Trans. on Systems, Man, and Cybernetics*, 19(5):1300–1306, 1989.
- [AM95a] I. Abuhaiba and S. Mahmoud. Fuzzy graphs to recognise handwritten arabic characters. *Arabian Journal for Science and Engineering*, 20(1):77–93, 1995.
- [AM95b] B. Albadr and S. Mahmoud. Survey and bibliography of arabic optical text recognition. *Signal Processing*, 41:49–77, 1995.
- [AM97] A. Amin and W. Mansoor. Recognition of printed arabic text using neural networks. In *Proceeding of The Fourth International Conference on Document Analysis and Recognition, ICDAR'97*, pages 612–615, Ulm, Germany, August 1997. IAPR.
- [AMG94] I. Abuhaiba, S. Mahmoud, and R. Green. Recognition of handwritten cursive arabic characters. *IEEE*

- Trans. on Pattern Analysis and Machine Intelligence*, 16(6):664–671, 1994.
- [Ami85] A. Amin. Arabic handwriting recognition and understanding. In *Proceeding of The Computer Processing and Transmission Of The Arabic Language Workshop*, pages 1–37, Kuwait, 1985.
- [Ami88] A. Amin. Ocr of arabic text. In *The 4th International Conference On Pattern Recognition*, pages 616–625, Cambridge, UK, 1988.
- [Ami97] A. Amin. Off-line arabic character recognition - a survey. In *Proceeding of The Fourth International Conference On Document Analysis and Recognition*, pages 596–599, Ulm, Germany, 1997. IAPR.
- [Ami98a] A. Amin. Off-line arabic character recognition : The state of the art. *Pattern Recognition*, 31(5):517–530, 1998.
- [Ami98b] A. Amin. Recognition of printed arabic text using machine learning. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 3305, pages 63–70. SPIE, 1998.
- [AMSH90] H. Abdelazim, A. Mousa, Y. Saleh, and M. Hashish. Arabic text recognition using partial observation approach. In *Proceeding of The 12th National Computer Conference*, pages 427–437, Riyadh, Saudi Arabia, 1990. King Saud Univeristy.
- [AU90] S. Alemami and M. Usher. On-line recognition of handwritten arabic characters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 12(7):704–710, 1990.

- [AU92] H. Alyousefi and S. Upda. Recognition of arabic characters. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(8):853–857, 1992.
- [Aud99] H. Auda. Nearest neighbor classification of handwritten numerals using class probability matrices. In *Proceeding of The Seventh International Conference On Artificial Intelligence Applications*, pages 175–183, Cairo, Egypt, 1999. Egyptian Computer Society.
- [AY87] H. Almuallim and S Yamaguchi. A method of recognition of arabic cursive handwriting. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 9(5):715–722, 1987.
- [Bai92] H. Baird. Anatomy of a versatile page reader. *Proceedings Of The IEEE*, 80(7):1059–1065, 1992.
- [Bel95] J. Bell. Arabic ocr. *Reader on The Internet*, <http://leb.net/archives/reader>, 1995.
- [BK94] C. Bose and S. Kuo. Connected and degraded text recognition using hmm. *Pattern Recognition*, 27(10):1345–1363, 1994.
- [BMZ96] J. Bell, A. Matveev, and P. Zemánek. Arabic ocr with al alamiah’s a–qari’ al-ali. In *The 5th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1996. Cambridge University Press.
- [Bou96] F. Bouslama. Neuro-fuzzy techniques in the recognition of written arabic characters. In *Proceeding of North American Fuzzy Information Process*, pages 142–146, Berkeley, CA, 1996. IEEE.
- [BRS95] H. Bunke, M. Roth, and E. Schukattalamazzini. Off-line cursive handwriting recognition using hidden markov models. *Pattern Recognition*, 28(9):1399–1413, 1995.

- [BSM99] I. Bazzi, R. Schwartz, and J. Makhoul. An omnifont open-vocabulary ocr system for english and arabic. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(6):495–504, 1999.
- [BZ94] J. Bell and P. Zemanek. Test of two arabic ocr programs. *Arabic and Computer on The Internet*, <http://www.hf-fak.uib.no/smi/ksv/arabocr.html>, 1994.
- [CDD97] C. Cracknell, A. Dowton, and L. Du. An object oriented form description language and approach to handwritten form processing. In *Proceeding of The Fourth International Conference on Document Analysis and Recognition, ICDAR'97*, pages 180–184, Ulm, Germany, August 1997. IAPR.
- [CF84] D. Casasent and D. Fetterly. Recent advances in optical pattern recognition. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 456, pages 105–115. SPIE, 1984.
- [CKS95] M. Chen, A. Kundu, and S. Srihari. Variable duration hidden markov model and morphological segmentation for handwritten word recognition. *IEEE Transactions On Image Processing*, 4(12):1675–1687, 1995.
- [CKZ94] M. Chen, A. Kundu, and J. Zhou. Off-line handwritten word recognition using a hmm type stochastic network. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(5):481–496, 1994.
- [CS83] D. Casasent and V. Sharma. Fourier-transform feature-space studies. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 449, pages 2–8. SPIE, 1983.
- [DFV97] G. Dzuba, A. Filatov, and A. Volgunin. Handwritten zip code recognition. In *Proceeding of The Fourth Inter-*

- national Conference on Document Analysis and Recognition, ICDAR'97*, pages 766–770, Ulm, Germany, August 1997. IAPR.
- [DPH93] J. Deller, J. Proakis, and J. Hansen. *Discerte-Time Processing Of Speech Signals*. Macmillen Publication Company, 1993.
- [DTLH92] A. Downton, R. Tregidgo, C. Leedham, and Hendrawan. Recognition of handwritten british postal addresses. In S. Impedovo and J. Simon, editors, *From Pixels To Features III: Frontiers In Handwriting Recognition*, pages 129–144. Elsevier Science Publishers B.V., 1992.
- [EA90] F. Elkhaly and M. Sid Ahmed. Machine recognition of optically captured machine printed arabic text. *Pattern Recognition*, 23(11):1207–1214, 1990.
- [EAIK94] A. Emam, H. Alkhatib, M. Ismail, and E. Korany. Character recognition of arabic script. In *The 4th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1994. Cambridge University Press.
- [EE89] T. Elsheikh and S. Eltaweel. Real-time arabic handwritten character recognition. In *The Third International Conference On Image Processing and It's Applications*, pages 212–216. IEE, 1989.
- [EE90a] T. Elsheikh and G. Eltaweel. Real-time arabic handwritten character recognition. *Pattern Recognition*, 23(12):1323–1332, 1990.
- [EE90b] T. Elsheikh and S. Eltaweel. Segmentation of handwritten arabic words. In *Proceeding of The 12th National Computer Conference*, pages 389–402, Riyadh, Saudi Arabia, 1990. King Saud Univeristy.

- [EG88] T. Elsheikh and R. Guindi. Computer recognition of arabic cursive scripts. *Pattern Recognition*, 21(4):293–302, 1988.
- [EGSS99] A. Elyacoubi, M. Gilloux, R. Sabourin, and C. Suen. An hmm-based approach for off-line unconstrained handwritten word modeling and recognition. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 21(8):752–760, 1999.
- [EH88] O. Emam and M. Hashish. Application of hmm of isolated arabic words. In *The 10th National Computer Conference*, pages 761–767. King Abdul Aziz University, 1988.
- [EH89] T. Elsadany and M. Hashish. An arabic morphological system. *IBM Systems Journal*, 28(4):600–612, 1989.
- [ElD98] A. Sharaf ElDin. Arabic manuscripts information system. In *Proceeding of The 6th International Conference and Exhibition on Multi-Lingual Computing*, pages 4.1.1–4.1.8, Cambridge, UK, 1998.
- [Eld99] A. Sharaf Eldin. Computer-assisted arabic transliteration. In *Proceeding of The Seventh International Conference On Artificial Intelligence Applications*, pages 99–105, Cairo, Egypt, 1999. Egyptian Computer Society.
- [EN98] A. Sharaf Eldin and A. Nouh. Arabic character recognition: A survey. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 3386, pages 331–340. SPIE, 1998.
- [ERK90] S. Eldabi, R. Ramsis, and A. Kamel. Arabic character recognition system : A statistical approach for recognizing cursive typewritten text. *Pattern Recognition*, 23(5):485–495, 1990.

- [ES87] M. Elwakil and A. Shoukry. On-line recognition of handwritten isolated arabic characters. In *Proceeding of The Symposium on Computer Arabiazation*, pages 109–120, Riyadh, Saudi Arabia, 1987. King Saud University.
- [ES89] M. Elwakil and A. Shoukry. On-line recognition of isolated arabic characters. *Pattern Recognition*, 22(2):97–105, 1989.
- [ESEA91] A. Eldesouky, M. Salem, A. Abd Elgwad, and H. Arafat. A handwritten arabic character recognition technique for machine reader. In *Proceedings of The Third International Conference on Software Engineering For Real Time Systems*, pages 212–216, Cirencester, UK, 1991. IEE.
- [ETV96] E. Erlandson, J. Trenkle, and R. Vogt. Word-level recognition of multifont arabic text using a feature vector matching approach. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 2660, pages 63–70. SPIE, 1996.
- [FA94] M. Fehri and M. Ben Ahmed. A new approach to arabic character recognition in multi-font document. In *The 4th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1994. Cambridge University Press.
- [FA98] M. Fehri and M. Ben Ahmed. An optical font recognising method for arabic texts. In *Proceeding of The 6th International Conference and Exhibition on Multi-Lingual Computing*, pages 5.15.1–5.15.7, Cambridge, UK, 1998.
- [FB99] E. Fadhel and P. Bhattacharyya. Application of steerable wavelet transform using neural network for signa-

- ture verification. *Pattern Analysis and Applications*, 2(2):184–195, 1999.
- [FCW98] K. Fan, D. Chen, and M. Wen. Skeletonization of binary images with nonuniform width via bloack decomposition and contour vector matching. *Pattern Recognition*, 31(7):823–838, 1998.
- [For73] G. Forney. The viterbi algorithm. *Proceedings Of The IEEE*, 61(3):268–278, 1973.
- [Fre61] H. Freeman. On the encoding of arbitrary geometric configurations. *IRE Transactions On Electronic Computers*, EC-10:260–268, 1961.
- [FS93] M. Fakir and C. Sodeyama. Machine recognition of arabic printed scripts by dynamic programming matching method. *IEICE Transactions On Information and Systems*, 76(2):235–245, 1993.
- [G98] G. Gérard. Building a kurdish language corpus: An overview of the technical problems. In *Proceeding of The 6th International Conference and Exhibition on Multi-Lingual Computing*, pages 5.3.1–5.3.11, Cambridge, UK, 1998.
- [GH92] Z. Guo and R. Hall. Fast fully parallel thinning algorithms. *Computer Vision, Graphics and Image Processing*, 55(3):317–328, 1992.
- [GMC⁺97] B. Gatos, S. Mantzaris, K. Chandrinos, A. Tsigris, and S. Perantonis. Recognition of printed arabic text using neural netwroks. In *Proceeding of The Fourth International Conference on Document Analysis and Recognition, ICDAR'97*, Ulm, Germany, August 1997. IAPR.
- [Gra89] R. M. Gray. Vector quantization. *IEEE ASSP Magazine*, 1:4–29, 1989.

- [GS90] V. Govindan and A. Shivaprasad. Character recognition - a review. *Pattern Recognition*, 23(7):671–683, 1990.
- [GS98a] D. Guillevic and C. Suen. Hmm-knn word recognition engine for bank cheque processing. In *Proceeding of The International Conference on Pattern Recognition, ICPR'98*, pages 1526–1529, Brisbane, Australia, August 1998. IAPR.
- [GS98b] D. Guillevic and C. Suen. Recognition of legal amounts on bank cheques. *Pattern Analysis and Applications*, 1(1):28–41, 1998.
- [GU94] H. Goraine and M. Usher. Printed arabic text recognition. In *The 4th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1994. Cambridge University Press.
- [GUA92] H. Goraine, M. Usher, and S. Alemami. Off-line arabic character recognition. *IEEE Computer Journal*, pages 71–74, July 1992.
- [GW92] R. Gonzalez and R. Woods. *Digital Image Processing*. Addison-Wesley Publishing Company, 1992.
- [Has94] K. Hassibi. Machine-printed arabic ocr using neural networks. In *The 4th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1994. Cambridge University Press.
- [HB94] H. Hosseini and A. Bouzerdoum. A system for arabic character recognition. In *Proceedings of Australian New Zealand International Information Systems Conference ANZIIS'94*, pages 120–124, Braisbane-Australia, 1994. IEEE.
- [HBT96] J. Hu, M. Brown, and W. Turin. Hmm based on-line handwritten recognition. *IEEE Trans. on Pattern*

- Analysis and Machine Intelligence*, 18(10):1039–1045, 1996.
- [HK91] Y. He and A. Kundu. 2-d shape classification using hmm. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 13(11):1172–1184, 1991.
- [Hoo95] J. Hoogland. Arabic ocr. *Reader on The Internet*, <http://leb.net/archives/reader>, 1995.
- [Hoo96a] J. Hoogland. Icemco and the day after. *Reader on The Internet*, <http://leb.net/archives/reader>, 1996.
- [Hoo96b] J. Hoogland. The use of ocr software of arabic in order to create a text corpus of modern standard arabic for lexicographic purposes. In *The 5th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1996. Cambridge University Press.
- [Hu62] M. Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8:179–187, 1962.
- [Jai89] A. Jain. *Fundamentals Of Digital Image Processing*. Prentice Hall, 1989.
- [JC92] B. Jang and R. Chin. One-pass parallel thinning analysis, properties, and quantitative evaluation. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(11):1129–1140, 1992.
- [JHF97] H. Jiang, C. Han, and K. Fan. A fast approach to the detection and correction of skew documents. *Pattern Recognition Letters*, 18(7):675–686, 1997.
- [JR85] B. Juang and L. Rabiner. Mixture autoregressive hidden markov models for speech signals. *IEEE Trans. Acoustics, Speech and Signal Processing*, ASSP-33(6):1404–1413, December 1985.

- [KA94] S. Kuo and O. Agazzi. Keyword spotting in poorly printed documents using pseudo 2-d hmms. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 16(8):842–848, 1994.
- [KA99] M. Kavianifar and A. Amin. Preprocessing and structural feature extraction for a multi-fonts arabic/persian ocr. In *Proceeding of The Fifth International Conference on Document Analysis and Recognition, IC-DAR'99*, Bangalore, India, September 1999. IAPR.
- [KC97] A. Kundu and G. Chen. An integrated hybrid neural network and hmm classifier for sonar signal. *IEEE Transactions on Signal Processing*, 45(10):2566–2570, 1997.
- [KD94] E. Kraus and E. Dougherty. Segmentation-free morphological character recognition. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 2181, pages 14–23. SPIE, 1994.
- [KG96] G. Kim and V. Govindaraju. Efficient chain code based image manipulation for handwritten word recognition. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 2660, pages 262–272. SPIE, 1996.
- [KG97] G. Kim and V. Govindaraju. A lexicon driven approach to handwritten word recognition for real-time applications. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 19(4):367–379, 1997.
- [KHB89] A. Kundu, Y. He, and P. Bahl. Recognition of handwritten word : First and second order hmm based approach. *Pattern Recognition*, 22(3):283–297, 1989.
- [KKKL97a] H. Kim, K. Kim, S. Kim, and J. Lee. On-line recognition of handwritten chinese characters based on hmms. *Pattern Recognition*, 30(9):1489–1500, 1997.

- [KKKL97b] H. Kim, S. Kim, K. Kim, and J. Lee. An hmm-based character recognition network using level building. *Pattern Recognition*, 30(3):491–502, 1997.
- [Kon94] N. Kondybaev. Maximum entropy approach in automatic classification of symbolic images. In *The 4th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1994. Cambridge University Press.
- [KP96] W. Kim and R. Park. Off-line recognition of handwritten korean and alphanumeric characters using hmms. *Pattern Recognition*, 29(5):845–858, 1996.
- [Lag93] K. Lagally. Arabtex a system for typesetting arabic. Technical report, Universitat Stuttgart, Germany, 1993.
- [Lar98] J. Larmagnac. Thinning and line segmentation by line following technique. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 3305, pages 210–219. SPIE, 1998.
- [LH90] C. Liao and J. Huang. Stroke segmentation by bernsten-bezier curve fitting. *Pattern Recognition*, 23(5):475–484, 1990.
- [LLS92] L. Lam, S. Lee, and C. Suen. Thinning methodologies - a comprehensive survey. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 14(9):869–885, 1992.
- [LRS83] S. Levinson, L. Rabiner, and M. Sondhi. An introduction to the application of the theory of probabilistic function of markov process to automatic speech recognition. *The Bell System Technical Journal*, pages 1035–1074, April 1983.

- [LS70] G. Lendaris and G. Stanley. Diffraction-pattern sampling for automatic pattern recognition. *Proceedings Of The IEEE*, 58(2):198–216, 1970.
- [MAG91] S. Mahmoud, I. Abuhaiba, and R. Green. Skeletonization of arabic characters using clustering based skeletonization algorithm. *Pattern Recognition*, 24(5):453–464, 1991.
- [Mah94] S. Mahmoud. Arabic characters recognition using fourier descriptors and character contour encoding. *Pattern Recognition*, 27(6):815–824, 1994.
- [MAS99] D. Motawa, A. Amin, and R. Sabourin. Segmentation of arabic cursive script. In *Proceeding of The Fifth International Conference on Document Analysis and Recognition, ICDAR'99*, pages 625–628, Bangalore, India, September 1999. IAPR.
- [MEF87] M. Mahmoud, M. Elhamalaway, and A. Fahmi. A statistical approach for arabic character recognition. In *Proceeding of The 12th International Conference For Statistics and Computer Science*, pages 243–250, Cairo, Egypt, 1987.
- [MFV98] G. Malandain and S. Fernandez-Vidal. Euclidean skeletons. *Image and Vision Computing*, 16(2):317–327, 1998.
- [MG96] M. Mohamed and P Gader. Handwritten word recognition using segmentation-free hmm and segmentation-based dynamic programming techniques. *IEEE Trans. on Pattern Analysis and Machine Intelligence*, 18(5):548–554, 1996.
- [MHY98] M. Manaf, A. Hamdan, and M. Yusof. Development of a system to recognise handwritten jawi scripts and

- conversion of jawi scripts to rumi scripts. In *Proceeding of The 6th International Conference and Exhibition on Multi-Lingual Computing*, pages 5.6.1–5.6.10, Cambridge, UK, 1998.
- [MLR⁺96] J. Makhoul, C. LáPré, C. Raphael, R. Schwartz, and Y. Zahao. Towards language-independent character recognition using speech recognition methods. In *The 5th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge, UK, 1996. Cambridge University Press.
- [Mos96] R. Mosfeq. Arabic character recognition using integrated segmentation and recognition. In *The 5th International Conference and Exhibition on Multi-Lingual Computing*, Cambridge-UK, 1996. Cambridge University Press.
- [MS97] J. Makhoul and R. Schwartz. What is a hidden markov model? *IEEE Spectrum*, pages 46–46, December 1997.
- [MSLB98] J. Makhoul, R. Schwartz, C. Lapre, and I. Bazzi. A script-independent methodology for optical character recognition. *Pattern Recognition*, 31(9):1285–1294, 1998.
- [MSY92] S. Mori, C. Suen, and K. Yamamoto. Historical review of ocr research and development. *Proceeding Of The IEEE*, 80(7):1029–1058, 1992.
- [Nag92] G. Nagy. At the frontiers of ocr. *Proceeding Of The IEEE*, 80(7):1093–1100, 1992.
- [NBL⁺99] P. Natarajan, I. Bazzi, Z. Lu, J. Makhoul, and R. Schwartz. Robust ocr of degraded documents. In *Proceeding of The Fifth International Conference on Document Analysis and Recognition, ICDAR'99*, Bangalore, India, September 1999. IAPR.

- [NS93] M. Nadler and E. Smith. *Pattern Recognition Engineering*. John Wiley & Sons Inc., 1993.
- [NST84] A. Nouh, A. Sultan, and R. Tolba. On feature extraction and selection for arabic character recognition. *Arab Gulg Journal For Scientific Research*, 2(1):329–346, 1984.
- [NUE88] A. Nouh, A. Ula, and S. Eldin. Algorithms for feature extraction: A case study for the arabic character recognition. In *The 10th National Computer Conference*, pages 653–666. King Abdul Aziz University, 1988.
- [NUN88a] A. Nurul-Ula and A. Nouh. Automatic recognition of arabic character using logic statements. *Journal of Engineering Sciences, King Saud University*, 14(2):343–352, 1988.
- [NUN88b] A. Nurul-Ula and A. Nouh. Automatic recognition of arabic character using logic statements. *Journal of Engineering Sciences, King Saud University*, 14(2):353–362, 1988.
- [OAB97] L. Owsley, L. Atlas, and G. Bernard. Self-organising feature maps and hmms for machine-tool monitoring. *IEEE Transactions on Signal Processing*, 45(11):2787–2798, 1997.
- [OHK95] S. Oh, J. Ha, and J. Kim. Context dependent search in interconnected hmm for unconstrained handwritten recognition. *Pattern Recognition*, 28(11):1693–1704, 1995.
- [Pao89] Y. Pao. *Adaptive Pattern Recognition and Neural Networks*. Addison Wesley Publishing Company, Inc., 1989.
- [Par97] J. R. Parker. *Algorithms For Image Processing and Computer Vision*. John Wiley and Sons, Inc., 1997.

- [Pav82] T. Pavlidis. *Algorithms For Graphics And Image Processing*. Computer Science Press, 1982.
- [PL98] H. Park and S. Lee. A truly 2-d hidden markov model for off-line handwritten character recognition. *Pattern Recognition*, 31(12):1849–1864, 1998.
- [PT81] B. Parhami and M. Taraghi. Automatic recognition of printed farsi texts. *Pattern Recognition*, 14(6):395–403, 1981.
- [Rab89] L. Rabiner. A tutorial on hmm and selected applications in speech recognition. *Proceedings of The IEEE*, 77(2):257–286, February 1989.
- [Rau94] T. Rauber. Two-dimentional shape description. Technical report, Universidade Nova de Lisboa, Portugal, 1994.
- [REK88] R. Ramsis, S. Eldabi, and A. Kamel. Arabic character recognition system. Technical report, IBM Kuwait Scientific Cenetr, 1988.
- [RJ86] L. Rabiner and B. Juang. An introduction to hidden markov models. *IEEE ASSP Magazine*, pages 4–16, January 1986.
- [RJ93] L. Rabiner and B. Juang. *Fundamentals Of Speech Recognition*. Prentice Hall, 1993.
- [RJLS85] L. Rabiner, B. Juang, S. Levinson, and M. Sondhi. Recognition of isolated digits using hmms with continuous mixture densities. *AT&T Technical Journal*, pages 1211–1233, July-August 1985.
- [RLS83] L. Rabiner, S. Levinson, and M. Sondhi. On the application of vector quantization and hmm to speaker-independent, isolated word recognition. *The Bell System Technical Journal*, pages 1075–1105, April 1983.

- [RSG92] T. Rauber and A. Steiger-Garcia. 2-d form descriptors based on a normalised parametric polar transform. In *Proceeding of The IAPR Workshop on Machine Vision Applications MVA '92*, Tokyo, Japan, 1992.
- [RW96] G. Ritter and J. Wilson. *Handbook Of Computer Vision Algorithms In Image Algebra*. CRC Press, 1996.
- [San96] H. Sanossian. An arabic character recognition system using neural network. In *Proceedings of 1996 IEEE Signal Processing Society Workshop*, pages 340–348, Kyoto-Japan, 1996. IEEE.
- [SBB⁺92] J. Schurmann, N. Bartneck, T. Bayer, J. Franke, E. Mandler, and M. Oberlander. Document analysis - from pixels to contents. *Proceeding Of The IEEE*, 80(7):1101–1119, 1992.
- [Sch77] E. Schwartz. Spatial mapping in the primate sensory projection: Analytic structure and relevance to perception. *Biological Cybernetics*, 25:181–194, 1977.
- [Sho89] A. Shoukry. Arabic character recognition state of the art. In *Proceedings of The 11th National Computer Conference*, pages 382–390, Dhahran, Saudi Arabia, 1989. King Fahad University Of Petroleum and Minerals.
- [Sim92] J. Simon. Off-line cursive word recognition. *Proceeding Of The IEEE*, 80(7):1150–1161, 1992.
- [SL95] S. Srihari and S. Lam. Character recognition. Technical report, Center of Excellence For Document Analysis and Recognition, January 1995.
- [SLG⁺92] S. Srihari, S. Lam, V. Govindaraju, R. Srihari, and J. Hull. Document understanding: Research direction. Technical report, Center of Excellence For Document Analysis and Recognition, May 1992.

- [Sri86] S. Srihari. Document image understanding. In *Proceedings Of ACM/IEEE Fall Joint Computer Conference*, pages 87–96, Dallas - USA, 1986.
- [SY85] S. Saadallah and S. Yagu. Design of an arabic character reading machine. In *Workshop On Computer Processing and Transmission Of The Arabic Language*, Kuwait, 1985. Kuwait Institute for Scientific Research.
- [SYS99] F. Said, R. Yacoub, and C. Suen. Recognition of english and arabic numerals using a dynamic number of hidden neurons. In *Proceeding of The Fifth International Conference on Document Analysis and Recognition, IC-DAR'99*, Bangalore, India, September 1999. IAPR.
- [TS90] M. Tolba and E. Shaddad. On the automatic reading of printed arabic characters. In *Proceedings of IEEE International Conference on Systems, Man and Cybernetics*, pages 496–498, Los Angeles, CA, 1990. IEEE.
- [Vas98] R. Vassie. Marrying the arabic and latin scripts conceptually. In *Proceeding of The 6th International Conference and Exhibition on Multi-Lingual Computing*, pages 5.9.1–5.9.14, Cambridge, UK, 1998.
- [VK92] J. Vlontzos and S. Kung. Hidden markov models for character recognition. *IEEE Transactions On Image Processing*, 1(4):539–543, 1992.
- [WMO92] T. Wakahara, H. Murase, and K. Odaka. On-line handwritten recognition. *Proceeding Of The IEEE*, 80(7):1181–1194, 1992.
- [YT90] S. Yu and W. Tsai. A new thinning algorithm for grayscale images by the relaxation technique. *Pattern Recognition*, 23(10):1067–1076, 1990.
- [YVA96] F. Yarman-Vural and A. Atici. A heuristic algorithm for optical character recognition of arabic script. In

- Proceeding of The Visual Communications and Image Processing*, volume 2727, pages 725–736. SPIE, 1996.
- [ZEEE86] F. Zaki, S. Elkonyaly, A. Abd Elfattah, and Y. Enab. A new technique for arabic handwriting recognition. In *Proceeding of The 11th International Conference For Statistics and Computer Science*, pages 171–180, Cairo, Egypt, 1986.
- [Zem96] P. Zemanek. Arabic ocr - omnipage. *Reader on The Internet*, <http://leb.net/archives/reader>, 1996.
- [ZR72] C. Zahn and R. Roskies. Fourier descriptors for plane closed curves. *IEEE Transaction On Computers*, 21(3):269–282, 1972.
- [ZS84] T. Zhang and C. Suen. A fast parallel algorithm for thinning digital patterns. *Communications*, 27:235–239, 1984.
- [ZS94] X. Zhao and S. Srihari. Word recognition using ideal word patterns. In *Proceeding of The International Society for Optical Engineers, SPIE*, volume 2181, pages 24–34. SPIE, 1994.
- [ZTF91] A. Zahour, B. Taconet, and A. Faure. A new method for recognition of arabic cursive scripts. In *The 1st International Conference on Document Analysis and Pattern Recognition*, pages 454–462, 1991.