

**ONLINE ARABIC TEXT RECOGNITION USING
STATISTICAL TECHNIQUES**

BY

BALIGH MOHAMMED AL-HELALI

A Thesis Presented to the
DEANSHIP OF GRADUATE STUDIES

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS

DHAHRAN, SAUDI ARABIA

In Partial Fulfillment of the
Requirements for the Degree of

MASTER OF SCIENCE

In

COMPUTER SCIENCE

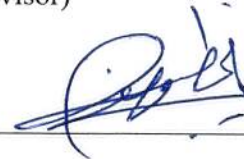
MAY 2016

KING FAHD UNIVERSITY OF PETROLEUM & MINERALS
DHAHRAN- 31261, SAUDI ARABIA
DEANSHIP OF GRADUATE STUDIES

This thesis, written by **BALIGH MOHAMMED AL-HELALI** under the direction of his thesis advisor and approved by his thesis committee, has been presented and accepted by the Dean of Graduate Studies, in partial fulfillment of the requirements for the degree of **MASTER OF SCIENCE IN COMPUTER SCIENCE**.



Dr. Sabri Mahmoud
(Advisor)



Dr. Wasfi Al-Khatib
(Member)



Dr. Abdulaziz Alkhoraidly
Department Chairman



Dr. Salam A. Zummo
Dean of Graduate Studies





Dr. Radwan Abdel-Aal
(Member)



Date

© Baligh Al-Helali

2016

Dedication

To my son Mohammed and my daughter Buthaina

ACKNOWLEDGMENTS

First I would like to thank almighty Allah for giving me the ability and the strength to work on and complete this thesis.

I would like to give my sincere appreciation to my advisor Prof. Sabri Mahmoud for all the hard work he has put in this work, and for the guidance and encouragement he provided throughout this research. Prof. Sabri has been a mentor, teacher and a father. I feel very lucky to have had Prof. Sabri as my thesis supervisor. I would also like to thank Prof. Sabri for helping me in my future endeavors. I would also like to thank the committee members, Prof. Radwan Abdel-aal and Dr. Wasfi Al-khatib, for dedicating time out of their busy schedule for this work and providing their feedback.

I am grateful to KFUPM for providing support for all these years. I would also like to sincerely thank and appreciate my home institution, Ibb university-Yemen, which donors me a scholarship to continue my graduate studies abroad.

I also acknowledge the support provided by King Abdul-Aziz City for Science and Technology (KACST) through the Science & Technology Unit at King Fahd University of Petroleum & Minerals (KFUPM) for supporting this work through project no. 11-INF2153-04 as part of the National Science, Technology and Innovation Plan.

I would like to express thanks to my parents, without them I would never be here, and they sacrificed so much to give me a better life. I'm thankful for my wife, for her patience and love. I appreciate my brothers and sisters for supporting me in pursuing my interests.

I would also like to thank all of my previous teachers, as their valuable work has laid the foundation for this work. I would like to thank all my friends and colleagues for their encouragement and moral support.

Last, but not least, there are many others who have been supportive of my efforts, I extend my deepest thanks to them all.

TABLE OF CONTENTS

ACKNOWLEDGMENTS	V
TABLE OF CONTENTS	VII
LIST OF TABLES.....	X
LIST OF FIGURES	XII
LIST OF ABBREVIATIONS	XIV
ABSTRACT	XV
ملخص الرسالة.....	XVII
CHAPTER 1 INTRODUCTION	1
1.1 Characteristics and Challenges of Arabic Online Text.....	2
1.2 Arabic Online Text Recognition Model.....	13
1.3 Motivation.....	17
1.4 Thesis Objectives and Outcomes	17
1.5 Thesis Organization.....	19
CHAPTER 2 LITERATURE REVIEW	20
2.1 Arabic Online Datasets.....	20
2.2 Preprocessing Approaches.....	25
2.3 Segmentation Approaches.....	36
2.4 Feature Extraction Approaches	44
2.5 Classification Approaches	51

2.5.1	Non-Cursive Text Recognition.....	52
2.5.2	Cursive Text Recognition.....	60
2.6	Post-Processing	74
 CHAPTER 3 A STATISTICAL FRAMEWORK FOR ONLINE ARABIC CHARACTER RECOGNITION		76
3.1	The Proposed Framework.....	77
3.2	Delayed Strokes Handling	79
3.3	Pre-Processing	82
3.4	Feature Extraction.....	86
3.5	Classification.....	89
3.6	Experimental Results	91
 CHAPTER 4 ARABIC ONLINE HANDWRITTEN TEXT RECOGNITION		102
4.1	Cursive Input Pre-Processing	102
4.2	Classification.....	106
4.2.1	HMM	106
4.2.2	Dynamic Bayesian network (DBN)	108
4.2.3	Segmentation-based Cursive Text Recognition.....	110
4.2.4	DBN-based Hierarchical HMMs (HHMMs).....	112
4.2.5	Segmentation-free Cursive Text Recognition Using HMM-HTK.....	114
4.3	Experimental Results	115
 CHAPTER 5 CONCLUSIONS AND FUTURE RESEARCH DIRECTIONS		123
5.1	Conclusions.....	123
5.2	Future Research Directions.....	125
 APPENDICES		127

Appendix A. Database details, transliteration of Arabic letters	127
Appendix B. Statistical Framework GUI Guide.....	130
Appendix C. Statistical Feature Extraction.....	137
Appendix D. Character Recognition Detailed Results.....	140
Appendix E. Cursive Text Recognition Detailed Results	154
REFERENCES	157
VITAE	169

LIST OF TABLES

Table 1.1 Arabic alphabet	3
Table 2.1 Summary of some online Arabic text databases	25
Table 2.2 Summary of pre-processing in some AOTR studies	35
Table 2.3 Summary of segmentation methods for online Arabic text.	43
Table 2.4 A summary of the features used in some AOTR studies.	49
Table 2.5 Summary of non-cursive AOTR	59
Table 2.6 Summary of cursive AOTR	71
Table 3.1 Statistics of Online-KHATT segmented characters.	92
Table 3.2. Some results of basic shapes classification using the validation set.	93
Table 3.3 Confusion Matrix of the tested data.	101
Table 4.1 Results of using some statistical features for cursive text recognition on Online-KHATT database.....	115
Table 4.2 Results of using PCA statistical features for cursive text recognition on Online-KHATT database.....	116
Table 4.3. Some recognition results on Online-KHATT text lines using HTK with different statistical features.	119
Table 6.1 Statistics of Online-KHATT database	127
Table 6.2: Transliteration codes for Online-KHATT database [125].....	129
Table 6.3 Mappings Relate Characters, Models, and Secondary Objects.	132
Table 6.4 Mappings Relate Delayed Strokes Models and Secondary Objects.	132
Table 6.5 List of extracted statistical features.	137
Table 6.6. Some results on annotated Basic Shapes classifier: All_100_30_100_30.	141
Table 6.7 Basic Shapes Confusion Matrix: All_100_30_100_30.....	141
Table 6.8 Basic Shapes Confusion Matrix: All_70_15_70_15.....	143
Table 6.9 Basic Shapes Confusion Matrix: All_100_30_inf_inf.	144
Table 6.10 Basic Shapes Confusion Matrix: All_70_15_inf_inf.	146
Table 6.11 Basic Shapes Confusion Matrix: All_40_10_inf_inf.	147
Table 6.12 Results on annotated Positional-Based Basic Shapes classifiers.	149
Table 6.13 Best results of positional classifiers on annotated balanced samples.	150
Table 6.14 Confusion matrix of I-position characters classification.	150
Table 6.15 Confusion matrix of B-position characters classification.	151
Table 6.16 Confusion matrix of M-position characters classification.	151
Table 6.17 Confusion matrix of E-position characters classification.	152
Table 6.18 Results on delayed strokes classification.	153
Table 6.19 Results of using some statistical features for cursive text recognition on ADAB database.	154

Table 6.20 Results of using PCA statistical features for cursive text recognition on ADAB database.	155
Table 6.21 Results of using PCA statistical features for cursive text recognition on Online-KHATT Lines.	155

LIST OF FIGURES

Figure 1.1 Ligature examples.	4
Figure 1.2 Difficulties in addressing the dots.	6
Figure 1.3 Main and secondary (delayed) strokes.	6
Figure 1.4 SEEN letter handwritten in different non-uniform writing styles.	6
Figure 1.5 Online internal discontinuity with (a) visual discontinuity, and (b) visual continuity (in two strokes)	9
Figure 1.6 Discontinuity in a PAW written in Naskh style.	9
Figure 1.7 Connectivity difficulties.	9
Figure 1.8 Handwriting variability.	10
Figure 1.9 Variability in the number of strokes.	12
Figure 1.10 The same word written with a different stroke order.	12
Figure 1.11 Handwriting movement variations.	12
Figure 1.12 Using the beginning form of the letter “Haa” هـ in the isolated and end positions.	12
Figure 1.13 Text recognition capabilities.	15
Figure 1.14 The general model of online text recognition, adapted from [2]	16
Figure 2.1 Example of detected baseline correction (green) [32]	30
Figure 2.2 Detection of the curvilinear velocity extremum points [100]	39
Figure 2.3 Topological characteristics used in the detection of specific points (above) and grapheme segmentation (below) [33].	39
Figure 2.4 An ambiguity that is caused when delayed strokes are removed	74
Figure 3.1 The proposed framework for character recognition.	78
Figure 3.2 Examples internal discontinuity.	80
Figure 3.3 Examples of delayed strokes larger than main strokes.	83
Figure 3.4 Internal discontinuity handling.	85
Figure 3.5 Freeman Chain Code.	88
Figure 3.6 Statistical Directional feature extraction.	88
Figure 3.7 Characters with similar basic shapes but different delayed strokes.	96
Figure 3.8 Examples of segmentation-based intra-errors.	97
Figure 3.9 Examples of segmentation-based inter-errors.	98
Figure 3.10 Examples of errors originated from writing distortion.	98
Figure 3.11 Example of confusions caused from the segmentation.	100
Figure 4.1 Preprocessing operations outputs.	104
Figure 4.2 Histogram-based baseline detection on a skewed text.	105
Figure 4.3 EM-based baseline detection on a skewed text.	105
Figure 4.4 Initial set of secondary strokes, correct (circle), incorrect (rectangle).	105
Figure 4.5 Simple HMM Bakis model with five states [128].	107
Figure 4.6 A DBN representation of HMM [126].	109

Figure 4.7 Segmentation-based Cursive Text Recognition Approach (adopted from [130]).....	111
Figure 4.8 A DBN for modelling a PAW.	113
Figure 4.9 DBN-HHMM modeling the PAW "لما" (LMA).	113
Figure 4.10 Architecture of a cursive online text HMM-based Recogniser.....	114
Figure 4.11 Samples of Online-KHATT PAWs.	118
Figure 4.12 Samples of the Online-KHATT lines.	122
Figure 6.1 Samples of text from Online-KHATT.....	128
Figure 6.2 Preparing GUI.	131
Figure 6.3 A gui for Statistical Character Recognizer.....	136
Figure 6.4 Samples of ADAB PAWs.	156

LIST OF ABBREVIATIONS

OCR	:	Optical Character Recognition
AOTR	:	Arabic Online Text Recognition
PAW	:	Part of Arabic Word
HMM	:	Hidden Markov Model
DBN	:	Dynamic Bayesian Network
ANN	:	Artificial Neural Network
KNN	:	K-Nearest Neighbor
TM	:	Template Matching
DT	:	Decision Tree
GMM	:	Gaussian Mixture Model
CRR	:	Character Recognition Rate
WRR	:	Word Recognition Rate
WI	:	Writer Independent
WD	:	Writer Dependent

ABSTRACT

Full Name : [Baligh Mohammed Ahmed Al-Helali]
Thesis Title : [Online Arabic Text Recognition Using Statistical Techniques]
Major Field : [Computer Sciences]
Date of Degree : [April 2016]

The widespread use of pen-based hand-held devices, such as PDAs, smart phones, and tablets, has increased the demand for online text recognition systems. This technology has great potential in markets that involve friendly learning environments, business applications, education and more.

The purpose of this thesis is to conduct research on Arabic online text recognition. This implies addressing the different phases of text recognition systems. In particular, the main focus is on using statistical features and techniques. We investigate the applicability of statistical-based techniques to Arabic online text recognition.

In this thesis, we present a comprehensive survey of the related work. We then develop recognition prototypes for both non-cursive and cursive online Arabic text recognition. We present several novel techniques for the different phases of online Arabic text recognition using statistical approach. One of the contributions of this research is the methodology of handling the delayed strokes. The delayed strokes are handled at the different phases of the recognition process differently to improve the overall performance. Another contribution is the intensive investigation of several novel statistical features using a developed framework for generating different statistical features. The framework consists of two main components. The first one is to extract the

point-based features (local features). A statistical layer is then added to form statistical features. Moreover, the used dataset is extracted from a database of unconstrained online cursive text. Using such dataset implies the need of addressing additional difficulties such as connectivity, variability, and delayed strokes challenges.

The results of the proposed statistical techniques are presented and analyzed. These techniques are applied to the recognition of Arabic online segmented characters, parts of words and text. In addition, the presented techniques may be utilized in many areas of scientific research such as writer identification/verification, forensic handwriting analysis, and signature verification systems. Finally, the thesis ends with summarizing the conclusions of our work and the future directions.

ملخص الرسالة

الاسم الكامل: بليغ محمد احمد الهلالي

عنوان الرسالة: التعرف على الكتابة الآتية العربية باستخدام تقنيات احصائية

التخصص: علوم الحاسب الآلي

تاريخ الدرجة العلمية: نيسان 2016

زاد الانتشار الواسع لاجهزة اللمس والادوات التي تدعم الكتابة الالكترونية بالقلم او باليد من الطلب على انظمة التعرف على الكتابة الآتية. ان لهذه التقنية تطبيقات مهمة من ضمنها بيئات التعليم الذكي و التطبيقات التجارية و غيرها.

ان الغرض من هذه الرسالة اجراء بحث علمي للتعرف على الكتابة الآتية العربية. يتضمن هذا البحث المراحل المختلفة من نظم التعرف على النصوص. على وجه الخصوص, مزيد من التركيز سيكون على التقنيات الاحصائية و قابليتها للتطبيق على التعرف على الكتابة الآتية العربية.

قمنا في هذه الرسالة بمسح شامل للدراسات السابقة. كما قمنا أيضاً بتطوير العديد من الأساليب لاستخدامها في المراحل المختلفة للتعرف على الكتابة الآتية العربية. احدى اسهامات هذه الرسالة هو الطريقة المطورة للتعامل مع النقاط و الهمزات و ما في حكمها والتي تسمى "الكتابة المتأخرة" وذلك باخذها بالاعتبار في مختلف المراحل من اجل تحسين الاداء الاجمالي. وقد قمنا بالاختبار المكثف للعديد من المميزات الاحصائية الجديدة والمكون من مرحلتين, الاولى لاستخراج المميزات المحلية والثانية لحساب العديد من الاحصائيات عليها. كذلك فإن البيانات المستخدمة هي من قاعدة بيانات كتابة عربية آتية بدون قيود مما يعني الحاجة للتعامل مع صعوبات اضافية مثل الترابط و التغيرات و صعوبات النقاط و علامات الترقيم.

هذا وقد تم عرض نتائج التقنيات والطرق المقترحة وتحليلها بعد تطبيقها على مستوى الحروف المقسمة و اجزاء الكلمات والنصوص المكتوبة. اضافة الى ذلك يمكن استخدام الطرق المقدمة في العديد من المجالات مثل التعرف والتحقق من الكاتب والتحليل والتقصي الجنائي للكتابة اليدوية وانظمة التحقق من التوقيع. اخيراً تختتم هذه الرسالة بتلخيص الاستنتاجات من عملنا هذا وكذلك اتجاهات البحث المستقبلية.

CHAPTER 1

INTRODUCTION

Arabic Online Text Recognition (AOTR) is an active research area that has potential markets in friendly learning environments, business applications, education and more. These applications are facilitated by the widespread use of pen-based hand-held devices, such as PDAs, smart phones, and tablet-PC's.

In the mid-1970s, digitizer tablets became available in which analog-to-digital conversion techniques were employed [1]. With these tablets, it was possible to track the pen tip. A number of technologies became available for writing pads or tablets that are based on electronic, electromagnetic, electrostatic or pressure sensitive devices. Handwriting in such devices is called online text, and the corresponding technologies facilitate the capture of the dynamic or temporal information of the handwriting. The main component of online writing is the stroke, which is the writing trajectory from pen down to pen up. Online devices represent each stroke as a one-dimensional, ordered vector of (x, y) coordinates of points.

In this chapter, we present the characteristics and challenges of Arabic online text, a general model for online text recognition systems highlighting their main phases, and the motivation, objectives and outcomes of our work.

1.1 Characteristics and Challenges of Arabic Online Text

In this section, we present the characteristics and challenges of AOTR and more details can be found in [2], [3].

Arabic characters, in addition to being used in Arabic, are used in Kurdish, Persian, Pashto and Urdu. The Arabic alphabet has 28 or 29 letters (the basic 28 plus the Hamza-on-the-line letter constructed from the Hamza letter form). In these counts, the Hamza letter marks are considered to be diacritical. Some researchers add other characters to form 40 Arabic letters (viz. the basic 28, Alif-Maqsurah, Ta-Marbuta, four Lam-Alif ligatures, and six Hamza letters) [4].

Arabic script is cursive in nature, as most Arabic letters are connected to their neighboring letters. Arabic letters can be written using two to four shapes depending on their position, as shown in Table 1.1. Note that characters with two forms have their initial and middle forms similar to the other forms, and their corresponding cell in the table is thus left blank, e.g., Alif “ا”, Za “ز”, and Waw “و”. A letter is said to be isolated if it is not connected from the left or from the right. Its position is middle if it is connected from both sides. It is in the beginning (end) form when it is connected only from the left (right). The set of connected letters is a sub-word, which is called a Part of Arabic Word (PAW). Arabic script is written from right to left. Although there is some overlap with Arabic off-line text recognition, it is harder to perform online text recognition because the writing in online devices is less controlled than a pen on paper. A PAW can be an isolated letter or a written script that begins with a beginning-shaped letter and ends with an end-shaped letter.

At certain positions of a word, some overlapping (horizontally or vertically) Arabic letters are represented by ligatures. In Arabic typing, the ligatures depend on the font. The ligatures are more challenging in Arabic handwriting because the writer might not be consistent in writing the ligatures (i.e., the same set of letters can be represented as a ligature or as a string of non-overlapping characters by the same writer at different parts of the handwritten text). Examples of some of the ligatures are shown in Figure 1.1.

Table 1.1 Arabic alphabet.

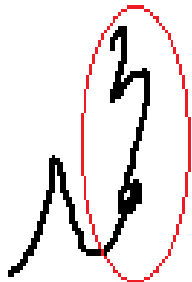
Character	Isolated	Initial	Middle	Final	Character	Isolated	Initial	Middle	Final
Alif	ا			ا	Dhad	ض	ضد	ضد	ض
Ba	ب	ب	ب	ب	Taa	ط	ط	ط	ط
Ta	ت	ت	ت	ت	Dha	ظ	ظ	ظ	ظ
Tha	ث	ث	ث	ث	Ayn	ع	ع	ع	ع
Jeem	ج	ج	ج	ج	Ghain	غ	غ	غ	غ
Ha	ح	ح	ح	ح	Fa	ف	ف	ف	ف
Kha	خ	خ	خ	خ	Qaf	ق	ق	ق	ق
Dal	د			د	Kaf	ك	ك	ك	ك
The	ذ			ذ	Lam	ل	ل	ل	ل
Ra	ر			ر	Meem	م	م	م	م
Za	ز			ز	Noon	ن	ن	ن	ن
Seen	س	س	س	س	He	ه	ه	ه	ه
Sheen	ش	ش	ش	ش	Waw	و			و
Sad	ص	ص	ص	ص	Ya	ي	ي	ي	ي



a)



b)



c)

Figure 1.1 Ligature examples.

Arabic text contains special symbols, such as dots and diacritics that are used as secondary units to complete the meaning of the main text. Some Arabic letters have dots above or below their basic shape. Diacritics are used in Arabic to resolve linguistic ambiguities in the text. Arabic letters that have similar basic shapes are distinguished from one another by the number of dots and their positions. It is common in Arabic handwriting to ignore the diacritics. In some handwriting, the dots are not carefully written, and the words are recognized from the context. Two dots can be written as a small horizontal stroke, and three dots can be written as a hat shape, as shown in Figure 1.2. Variations in the handwriting of these units can result from the stroke size, number, order, location, shape, or writing direction, as shown in Figure 1.2. The diacritics are written above or below the main text. Dots and diacritics can be a source of confusions, particularly with unconstrained natural Arabic handwriting. These objects are typically called secondary components and are normally handled as delayed strokes, as shown in Figure 1.3. Such strokes are called “delayed” as they are normally written after completing the strokes that represent the main body of the online text.

Arabic script can include digits and punctuation marks, which can lead to some confusion (e.g., a zero (٠) vs. a full stop mark (.), the digit one (1) vs. the letter Alif (ا)). In addition, Arabic numbers are written from left to right, whereas the characters are written from right to left.

Arabic has different writing styles. In general, the main styles are Naskh, Ruqyah and several others, notably Thuluth, Kofi, and Diwani, which are used for decorative calligraphy. In general, writers do not follow handwritten style rules, which increases recognition difficulty, as shown in Figure 1.4.

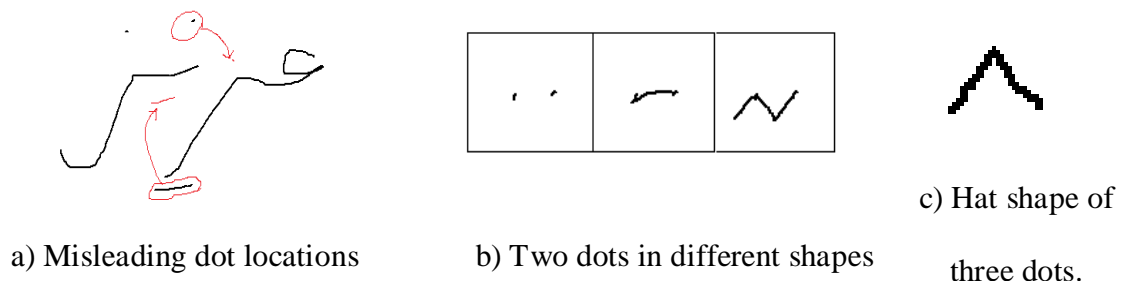


Figure 1.2 Difficulties in addressing the dots.

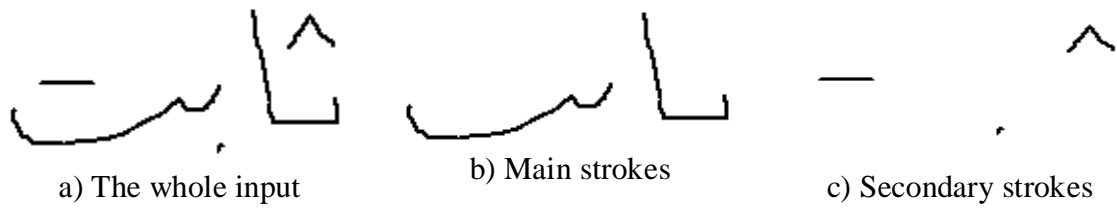


Figure 1.3 Main and secondary (delayed) strokes.



Figure 1.4 SEEN letter handwritten in different non-uniform writing styles

The connectivity of the handwritten text can lead to several difficulties, such as the presence of internal discontinuity and external connectivity. *Internal discontinuity* is a cut in the writing trajectory of a handwritten input that is expected to be represented by a connected component. An *online internal discontinuity* is different from an offline handwriting internal discontinuity in terms of defining the writing component. An offline writing component is spatially connected, whereas a stroke connects online components. An *offline discontinuity* is caused by spatial gaps in the input trajectory. This type of discontinuity may exist in a connected online writing component (a single stroke). An online internal discontinuity occurs when more than one stroke is used to write a handwritten input that is expected to be represented by a single stroke. Some writers might move their hands up while writing one component. An online internal discontinuity can result in a visual discontinuity, as shown in Figure 1.5.a (where the letter consists of two strokes instead of one), whereas it is visually connected in Figure 1.5.b.

Internal discontinuity is more problematic in systems that assume that PAWs will be represented by connected components. For example, the middle forms of some letters (e.g., "ح") are difficult to draw without lifting the pen when using the Naskh style, as shown in Figure 1.6.

External connectivity can be a source of several difficulties. The separation between consecutive units might not be clear. These units can be strokes in the online case or letters, PAWs, or words in the offline text. *External connectivity* is apparent when consecutive units are touching, as shown in Figure 1.7.a. Online connectivity depends on the recording process of the number of strokes, input order and writing time. Online touching occurs when consecutive units are handwritten in one stroke when they should be represented by separate strokes, as shown in Figure 1.7.b, in which the letter ALEF is connected to the next letter when writing "لا". Another difficulty occurs when the trajectories of different units overlap with regard to the x-axis, as shown in Figure 1.7.c.

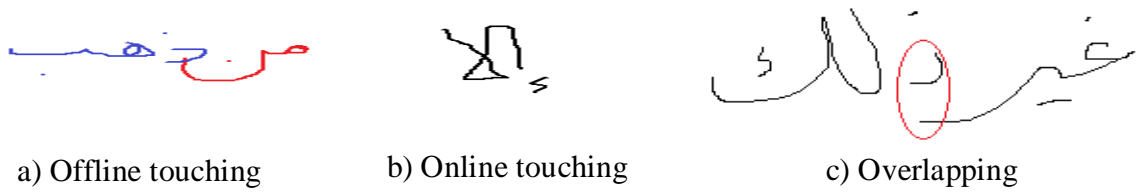
The acquired input might contain some variations that originate with the handwriting process. Most of the variations are spatial in nature, such as different character shapes and alignments, handwriting drifts, skews, slants, hooks, and curvatures. Figure 1.8 shows examples of geometric and visual variations.



Figure 1.5 Online internal discontinuity with (a) visual discontinuity, and (b) visual continuity (in two strokes)



Figure 1.6 Discontinuity in a PAW written in Naskh style.

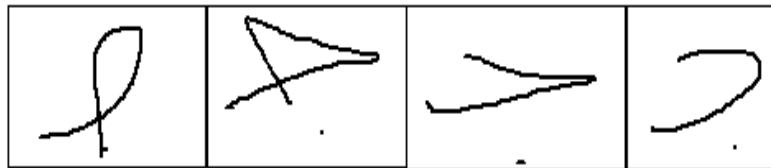


a) Offline touching

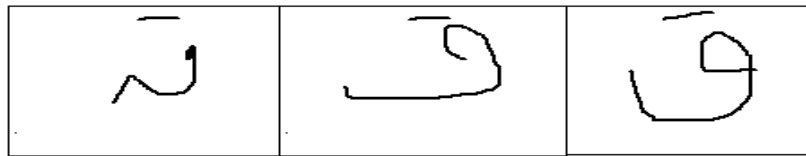
b) Online touching

c) Overlapping

Figure 1.7 Connectivity difficulties.




a) Geometric variations in GEEM

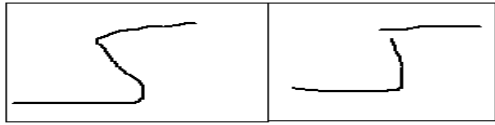


b) Variability in the loop shape.

Figure 1.8 Handwriting variability.

The variations can originate from the online-based characteristics, such as the number of strokes, the order of strokes, and the writing direction. The same unit can be handwritten with a different number of strokes as a result of the handwriting style adopted, as shown in Figure 1.9.a, or because emphasized strokes were added, as shown in Figure 1.9.b. The strokes that represent the handwritten text can be input in various orders, as shown in Figure 1.10. The variations may result from the direction of the pen movement when drawing strokes, as shown in Figure 1.11.

Although these variations are common among different writers, online text styles can also vary with the same writer, which might result from writing conditions, the writer's mood, familiarity with software or hardware, or other factors. Moreover, some writers have their own handwriting style. For example, the beginning form of the letter "Haa"  is used in the isolated and end positions, although this type of usage is relatively unusual, as shown in Figure 1.12.



a) The letter Kaf with a different number of strokes

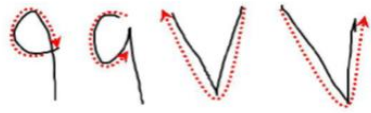


b) Over-tracing [5]

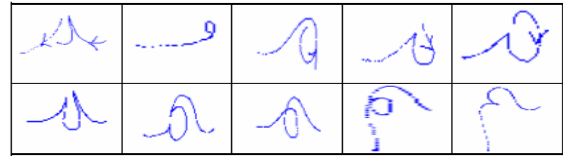
Figure 1.9 Variability in the number of strokes.



Figure 1.10 The same word written with a different stroke order.



a) Digit writing directions in [5]



b) Different writings of MEEM [6]

Figure 1.11 Handwriting movement variations.

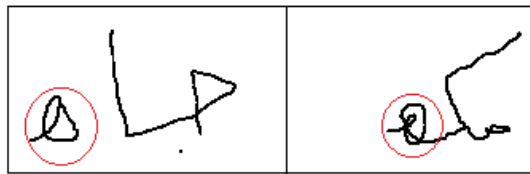


Figure 1.12 Using the beginning form of the letter “Haa” in the isolated and end positions.

There are several challenges in addressing online Arabic text recognition, which can be classified into three main categories: general issues, Arabic-text-specific issues, and Arabic-handwriting-specific issues [7]. The general issues are those that Arabic shares in common with the online recognition systems of other languages, such as the hardware devices, data availability, linguistic problems, presence of noise in the input and some non-character objects. Arabic-text-specific issues refers to the challenges that arise from the nature and characteristics of Arabic text, such as its cursive nature, ligatures, delayed strokes, stylistic variations (corresponding to fonts in printed text), and the presence of dots and diacritics and different writing forms, to name a few. Some of the difficulties result from the variability in personal writing styles, size, direction, slant, separating spaces, and cut text. The personal factors that affect online text writing include idiosyncrasies such as the writer's handedness. Right- and left-handed people may use different directions and positions when writing. Situational factors depend on the presentation of the handwriting, which might have been executed in haste or under duress [8], [9]. The material factors are based on the device used, which might cause comfort/discomfort to the writer and lead to handwriting variations, such as the size of the writing board, inaccuracies of the pen-down indication, and features of the device that might limit its accuracy [1], [8]. More details regarding the problems of writing on pen-based devices in online handwriting recognition systems can be found in [10].

1.2 Arabic Online Text Recognition Model

Text recognition systems can be categorized as online and offline, based on input. Online input is handwritten in nature, whereas offline input can be handwritten or machine printed. Printed text can be single font or Omni-font, depending on the font restrictions.

Text recognition systems are classified based on the input text connectivity into isolated characters and cursive text. The different capabilities of text recognition systems are illustrated in Figure 1.13.

Handwriting recognition is also categorized into writer-dependent and writer-independent systems. The main difference is that the writers of the testing dataset in writer-independent systems are disjoint from those writing the training data. Constrained handwriting refers to handwritten text that conforms to predefined writing rules, such as untouched, discrete-spaced characters, base-lined and aligned texts. Typically, unconstrained handwriting refers to cursive or mixed cursive handwriting script without restrictions on the writing. Dealing with constrained handwriting is easier, but it is not realistic for Arabic cursive text. Unconstrained handwriting is a more challenging problem. In general, research data is collected for use in experimental setups rather than being collected from daily life applications, which makes the data less natural, even if it is unconstrained.

Online text recognition employs a general model that is similar to that used for offline text recognition, as shown in Figure 1.14. The input is acquired from a pen-based device, which may require preprocessing. The online text may then be segmented into smaller units (e.g., strokes into letters or graphemes). Then, features are extracted to build the classification models in the training phase and to recognize the input by a trained classifier in the testing phase. Finally, the recognition results can be improved by an optional post-processing step.

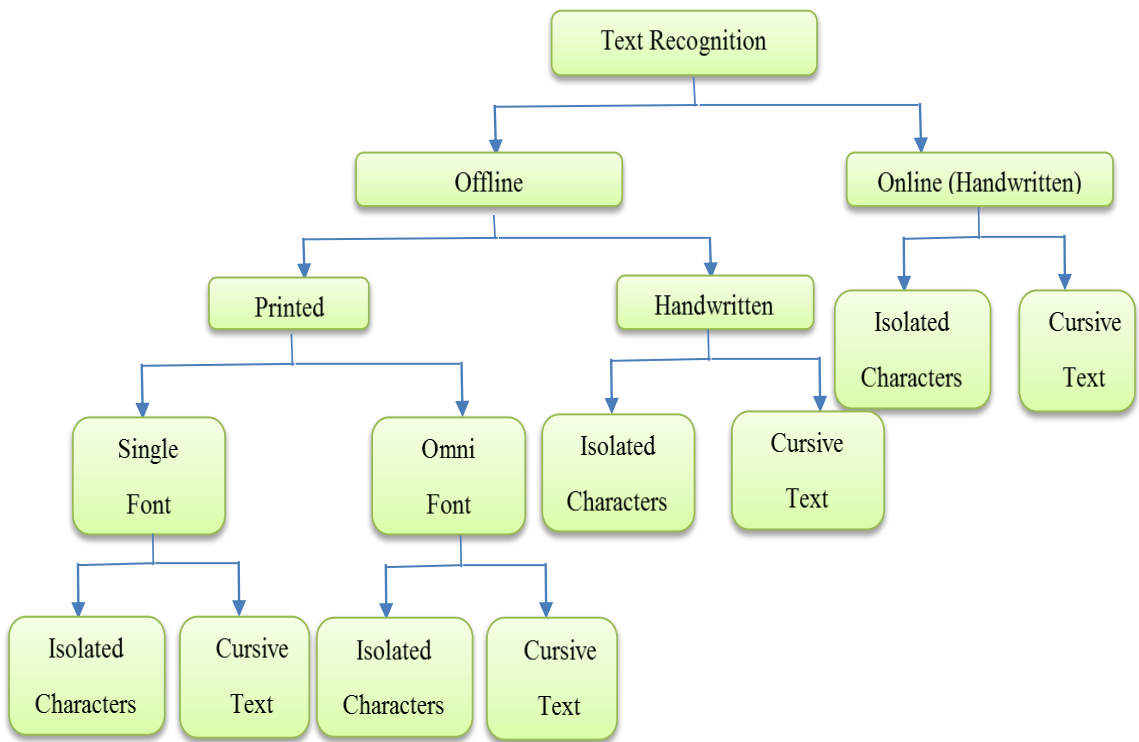


Figure 1.13 Text recognition capabilities.

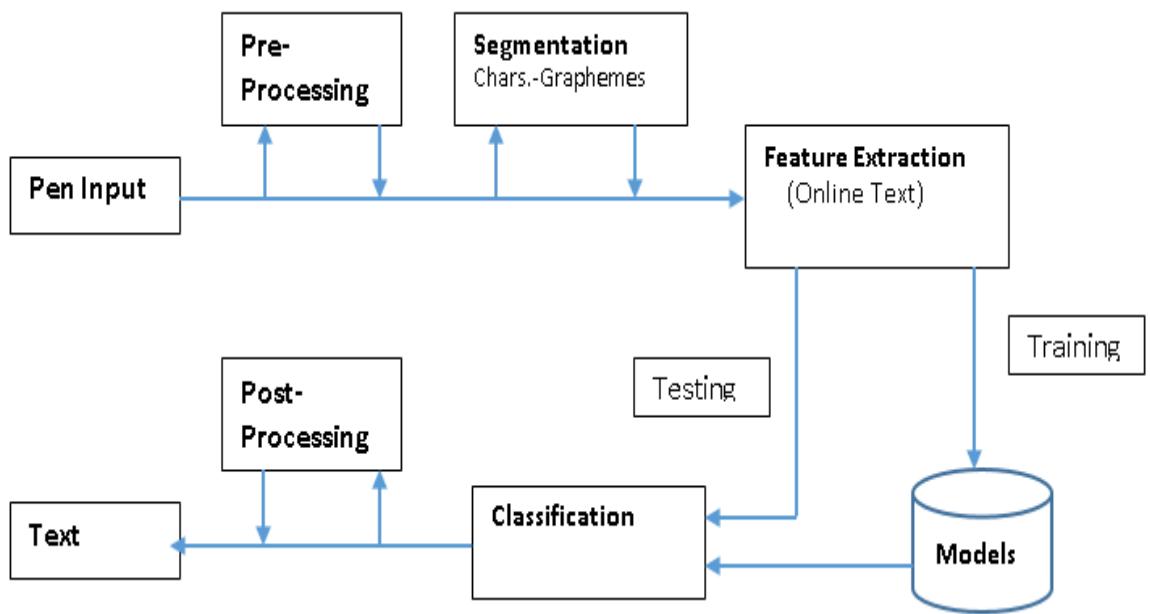


Figure 1.14 The general model of online text recognition, adapted from [2]

1.3 Motivation

The wide spread use of pen-based hand held devices such as PDAs, smart phones, and tablet-PC, increases the demand for online text recognition systems. These man machine interface systems are alternative to the traditional keyboard with the advantages of being easier, user friendly, and natural. This technology has great potential markets in friendly learning environments, business applications and more. Also, there is need for systems that support free cursive handwriting with multilingual capabilities. Besides the above mentioned facts, there are needs for research on the recognition of online Arabic text. Most of the used techniques of online Arabic text recognition are derived from those used for other languages. However, techniques of other languages may not suit Arabic text recognition. Hence, it is useful to develop systems that utilize the characteristics of online Arabic text.

The cursive nature of online Arabic text and the variability among a large number of writing styles makes the recognition of online Arabic text a challenging problem. Although this overlaps with Arabic offline text recognition, it is harder in online text recognition as the writing in online devices is less controlled than a pen on paper.

1.4 Thesis Objectives and Outcomes

The main objective of this thesis is to conduct research on automatic recognition of online Arabic text. This implies addressing the different phases of building a recognition prototype. Achieving the objective of this thesis may be useful in many applications such

as data entry using tablets and touch screen devices, data entry using smart phones and PDAs.

The outcomes of this thesis can be utilized in other areas of scientific research such as writer identification/verification, forensic handwriting analysis, and signature verification. The main outcomes of this thesis are as follows:

- **Literature Review:** A comprehensive survey of the published studies related to our topic has been conducted. This survey gives more focus on the studies that are not included in the existing published surveys (latest published survey addressed research up to 2011) and describes the limitations and restrictions of the reviewed studies and future directions.
- **Algorithms and Procedures:** The most important outcomes of the thesis are the methods and procedures that have been developed for online Arabic text recognition. Different preprocessing methods were developed to address the difficulties facing the existing methods. A large number of features were extracted and used for developing the prototypes. The main focus is on using statistical patterns. For classification, the approaches that are suitable for Arabic online text are adopted. We focused on classifiers based on Bayesian approach (like Hidden Markov Models).
- **Possible Publications:** The developed work is a result of research activities integrated into a prototype. The results of our research are reported in papers for possible publication. Some of the submitted papers are:

- Baligh M. Al-Helali and Sabri A. Mahmoud, ““*Arabic Online Handwriting Recognition (AOHR): A Survey*,” submitted.
- Baligh M. Al-Helali and Sabri A. Mahmoud, “*A Statistical Framework for Online Arabic Character Recognition*,” Cybernetics, submitted.
- Mahmoud, Sabri A., Hamzah Luqman, Baligh M. Al-Helali, Galal BinMakhashen, and Mohammad Tanvir Parvez. 2016. “Online-KHATT: An Open-Vocabulary Arabic Online Text Database,” submitted.
- Mohammad Tanvir Parvez, Hamzah Luqman, Baligh Al-Helali, Sabri A. Mahmoud, “*ICFHR2016 Competition on Arabic Online Text Recognition using Online-KHATT Database*”, the 15th International Conference on Frontiers in Handwriting Recognition, 23-26, October, Shenzhen, China, accepted.
- Baligh M. Al-Helali, Hamzah Luqman, and Sabri A. Mahmoud, “*Extension of Arabic online text database Online-KHATT*,” submitted.
- Patent: Baligh M. Al-Helali and Sabri A. Mahmoud, “*A Statistical Framework for Online Arabic Character Recognition*”, submitted.

1.5 Thesis Organization

The rest of the thesis is organized as follows. Chapter 2 presents a comprehensive literature review of research on online Arabic handwriting recognition. The developed character’s recognizer is described and the related experimental results are given in Chapter 3. Chapter 4 presents the details of the online text recognition prototype. Finally Chapter 5 concludes this thesis and summarizes the outcomes and future work.

CHAPTER 2

LITERATURE REVIEW

In this chapter, we present a comprehensive survey of AOTR that focuses on published work not covered by other surveys and reviews, such as [11]–[21]. This survey presents the characteristics of Arabic text as related to AOTR. Reference is made to [2], [22] for surveys on Arabic offline handwritten text recognition and to [3] for a general and comprehensive survey of Arabic text recognition. This chapter is organized based on the recognition phases. It reviews related published works to present their contributions and limitations. Summary tables are presented for each phase for the purpose of reference and comparison to facilitate easy comparison of the different techniques. At the beginning of each phase, we offer a general overview of the surveyed work to highlight capabilities and limitations and then present our detailed discussions.

2.1 Arabic Online Datasets

This section describes the main databases that are used in AOTR research. The other datasets used are described in the classification section because their use is limited to the cited references.

LMCA: An on/off-line dual Arabic handwriting database is presented in [23]. It is called LMCA (from the French "Lettres, Mots et Chiffres Arabe") and contains both on/off line samples of 500 Arabic words, 100,000 Arabic letters and 30,000 digits collected from 55 writers. This database is limited to a small set of words and is not natural Arabic text. Instead of extracting them from the collected words, the letters are collected separately;

hence, it does not reflect natural Arabic handwriting. The digit samples of the LMCA are used in [24], and the words' subset is used in [25].

AOD: AOD is a database of online Arabic Digits [5]. It is collected from 300 writers who were 11 to 70 years of age. Each of these writers wrote an average of 10 samples for each digit. There were no constraints on the writing style, such as orientation, size, or number of strokes for each digit. The total number of collected samples is 30,000, with 300 samples per digit in which the samples of 80% of the writers are grouped into the training set and the remaining 20% into the testing set. This database is limited to digits and is freely available at (<http://www.aucegypt.edu/sse/eeng/Pages/AOD.aspx>).

ADAB: ADAB (Arabic DataBase) was developed in a cooperative arrangement between the Institut fuer Nachrichtentechnik (IfN) and the Research group on Intelligent Machines (REGIM). This database contains online samples for 937 Tunisian city names and is used in competitions [26], [27]. It contains 33,164 Arabic words (174,690 characters) written by approximately 166 different writers. Most of the selected writers are from the narrower range of the l'Ecole Nationale d'Ing'nieurs de Sfax (ENIS). Although this database is used more often than other databases, it has several limitations. First, it has a small lexicon with limited coverage and levels. The data is available in isolated word samples, and no segmentation of the words into letters or PAWs is provided. The database is limited to city names, and it is thus not a natural Arabic online text. Despite these limitations, the ADAB database is widely used in the AOTR literature, such as in [28]–[41].

OHASD: An Online Handwritten Arabic Sentence Database (OHASD) is presented in [42], which was inspired by the IAM-Database [43] and the IAM-OnDB datasets [44]. Constructing OHASD is undertaken by collecting samples of paragraphs of complete sentences that range from 15 to 46 words. Erratic/illegible handwriting is excluded, which results in 154 paragraphs written by 48 writers, containing 3825 words and 19,467 characters. This database has a limited lexicon, limited data and a limited number of writers.

MAYASTROUN: This Multilanguage database for both online and off-line unconstrained handwriting is called the “MAYASTROUN-database” and was developed in the REGIM laboratory. This database contains cursive Arabic and Latin texts, words, characters, digits, signatures and mathematical expressions and it is presented in two versions. The first version is “MAYASTROUN set1”, which was collected from 100 writers in [45] and consists of 2600 letters, 1000 words, 1000 digits and 200 Arabic texts. The dataset has been used in previous studies [46], [47]. In the second version, set 2, the database is extended in [48] and has more than 67,825 data samples written by 355 writers. The Arabic text lexicon is limited in this dataset.

ALTEC: A large lexicon in an online Arabic text database is produced by the Arabic Language Technology Center (ALTEC) [49]. It consists of 152,680 samples of 39,945 unique words, including 325,477 samples of 14,740 unique PAWs. The database is collected from approximately 1000 writers and contains samples for digits, characters and punctuation marks. A new version of the database is AltecOnDB: A Large-Vocabulary Arabic Online Handwriting database with an added set (viz. Set-H)[50]. This set is more suitable for writer-dependent research. The main drawback is that the data is collected by

writing on paper, and the x, y co-ordinates of writing are collected by a wireless signal. Writing on paper is much more controlled and looks better than online writing on devices. Hence, systems that are built based on this database may be less accurate when used with online writing devices. In addition, this database is not freely available.

QHW: An online handwritten Arabic word dataset is collected in [51] that is called the Quranic Handwritten Words (QHW) database because the collected words are the most commonly used words in the holy Quran. There are 120 words selected that are divided into two equal-size sets. Two hundred writers from several countries from 6 to 50 years of age were asked to contribute by writing 12,000 word samples, which included more than 23,300 PAWs and 42,800 characters. This database is a closed vocabulary data set and has samples of a limited number of words.

To overcome the difficulty of obtaining a comprehensive natural online database, there have been some attempts to generate comprehensive databases synthetically. Saabni and El-Sana [40], [52] presented methods for generating synthetic online Arabic script from a given lexicon that determines the set of words and PAWs in addition to a set of handwriting prototypes. This lexicon could be generated manually by human writers or extracted automatically from a given small dataset of word shapes. These methods are used to synthesize large sets of shapes for each PAW in the lexicon. A collection of Arabic texts is explored to extract 300,000 different words with 48,000 unique PAWs. Ignoring additional strokes reduced the number of unique PAWs to 28,500. The generated data is synthetic and hence not real online data. This data can be used in the case of limited online data in the training phase to improve the trained models. Systems built with this data may have poor results when used with real online devices because the

synthesized data is not natural, and the errors of the synthetic generation affect the performance of the entire system.

Based on the analysis of these databases, there is no benchmarking database of natural Arabic online text for AOTR that is freely available to researchers. Hence, different researchers use their own or available databases. Some databases are limited by size, whereas others are limited to digits or words. As shown below, the closest to the required benchmarking database are two databases, the first is AltecOn which has the major drawback that the data collection was undertaken by writing on paper and the x, y coordinates of the writing are collected by a wireless signal. Writing on paper is much more controlled and looks better than online writing on devices. In addition, the database is not freely available. The second database, ADAB, is limited to city names, and it is thus not a natural Arabic online text. As shown below, other databases have more limitations.

Table 2.1 shows a summary of online Arabic text databases. The statistics for these datasets are extracted from the cited references. We found that statistics derived from some of the actual databases have discrepancies with published statistics, and some of these discrepancies are based on the source of the text data rather than on the collected handwritten samples, e.g., the number of characters shown in the ADAB database refers to the characters included in the words of the ground truth text, although this database does not contain segmented samples of characters. An overview of online Arabic databases and applications that focus on LCMA and ADAB can be found in [53].

Table 2.1 Summary of some online Arabic text databases ¹

Dataset	Digits	Chars	PAWs	Words	Lines	Pages	Writers
LMCA [23]	30,000	100,000	NM	500	-	-	55
OHASD [42]	-	19,467	NM	3,825	NM	154	48
ADAB [26]	NM	174,690	NM	33,164	-	1,575	166
AOD [5]	30,000	-	-	-	-	-	100
MAYASTROUN [48]	6500	5600	NM	1500	NM	200	355
ALTECOndb [50]	NM	106,433	325,477	152,680	31,124	4,512	1000
QHW [51]	-	42,800	23,300	12,000	-	-	200

As shown above, most of the databases are limited to a certain level of text (e.g., digits, words) and are also limited in size and number of writers. Moreover, databases that are more generic and representative of the Arabic language are not freely available. As a result of the lack of freely available online Arabic text databases that satisfy their requirements, several researchers have used their own collected data [51]. These datasets are described in the classification section because their usage is limited to the references cited.

2.2 Preprocessing Approaches

The preprocessing phase aims to prepare the input to be more suitable for subsequent phases. This preparation typically includes enhancing the input, processing the delayed strokes and identifying the baseline. In this section, we review different preprocessing techniques, including online text simplification, smoothing, interpolation and resampling, normalization, de-hooking, processing of delayed strokes, and baseline identification.

The raw input is typically refined by reducing the noise and distortion that is caused by hardware and software limitations or by the writer's handwriting style and erratic hand motion. The noise that originates from the devices used includes missing points, irregular

¹ "NM" means not mentioned, and "-" not included

text size, jitters in the text, and uneven distances between the collected points. The distortion that originates from the handwriting includes variations in style, size, spaces, and hooks.

Simplification of the input point sequences aims to discard redundant points that are irrelevant for pattern classification. The simplification is performed by eliminating duplicate points and thus reducing the number of total points, which can be undertaken by simply removing any number of successive points in a stroke that have the same indices, leaving only one [37]–[39], or by point clustering [54]. Douglas and Peucker’s algorithm [55] is used to simplify the input points in [56]–[60]. Simplification is performed by forcing a minimum distance between consecutive points, as in [61], [62].

Smoothing is used to reduce the noise and eliminate hardware imperfections and irregularities in the input handwriting signal caused by the acquisition devices. A common technique is to replace a point in a stroke by a weighted average of its neighbors, as in [5], [37]–[39], [63]–[68]. Smoothing is also performed by using low-pass filters, as in [54], [56], [57], [60]. Such filters include the Laplacian filter, as in [69], [70], a local regression called the “loess” filter in [59], and the Chebyshev second type filter, which is applied to the normalized trajectory, as in [25], [32]–[35], [71]. In [72], smoothing is based on the orthogonal decomposition of the online data into the Haar basis. The smoothed trajectory is the approximation coefficients of a single-level one-dimensional wavelet decomposition for each of the x and y streams.

Interpolation and re-sampling operations are used to recover missing data or to force points to lie at a uniform distance. Due to variations in writing speed, the acquired points

are not distributed evenly along the stroke trajectory. Interpolation is used to restore the missing points, which can be performed with several methods. The linear interpolation introduced in [73] is used in [5], [28], [37]–[39]. In [41], Spline interpolation is used to obtain equidistant smoothed data sequences. Resampling can be performed to obtain a sequence of points that are equidistant with respect to the coordinates (equally spaced points in terms of area), as performed in [37]–[39], or with respect to time (equally time-spaced points) [74].

A normalization step is typically performed to reduce some handwriting variations and simplify the inputs because normalization adjusts the input size in a manner that preserves the writing's spatial structure to achieve scale invariance. This task is performed in [60], [74] using linear transformation – which includes scaling – while preserving the aspect ratio. The same authors used a transition invariance step that was undertaken by shifting the points in such a way that the minimum x and y coordinates become zero. In [61], each character is mapped into a rectangle of fixed size. In [63], the input stroke fit maximally in a 100×100 square that was centered at the origin. In [41], the input size is normalized to one. The vertical dimension of the handwritten line sentence is adjusted to a fixed value to obtain a normalized size script in [33], [34]. Shifting is used to center the input as in [51], [59]. Stroke length normalization is useful for easy alignment and subsequent classification. To achieve scale invariance, re-sampling is used to make all the instants the same length, as in [64], which is accomplished by replacing the captured point sequence with a sequence that has a fixed number of equidistant points in [67], [68], [75], [76]. The writing speed is normalized by using vertex removal in [77] and using re-sampling in [56], [57].

De-hooking entails the elimination of the hook-shaped parts from the start or end of strokes. Hooks result from rapid or erratic motions with respect to placing the stylus on – or lifting it off – the tablet. De-hooking can be performed in two steps. The first step is finding the sharp points to extract the segments that form the input stroke. The second step is to remove the end segments that are straight and have a short length relative to the whole stroke. Hooks are detected and removed in the preprocessing phase, as in [39], [50], [54], [64], or they can also be modeled as a legal pattern of a ligature in natural cursive handwriting.

The baseline is the virtual line on which semi-cursive or cursive text is aligned/joined, which is an ideal parameter to use in simplifying handwritten text [78]. Baseline detection can be utilized for slant correction [21], baseline drift correction [20], localizing and removing delayed strokes [28] and in feature extraction, as in [37]. Hence, errors in baseline detection may impact other processes.

The traditional histogram method for baseline detection is used in [37]–[39], [65], [66], [79], [80]. The baseline is detected by projecting the input points onto a vertical line, and the maximal peak in the histogram is then used to locate the baseline, as described in [73]. A method that is based on dynamic programming is presented in [81]; it attempts to find the paths that have the minimum cost between the collections of text line segments.

In [78], an algorithm of straight or curved baseline detection for short Arabic handwriting is presented. The baseline detection is performed in two stages. In the first stage, a set of points of an aligned neighborhood is detected. The second stage measures the level of verification of some of the topological conditions by the most numerous set of points

found in the first stage. The conditions that characterize the baseline are based on the intersection points, with some cases of the tracings of the trajectories (e.g., ‘legs’) and the curvature angles of the grapheme trajectories below/over the baseline. This algorithm is used in [32], [34] for baseline detection and correction, as shown in Figure 2.1. The baseline is detected by checking combined geometric and logic conditions in [30], [33], which is performed by inspecting the alignment and the tangent direction of each point according to the neighborhood points.

Local baseline detection for online Arabic cursive script is presented in [82]. The proposed method is divided into three steps: diacritical marks segmentation, primary baseline estimation and local baseline estimation. The local baseline is estimated using the features extracted from the ending shapes of words. Different rules are used for baseline estimations in the Nasta'liq and Naskh style because of structural differences of the styles. The vertical density histogram of the normalized and resampled sub-strokes is calculated to determine the baseline in [58]. The height of the stroke (y-axis) is partitioned into ten intervals of equal length. The center of the most frequent interval is taken as the baseline location.

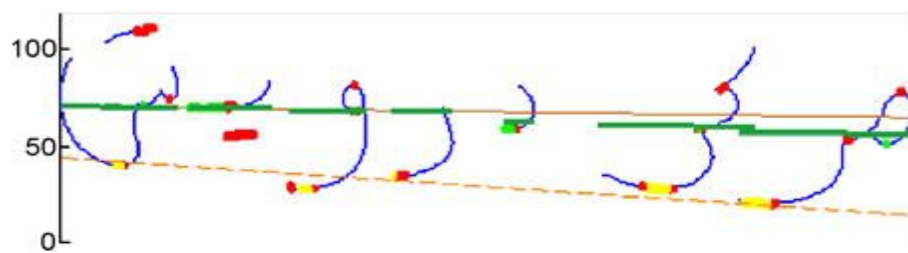


Figure 2.1 Example of detected baseline correction (green) [32]

Delayed strokes are those strokes that are added by the writer to the main strokes of the handwritten text. Delayed strokes may be dots, diacritics, or complements of handwritten text such as the Assa “” of the letter Taa “ﺕ”. Dealing with delayed strokes is a challenging issue in AOTR. Several approaches have been used to deal with delayed strokes. These approaches can be sorted into two main categories based on the involvement of the delayed strokes in the different phases. In some methods, the delayed strokes are eliminated, and the main classification processes are performed on the main strokes. In other methods, the delayed strokes are integrated into the input.

In the first approach, the delayed strokes are not considered at all. This approach is employed because of certain restrictions on the collected data in which the diacritical marks are ignored, as in [54], [64], [66]–[68], [75], [76], [83]–[86]. The delayed strokes are eliminated manually from the dataset when preparing the data in [41]. In [87], they are removed in the preprocessing phase. In these methods, a single stroke input is expected, and only the first input is considered when there are more than one. Thus, this approach is very restrictive.

In the second approach, the features are extracted from the entire input regardless of the stroke types, as in [88]. Such methods do not utilize the special characteristics of diacritical marks, and they suffer because of the writing variations of these symbols.

In the third approach, delayed strokes are detected, identified and utilized in feature extraction. In [33], the identified diacritics are associated with the segmented graphemes, and their association rates are concatenated with the features of the corresponding graphemes. In [31], [32], a method for diacritic detection and fuzzy affectation is

presented. Delayed strokes are modeled by fuzzy parameters and are associated with the segmented main graphemes using fuzzy membership function. This approach propagates the preprocessing errors to the extracted features as the diacritic identification and associated errors lead to problematic feature extraction.

In the fourth approach, delayed strokes are detected, identified, crisply assigned, removed, and then used for lexicon reduction in the classification phase. The delayed strokes are identified and used to reduce the number of candidate letters, as in [77], [89], which is undertaken with a reduced lexicon dictionary, as in [28], [29], [84], or with a hierarchical tree decision, as in [90]. In [91], once a delayed stroke is detected, it is crisply assigned to the grapheme with the maximum x-axis histogram overlap. The delayed strokes are crisply interpreted and used for lexicon reduction. In this approach, the classification process is simplified based on the results of delayed stroke preprocessing. However, the errors of this preprocessing are difficult to repair in subsequent phases, and the overall recognition accuracy is thus affected.

In the fifth approach, delayed strokes are detected, removed, and then restored in a post-processing phase to improve the classification results. The delayed strokes are used to distinguish different letters that have the same basic shape as in [69], [92], [93]. In [6], the delayed strokes are removed in the preprocessing phase and then restored in the post-processing phase to remove the classification decisions that conflict with Arabic language characteristics. Sternby et al. [94] propose a dual-graph approach using a dynamic algorithm to handle the diacritical mark variations. Branch-and-bound search techniques are used to discriminate between word hypotheses that have similar basic shapes but different diacritics. This approach utilizes delayed strokes to enhance the recognition

results. However, this approach increases the hypothesis space and decreases the discriminating power of the classifier, as will be discussed in the classification section.

In the sixth approach, delayed strokes are connected to the main strokes, which makes the entire handwritten input a continuous sequence of strokes. In [94], delayed strokes are connected to the end of a main stroke by a special stroke. In [5], linear interpolation is used to concatenate the strokes of the digit samples after reversing the points of the strokes with a directional change in the writing flow. Detected delayed strokes are projected to the nearest word part in [56], [57]. This method does not work properly when the strokes are not well located. In this study, the writers are asked to write the delayed strokes after completing the main stroke and to align them carefully. A similar method is used in [34]. The primary difference is that the delayed stroke end points (corresponding to the velocity profile) are vertically projected to the nearest main body segment. In this approach, in addition to the errors in the detection of the delayed strokes, these end points could be connected to the wrong locations.

In the seventh approach, delayed strokes are handled by rearranging the input segments to match the system models, as in [37]–[39]. This approach is used to overcome the problems of the other approaches and does not require the initial detection of the delayed strokes because all the strokes of the input are handled similarly. However, it adds to the complexity of the delayed stroke rearrangement step.

In the approaches in which it is necessary to detect the delayed strokes, the detection is processed based on certain features, such as the location, sequential order, size, and bounding box shape with respect to the entire input, as in [56], [57], [69], [77], [92], [93].

Delayed strokes are determined based on their position with respect to the detected baseline, as in [31]–[34], [79].

Several methods are used to identify the detected delayed strokes. The identification of the delayed strokes is performed by fuzzy classification using parameters representing dimensions and shapes in [31], [32]. In [28], [29], [91], a holistic approach is used to detect the delayed strokes using a set of Boolean expressions based on the strokes' dimensions, vertical distance from the baseline, the number of points, the shape, and the trace duration for each category of the eight diacritics. In [95], the geometric features from all the strokes are used to remove the secondary strokes in two stages. In the first stage, small size strokes (e.g., single dots) are filtered out, and in the second stage, an estimation for the baseline is used to filter the larger delayed strokes (e.g., Hamza, Mada, Kaf-hat). In [33], identification of the detected delayed strokes is performed using a k-nearest neighbor classifier based on the sizes and shapes of the strokes (modeled by Fourier descriptors).

Handling the delayed strokes presents several challenges. One problem is detecting the delayed strokes, particularly in the case of internal discontinuity. Another difficulty is identifying the detected delayed strokes. Moreover, in unconstrained writing, it is difficult to associate such strokes with the corresponding characters of the main strokes. The effects of delayed strokes on the recognition in an online Farsi handwriting study are presented in [96]. Some of the small characters and sub-strokes are confused as delayed strokes because of the discontinuity and vice versa (see Figure 1.5). Spatial-based decisions may cause errors because the delayed strokes are not typically located carefully during the natural handwriting, as shown in Figure 1.2. Handling the delayed strokes

based on a sequential order might not be suitable because the stroke order is not necessarily fixed. For example, delayed strokes may be written before completion of the main text, and some words can be written with different orders of strokes, as shown in Figure 1.10. Table 2.2 shows a summary of the pre-processing operations employed in the literature.

Table 2.2 Summary of pre-processing in some AOTR studies

Study	Method
Alsallakh [63]	Smoothing, normalization, then resampling.
Mezghani [67], [68], [85]	Smoothing, then resampling.
Izadi [64]	Smoothing, de-hooking, and point re-sampling.
Daifallah [54]	Smoothing, simplification, and de-hooking.
Omer [79]	Detecting baseline, delayed stroke identification.
Biadisy [56], [57]	Smoothing, simplification, re-sampling, and connecting delayed strokes.
Khodadad et al. [74]	Resampling, shifting, and scale normalization.
Ahmed, Abdelazeem, Eraqi [28], [29], [91]	Interpolation, smoothing, baseline detection, then delayed stroke association and removal.
Hosny [37]	Simplification, interpolation, smoothing, re-sampling, reordering.
Elanwar [6], [95]	Smoothing, resampling, normalization, baseline, delayed stroke removal.
Azeem [5]	Resampling, smoothing.
Tagougui [34]	Normalization, smoothing, baseline detection, delayed stroke connection.
Abdelaziz [50]	Simplification, interpolation, smoothing, resampling, dehooking, reordering.
Kour [41]	Size normalization, simplification, re-sampling, smoothing.
Ramzi [60]	Online: shifting, normalization, smoothing, resampling, simplifying. Offline: binarization, cropping, scaling, filtering, edge detection.
Abdelaziz [39]	Reordering, resampling, interpolation, smoothing, simplification, dehooking (best combination).
Abuzaraida [51], [59]	Smoothing, simplification, size normalization, centering.

As shown in this table, different preprocessing operations and orders are experimented with in [39], and the best results are achieved when using the reported combination, as

shown in Table 2.2. Offline preprocessing is also used after transforming the online input into a binary image, as in [57], [93], or by combining online and offline preprocessing as in [60], [91], [92]. The offline operations include contrast enhancing and noise removal as in [97]. In [60], the offline preprocessing includes binarization, image cropping, image scaling, low-pass filtering, and edge detection.

2.3 Segmentation Approaches

This section presents a summary of the segmentation techniques used, focusing on how the segmentation is performed rather than on how it is employed for recognition. We present the segmentation at the grapheme, character, PAW, word and line levels. The methods that do not include segmentation are categorized as global or holistic approaches, in which the input is processed as a whole during the different recognition phases. This approach avoids the error-prone segmentation step, which is a challenging problem. However, the entire vocabulary must be considered when building such systems, which is suitable for small lexicon applications, such as non-cursive text recognizers (e.g., characters and digits), or for small vocabulary applications, such as check processing. The main drawback of this approach concerns recognizing cursive text (e.g., word and PAW) in large vocabulary applications. In other words, a large training dataset is required, and an efficient recognizer should be used to discriminate a large number of classes. For this reason, an analytic recognition approach is used for such applications.

In the analytic approach, the classification level is smaller than the input level. This approach reduces the search space when dealing with cursive handwriting recognition, and as a result, the need for a large training data set is reduced. However, it propagates

segmentation errors to subsequent phases, which increases the likelihood of recognition errors for the entire system. The segmentation methods are categorized into "external" vs. "internal" segmentation, depending on whether the recognition is required in the process [20]. In [98], the segmentation strategies are classified into a classical approach called "dissection" segmentation versus recognition-based segmentation based on how the classification and segmentation phases interact in the overall process.

In the internal approach, the main interest is to bypass the segmentation problem (i.e., building a complex segmentation algorithm is unnecessary), and the segmentation is adjusted by the classification phase. Moreover, the results can be improved by including language models. However, this approach is sensitive to the training data, and a more complex training stage is required. Overall, the recognition errors are basically due to classification failures. Some of the internal segmentation techniques used are not typically thought of as segmentation methods, particularly those that are thought of as segmentation-free approaches. Most of the details of such methods are related to the classification phase, as will be shown below.

External segmentation is the classical approach in which segmentation is explicitly performed prior to the classification phase. In this approach, the dissection of the input into smaller components is based on "component-like" characteristics. In [99], extreme points, direction, intersection points, and pen speed are computed between consecutive points to distinguish between the segments. Finding the extreme points is also used in [88] for dividing letters into strokes (pre-segmentation), and each stroke is then divided into sub-strokes (tokens). In [94], the input is segmented at the vertical extreme points with respect to the writing direction. The Beta-Elliptical strategy is used to segment the

input stroke in simple movements by inspecting the extreme points of the curvilinear velocity in [25], [34], [100], as shown in Figure 2.2.

In [101], the segmentation is based on the perceptual encoding system by using genetic algorithms (GAs) to detect the global perceptual codes (GPCs) and fuzzy theory for the elementary perceptual codes (EPCs). A similar approach is used in [45]–[47]. The details of using GAs for GPC extraction are presented in [102]. A fuzzy approach is also adopted in [30] to develop a grapheme segmentation-based model. Fuzzification is achieved by overlapping the graphemes' segments based on the confidence degrees associated with the detected separating points. The fuzzy membership of the points of the extracted fuzzy graphemes are then considered for modeling the fuzzified boundary shapes. In [32], the detected baseline is used to detect the valley bottoms and the angular points to segment the input trajectory into graphemes. Such methods suffer as a result of the geometric- and spatial-based variations that are common in unconstrained Arabic handwriting, particularly online writing.

The grapheme segmentation algorithm presented in [91] is based on the local writing direction. The algorithm aims to segment each PAW's main stroke into its basic graphemes (at least one grapheme). It begins with arbitrary segmentation based on the angle between consecutive points. These points are then filtered and used to extract small, hill, circular, tail, and early segmentation junctions. The center points of the valid junctions are considered the final segmentation points. In [33], the grapheme segmentation is performed by detecting the baseline and estimating the width of the median zone. The detected baseline is then used as a topologic reference to extract specific points that delimit the grapheme trajectories, as shown in Figure 2.3.

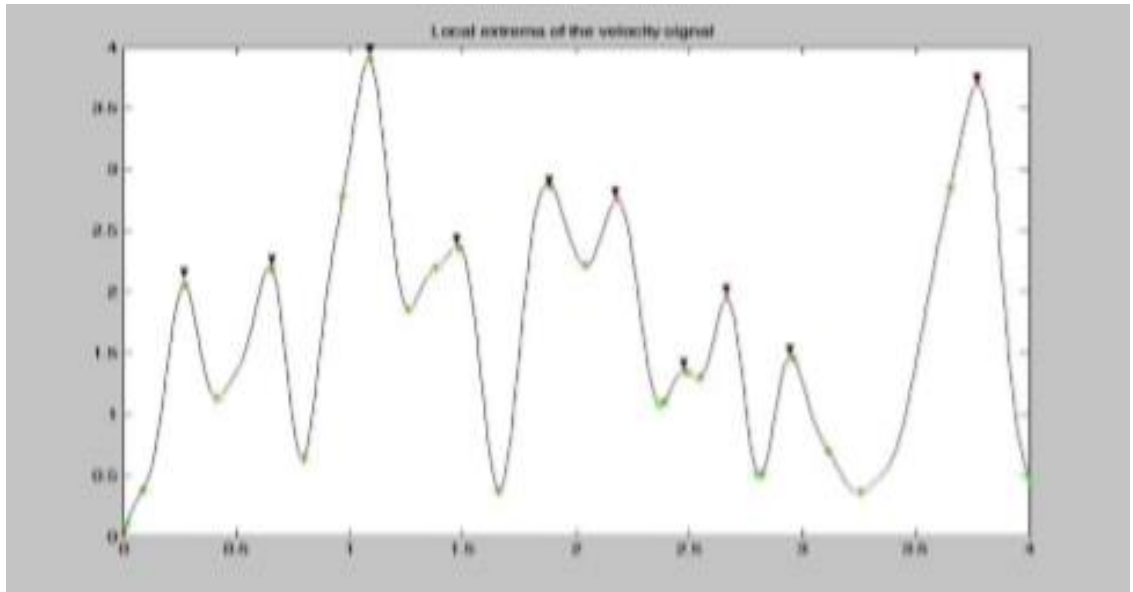


Figure 2.2 Detection of the curvilinear velocity extremum points [100]

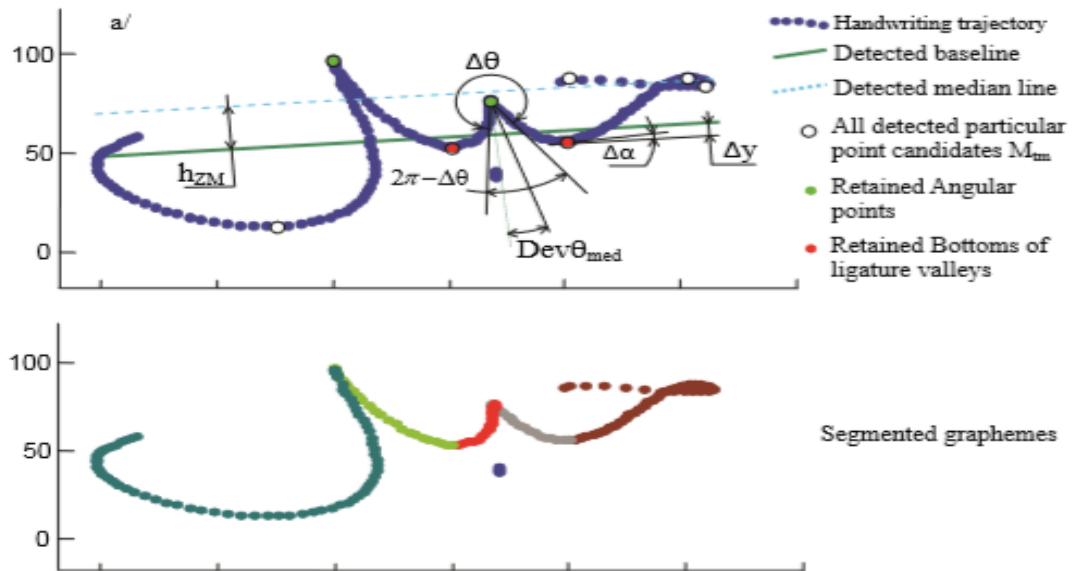


Figure 2.3 Topological characteristics used in the detection of specific points (above) and grapheme segmentation (below) [33].

Baghshah et al. presented a method for segmenting each stroke into tokens of lines, arcs, or loops [62]. Each token is specified by two end points that are determined by computing the average of the changes in the angles of successive points. In [65], character boundary-based segmentation is proposed. The algorithm consists of two stages. The first stage specifies the three types of straight writing lines: right-to-left horizontal lines, bottom-up vertical lines, and top-down vertical lines. The second stage is for detecting the beginning and end boundaries of the characters. In [103], a statistical segmentation method is presented. The main idea is to flatten the words to represent multi-connected lines, which are then filtered to obtain the ligature positions. Segmentation points are determined based on normalization, direction transformation and clustering. Then, filters are used to correct the segmentation errors at the first, last and overlapped character.

Segmentation of unconstrained cursive Arabic online handwritten documents is presented in [6]. The document is broken up into text lines, words or sub-words by a rule-based method for grouping the input strokes. Successive input strokes are compared spatially to determine whether they belong to the same word. Different strokes are considered to belong to the same word if they touch or have an x-axis histogram overlap or if the inter-stroke distance is less than the average stroke width. Otherwise, they are assumed to belong to different words. This method is not suitable for unconstrained data because of connectivity problems and incorrect segmentation that is obtained for inputs similar to those shown in Figure 1.7. In [56], [57], a straightforward PAW segmentation is performed because the input is restricted such that a PAW body should be written in a single continuous stroke. An automatic text line-detection method is proposed in [81].

Those authors used dynamic programming to find the paths with the minimum cost between collections of text lines that were segmented based on their spatial information.

In the internal segmentation approach, segmentation and classification are integrated and are thus called recognition-based segmentation. The connection degree between them depends on the scheme used. An initial segmentation can be performed and validated by the classification. The segmentation can be performed implicitly when the classification phase is involved in finding the input segments that match the classes in its alphabet. Recognition-based segmentation is performed in two stages. In the first stage, an initial segmentation points are provided. The segments obtained are then examined by a recognizer, and the unacceptable recognition results are re-segmented. Daifallah et al. [54] presented a recognition-based segmentation method for segmenting the input strokes into letters in four stages. The first stage is nominating arbitrary segmentation points (SPs). The segmentation is then enhanced by locating semi-horizontal right-to-left moving lines. This step is followed by consecutively connecting segments using a predefined set of rules to ensure that the joints do not have any writings above or below in the same stroke. Finally, the sub-strokes are classified into letters using HMM with Hu features, and the letter candidates and their scores are used to locate the best set of segmentation points. A real-time approach for segmenting open-dictionary Arabic handwritten script is presented in [58]. The segmentation is performed at the stroke level. Morphological features are employed to nominate potential SPs. The sub-strokes that result from the segmentation points are classified using a k-NN letter classifier, and the final segmentation points are then selected by finding the best-scored segmentation path.

Another interaction between classification and segmentation occurs when the classifier is used to select the best segments from a set of possibilities. In [87], the segmentation is performed manually during the training phase. For testing, a genetic algorithm with a fitness function that computes the match degree between a gene and the real handwritten word is used to find the best combination of characters. In [6], the cut points of the input feature vector are found using a dynamic programming algorithm that minimizes a defined cost function for segmenting the strokes into letters. Saabni and El-Sana [77] used a series of filters hierarchically and extracted global geometrical features that were used to determine and order the trained models (candidates) matching the input sequence using a dynamic time warping recognizer.

In [56], [57], the training is performed at the letter level, and the recognition is performed at the continuous PAW level. The writers are asked to specify demarcation points among letter shapes and align all the delayed strokes horizontally based on those demarcation points. A similar approach is adopted in [29], [37]–[39]. In [38], the alignment mode of the Hidden Markov Models tool is used to find the best character segmentation of a word. This segmentation is undertaken by checking all the possible segmentation hypotheses and retrieving the combination with the highest score according to the trained models. In [95], the segmentation process has two stages. In the first stage, words are segmented into letters using HMM-based simultaneous segmentation-recognition. The second stage involves a rule-based validation for the proposed segmentation points to solve the different segmentation errors without contextual information.

Table 2.3 Summary of segmentation methods for online Arabic text.

Level	External	Internal
Character	Alsallakh [63], Mustafa [65], Al-Emami [99], Potrus [103], Harouni [104].	Elanwar [6], Al-Barhamtoshy [38], Daifallah [54], Biadsy [56], Biadsy [57], Alimi [87], Elanwar [95].
Graphemes	Boubaker [30], Boubaker [33], Baghshah [62], Al-Taani [80], Harouni [88], Eraqi [91], Sternby [94], Boubaker [100], Njah [101].	Abdelazeem [29], Hosny [37], Al-Barhamtoshy [38], Abdelaziz [39], Kour [58], Al-Habian [83].
Word/WP	Elanwar [6], Biadsy [56], Biadsy [57].	Kour [58], Elanwar [95].
Line	Elanwar [81].	

As shown in Table 2.3, there is limited research at higher text levels (e.g., lines, and words). The input is segmented into graphemes rather than letters in some methods. A grapheme can be a combination of 2 or 3 letters, a letter or a part of a letter depending on the used method. Graphemes are sub-strokes that are extracted based on geometric primitives and shapes as in [72], [105], and they include primitive skeleton patterns in [6]. Segmentation into graphemes is also performed on non-cursive text, as in [71], [72], [80].

The classical external segmentation extracts the segmentation points using features such as the extreme points of the handwriting trajectory/velocity and points with low slopes. The main limitation of this approach is that Arabic letters may not have distinct segmentation boundaries. For example, the under-segmentation problem is typically a result of ligatures. When handwritten as shown in Figure 1.1.a, the PAW ل (Lam-Mim) does not contain a horizontal hyphen-like segment between the letters (ل) and (م). This method is less limited when segmenting into ligature graphemes. Segmentation is highly influenced by several challenges, such as Arabic handwriting variations, connectivity, and ligatures. In this approach, the obtained segments are passed to the next phases, and

no feedback from those later phases is used to enhance the segmentation phase. Hence, the segmentation performance influences the performance of the subsequent phases.

2.4 Feature Extraction Approaches

The goal of the feature extraction phase is to generate attributes that can be used to represent the input data compactly. These attributes or “features” are used in the recognition phase to discriminate between the inputs that belong to the different classes. The features can be classified into global vs. local features. Global features are those that represent the entire sample within a specific domain (e.g., stroke, character, word), whereas local features are computed for part of a sample and represent part of a stroke or character or sub-character. Another method of classifying features is based on the nature of the extracted feature into structural vs. statistical features. Structural features describe the structure of the input data, such as geometric attributes. Statistical features are computed from the raw data or from other features, such as ratios and histograms. Features are classified into online features and offline features based on the extraction processing time. The online features are extracted from the original input signal directly, whereas the offline features are extracted from a corresponding offline image of the original input. The features can also be classified based on the raw data that are used for the extraction. Spatial features are extracted from the coordinates of the input signal, and the features that are extracted using the timing of the handwriting are said to be temporal. In this section, we present the feature extraction approaches that are used by AOTR researchers.

Some researchers model the online text before feature extraction. In [87], the input is modeled based on the motor theory of movement generation and the neurophysiological

and biomechanical parameters of the equation that describes the curvilinear velocity of the script. The kinematics and geometry in the trajectory modeling are combined by representing each stroke using dynamic Beta parameters and static circular parameters [23], [106], [107]. The parameters are extracted from the curvilinear velocity of the points. The approach is improved by introducing the elliptical parameters to generate the Beta-elliptical modeling [25], [71], [100], [101], [108]. A neuro-Beta-elliptical model for handwriting generation movements is presented in [109]. Chaabouni et al. [110] used multi-fractal modeling for online text-independent writer identification. Haddad et al. [111] built a system using an adaptation module (AM) and found that it decreased the error rate without altering the writer-independent system. In [30], Fourier descriptors are used to model open forms of segmented graphemes, and the association of points with graphemes is accomplished using fuzzy membership degrees.

Visual features are extracted based on the visual appearance of the handwritten text, such as loops, valleys, and primitive shapes. The loop shape is detected by finding a crossing point in the writing trajectory, which is used as a local feature to indicate whether the point belongs to a loop, as in [39], [56], [57], and it is used globally to indicate whether the trajectory (e.g., stroke) contains a loop, as in [92], [97]. In [77], global ascender and descender loops, local features, and local relations between adjacent points are extracted from the main strokes of the body shape. However, the secondary strokes are described based on size, location, and order. The hat feature is used to indicate whether the current point is part of a delayed stroke [39].

Geometric features are widely used in AOTR. One of the simplest geometric features consists of the x-y coordinates of the points that constitute the stroke, as in [63], and their

relative positions are used in [89], [94]. Tangents and slopes are used in [30], [63], [68], [75], [76], [100]. Distances between consecutive points (delta features) are used in [28], [64], [83], [112]. The angles of the line segments that constitute the stroke are used in [29], [37]–[39], [56], [57], [62], [64], [90], [112]. The curvature of the handwriting trajectory at a point is a measure of how sensitive its tangent line is compared with the largest line of other nearby points, as in [37]–[39], [50]. Curliness is used to describe the deviation from a straight line in the vicinity of the current point in [29], [39], [50], [95]. In [64], the authors obtained features from the relative pairwise distances and from the angles of the writing trajectory, which they call the Relative Context (RC). In [95], the line length and the angle for each point are called the "Polar Moving Eye" feature when computed with respect to the current point. Similar features are also computed with respect to the first point in the current stroke or with respect to the first point in the first stroke, which these authors called "Polar PAW Fixed Eye" and "Polar Word Fixed Eye", respectively.

Several challenges arise when using structural features. The geometrical features are sensitive to the spatial-based variations of the handwritten text. The angles, sharpness, curvature and straightness attributes can vary for the same letter, as shown in Figure 1.8.a. The visual appearance of the handwritten text is not guaranteed to be as expected; for example, the loop shape can be handwritten in different ways, as shown in Figure 1.8.b. Arabic text style variation is another challenge that arises from using structural features. For example, the descriptors of the letters 'SEEN' in Nasakh style are different from those extracted when written in Ruqqah. Moreover, even if the common font styles are considered, a non-uniform writing style can be found, as shown in Figure 1.4.

Considering all the visual characteristics and their variations complicates the feature extraction phase.

Pen movements and writing direction features are those attributes that are extracted to describe the direction of the pen movement during handwriting. The cosine and sine functions are used to describe the local writing direction at each point in [39], [65], [66], [91]. Such functions are used to represent the instantaneous and relative directions in [95]. The instantaneous feature is the local writing direction “Moving Eye”. The relative feature is the writing direction with respect to one fixed sample and is called “Word Fixed Eye” when the fixed point is the first point on the first stroke in the word (i.e., the word head). It is called “PAW Fixed Eye” when it is the first point on the current stroke (i.e., the PAW head). The instantaneous feature is considered to represent the role of the online features, whereas the fixed relative features represent the role of the offline features. The writing direction can also be represented by Freeman chain code, as in [6], [113]. The direction code is computed by quantizing the angles that are retrieved from the inverse tangents. A 32-directional chain code is used in [29], [37], [39], [50], [79]. This chain code follows the trajectory in a counter-clockwise manner to keep track of the directions from one point to the next. Sixteen directional codes are used in [61], whereas the direction codes are used as global features in [93], [104].

Time-dependent features such as curvilinear and angular velocities are used in [24], [32], [78], [101], Neuro-physiological and bio-mechanical features are used in [87]. The beta-elliptical features are combined with visual coding in [25]. In [34], the dynamic features are Beta velocity profiles of the successive strokes, and the static features are the parameters of a continuous stroke segment modeled by the two arcs of an ellipse.

Acceleration coefficients are used in [28]. In [39], some dynamic sequences are used as extended functions, such as the path velocity magnitude and the total acceleration magnitude.

When using the directional features, there are some cases that should be considered. For example, the same object can be handwritten with movements in different directions, as shown in Figure 1.11.a. The delayed strokes may alter the expected movement direction pattern, particularly those tracing (writing) a previously handwritten script, as shown in Figure 1.11.b. The time-dependent features are more related to the writer than to the handwritten text.

Statistical features are extracted by computing certain statistics using other features or of the raw data. The stroke length and its relative ratio to the length of the entire input are used in [94], [104], and the number of strokes is used in [89], [92], [97], [114]. The density ratio can be used as a feature and extracted by computing the ratio of the black pixels after transforming the input into an offline image [92], [93]. The bounding box is extracted, and its height-to-width ratio (aspect ratio) is used in [50], [63], [93], [95]. The winding value, which is the algebraic sum of the direction changes, is used in [63]. In [68], [76], histograms of tangent angles and tangent angle differences are generated to represent the input character. In [69], [70], a horizontal and vertical projection profile as well as a Laplacian filter are used as features of the characters.

The characteristics of the statistical features depend on the attributes used to compute them. For example, histograms of tangent angle differences are invariant to scaling, translation and rotation because the tangent angle differences are invariant to these same

transformations. However, some of the statistics may lead to wrong results. For example, if the number of strokes is used as a feature, then different styles for some diacritics should be considered (e.g., Kaf can be written with a different number of strokes, as shown in Figure 1.9.a).

Global transforms are used to extract the features from online handwritten text. Fourier descriptors of the x and y components of the pen positions are used as a global representation in [30], [33], [68], [75]. The Discrete Cosine Transform (DCT) of the x and y coordinate signals takes advantage of the temporal features in [74], [115].

Some of the features are extracted from an offline image that is constructed from the online input. In [57], the features used include the binary pixel values. In [54], Hu's moments are extracted as a feature vector. In [115], a sequence of images is generated, and differences in the motions of the sequence images are then accumulated and analyzed. In [93], the offline features used include the density, aspect ratio, and character alignment ratio. In [60], the extraction of offline features is accomplished over three steps: zoning, traversal, and determining the types of line segments (horizontal, vertical, right diagonal, and left diagonal lines). The feature vector is formed by combining the features of the zones. The features of each zone are the normalized length of each line type, the normalized area, Euler number, regional area, and eccentricity. A summary of the features used in some studies is given in Table 2.4.

Table 2.4 A summary of the features used in some AOTR studies.

Study	Features	Statistical	Structural
El-Wakil [89]	Number of dots, relative positions, secondary stroke counts and slopes.	✓	✓
Mezghani [75]	Global: Fourier descriptors. Local: tangents.	✓	✓

Mezghani [68], [76]	Histograms of tangents and tangent differences.	✓	
Alsallakh [63]	Coordinates series, tangents, winding value, aspect ratio.		✓
Elanwar [6]	Three Freeman chain codes.		✓
Izadi [64]	Relative Context (RC): from distances and angles of the trajectory.		✓
Daifallah [54]	Hu's moments.	✓	
Sternby [94]	Angle, arc type, connection angle, length ratio, and relative positions of points.	✓	✓
Al-Taani [92]	Segment count, loops, sharp edges, secondary segments, density, orientation.	✓	✓
Biadsy [56], [57]	Local-angle, super-segment, and loop-presence.		✓
Biadsy [57]	Offline pixel values by sliding window.		✓
Ahmed [28]	Delta and acceleration coefficients, direction.	✓	
Abdelazeem [29]	Moving window. Local: delta, direction, angle. Vicinity: aspect, curliness, slope.	✓	✓
Hosny [37]	Chain code, Turning angles, curvature, baseline, and vertical position.		✓
Azeem [5]	Temporal: writing direction. Spatial: global aspect and concavity features.	✓	✓
Elanwar [95]	Moving window. Local: Moving Eye, PAW Fixed Eye, Word-Fixed Eye, Polar Moving Eye, Polar PAW Fixed Eye, Polar Word Fixed Eye. Vicinity: Chords.	✓	✓
Alijla [93]	Online Features: Number of strokes, letter direction. Secondary object Off-Line Features: Density, aspect ratio, character alignment ratio.	✓	✓
Addakiri [97]	Number of segments, loops, sharp edges.	✓	✓
Abdlshafy [66], [91]	Handwriting direction via sine and cosine sequences.		✓
Ramzi [60]	Online: directions, 8-directional freeman code. Offline: zones, geometric features.		✓
Abdelaziz [50]	32-directional chain code, curliness, aspect ratio, curvature.		✓
Abdelaziz [39]	Chain code, curliness, aspect, direction, curvature, baseline and zones, loops, hat, path-tangent, curvature radius, velocity and acceleration		✓
Omer, Abuzaraida [59], [79]	Freeman Chain code		✓

As shown in Table 2.4, different types of features are typically combined because of the shortcomings of each type individually. In [60], several experiments with different types of features are conducted to select the best features. The best results are obtained when using the chain code online features combined with geometric features extracted from the offline image (which is divided into 9 zones). The features can be extracted for the point, set of points, stroke, or set of strokes, or for the entire input. The sliding window technique, which is used for feature extraction in offline systems – and particularly when adopting HMM classification – is also used for online systems. This technique is used in

[83], for recognizing the sub-letter strokes; a feature vector that is adopted in [116] is extracted from a sliding window that has a length of 10 points (the stroke's average samples' lengths) with 50% overlap. In [57], the features are the pixel values extracted from a rectangular window with a three-column width shifted from right to left across a binary image generated from the online input script. In [29], [95], the local features are extracted from the trajectory points, and the vicinity features are extracted from a set of points within a sliding window. Some techniques are used to improve the feature extraction phase. Z-score normalization is used to reduce the effect of the range value differences of the extracted features in [37]. The feature vectors are embedded into a normed wavelet coefficient domain in which the Earth Movers Distance metric is approximated using the Manhattan distance in [41].

2.5 Classification Approaches

This section presents the classification techniques that are used in AOTR. Different types of classifiers are used, such as structural approaches, SVM, Fuzzy SVM, Neural Networks, HMM, Genetic algorithms, decision trees, and rule-based systems. This section is organized into two main sub-sections. One sub-section addresses recognition approaches for non-cursive text, such as for recognizing digits and characters, whereas the second sub-section addresses recognition approaches for cursive text, such as for words and lines of text. Global and analytic approaches of cursive text recognition are reviewed and then analyzed.

The order of presentation is based on the recognition level—non-cursive studies and then studies that recognize cursive text. The information reported includes the classification method used, details about the data set used, the performance evaluation methodology,

and the accuracy. The recognition level is considered the level of the input rather than the level that is used in the basic classification unit. These details are presented when describing the classification method used for each study. The dataset details, such as the vocabulary, size of the dataset, number of writers, and method of collecting the data are stated. The performance evaluation methodology describes how the experiments were conducted, how the performance was evaluated, and the level at which the accuracy is computed. We use “WD” to denote that the experiment is writer-dependent and “WI” if it is writer-independent. The details are presented based on the explicit or implicit information in the study that is reviewed. However, sometimes the required information is difficult to extract, such as the use of separate training and testing datasets does not imply that the evaluation is writer-independent if the separation of the writers is not specified.

The goal of the classification is to classify unknown objects into one of a finite set of classes. These classes could be whole word, sub-word, stroke, or sub-stroke, depending on the approach used. A number of classification techniques have been used in AOTR, such as Hidden Markov Model (HMM) [29], [33], [39], [50], [54], [57], decision trees [79], [92], [99], template matching [89], dynamic programming [6], [61], Artificial Neural Networks (ANNs) [34], [67], [74], [93], [97], k-nearest neighbor [115], and combinations of classification techniques [5], [71], [84]. A survey of recognition systems that focus on the classification techniques for AOTR can be found in [16].

2.5.1 Non-Cursive Text Recognition

Recognition of non-cursive text assumes that the inputs are digits or characters. Such recognizers can also be integrated into a cursive text recognition system.

Using a Beta-circular modeling approach, Kherallah et al. [117] presented an Arabic online handwritten digit recognition system that is based on neural networks. Although the circular strokes are superposed, they do not have overlapping shapes and cannot thus perfectly reconstruct the handwritten trajectory. The reported performance is (95%) on a dataset of 10,000 samples (1000 for each digit) divided into a 7:3 ratio for training and testing, respectively. An online Arabic digits structural recognizer is presented in [80], [105]. A Finite Transition Network containing digit grammar is used for matching primitive strings in each test sample. Then, the ambiguities for some digits are resolved by checking the predefined constraints. An average recognition rate of approximately 95% is reported on their own dataset, which contains 30,000 samples of the digits 0–9 collected from 100 writers. The method does not require training. Each digit is modeled using different patterns to handle the handwriting variability. A fuzzy approach is used to develop a multilayer perception neural network classifier (MLPNN) in [24]. The training stage is performed by an association of fuzzy k-nearest neighbor algorithms (FKNNA) with self-organization maps (SOMs) to obtain maps (subsystems) of digit classes with the membership degrees of those that are in the same cluster. This information is used as the desired output of the MLPNN. A database of 30,000 Arabic digits (from LMCA) was used to test the system, which resulted in a reported recognition rate of approximately 95.08%. The classification is performed in two stages in [5]. In the first stage, a single linear Support Vector Machine (SVM) classifier is used to recognize the digit Zero. In the second stage, a multi-class nonlinear SVM using the One Versus One (OVO) scheme is used to discriminate the other nine digits. The reported recognition rates are 99.69% and 98.89% in the two stages, respectively, with an overall reported accuracy of 98.73%

on the AOD dataset in a writer-independent setup. In [59], a matching algorithm called Global Alignment Algorithm (GAA) is used to recognize Arabic digits. A recognition rate of approximately 98% is reported on a dataset of 100 samples for each digit collected using a touch screen laptop from 100 writers. The data used are too small for practical applications.

A comparison of three classifiers used to recognize online Arabic letters is presented in [84]. The diacritics are ignored, leaving only 15 distinct ‘primary’ letter classes. The investigators used the perceptron, multi-layer perceptron (MLP), and genetic programming (GP) classifiers with reported recognition rates of 95%, 95%, and 83.3%, respectively. The dataset is collected from 25 writers divided into training (≈ 1680 samples), validation (≈ 1080 samples) and testing (≈ 1000 samples) disjoint sets. The Kohonen memory classifier was developed and compared with the nearest neighbor classifier (NN), which led to obtaining the reported recognition rates of 83.43% and 86.02%, respectively, in [67]. The Kohonen memory classifier results were improved by several runs, which led to a memory that increased the accuracy to approximately 88.38%. The dataset used contained approximately 7400 samples for 18 basic shapes of Arabic isolated characters collected from 17 writers. Each writer was asked to write each character 24 times. Unconstrained data were collected, which led to a wide variety of sizes and orientations. The database is divided into two sets: 5000 samples for training and 2400 for testing. This dataset was used in other studies but with only 17 distinct classes of shapes, considering that the “Fa” (ف) and “Qaf” (ق) letters have the same basic shape. The basic shapes of “Fa” (ف) and “Qaf” (ق) are not the same when written in the isolated or ending forms. It is expected that the recognition rates will be lower when the

entire Arabic character set and all forms of the characters are used. Mezghani et al. [75] developed a classifier that was based on a combination of two Kohonen maps for two different representations of letters. The reported recognition rate is 93.54%. An associative memory with the Hellinger distance is used in [76] with a reported recognition rate of 94.56%. In [85], [86], the recognition is conducted by associative memories using several distance measures. The reported recognition rates were 94.07%, 94.56%, and 94.48% when using the Euclidean, Hellinger, and Kullback–Leibler divergence measures, respectively. These measures are also used with the nearest neighbor classifier, and the corresponding reported accuracy rates were 95.26%, 95.09%, and 95.34%. An SVM classifier using the “one-vs-one” approach is used for multi-class classification, and it obtained a reported writer-independent recognition rate of 97.8% in [64]. Mezghani et al. [68] adopted a Bayesian approach for online Arabic character classification, using Gibbs modeling of the class-conditional probability density. The evaluation was performed on the dataset of [67] after adding 5 writers, which resulted in a total of 22 writers. They compared different classifiers, namely direct Bayes, indirect Bayes, Combination of Bayes, Kohonen neural network, and nearest neighbor, with reported recognition rates of 84.85%, 90.19%, 92.61%, 90%, and 94.00%, respectively. The 17 classes used in [75] are used in addition to an extra class for the letter Kaf (ك), based on the number of strokes.

Alimi and Ghorbel [61] developed a recognition system of 28 isolated Arabic letters using dynamic programming. The training set consisted of 20 sets that contained sample replications of each letter. For testing purposes, 280 additional samples were used, which resulted in a reported recognition rate of 93%. The number of characters and samples

were limited. A decision tree is used in [92], which reports an accuracy of approximately 75.3% using a dataset that contained 1400 samples collected from 10 writers in which each writes the 28 letters five times. The same dataset is used in [97], with similar online features and excluding offline features. For classification purposes, neural networks with feed-forward back propagation is used. A recognition rate of approximately 83% is reported. Decision tree and matching algorithm are used based on matching the stroke directional string in [79]. The training dataset is collected from four writers; each wrote the 28 letters three times, thus producing 336 samples. For testing, five writers were asked to write the 28 letters randomly, and the reported recognition rate was 97.6%.

A rule-based classifier for recognizing isolated Arabic letters was proposed and compared with MLP neural networks and decision tree classifiers, with reported classification results of 97.6%, 98.85%, and 97.5%, respectively [69]. The dataset used contained 840 samples, 504 for training and 336 for testing, and it was written by different writers. The same dataset is used in [70] with a reported recognition rate of 96.7% when a decision tree is combined with an MLP Neural Network. The size of the dataset and the number of writers are limited. In [93], the letters are clustered into four groups based on the number of strokes. For each cluster, a multi-layer feed-forward neural network with a different structure is used. Approximately 500 samples are used for evaluation, and the reported recognition rates are 99.1% for trained writers and 95.7% for untrained writers. These authors considered the writing variations that are in the number of strokes in the letters Taa (ت), Thaa (ث), and Qaaf (ق). As a practical matter, other characters might have variability in the number of strokes. A multi-layer neural network is used for isolated Arabic/Persian character recognition in [74], which used a 3-layer

feed-forward neural network with 35 neurons in the hidden layer (chosen empirically) and 32 output nodes (which represent the 32 classes: 28 Arabic + 4 additional Persian). The dataset used consists of 120 samples for each letter: 100 for training and 20 for testing, by different writers. The reported accuracy is approximately 95.69%. Similar to other research, it is expected that recognition rates will drop when all forms of Arabic and Persian characters are used. In [60], the classification is performed in two stages. In the first stage, a back propagation neural network is used to classify 15 non-dotted basic letter shapes. The second stage is performed using logic programming for handling delayed strokes based on their counts and locations. The evaluation is performed on a database with a similar number of online and offline character samples. For training, 1050 samples are collected from five users. The best recognition rate is 74.8% on 420 test samples. The writers are asked to write with stroke number and stroke order restrictions. In a natural setting, different writers may use different numbers of strokes for the same character, and sometimes the same writer could write the same character with different number of strokes.

El-Wakil and Shoukry [89] used a three-stage classifier that combined a hierarchical classifier, template matching, and K-NN for recognizing position-dependent Arabic letter shapes. The reported recognition rate is 84% and reached 93% when the features were manually weighted on a dataset that was collected from 7 writers. A real-time Arabic handwritten character recognition system is developed in [114], using a hierarchical classifier with a tree structure. The character shapes are grouped into four sets based on the position of the character, and each set is divided into four subsets based on the number of strokes. The reported recognition rate is 99.6% on data that are collected from

one writer (data from only one writer are limited and explain the high recognition rate). Harouni et al. [104] proposes a deductive method for online handwritten Persian/Arabic character recognition. The input strokes are segmented into tokens, and the features that are extracted from those tokens are fed into the Back Propagation and Multilayer Perceptron (BP/MLP) neural network classifiers. The evaluation is performed on a dataset of 31,620 samples with 102 different position-dependent Persian/Arabic character shapes. The recognition rates are computed for the four writing positions, for which the reported recognition rates are between 92.60% and 96.84%. The data is divided into training and testing sets. It is unclear if there is overlap between writers. The recognition of the segmented characters of the ADAB database is presented in [118], using SVM with an RBF kernel classifier and reporting 97.11% recognition rate. The number of the segmented characters are: 25,251 samples for 19 isolated letter classes and four ligatures, 29,757 samples for 12 beginning letter classes, 23,536 samples for 14 middle letter classes, and 29,538 samples for 19 end-form classes.

Alsallakh and Safadi [63] proposed AraPen, an Arabic online handwriting recognition system. This system is designed to handle non-cursive isolated letters and digits and is then adapted to the cursive case. A two-level classifier is used. First, the top three candidates are obtained using Dynamic Time Warping (DTW). A simple neural method is then used to classify those candidates. Although AraPen is not writer independent, users can train it with their own style and add new patterns; thus, AraPen is trainable and Alphabet-Extensible. To evaluate the system's performance, a small corpus is collected. For the non-cursive data, the reported accuracy is 91% on the basic patterns and 98% after training. A recognition rate less than 50% was obtained when the system was

adapted to cursive text. Assaleh et al. [115] proposes a video-based approach for online Arabic letter recognition via hand motion tracking. Two classification techniques are used: Hidden Markov Models (HMM) with time-dependent features and K-nearest neighbor (K-NN) with time-independent features. The highest reported recognition rate is 99.11% on a dataset that consists of videos of each letter being written 8 times by two writers. The video-based approach for recognizing handwriting implies several difficulties, such as processing videos and extracting text from images, in addition to the other phases of text recognition. A summary of some studies of online Arabic handwritten digit/isolated character recognition is shown in Table 2.5.

Table 2.5 Summary of non-cursive AOTR²

Data	Study	Method	Results %
10,000 digits. Train: 7000. Test: 3000	Kherallah [117]	Neural Networks	95
30,000 digits, 100 writers.	Al-Taani [80]	Structural approach	95
LMCA digits	Kherallah [71]	Fuzzy+ MLP+ SOM+ K-NN	95.08
AOD digits	Azeem [5]	SVM	98.73
Digits each by 100 writers	Abuzaraida [59]	Matching Algorithm	98
15 letter shapes, 25 writers	Klassen [84]	Perception, MLP and GP	WI:95,95,83
18 letter shapes, 7400 samples, 17 writers.	Mezghani [67]	Kohonen memory, K-NN	88.38, 86.02
	Mezghani [75]	Kohonen maps	93.54
	Mezghani [76]	Associative memory	94.56
	Mezghani [86]	Associative memory, K-NN	94.56, 95.34
18 letter shapes, 528 samples for each letter * 22 writers.	Mezghani [68]	Bayes direct, indirect Bayes, combine Bayes, ANN, K-NN	84.85, 90.19, 92.61, 90, 94
	Izadi [64]	SVM	WI: 97.8
28 letters, Train: 20 sets of samples, Test: 280 samples	Alimi [61]	Dynamic programming	WI: 93
28 Letters, 8 samples each, 2 writers	Assaleh [115]	K-NN, HMM	99.11(K-NN)
28 letters. Train: 336 samples, 4 writers. Test: random, 5 writers	Omer [79]	Decision tree	97.6
28 Letters 1400 samples, 10 writers	Al-Taani [92]	Decision tree	75.3
	Addakiri [97]	FFBP ANN	83

² WD: writer dependent, WI: writer independent

28 letters, 500 samples	Alijla [93]	ANN	WI:95 WD:99
28 letters, 504 samples for training and 336 testing	Ismail [69]	Rule based, Decision tree, ANN	WI: 97.6, 97.5, 98.85
	Ismail [70]	Decision tree + ANN	WI: 96.7
29 letters, each 250, 25 writers	Harouni [88]	BP/MLP	96.50
28 letters. Online and offline Train: 1050, 5 writers. Test: 410.	Ramzi [60]	BPNN + logic programming	74.8
32 Arabic/Persian Letters, 100 train and 20 test samples each	Khodadad [74]	ANN	WI:95.69
64 characters, 108,082 samples of ADAB, manually segmented.	Azeem [118]	SVM	97.11
102 Arabic/Persian characters, 31,620 samples, 31 writers	Harouni [104]	ANN	92.60-96.84

Recognizing non-cursive text can be used in special purpose applications, such as educational learning, as in [38], [119]–[121], or utilized in cursive text recognition systems, particularly with analytic recognition approaches (described later). There are two main limitations in the reviewed studies: the first limitation is related to the method of acquiring the data, and the second is related to the collected data themselves. In most studies, the samples are collected individually in their non-cursive forms, which limits the usability of the system in cursive text recognition. Another limitation that is found in almost all the surveyed non-cursive recognition studies is the size of the used data. The size of the data and the number of writers do not seem to be sufficiently large to guarantee applicability to real-life systems. Although the first limitation is avoided in [118], where the samples are extracted from cursive text, the limitations are inherited from the cursive text database (ADAB). In addition, most of the studies discussed above addressed a limited number of Arabic character classes and not all characters in all positional forms.

2.5.2 Cursive Text Recognition

There are two main approaches for recognizing Arabic cursive text: the global approach and the analytic approach. The global approach recognizes the input as a whole, without

segmentation, whereas the analytic approach recognizes the input based on its constituents. The analytic approach requires some type of segmentation, depending on the scheme adopted. Two main schemes are used in this approach. In the first, the segmentation is performed explicitly, whereas it is performed implicitly in the second scheme.

The global approach is also called the holistic recognition approach, in which the classification is performed on the input without segmenting it into smaller units. This scheme avoids errors that result from the segmentation. However, the classifier is trained on the entire dictionary, which is impractical for large vocabulary applications [19]. Global recognition of Arabic words using genetic algorithms and visual encoding is proposed in [107]. A set of 100 Arabic words is used for training, and two sets are used for testing; all the sets are collected from one writer. The first set of test words contains a list of words that are included in the training set. The second test set consists of pre-selected words with syllables that are included in the learned word set. The reported recognition rate using the first test set is 92%, and that rate is approximately 70% on the second test set. A similar approach is adopted in [25] on a dataset of 500 words written by 24 writers, with a reported recognition rate of 97% for isolated words. An online Arabic PAWs recognition system was developed based on an elastic matching technique in [52]. The performance of this systems is evaluated using two datasets of 500 selected unique PAWs. In the first dataset, the samples were collected from 6 writers. The second dataset is constructed by generating the samples synthetically. The reported recognition rates are 81% and 82% for the manual and synthetic datasets, respectively. In [77], a multi-level recognizer for cursive text is presented in which hierarchical filters are used

to reduce the search space. In the first filter, delayed strokes and global features were used to reduce the PAW candidates. The second filter utilizes local features in a dynamic time warping (DTW) classification. A set of Arabic words that includes the different shapes of Arabic letters is used for evaluation purposes. For training, samples of these words are collected from 10 writers. In addition, for each writer, the shapes of all the words are generated by a semi-automatic system. For testing, six writers were asked to write 100 random PAWs from the database 10 times. Three of the writers participated in the training set, and the reported results after applying a geometric filter were between 83 and 88%. The recognition results after using the shape context filter with five candidates improved to between 86 and 90%. This system is used in [40] on three datasets: a manually modified ADAB database (MM-Adab), synthetic generation from ADAB (SG-Adab), and synthetic generation from users (SG-ON-User). MM-Adab is a modified version of the ADAB database that includes 2,200 PAWs with 16,356 different shapes. The modification aims to ensure the correctness of the main component; in other words, each PAW is represented as a single connected component. The reported precision rate on MM-Adab is 78.21%, and the recall is 79.21%. SG-Adab consists of 22,000 different shapes that are generated using the letters and PAWs of the MM-Adab. The precision rate using SG-Adab is 78.64%, and the recall rate is 77.96%. In SG-ON-User, 48 writers were used to train the system to generate three sets of prototypes based on three proposed style schemes in which 31,230 different shapes for the PAWs of the MM-Adab set were generated. A precision rate of 80.43% and a recall rate of 78.12% were reported when using the SG-ON-User dataset.

In the explicit analytical scheme, a segmentation phase provides the classifier with the basic units, which is the reason it is called segmentation-based recognition. Using this approach has the advantage of being adaptable to large and even unlimited vocabulary applications, and more types of classifiers can be used. However, it suffers from the challenging nature of connectivity in Arabic cursive writing, particularly with online Arabic text, which makes perfect segmentation difficult, if not impossible. As a result, the segmentation errors propagate into the subsequent phases, which increases the likelihood of recognition error over the entire system. In this approach, an explicit segmentation step is employed to provide small units that can be used to construct the entire cursive text. These units can be letters, as in [41], [66], [99], or graphemes that represent sub-strokes, as in [30]–[32], [35], [91], [94].

Al-Emami and Usher [99] used Genetic algorithms for recognizing postal codes based on 13 Arabic characters, with reported recognition rates of 86% and 100% for writer-independent and writer-dependent classification, respectively. In [94], a template matching scheme is used for recognizing single characters and cursive words. A set of 66 Arabic words were selected such that the different shapes of the Arabic letters were included. The dataset is collected from 40 writers, which resulted in 1578 word samples and samples for the isolated characters. Half of the word samples (839) and 27 of the writers of isolated single characters are used for training. The single-character recognition experiments are writer-independent, and the highest reported accuracy was 94.8%, whereas the best reported word recognition rates are 91.2% (WI) and 91.6% (WD). In [91], one-vs-one (OVO) multiclass fuzzy support vector machines are used to the pre-

segmented graphemes using the RBF kernel. The evaluation is performed on the ADAB database with a reported word-based recognition rate of 87%.

The classification methodology discussed in [122] is adopted in [30]–[33], for recognizing Arabic words in the ADAB database. The input is segmented explicitly into graphemes, and a network of interconnected left-to-right discrete HMMs is then used to recognize the character sequences. They used a codebook size of 256 using the HTK tool, where the Viterbi algorithm is used to improve the training models. A recognition rate of approximately 86% is reported in [30]. The results are reported for three experiments conducted on ADAB database sets 1 and 2 in [31], [32]. The first reported a recognition rate of 57.87% on set 1 and 54.26% on set 2, without diacritics [123]. The recognition rate in the second experiment is 79.46% on set 1 and 77.61% on set 2 after filtering and without diacritics. The third experiment is performed with a fuzzy affectation of diacritics after filtering, which obtained a recognition rate of 87.13% on set 1 and 84.79% on set 2. The diacritics considered are limited to simple dots, double merged dots, three merged dots, or ‘shadda’. In [33], the ADAB sets 1, 2, and 3 are used for training, and sets 4, 5, and 6 are used for testing. The reported accuracy is 87.46% for the training sets and 85.37%, 85.37%, and 87.62% for testing sets 4, 5, and 6, respectively.

Tagougui et al. [34], [35] proposed a hybrid MLPNN/HMM system for Arabic online text recognition. The input is segmented into continuous sub-strokes called segments based on the Beta-Elliptical strategy by inspecting the extremums’ points of the curvilinear velocity profile. They used Multi-Layer Perception Neural Networks (MLPNNs) trained on segment-level contextual information to extract character class probabilities. The output of this network is decoded into 120 characters that are modeled

by a 4-state left-to-right discrete HMM with a codebook size of 256 to provide character-level recognition. These 120 models are the positional-based character shapes in addition to digits and some ligatures found in the ADAB database. For the training, 6000 words were chosen randomly from the first three sets of the ADAB and segmented into characters, which yielded 378950 segment strokes. These strokes are injected into the MLPNN to assemble different Arabic character skeletons that are used in HMM modeling. The training is supervised, and some steps are performed manually. For testing, sets 4, 5 and 6 of the ADAB are used to evaluate the proposed system performance, with a reported 96.4% character-based recognition rate. The system is designed in a character-based manner to be used for open lexicon applications. In [66], the classifier is designed based on HMMs with Gaussian-mixture models (GMM). In the training phase, the segmentation algorithm in [65] supplies the classifier with exact letter segments to train the corresponding HMM models. However, the algorithm proposes many alternatives of the segments in the testing phase, and the best sequence is selected as a classification result. The evaluation is performed using 100 samples of two personal names with different writing styles.

In [41], a rapid Arabic online handwritten character recognizer is proposed that aims at facilitating real-time handwritten script analysis tasks. For character-based classification, a k-d tree is built to be used by k nearest neighbors for retrieving a given query character. DTW is employed in refining the similarity scoring of the candidates. The PAW recognition is performed by a holistic approach that employs the character classification information that is obtained using real-time segmentation. The system is trained and evaluated on characters of all positions using letter samples that are extracted manually

from the ADAB database. The performance of the classifier is measured using 10-fold cross-validation, and the highest reported recognition rates are 91% for characters and 90.8% for PAWs.

In the implicit analytic scheme, recognition and segmentation are performed simultaneously. This scheme avoids the main disadvantages of the other two methods: the need for large amounts of training data and the error-prone explicit segmentation process. This approach can be performed using a parallel or serial optimization scheme. The serial method searches for a "satisfactory" recognition result iteratively in a left-to-right scan of the input. In the parallel scheme, the process is performed more globally by generating a lattice of all (or many) possible combinations, and the final decision is found by choosing an optimal path through the lattice. Typically, the input is systematically divided into overlapping segments and is used to find a coherent segmentation/recognition result. This approach is also called "segmentation-free" recognition. "Segmentation-free" is considered to be misleading as terminology [98] because recognition necessarily involves segmentation, whether explicitly or implicitly. In [87], a genetic algorithm is used to select the best combination of the recognized characters using a fuzzy neural network. The models considered are the position-dependent basic shapes of Arabic letters that can be represented by one stroke and the Kashida (–) connecting character. The dataset used is collected from one writer, and a training set of 2,000 character samples and a testing set of 100 samples of one word achieved 89% recognition rate. The data is limited and is from only one writer.

Biadsy et al. [56] used recognition of un-diacritized (un-vocalized) Arabic words based on Hidden Markov Model (HMMs). Each position-dependent letter shape is modeled by

a discrete left-to-right HMM with various numbers of states based on the geometric complexity of the characters. These models are embedded in a network that represents the PAW dictionary, which is optimized by grouping all the shared suffixes. For word recognition, the best PAW sequence is found from the input PAW observation sequences. The training data is collected from four writers, where each is asked to write a list of 800 words that were selected to include all Arabic letter shapes with an almost uniform distribution. The testing dataset contains samples of 280 other words that were collected from ten writers (the four trainers and six new writers). Several experiments are conducted with different dictionaries, where all contain the 280 test words. The results are reported for word and PAW recognition. The highest reported recognition rates are 96.47% (WD) and 96.28% (WI) for words and 98.44% (WD) and 98.49% (WI) for PAWs. This work is extended in [57], and a segmentation-free system is introduced by performing the recognition at the continuous PAW level and the training at the letter level. The presented results in [56] are compared with the results derived from using different datasets, features, and classifiers. To validate the results of using the manual database, the system is applied using a synthetic database. The methods given in [52] are used to synthetically generate the same 280 test words. In total, the evaluation set included 3282 multiple shapes of the same 280 words that were synthetically generated and 1200 shapes written by the four trainers. The results of using the HMM classifier on the synthetic database are 92.14% (WD) and 91.44% (WI). To evaluate the effectiveness of the features used, the same system and dataset (manual database) are used while the geometric features set are replaced by sliding window offline features, for which recognition rates of 91.21% (WD) and 92.11% (WI) were reported. A Dynamic Time

Warping (DTW) technique is used for comparison with the HMM classifier, which yields recognition rates of 91.24% (WD) and 96.18% (WI) using the geometric features of the manual database.

Al-Habian and Assaleh [83] presented a recognizer for online Arabic cursive handwriting using HMMs with Gaussian-mixture continuous emissions in which the basic recognition units are the sub-letter strokes. HMMs are used for stroke recognition, and a decision logic tree is then used to convert the output into recognized words. The dataset used is collected from six writers, where each is asked to write a number of words six times (which are chosen to include all strokes in a balanced manner). A writer-dependent evaluation is performed in which half of the data for each writer are used for training, and the remaining half are used for testing. The reported recognition rates for the strokes and letters are 78.25% and 75.25%, respectively. Daifallah et al. [54] presented a recognition-based segmentation of Arabic words and letters that was based on HMM. The number of states and codebook size were fixed at 7 and 4096 for all the models, respectively. The system is evaluated using 150 non-pointed word samples that consist of 720 letters. The results of the writer-dependent experiment on the same training user samples are 92.6% for words and 97.2% for letters. However, the testing results for samples written by a new user who knows how to use the tablet and pen are 85.3% for words and 88.8% for letters. Moreover, the testing for samples that are collected from new users who do not know how to use the tablet and pen yields reported recognition rates of 71.3% for words and 79% for letters.

In [6], a simultaneous recognition and segmentation system of words in an unconstrained cursive online text using Rule-based dynamic programming is presented. The input

strokes are manually segmented according to pre-defined pattern shapes for training. These patterns are clustered and stored in a registry. In the testing stage, words are tested sequentially, and the extracted skeleton patterns are compared with all the training patterns using a dynamic programming technique using Minimum Edit Distance to find a globally optimal set of cuts of the input test string (feature vector), which minimizes the defined cost function. The training set consists of 317 words (1,814 characters), which were written by four writers, and the testing set consists of 94 words (435 characters), which were written by another four writers. The reported recognition rates are 95% and $\approx 75\%$ for character- and word-based recognition, respectively. The HTK Toolkit is used to implement a HMM-based simultaneous segmentation and recognition in [28], [29]. Left-to-right HMMs with 16 Gaussian Mixtures are used for modeling the primary shapes of the position-dependent Arabic characters after removing the delayed strokes. The delayed strokes are then restored to reduce the candidates. In [28], the first three sets in ADAB are used to conduct three experiments; in each experiment, two sets are used for training and the third for testing. The highest reported word-based recognition rate is 95.27%, which is obtained when the testing set is set 3. In [29], the training is performed on the ADAB database using non-segmented words and manually segmented characters. For testing, 300 Arabic personal names are collected from 10 writers; each was asked to write 30 names randomly. The reported recognition rate is 92.5% for word-based recognition. In [36], a fusion of two HMM-based classifiers was applied, which resulted in a word-recognition rate of 97.78% using ADAB sets 1, 2, and 3 for training and set 4 for testing. One classifier is designed to recognize the online input by directly utilizing

online features, and the other classifier recognizes the input after converting it into an offline image.

In [37]–[39], a more sophisticated HMM-based approach is used. Hidden Markov Models are trained with advanced modeling techniques adopted from speech recognition, such as context-dependent modeling, speaker-adaptive training, discriminative training and Gaussian mixture splitting. The training process is initiated by building 115 mono-grapheme models that represent the different characters in the ADAB database (103 Arabic letter position-dependent shapes, 10 English digits, Arabic MAD symbol '~' and the English letter Capital V). The HMMs used are applied left to right with the number of states varying from 3 to 9, depending on the complexity of the character's shape. A more sophisticated HMM modeling method is then used by building tri-grapheme context-dependent models. This type of modeling method requires a large dataset. To address this problem, a decision tree-based clustering technique is used. When ADAB database sets 1, 2 and 3 are used for training, and when set 4 is used for testing, the system performance is evaluated using Mono Grapheme, Writer Adaptive Training (WAT), Tri-Grapheme, DT, and Gradual Gaussian techniques with reported word-based recognition rates of 88.71%, 90.84%, 93.48%, 94.44%, and 94.63%, respectively, in [37], and 94.43%, 94.83%, 96.18%, and 97.13% in [39]. In [39], the system is evaluated using a large vocabulary database (i.e., the ALTEC Arabic Handwriting database (ALTECOnDB)) [50]. Two passes are used. In the first pass, a word-internal tri-grapheme HMM model with a bi-gram language model is used to generate a word lattice that represents a reduced search space of the hypothesis sets. In the second pass, this lattice is re-scored with a cross-word tri-grapheme HMM model and a fifth-gram

language model. The performance of pass one that is obtained is improved by using pass two, where the reported accuracy is increased from 68.76% to 80.07% for the Writer-Independent models and from 79.40% to 87.47% for the Adapted models. A summary of selected online Arabic cursive text recognition research is shown in Table 2.6.

Table 2.6 Summary of cursive AOTR³

Data	Study	Method	Results %
120 words of 13 characters	Al-Emami [99]	Genetic algorithm	WI: 86 WD:100
Train: 2000 letter Test: one word 100 samples, one writer	Alimi [87]	Genetic algorithm	89 WR.
100 words; one writer	Kherallah [107]	Genetic algorithm	92 WR.
LMCA words	Kherallah [25]	Genetic algorithm	97 WR.
Words, 2916 letters, 6 writers	Al-Habian [83]	HMM +decision tree	WD: 75.25 CR
66 words, 1578 samples, 40 writers	Sternby [94]	Template matching	WD: 91.6 WR WI: 91.2WR,94.8 CR
150 word samples, 3 writers	Daifallah [54]	HMM	WD:92 WR, 97CR WI: 85 WR, 88 CR
100 samples for two names taken from SUSTOLAH.	Abd Alshafy [66]	HMM	90
Train: 800 words, 3200 samples, 4 trainers. Test: 280 words, 2358 samples, 4 trainers + 6 testers	Biadisy [56]	HMM	WD: 96.47 WR,98.44 WPR WI: 96.28 WR, 98.49 WPR
	Biadisy [57]	DTW	WD:91.24,WI:96.18
	Biadisy [57]	HMM, Offline	WD: 91.21 WI: 92.11
Synthetic words. Train: 800, Test: 280; 4,480 samples	Biadisy [57]	HMM	WD: 92.14 WI: 91.44
ADAB sets 1, and 2.	Boubaker [32]	HMM	1: 87.13, 2: 84.79
	Boubaker [30]	HMM	1: 86.39, 2: 83.56
MM-Adab, SG-Adab, and SG-ON-User.	Saabni [40]	DTW	PrR: 78.21, 78.64, 80.43. ReR: 79.21, 77.96, 78.12
5056 words in 1023 lines from 5 writers.	Parwej [124]	HMM (Offline, Online, Combined)	PrR: 62.8, 66.3, 68.2
ADAB sets 1,2 and 3	Eraqi [91]	Fuzzy SVM	87
	Ahmed [28]	HMM	95.27
	Kour [41]	k-d tree+K-NN+DTW	91 CR, 90.8 WPR
ADAB Train: 6000 words from sets 1,2,3.Test:4,5, 6	Tagougui [34]	hybrid MLPNN/HMM	96.4 CR
ADAB Training: sets 1, 2, 3; Testing: sets 1+2+3, 4, 5, 6	Boubaker [33]	HMM	87.46, 85.37, 85.37, 87.62
ADAB Training: sets 1, 2, 3; Testing: set 4	Hosny [37]	HMM	94.63
	Azeem [36]	HMM (Offline, Online, and Fusion)	95, 95.5, 97.78
	Abdelaziz [39]	HMM	97.13

³ WD: writer-dependent, WI: writer independent, WR: word recognition rate, WPR: word-part (PAW) recognition rate, CR: character recognition rate. PrR: Precision Rate, ReR: Recall Rate.

Train: 317 words, 4 writers. Test: 94 words, other 4 writers	Elanwar [6]	Rule-based, DP	WI: 95 CR, 75 WR
ALTECOonDB	Abdelaziz [39]	HMM	WI 80.07, 87.47 WD

The results are organized based on the data used, including the vocabulary, number of writers, dataset size, and separation of training and testing sets. This is useful for comparing the different techniques. Some of the results are different, although the same data and classifier are used, which is due to the difference in the preprocessing steps, in the features used, or in the setup. Most databases are difficult to obtain. Moreover, the statistics are not available for other databases, such as SUSTOLAH [66] and APOHCD [74]. Most researchers use their own databases, and some present results using databases that are not available to the public [60]. Each database is mostly used by its originators, which makes it difficult to compare different researchers' results, and such comparisons are important to improve research. One exception is the ADAB database, which is used as a benchmark in several studies as well as being used in competitions [26], [27]. However, this database has several limitations, as described in Section 2. The evaluation of the experiments depends on the methodology used, and in some studies, the accuracy is computed at the character level for recognizing cursive text.

It is difficult to achieve high recognition rates without considering the variations in Arabic text because there is significant variability when writing some of the characters with different styles. Some of the studies avoid this obstacle by restricting the allowed style (constraint) as in [6], which used only Naskh. One method for handling the intra-variation is to use multi-patterns for some classes. Determining those patterns is difficult. Such patterns can be constructed manually based on certain characteristics. In [93], the letters Taa (ﺕ), Thaa (ﺙ), and Qaaf (ﻕ) have patterns based on the number of strokes,

which can be accomplished automatically during training by using SOM with fuzzy k-nearest neighbor algorithms (FKNNA) as in [24]. Another method for handling the intra-variation is to rely on the classifier to tolerate the variations in the samples that belong to the same class, based on the variability of the training dataset, as in [37]–[39], [74], [121].

Statistical approaches are useful for handling class variations. However, they require large amounts of training data. Structural approaches are robust even with limited training data, and no training is required with some methods, as in [80]. The difficulty comes from the need to consider different patterns for the same class when there is intra-class variation. HMMs are widely used for cursive text recognition because of their ability to learn the segmentation behavior from a training set. The HMM-based “segmentation-free” approach that is used in speech recognition is employed in AOTR by utilizing the similarity in the serial digital representation of both speech and online text.

The basic units of classification can be an entire word (global scheme) or a character or even a sub-character. Recognition based on sub-character graphemes reduces the number of basic classes, as in [83]. However, some effort is required to produce meaningful text from these graphemes. Processing the diacritical marks, special symbols, punctuation marks, and non-text objects is mostly addressed when they are available in the dataset used. For example, when the ADAB database is used, there are certain digits and the Latin letter “V” that are modeled that are in addition to the Arabic letters.

Considering diacritics when modeling letters in cursive text assumes that the delayed strokes are integrated with the main strokes and that the features are extracted from the

result. This approach helps to improve the discrimination power of the classifier. However, the main problem is the error-prone integration methods, as such methods are expected to have errors when the handwriting is unconstrained. Alternatively, when modeling letters without diacritics is adopted, the classification complexity is reduced because different letters are represented as one model (class) if they share the same main body (i.e., they differ only by the number and position of the delayed strokes). This approach has the drawback of losing some information that is required when building a general recognizer. Even if the application is limited and does not have to differentiate between different units with the same main body, ambiguity can result if some Arabic letters have a shape that is similar to the composition of other letters. For example, the word "بين" which consists of the three letters "ب", "ي", and "ن" (Figure 2.4.b), is similar to the letter "س" (Figure 2.4.a) when the delayed strokes are removed (Figure 2.4.c). Hence, there is a need for an extra phase to utilize these removed strokes to enhance recognition results.

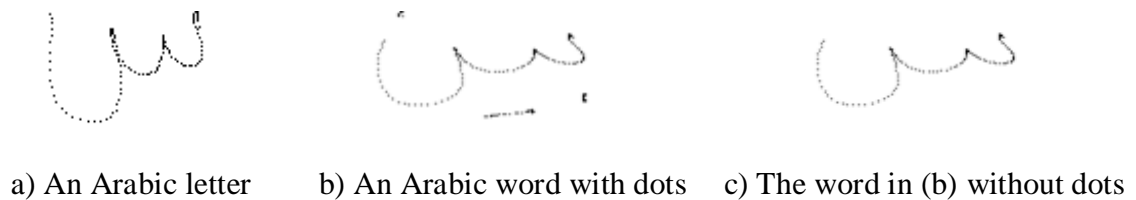


Figure 2.4 An ambiguity that is caused when delayed strokes are removed

2.6 Post-Processing

The output of the classification phase is normally a list of hypotheses or the recognized unit. The post-processing phase is used to enhance the recognition accuracy by utilizing the context to improve the results of the classification phase. As shown below, few research studies have used post-processing techniques to improve the results. The most

commonly used approach is to use a dictionary of frequently used words to find the best string that matches the classification output [87].

Post-processing is conducted in two steps in [83]. First, rules are imposed on the letter shapes to exclude the recognized candidate letter shapes that are invalid for a corresponding position in a word. Then, a word dictionary is used to manage forming the words by joining the recognized letters. In [57], a PAW dictionary network for letter models is constructed. This network is utilized to verify the PAW recognition and construction of the recognized words from the recognized PAWs. Testing is performed with five different dictionary sizes to evaluate the performances under different conditions of ambiguity.

In [39], the system results are enhanced using an additional post-processing step to re-score the multiple hypotheses of the system's results with higher order language models and cross-word HMM models. A dot restoration step is added in [6]. The restored dots are used to remove the inconsistent decisions from the list based on Arabic-language characteristics. A ranked list is created after filtering the initial list, which is built based on the classification results. Filtering is performed by removing the candidate characters that are inconsistent with the numbers and locations of the dots. In [94], the word recognition results are given with and without dictionaries of different sizes. Using a reasonable size dictionary improves the performance. The dictionary used is limited to static matching, and it is better if it is expanded to handle other cases. Spell checking can be used to enhance the classification results where a dictionary is utilized in addition to natural language processing techniques.

CHAPTER 3

A STATISTICAL FRAMEWORK FOR ONLINE ARABIC

CHARACTER RECOGNITION

Research on recognition of segmented (non-cursive) characters has several advantages. It can be used to validate the techniques employed in the different phases of the recognition systems. It can be utilized in cursive text recognition systems, especially those using the analytic approach as in [41], [66]. It also can be used for specific purpose applications such as educational systems, forms filling, and digits recognition (digits in Arabic are non-cursive).

In this chapter, a recognizer for unconstrained Arabic online character is developed. One of the contributions reported is the methodology of handling the delayed strokes. The delayed strokes are handled differently at the different phases of the recognition process to improve the overall performance. Another contribution is the intensive investigation of several novel statistical features using a developed framework for generating different statistical features. The framework consists of two main components. Most of the point-based features (local features) that are found in the related literature are extracted. A statistical layer is then added to form statistical features. Those features can be at the level of points, sub-strokes, the whole stroke, or a combination of the different levels.

Moreover, the used dataset is extracted from a database of unconstrained online cursive text and not written in isolated form. Using such dataset implies the need of addressing additional complexities as will be discussed later.

3.1 The Proposed Framework

In this section, we briefly describe the proposed framework shown in Figure 3.1. The details of preprocessing, feature extraction and classification phases are given in their corresponding sections. A character sample is input into a pre-processing phase which involves enhancing the input signal, detecting the primary and delayed strokes, and then merging the primary strokes. The output is the main stroke and a set of secondary strokes (if any). The attributes of those strokes are obtained using the proposed statistical features extraction method. The extracted features are then fed into the classification phase to identify the primary and secondary candidates with their likelihoods and combine them to select the most probable character.

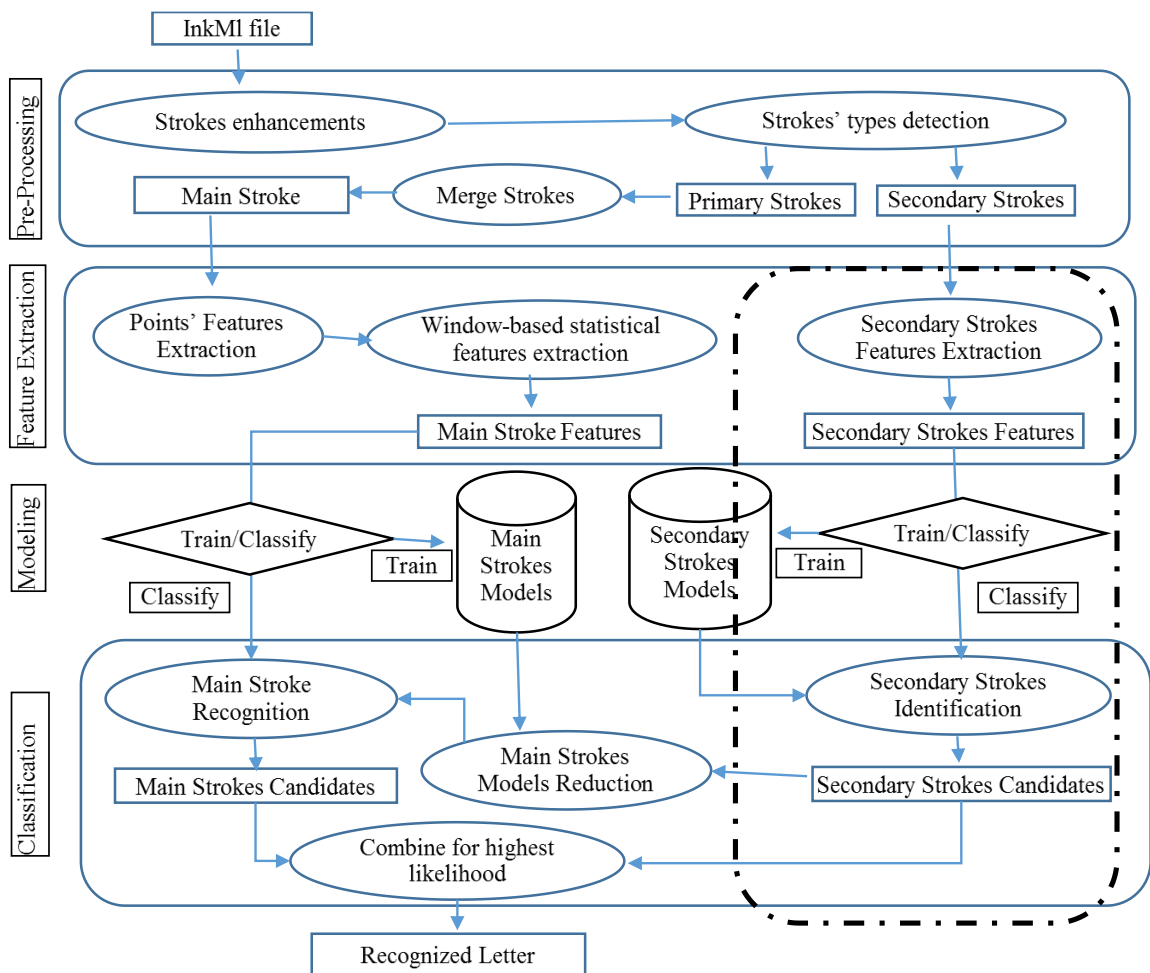


Figure 3.1 The proposed framework for character recognition.

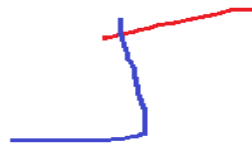
3.2 Delayed Strokes Handling

Handling the delayed strokes is described in this section as it is involved in the different phases (dashed rectangle Figure 3.1). Delayed strokes handling has several challenges. Such strokes are assumed to be “delayed” during handwriting (i.e. written after completing the main strokes). Thus, processing of the delayed strokes is complicated when the delayed strokes are written before the main strokes. The main stroke represents the main body shape of the character whereas the secondary strokes, normally, represent the dots and diacritics. This assumption has drawbacks when the delayed stroke is a complement grapheme and in the presence of internal discontinuity as shown in Figure 3.2. For instance, the end form of the letter Dal “ﺩ” can be written using two strokes as shown in Figure 3.2.a. In this figure, none of the strokes represents a diacritical mark. Another cause of complication is due to the variability of the shape, size, and location of the graphemes that represent them.

The difficulties of handling delayed strokes can be avoided by imposing constraints such as writing the main stroke before the secondary strokes as in [60] or by forcing the main body of the input to be written in one stroke as in [61]. Most reviewed studies considered the delayed strokes as diacritical marks. However, such constraints limit the applicability of the system on realistic data and impact extending the character recognizer to the cursive case. More discriminative features may address the variation problem and possibly using more models for the diacritics.



a) End form of Dal “د”



b) Beginning form of Kaf “ك”



c) Middle form of Ha “ح”

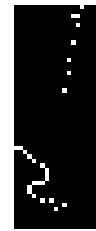
Figure 3.2 Examples internal discontinuity.

In this work, the delayed strokes are considered in the different phases of the proposed recognizer. For example, detection and merging operations are performed when needed in the preprocessing phase. In the feature extraction phase, some features are extracted for the delayed strokes such as number, position, dimensions, aspect ratio, and changeability of writing directions. A simple hierarchical classifier is used to identify the detected delayed strokes. The classifier returns the candidate types along with the corresponding likelihoods. In the first level, the candidates are filtered according to their position and strokes' number. For example, three secondary strokes lying above the main stroke are considered as three dots and the other options are ignored. A second level is involved in the case of having more than one candidate returned from the first level (e.g. an above secondary stroke which can be one dot, two dots, three dots, Hamza ...). This process assigns confidence values for the delayed strokes. The identified delayed strokes candidates are then used for lexicon reduction by eliminating the main shape models that are unlikely to have any of the detected secondary units. An extra step is performed to select the best main-secondary stroke combination. The selection is based on the classification likelihood of the main stroke candidate, the secondary unit identification likelihood, and the association likelihood between them.

3.3 Pre-Processing

This phase involves single-stroke and multi-strokes preprocessing. Single-stroke preprocessing enhances the input signal using simplification, resampling, smoothing, and interpolation. The simplification process is performed by eliminating any number of duplicate successive points in a stroke, leaving only one. A weighted average smoothing filter is used to reduce the noise and eliminate hardware imperfections and trembles in the input handwriting signal caused by acquisition devices. We applied interpolation and resampling operations to recover missing data and to make points lie at uniform distances. The linear interpolation introduced in [73] is used for this purpose.

Multi-strokes preprocessing stage is employed for inputs with more than one stroke. It implies detecting the primary and delayed strokes and then merging the primary strokes. The output is the main stroke and a set of secondary strokes (if any). The detection of the primary and secondary strokes is based on some global features (viz. the size, distances, and the ratio of points lying above, below, left, right of the main stroke, and the order of input). Depending on these attributes individually causes errors. For example, the size-based detection is not suitable since the delayed stroke can be larger than the main stroke as shown in Figure 3.3. Strokes' location and relative position are not useful in the case of segmented characters. Such features are more suitable for the cursive text case since some methods can be utilized such as the baseline detection. The expected strokes' input order has some variations, i.e. the delayed strokes can be input before the main stroke in some cases.



a) Up, the delayed stroke b) Down, the delayed stroke b) Down, the delayed stroke
“hamza” in the letter "ا". “two dots” in the letter "آ". “hamza” in the letter "أ”.

Figure 3.3 Examples of delayed strokes larger than main strokes.

The merging process is developed in a manner that avoids difficulties resulting from the internal discontinuity problems like those shown in Figure 3.2, where the main body is represented by more than one stroke. The order of the merge process during the preprocessing is important. If the merge is performed before the geometric pre-processing then some unwanted results may occur. For example, if the strokes of the segmented characters representing the end form of the letter "ha" ح in Figure 3.4.a are concatenated using a linear interpolation then the output will be similar to the end form of the letter Ain "ع" as in Figure 3.4.b. This problem is avoided in [118] by removing the small strokes that are on the left or the right sides of the main stroke. However, such deletion operation has its own problems. Removing the delayed stroke can cause confusion in some cases. For example, ignoring the "Kashida" stroke makes the character similar to the isolated form of the letter "ha" ح as shown in Figure 3.4.c. Similarly, removing the red strokes in Figure 3.2.(a-c) result in shapes that are similar to beginning forms of Dal "د", Lam "ل", and Ha "ح" rather than the end form of Dal "د", beginning form of Kaf "ك", and middle form of Ha "ح", respectively. In this example separating the two strokes can be adopted when modeling the Kaf letter as having a delayed stroke (La+AbvLine). However, this will cause errors when similar cases occur for letters that have no delayed strokes as in the other cases.



a) End form of “ha” ح.



b) The result of linear interpolation.



c) The result of ignoring 'Kashida'.

Figure 3.4 Internal discontinuity handling.

3.4 Feature Extraction

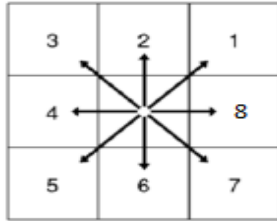
In this work, we propose a framework to investigate novel statistical features that are not used, to the best of our knowledge, by other researchers. We use different statistics of several raw features at different levels producing a large set of statistical features.

The raw features are the local features that are extracted at the points' level. Some of these features describe the geometric characteristics such as the axis coordinates, distance (horizontal, vertical, and Euclidean), the relative position, aspect ratio, curvature, slope, tangent, cosine, sine, angle with their differences. The temporal features are time, pressure, velocity, acceleration -w.r.t x, y, and both-, and their changes. In addition, the chain codes of writing direction (ingoing and outgoing) are extracted. Such features represent the raw data that is used to compute the statistical features.

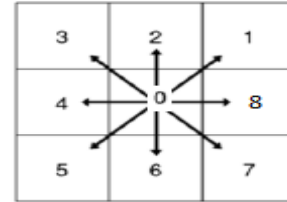
The statistical features are extracted by computing some statistics such as histogram, mean, mode, maximum, minimum, change (range), variance, and standard deviation from the raw point-based features for the sub-segments of the main strokes. The sub-segments are determined by the length and the overlapping parameters. In general, the statistical features represent global features that are based on the local ones. The global features are those extracted from a set of points that may constitute the whole stroke or sub-stroke (sliding segments of the stroke) like aspect ratio, bounding box, length. The used feature extraction method allows obtaining novel statistical features such as histogram of Freeman code, mean of tangent, mode of angles, variance of x distances, mean of Freeman code, histogram of cosine, variance of angles, mean of y distances, mode of

Freeman code, mean of sine, variance of x-axis, mean of curliness, max of Freeman code, variance of cosine, change of Y-axis, variance of curvature, variance of Freeman code, histogram of angles, mean of relative X-axis, mean of acceleration, mean of angles, change of relative Y-axis, and mean of aspect ratio. More details on statistical features extraction can be found in Appendix B and Appendix C.

The manner of obtaining the statistical features has several advantages besides using them as global features. It performs preprocessing operations, can be used as a codebook and enhance the extracted features (e.g. mean of writing direction). For example, the Freeman Chain Code feature is extracted for each point as shown in Figure 3.5.a. To capture the emphasis nature of writing we use a modified Freeman Chain code as shown in Figure 3.5.b. The modified code can be used to represent the writing direction without removing the duplicated points. For instance, if the original input is as shown in Figure 3.6.a, then the result of the classical preprocessing is shown in Figure 3.6.b and the result of taking the direction code after preprocessing is 4s. This segment is then represented as one writing movement as in [104], or as a primitive shape in [89]. However, the whole set of these operations can be alternated by simply taking the mode of the raw direction code feature extracted from the original input as shown in Figure 3.6.c, i.e. $\text{mode}([5,4,4,4,3,5,4,4,4])=4$.

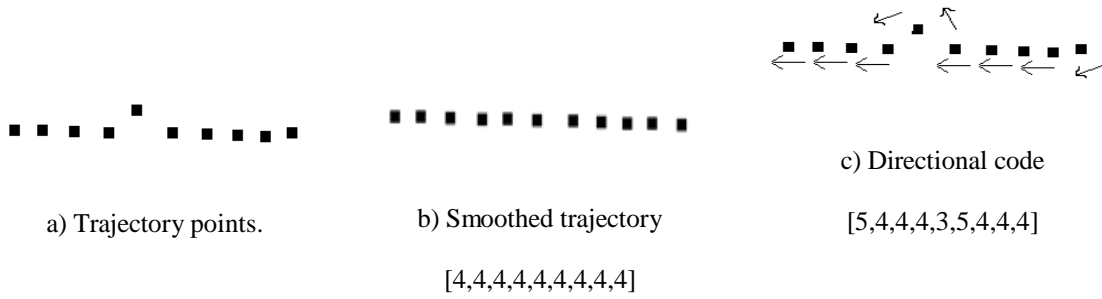


A. Freeman Chain Code



B. Modified Freeman Chain Code.

Figure 3.5 Freeman Chain Code.



a) Trajectory points.

b) Smoothed trajectory

c) Directional code

[5,4,4,4,3,5,4,4,4]

[4,4,4,4,4,4,4,4,4]

Figure 3.6 Statistical Directional feature extraction.

3.5 Classification

The classification phase consists of three main processes. The first process is the identification of the delayed strokes. The outcome of this process (secondary models candidates) is used for reducing the main stroke models. In the second process, the input main stroke is classified against the reduced main models. Finally, the results of the classification of both main stroke and delayed strokes are combined to obtain the overall result of the character classification.

For secondary stroke identification, different classifiers are used such as K-nearest neighbor's (KNN), Bayes network (Bnet) and Dynamic Bayes Networks (DBN). However, the main strokes classification is carried using DBN-based HMM classifier. The parameters of each classifier are chosen empirically as will be detailed in the experimental results section. The success of using HMM for speech recognition and the similarity of the digital representations of both speech and online text and its robustness to the intra-variability motivate us to use HMM classifier.

After the classification is performed on the main and delayed strokes, we have a set of main model candidates and another set of secondary strokes model candidates. Each set has a corresponding set of likelihood values. The final recognition result is a set of character candidates, each identified by both main and secondary stroke models. For those characters having no secondary strokes a virtual secondary model is used to refer to none. The likelihoods of those character candidates are computed according to the corresponding main and secondary candidates. Let $M = \{M_i\}_{i=1}^m$ be the set of the main

models' candidates from the main classifier and $S = \{S_j\}_{j=1}^l$ is the secondary models' candidates from the secondary classifier where the corresponding likelihoods are computed, normalized, and denoted as $ML = \{ML_i\}_{i=1}^m$, and $SL = \{SL_j\}_{j=1}^l$, respectively, then the likelihoods of the characters' classes candidates $CL_{i,j}$ are computed as in Equation 1 where λ is a scalar factor of the weight of the secondary candidates.

$$CL_{i,j} = ML_i + \lambda(SL_j) \quad (1)$$

The non-zero character likelihood candidates are denoted as $C_{i,j}$ and refer to the character class that has the main model M_i and the secondary object S_j . The selected class is the one having the maximum aggregative likelihood, i.e. $C_{i,j}$ where,

$$\widehat{C_{i,j}} = arg \{ \max_{i,j} CL_{i,j} \} \quad (2)$$

The grouping of the letters based on their basic shape is usually performed regarding their isolated forms. However, considering the position-dependent shapes provides a more useful clustering methodology. For example, the beginning and middle basic forms of the letters ن, ي are similar to their corresponding forms of the commonly clustered letters ب, ت, ث. This reduces the number of the used classification models and simplifies the main stroke classification process.

Intra-variation is the variability of the samples that belong to the same class. This variation can originate from the Arabic text styles (e.g. Naskh, Ruqaa) or from the handwriting process itself. This variation can be left to be handled implicitly by the used classifier or can be addressed explicitly when modeling the desired classes. In this work, the two approaches are combined to address the intra-variability. For some characters,

different classification models are considered when there is a significant variability regarding the writing styles (e.g. “س” vs. "س") or when it can be represented by a different number of strokes (e.g. “Kaf”). On the other hand, variations that are subject to the handwriting process itself such as size and direction are handled by the used statistical classifier. A semi-automatic methodology is adopted to extract the classification models for each character and to annotate the delayed strokes. The training samples of each character are utilized for building its model based on the global features representing strokes’ number, position, and dominating writing direction.

3.6 Experimental Results

The experimental work includes data preparation, data labeling, modeling, and finally testing the integrated system and analyzing the results. The main and secondary strokes are classified separately, then the results of the two classifiers are combined to identify the input character. The used dataset is a subset of the segmented characters of the Online-KHATT database [125]. The database consists of 10,040 lines of online Arabic text written by 623 writers using Android- and Windows-based devices. The characters’ samples were acquired by segmenting online Arabic text into the corresponding characters. Besides the position-dependent character labeling of the samples, a semi-automatic stroke-based annotation is performed. For evaluation, we split the data into 70%, 15%, and 15% training, validation, testing sets, respectively. The statistics of this dataset are shown in Table 3.1. More details on Online-KHATT can be found in Appendix A.

Table 3.1 Statistics of Online-KHATT segmented characters.

	Isolated	Begin	Middle	End	Total	Writers
Training	4013	4101	3158	4298	15570	48
Validation	908	994	727	1033	3662	13
Testing	946	942	686	989	3563	12
Total	5867	6037	4571	6320	22795	73

In the first stage of the experimental work, the training and validation datasets are used in intensive experiments to select the suitable parameters (viz. preprocessing, feature extraction, and classification settings). In this stage, we addressed single and multi-strokes' classification and generated models for both basic shapes and delayed strokes. We utilized single-strokes' models for multi-strokes' classification. The different parts and modules are integrated to form the desired recognizer.

There are 28 basic shape classes each has at least 100 training samples and 30 validation samples and the corresponding datasets contain 11801 training samples, 2880 validation, and 2565 testing samples. To conduct a balanced evaluation, in each experiment we pick randomly 100 samples for training and 30 samples for validation and averaged the results of each class. However, the testing is performed on the all available test data for the considered classes.

Our system is implemented using Bnet Matlab tool version 7, developed by Kevin Murphy [126] and acquired from the link: <https://code.google.com/p/bnt>. Several preprocessing operations are examined in different orders and the best results are obtained when applying simplification, smoothing, interpolation, then resampling. We extracted 103 features (some unreasonable features are ignored, e.g. max and min X-Y

coordinates when no size normalization pre-processing is performed). Due to the large number of features, it is difficult to examine all possible combinations (2^{103} combinations). Hence, selected combinations are examined and principle components analysis (PCA) approach for features selection is used in the analysis. We extracted several features from the online points trajectory input and statistical features from a sliding window of the input. Table 3.2 shows the results achieved using different features where RR refers to the recognition rate and Top3 is the accuracy of retrieving the desired output among the first three candidates. We used one of the most used features in the literature (i.e. the local writing direction represented by cosine and sine trigonometric functions). This feature is used in combination with other features as in [93], [118] and it is also used alone as in [91]. We computed also the average of the local direction in a sliding window. The results show that the statistical feature outperforms the points based feature. Hence, we adopted the statistical features in the other experiments. The following table shows some of the obtained results using the validation set.

Table 3.2. Some results of basic shapes classification using the validation set.

Features	RR	Top3
Local writing direction Cos; Sin	32%	52%
Mean(Cos); Mean(Sin) from a sliding segment window	54%	83%
In_FreeMan (Histogram, Mode, Mean, Variance); X (Change); Y(Change); Curvature(Mean)	56%	85%
In_FreeMan (Histogram, Mode, Mean) Curvature (Mean).	57%	84%
Cos (Mean); Sin (Mean); In_FreeMan (Mean); Out_FreeMan(Mean)	57%	85%
PCA (10 features)	61%	86%
Cos (Mean); Sin (Mean); In_FreeMan (Mean, Histogram); Out_FreeMan	66%	

(Mean); Curvature (Mean) RX (Mean); RY (Mean)		88%
PCA (15 features)	73%	92%





The best results are obtained when using PCA feature selection method. In this experiment, we used 103 statistical features (a 138-dimensional feature vector). We applied the PCA method to select 15 components (several values are tried). The other parameters are selected empirically. The statistical features are extracted using a sliding window of size 18 points with 6 points overlapping. The numbers of hidden states and Gaussian mixtures are fixed for all models at 15 and 64, respectively.

In the classification of the delayed strokes, simpler classification methods are used. We used different classifiers such as K-nearest neighbor's (KNN), Bayes network (Bnet) and Dynamic Bayes Networks (DBN) with several features. The best result is 92% when using KNN classifier with delta x, delta y and aspect ratio features.

A recognition rate of 82% is obtained on a test set of 2565 samples that are not uniformly distributed as some characters are more frequent and some characters have no test samples (e.g. se-B “سـ” and ya-I “ي”). However, they are shown in the confusion matrix (Table 3.3) because samples of other characters are recognized as those characters. More details on character recognition can be found in Appendix D.

On the analysis of the confusion matrix we note the following:

- The inclusion of the delayed strokes resulted in resolving several confusion cases of the basic shape classifier. Using the delayed strokes position and count distinguishes between characters with the same basic shapes such as “fa-B” ف and “wa-I” و as shown in Figure 3.7.

- The accuracy of the complete recognizer depends on the performance of the basic classifiers. There are some errors that are due to basic shapes misclassification like recognizing ha-B as da-I. There are errors in the classification of the delayed strokes which lead to wrong character recognition. For example, an error in detecting the delayed stroke in the ya-M “”  results in recognizing it as th-M “”. Several classification inter-errors in the characters ba, ta, th are caused by errors in identifying the delayed strokes. These results from writing the two and three dots in one stroke. For example, the merged two dots sample  is recognized as three dots.
- Segmentation-based intra-errors come from confusion between different positional forms of the same character. The beginning (isolated) form is similar to the middle (end) form for most characters as the main difference is a connecting stroke called "Kashida". Examples of such errors are shown in Figure 3.8. These errors are resolved by combining the codes of the same characters and the shape of the different positions of the characters are automatically addressed by contextual processing of the editors.
- Segmentation-based inter-errors are those errors caused by the confusion between different positional classes of different characters that seem to be similar after segmentation. Examples of such errors are shown in Figure 3.9.
- Errors due to writing distortion come from the confusion that is originated from the writing quality which can be influenced by some variations such as writing movements' direction (e.g. ma-B vs. ha-B) or curvatures (e.g. da-E vs. ra-E). Examples of these errors are shown in Figure 3.10.



a) The character fa-B “ف”



b) The basic shape of fa-B confused with the wa-I “و”

Figure 3.7 Characters with similar basic shapes but different delayed strokes.



a) Letter (ha) “ح” intra-errors, left: a middle sample (ha-M) “ح” recognized as begin (ha-B) “ح”,
 right: a begin sample (ha-B) “ح” recognized as middle (ha-M) “ح”



b) Letter (ra) “ر” intra-errors, left: an isolated sample (ra-I) “ر” recognized as end (ra-E) “ر”,
 right: an end sample (ra-E) “ر” recognized as isolated model (ra-I) “ر”



c) Letter (wa) “و” intra-errors, left: an isolated sample (wa-I) “و” recognized as end (wa-E) “و”,
 right: an end sample (wa-E) “و” recognized as isolated model (wa-I) “و”

Figure 3.8 Examples of segmentation-based intra-errors.



a) da-I “د” as ha-B “ح” b) wa-I “و” as ma-B “م”

Figure 3.9 Examples of segmentation-based inter-errors.



a) gh-M “غ” as fa-M “ف” b) ra-E “ر” as da-E “د”

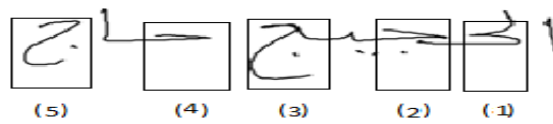
Figure 3.10 Examples of errors originated from writing distortion.

Segmentation-based intra-errors account for 116 errors (i.e. about 25% of the total errors) and segmentation-based inter-errors account for about 210 errors (i.e., 45% of the total errors). Such errors are reasonable and are due to the high similarity between the confused characters and they disappear when computing top-3 recognition rate. The rest 30% of errors are originated from writing distortion.

Some difficulties may arise when extracting the characters' samples from the cursive text manually. The determination of the segmentation points and the implication of parts of the different strokes are some examples. Let us take as an example the input in Figure 3.11.a. Some of the characters that should be extracted after segmentation are shown in Figure 3.11.b:(1) ha-M, (2) ja-M, (3) ja-E, (4) ha-B, (5) ja-I. One segmentation can be seen as in Figure 3.11.c (in order). This segmentation is most probably obtained when using segmentation similar to those adopted in the offline case where the spatial features are utilized to extract rectangular segmentation which makes the segmentation points as the red ones in the in Figure 3.11.d. However, in the online case the segmentation can be done based on the temporal trace trajectory of the points which may lead to segmentation points as colored in blue, and in turn gives different segmentation for some characters as shown in Figure 3.11.e. This makes the segmented character of (1) ha-M be similar to (4) ha-B and (3) ja-E similar to (5) ja-I.

الحجيج حاج

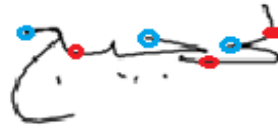
a) Cursive text input.



b) Some desired segmented characters (1) ha-M, (2) ja-M, (3) ja-E, (4) ha-B, (5) ja-I

ح - ح - ح - ح - ح

c) Possible extracted characters.



d) Two possible segmentation points (one in red and another in blue).

ح - ح - ح

e) Different segmentation for some characters.

Figure 3.11 Example of confusions caused from the segmentation.

CHAPTER 4

ARABIC ONLINE HANDWRITTEN TEXT

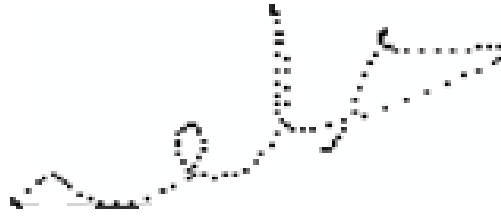
RECOGNITION

In this chapter, we address Arabic online cursive text recognition at three levels. In the first we utilize the DBN-based HMM character classifier to recognize the characters and then combine them for cursive text. The second one is to classify the PAWs using DBN-based HHMM recognizer. Thirdly, a segmentation-free recognition online handwritten cursive text recognition using HMM classifier is adopted.

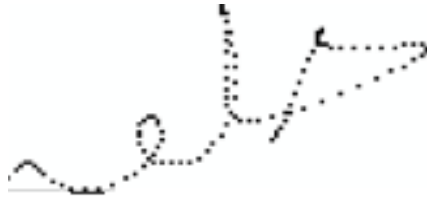
4.1 Cursive Input Pre-Processing

The preprocessing of cursive text input consists of several operations. Firstly, geometric operations are performed to obtain stroke-based enhancements. The results for implementing some of the geometric preprocessing operations are shown in Figure 4.1. The second stage is to detect the delayed strokes. For this purpose, we identify the main and secondary strokes. Unlike the segmented characters case, it is useful to utilize the baseline detection in cursive text recognition. Using the histogram method for baseline detection may not be adequate in some cases (see Figure 4.2) because of the baseline shifts and undulations, the inter-line distance variability and the baseline-skew variability. In addition, the diacritics and holes in the characters may yield false maximums and false minimums, respectively. Hence, we adopt expectation–maximization (EM) method

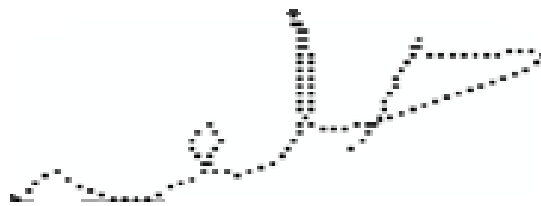
proposed by [127] for baseline detection in the offline case. An example of its application on online input is shown in Figure 4.3. Once the baseline is detected, an initial set of the main strokes is considered to be all strokes that intersect the baseline. The other strokes are initially grouped in the secondary strokes set. This set is indicated by the strokes surrounded by circles and rectangles as shown in Figure 4.4. Some of these strokes are correctly detected (indicated by circles) but there are other strokes that are incorrectly detected like those indicated by rectangles. Hence, an extra filtering step is performed by examining the overlapping and the size (i.e. each secondary stroke is compared to the non-secondary strokes and it is considered as main stroke if there is no overlapping with other strokes with respect to the x-axis).



a) An image of the input



b) Average smoothing



c) Writing speed normalization.

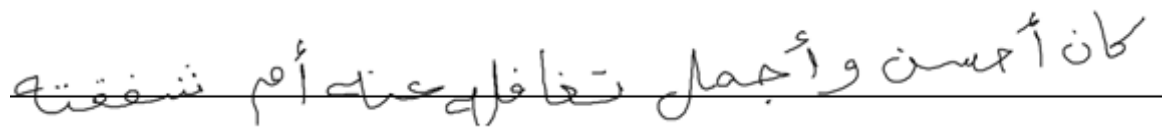


d) One pixel equidistant resampling



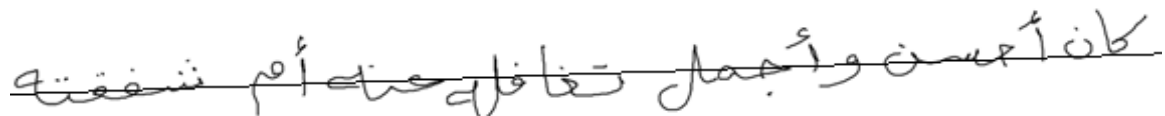
e) Simplification then minimum equidistant resampling

Figure 4.1 Preprocessing operations outputs.



كان أحسن وأجمل تغافل عنه أم شفقتك

Figure 4.2 Histogram-based baseline detection on a skewed text.



كان أحسن وأجمل تغافل عنه أم شفقتك

Figure 4.3 EM-based baseline detection on a skewed text.



كان أحسن وأجمل تغافل عنه أم شفقتك

Figure 4.4 Initial set of secondary strokes, correct (circle), incorrect (rectangle).

4.2 Classification

In this section, the methods of cursive text recognition are described.

4.2.1 HMM

An HMM is a stochastic finite automaton that is denoted in the first order by λ and defined by the triple (π, A, B) , where π is the vector of the initial state probabilities, A the state transition matrix, and B the observation probability distribution:

- $\pi(i) = \{\pi_i | \pi_i = P(S_1 = i)\}$
- $A = \{a_{ij} | a_{ij} = P(S_t = j | S_{t-1} = i)\}$
- $B = \{b_j(o_k) | b_j(o_k) = P(O_t = o_k | S_t = j)\}$.

Figure 4.5 shows a simple HMM Bakis model with five states. The three internal states are emitting states and the first and last states have no output probability distributions. The dimension of the transition matrix for this model is 5×5 . Each row sums to one except for the final row which is zero since no transitions are allowed out of the final state. For discrete HMM, a codebook of the output of the input feature vector quantiser is used. When the observations are continuous, each observation probability distribution can be represented by a Gaussian mixture. Reference may be made to [128] for more details on HMM.

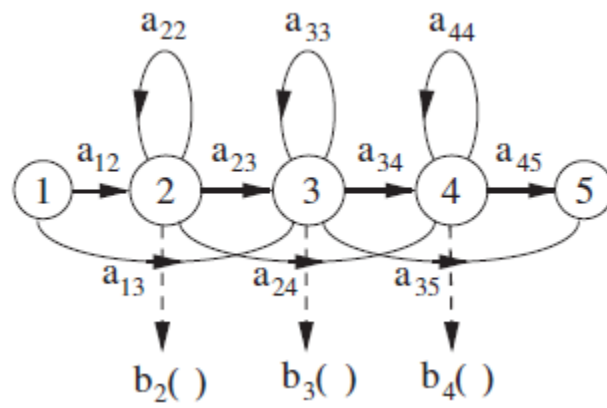


Figure 4.5 Simple HMM Bakis model with five states [128].

4.2.2 Dynamic Bayesian network (DBN)

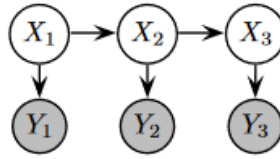
Bayesian networks (BNs) or belief networks, also known as probabilistic networks (PNs) are representations of domains involving uncertain relations among a group of random variables. The extension of Bayes nets can be done using Dynamic Bayesian network (DBN) [129] to model semi-infinite collections of random variables, Z_1, Z_2, \dots . The partition of the variables are $Z_t = (U_t, X_t, Y_t)$ representing the input, hidden and output variables.

A DBN is a pair (B_1, B_{\rightarrow}) , where B_1 is a BN which defines the prior $P(Z_1)$, and B_{\rightarrow} is a two-slice temporal Bayes net (2TBN) which defines $P(Z_t | Z_{t-1})$ by means of a DAG (directed acyclic graph) as follows:

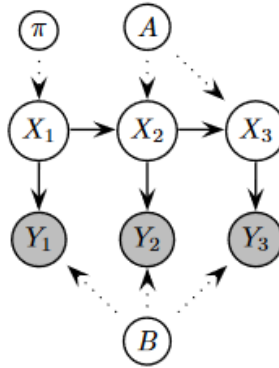
$$P(Z_t | Z_{t-1}) = \prod_{i=1}^N P(Z_t^i | \text{Pa}(Z_t^i))$$

where Z_t^i is the i^{th} node at time t , which could be a component of X_t, Y_t or U_t , and $\text{Pa}(Z_t^i)$ are the parents of Z_t^i in the graph.

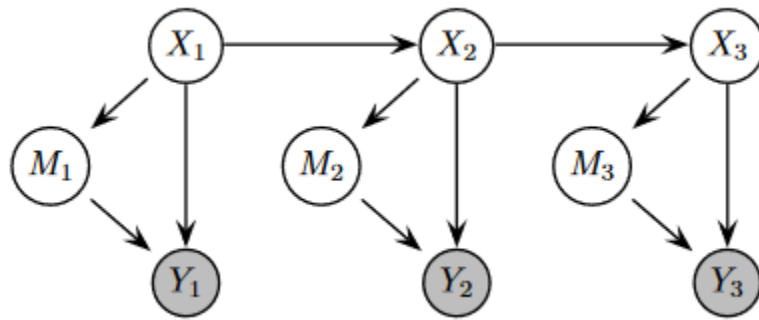
An HMM can be represented as an instance of a DBN unrolled for 3 slices as shown in Figure 4.6.a where the shading refers to observation nodes, clear nodes are hidden, and the arcs represent the assumptions: $X_{t+1} \perp X_{t-1} | X_t$ (the Markov property) and $Y_t \perp Y_{t'} | X_t$, for $t' \neq t$. Figure 4.6.b shows an HMM in which the parameters are explicitly represented. The parameters are $P(X_1=i) = \pi(i)$, $P(X_t=j | X_{t-1}=i) = A(i,j)$, and $P(Y_t=j | X_t=i) = B(i,j)$. If the conditional probability distribution (CPD) for Y is a Gaussian, the B node can be replaced with the mean and covariance or with a Gaussian mixture matrix M as in Figure 4.6.c. Reference may be made to [126] for more details on DBN.



a)



b)




c)

Figure 4.6 A DBN representation of HMM [126].

4.2.3 Segmentation-based Cursive Text Recognition

In this approach, the designed non-cursive character recognizer is utilized in cursive text recognition. The adopted methodology is shown in Figure 4.7. The writing trajectory is traced and different segmentations are extracted using sliding window method. The features are then extracted from those segments and fed into the character classifier to return list of character hypothesis. The character hypothesis are used to generate paths of words representing word hypothesis. Finally, the word paths are scored and ranked to come up with a final ordered list.

Several methods are employed in this approach. The time-dependent features (velocity and pressure) are used for initial hypothesis generation. Global features are utilized for hypothesis reject such as: variability in direction, curvature, and dimensions. For example segments like  which is a sample of the basic shape of middle “na” ن has no chance to be compared with the middle basic shapes of several models such as “sa” ص, “ma” م, “la” ل. Statistics from the training data are considered when generating the candidate paths such as: max, min, and mean of the training dataset of positional-based segmented samples, the maximum length of the Arabic PAWs and words.

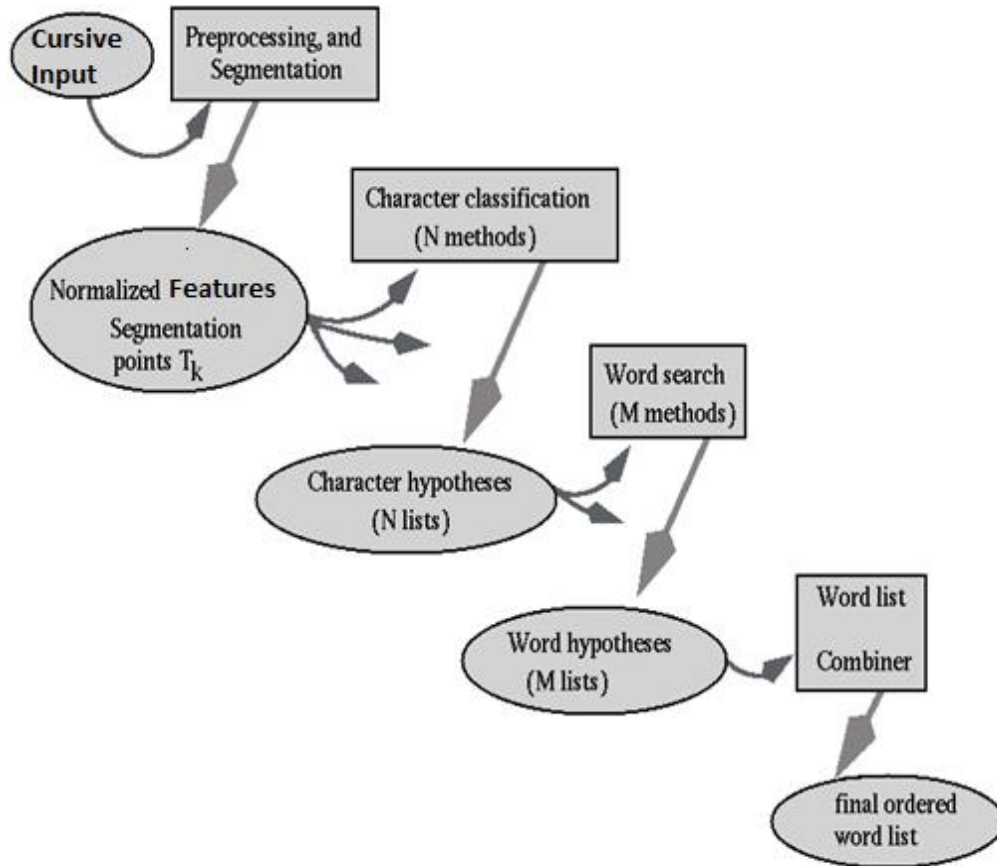


Figure 4.7 Segmentation-based Cursive Text Recognition Approach (adopted from [130])

4.2.4 DBN-based Hierarchical HMMs (HHMMs)

The Hierarchical HMM (HHMM) [131] is an extension of the HMM that is designed to model domains with hierarchical structure and/or dependencies at multiple length/time scales. In an HHMM, the states of the stochastic automaton can emit single observations or strings of observations. Those emitting single observations are called “production states”, and those emitting strings are termed “abstract states”. The strings emitted by abstract states are themselves governed by sub-HHMMs, which can be called recursively. When the sub-HHMM is finished, control is returned to where it was called from; the calling context is memorized using a depth-limited stack. Based on [132], a dynamic Bayesian net for modelling the writing of a PAW is shown as in Figure 4.8. Q^h is the state (position) in the PAW HMM; Q is the character; S is the state (position) within the character HMM, Y is the acoustic vector. F^s is a binary indicator variable that turns on when the character HMM is transmitted. Figure 4.9 shows a DBN modeling instance for the PAW لـما (Lma) written using three characters: /la/-/ma/-/aa/. It uses the deterministic variables “Position” and “Character”, and the stochastic variables “Transition” and “Observation” as follows.

- **Position** refers to the current position in the PAW model. It takes values $1, \dots, N$, where N is the maximum length of a PAW model.
- **Character** refers to which character is associated with the current Position.
- **Transition** refers to whether a transition is being taken out of this character using only two possible values: 1 or 0.
- **Observation** refers to the acoustics online PAW input. In the case of multiple acoustic streams, it can be replicated for each stream for each time frame.

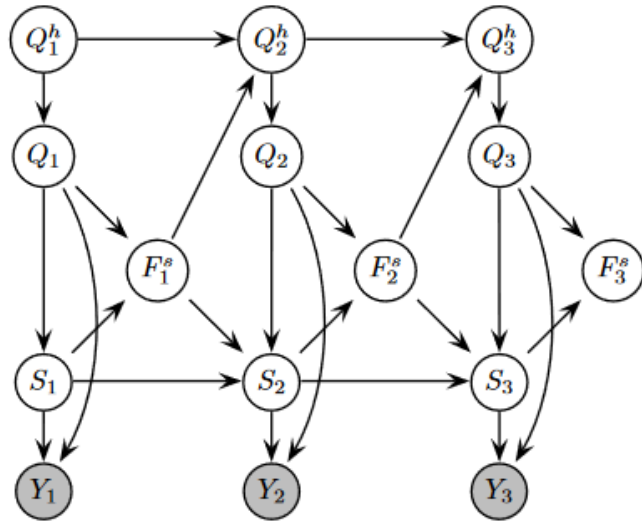


Figure 4.8 A DBN for modelling a PAW.

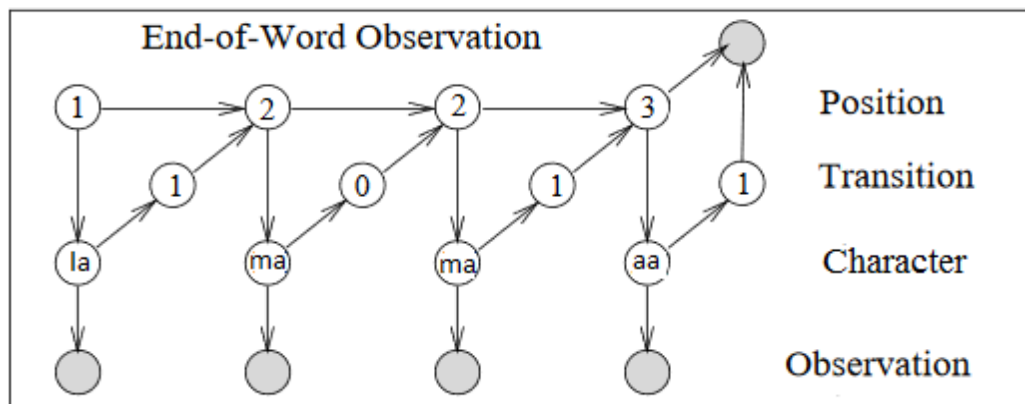


Figure 4.9 DBN-HHMM modeling the PAW "لما" (LMA).

4.2.5 Segmentation-free Cursive Text Recognition Using HMM-HTK.

The principal components of a cursive online text recognizer are illustrated in Figure 4.10. The input from an online device is converted into a sequence of fixed size acoustic vectors $Y_{1:T} = y_1, \dots, y_T$ in a feature extraction phase.

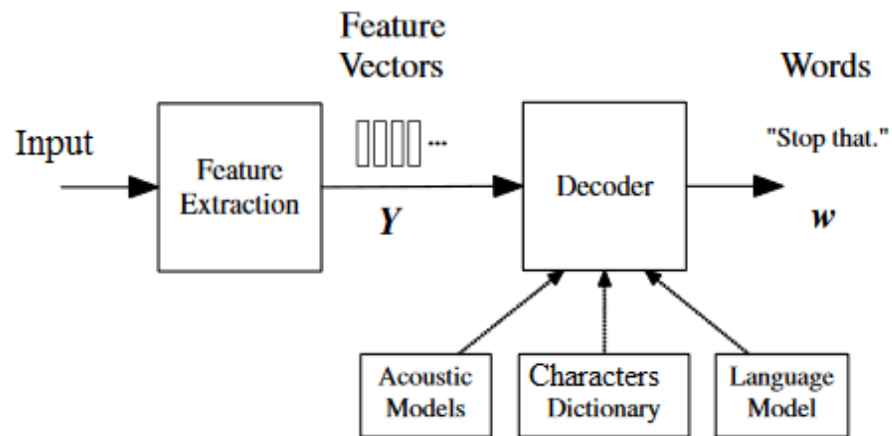


Figure 4.10 Architecture of a cursive online text HMM-based Recogniser.

HTK is a toolkit for building Hidden Markov Models (HMMs) involving two main stages. The first stage is the training process which uses training utterances and their associated transcriptions to estimate the parameters HMMs' models. Secondly, the HTK recognition tools are used to recognize (transcribe) unknown utterances. HTK is initially developed for speech recognition. However, it is used for several other tasks like handwriting recognition as we do in this work. Reference may be made to [128] for more details on HMM-htk.

4.3 Experimental Results

To evaluate the recognition of unconstrained cursive text we use Online-KHATT database. We started by a subset of segmented cursive units extracted from Online-KHATT database for evaluating the proposed methods. In this dataset, the cursive text is segmented using a semi-automatic method. The available dataset contains 4,814 letter samples, 1123 PAW samples and 731 words' samples. Samples of this dataset are shown in Figure 4.11. Several experiments have been conducted, however, some of the achieved results with different statistical features are shown in Table 4.1.

Table 4.1 Results of using some statistical features for cursive text recognition on Online-KHATT database.

Features	SB	HHMM	HTK
Average of Freeman Code	37.1%	47.5%	54.7%
Averages of Sine and Cosine	40.3%	49%	53.2 %
Average of Relative Position	37.7%	47%	53.5%
Average of Curvature	38.7%	40%	53.7%
Average of Curliness	38.2%	39.6%	52.6%
Histogram of Freeman Code+ Variance of Freeman Code	41.3%	53.5%	51.4%
Histogram of Freeman Code +average of Curliness +averages of Sine and Cosine	51.3%	57.3%	59.6%
Histogram of Freeman Code	45.6%	60.3%	51.3%
Mode of Freeman Code	42.3%	50%	50.7%
Variance of Freeman Code	33.3%	47.8%	51.7%
Average of Tangent	32.2%	43.1%	33%
Average of Velocity	27%	36.6%	36.1%
Average of Acceleration	27.9%	21.3%	35.3%
Averages of Sine, Cosine, Velocity, Acceleration, Freeman Code, Curvature	50.1%	53.3%	51.5%
Histogram of Freeman Code + Average of Freeman Code	54.3%	47.1%	66.6%

Due to the difficulty of examining all possible extracted statistical features, we used PCA method for feature dimension reduction. We used different dimensions and some of the achieved results are shown in Table 4.2 to compare the different classification methods. More details on cursive text recognition can be found in Appendix E.

Table 4.2 Results of using PCA statistical features for cursive text recognition on Online-KHATT database.

Features	SB	HHMM	HTK
PCA (10)	50.5%	44.2%	57.3%
PCA (15)	57.6%	60.9%	67.3%
PCA (20)	55.1%	54 %	61.7%

On the above cursive text recognition results we have the following comments.

- The accuracy is computed at the character level, i.e. character recognition rate within the cursive text.
- The method SB is segmentation-based cursive text recognition approach using DBN-HMM, the method HHMM is the DBN-HHMM, and HTK refers to the HTK-HMM method.
- DBNs are typical for temporal processes modeling with the advantages: nonlinearity, interpretability, efficiency, and extensibility [132].
- The segmentation-based recognition approach has some advantages. The extensive work on character recognition is utilized. The need for large verified cursive text dataset is avoided. The flexibility in improving the recognizer by employing some methods such as: position detection, alternative results, delayed strokes association, and reduction. However, there are several limitations of using segmentation-based cursive text recognition. One of these limitations is the time-consuming recognition

process. Moreover, the performance of the character recognizer impacts the cursive text recognizer. Hence, any problem of the character recognizer is propagated to the cursive case and improving the performance of the cursive text recognizer requires modeling the character recognizer which is a complicated process.

- To train the HHMM model, semi-supervised learning techniques are used. A semi-automatic method is used to label the hidden states in the first level and to set the values of transition flags among the sequent HMMs.
- When using each category of features separately, the statistics of the directional features are more descriptive than the other features and the statistical time-dependent features are the less descriptive features. This is due to the nature of the handwriting, that is, the time-dependent features are more related to the writer identification rather than to the handwritten text recognition.
- For evaluating the segmentation-based (SB) approach, the models are built in the training stage on the letter samples and the results are obtained on the cursive PAW samples in the testing stage.
- The DBN-HHMM approach may need more effort in a pre-training process. Each cursive PAW is considered as a separate model and hence it requires training, hence, its applicability is limited. Such approach is useful in small vocabulary application but not with open vocabulary applications.
- The DBN-HHMM approach can employ the HMMs trained in the segmentation-based method when constructing the cursive models by initializing the underline character HMM models. However, this scheme requires extra effort for pre-training and deterministic variables setting.

ع
سلسلة
على

Figure 4.11 Samples of Online-KHATT PAWs.

The above experimental work aims at evaluating the proposed methods for cursive text recognition and it is conducted on the available segmented data. The next step is to apply the recognition on cursive text lines. DBN-HHMM and SB approaches require the training data to be segmented in a pre-training process. Due to the lack of fully segmented text lines, we use the segmentation-free approach provided by htk toolkit [128] to conduct the experiments of text lines recognition. Table 4.3 shows some recognition results on Online-KHATT text lines using HTK with different statistical features. The results are obtained by using 700 lines for training, 150 validation lines, and 150 testing results where the parameters are chosen empirically. Samples of the Online-KHATT lines are shown in Figure 4.12.

Table 4.3. Some recognition results on Online-KHATT text lines using HTK with different statistical features.

Features	Correct	Accuracy
Average of Freeman Code	38.6%	27.5%
Averages of Sine and Cosine	34.3%	25.2%
Average of Curvature	25.%	17.2%
Histogram of Freeman Code + Variance of Freeman Code.	24.3%	16.7%
Histogram of Freeman Code + average of Curliness + averages of Sine and Cosine	35.5%	26.5%
Histogram of Freeman Code + Average of Freeman Code	36.1%	26.2%
PCA (10)	46.2%	34.4%
PCA (15)	50.1%	40.1 %
PCA (20)	50.2%	40.3%
PCA (25)	47.1%	36.1 %

Once the optimal alignment has been found, the number of substitution errors (S), deletion errors (D) and insertion errors (I) can be calculated. The percentage of correctness is then

$$\text{Percentage of correctness} = \frac{N - D - S}{N} \times 100\%$$

where N is the total number of labels in the reference transcriptions. Notice that this measure ignores insertion errors. For many purposes, the percentage of accuracy defined as

$$\text{Percentage of accuracy} = \frac{N - D - S - I}{N} \times 100\%$$

is a more representative figure of recogniser performance.

There are two difficulties when dealing with text lines, delayed strokes and the connectivity issues. To handle the delayed strokes, the features' vector is extracted in a manner that considers the upper and lower delayed strokes after detecting and associating them. This is done by firstly detecting and rearranging the delayed strokes then concatenating the features extracted from the delayed strokes with the features extracted from the main strokes using the sliding window technique. Let $F[1..N]$ be the features' vector then $F[1, N_1]$ is the vector of features extracted from the main stroke, $F[N_1+1, N_1+N_2]$, and $F[N_1+N_2+1, N]$ are the features' vectors extracted from the upper, and lower delayed strokes, respectively. Where, N_1 is the dimension of the features' vector extracted from the main stroke using sliding window, N_2 is the dimension of the features' vector extracted from the delayed strokes, hence $N = N_1 + 2N_2$ is the dimension of the

concatenated features' vector. When there is no upper (lower) delayed strokes then the corresponding vector is assigned to zeros. For the connectivity issue, a virtual stroke is used to connect the consecutive main strokes which is then considered when extracting the features to get a separating null feature vector using the adopted sliding window method.

من عنبر الكبريت الذي يؤدي احتراقه الى اطلاق

ذهب نوح مظفر خمر غام بصحبة رؤوف بن لؤي

كان أحسن وأجمل تغافل عنه أم شفقتة

Figure 4.12 Samples of the Online-KHATT lines.

CHAPTER 5

CONCLUSIONS AND FUTURE RESEARCH

DIRECTIONS

In this chapter, we conclude the work of the thesis and summarize its outcomes and then present future research directions.

5.1 Conclusions

Automatic recognition of Arabic online text has several applications like friendly learning environments, business applications, communication and more. There are many challenges facing research on Arabic online text recognition such as the lack of benchmarking databases, the cursive nature of Arabic text, overlapping characters and ligatures, and the delayed strokes' presence and variability.

In this thesis, research on Arabic online text recognition was conducted. Our approach is based on statistical techniques. Novel statistical features are presented and several methods and algorithms are proposed.

The contributions of this thesis can be summarized as follows.

- **A comprehensive survey of Arabic online text recognition:** This thesis provides a detailed literature review of the related research in the different phases of Arabic online text recognition. It presents critical comments, conclusions and future directions.

- **Unconstrained Arabic online character recognizer.** In this thesis, a recognizer for unconstrained Arabic online character was developed. The recognizer integrates the different phases of online Arabic character recognition. The system is tested with unconstrained online Arabic characters that are segmented from online text. This implies several challenges like delayed strokes handling, connectivity problems, variability, and style change of text.
- **Statistical feature extraction technique:** Another contribution of this work is the intensive investigation of several novel statistical features using a developed framework for generating different statistical features. The framework consists of two main components. Most of the point-based features (local features) that are found in the related literature are extracted. A statistical layer is then added to form statistical features. Those features can be at the level of points, sub-strokes, the whole stroke, or a combination of the different levels.
- **Delayed strokes handling approach.** We proposed and implemented an approach of handling delayed strokes which includes several methods to process the delayed strokes at the different phases of the recognition process to improve the overall performance.
- **Recognition of Arabic online cursive text.** The Arabic online cursive text recognition is addressed at three levels. In the first we utilize the DBN-based HMM character classifier to recognize the characters and then combine them to

cursive text. The second one is to the recognition at PAWs level using DBN-based HHMM recognizer. Thirdly, a segmentation-free recognition is addressed at cursive text handwritten lines using HMM classifier. The results show that the HMM approach outperforms the other approaches because it avoids the errors originating from the segmentation.

5.2 Future Research Directions

There are several extensions for future work to improve the performance of Arabic online text recognition.

- **Handling more delayed strokes.** In our work, we considered a number of delayed strokes that are used frequently in Arabic online text. Applications that deal with official documents or with holy texts such as QURAN and HADITH require the use of additional types of the delayed strokes.
- **Investigating the effect of writing styles.** This implies applying our techniques and possibly adding more features to address wider range of styles of writing Arabic text.
- **Using advanced classification techniques.** Other classification techniques such as adaptive training, deep learning may be investigated.
- Combination of statistical and syntactical features is a natural extension of the work which expected to improve the performance considerably.

- **Post-processing.** The recognition performance can be improved by adding a post-processing phase. Such phase may utilize lexicon/dictionary, language models, linguistic information, etc. Moreover, natural language processing techniques can be utilized to further improvement of the recognition performance.
- **Real-time recognition.** Unlike offline systems, online recognizers require real-time processing and more research is required in this area in which the trade-off between accuracy and computational complexity must be addressed carefully.

Given that our work is applied on unconstrained Arabic online text, the achieved recognition rates are acceptable. However, there is big room for improvements and more research is needed to enhance the current state of the art.

With this discussion, we conclude our thesis. All praises and thanks due to Allah who has helped us and permitted us to complete this thesis.

APPENDICES

Appendix A. Database details, transliteration of Arabic letters

This section presents some details of Online-KHATT database. Table 6.1 shows statistics of Online-KHATT database. Figure 6.1 shows samples of text from Online-KHATT. Table 6.2 shows Transliteration codes for Online-KHATT database [125].

Table 6.1 Statistics of Online-KHATT database

Set	Word counts	Character counts	Line counts	Unigrams	Bigrams	Trigrams
Training	56950	564241	6996	19775	47464	55730
Validation	12004	126125	1482	4635	9998	11530
Testing	12205	127196	1645	5450	10561	11839
Whole Database	81159	817562	10123	29860	68023	79099

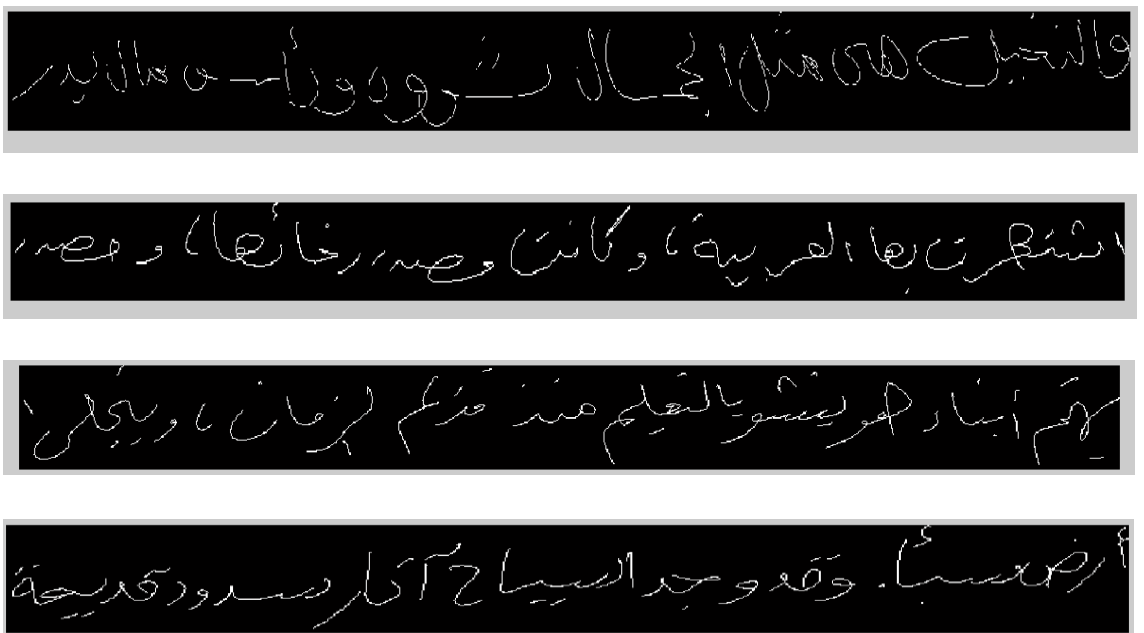


Figure 6.1 Samples of text from Online-KHATT.

Table 6.2: Transliteration codes for Online-KHATT database [125].

Arabic Char	ء	أ	أ	إ	ا	ب	ت	ة	ث	ج	ح	خ	د
Transliteration	hh	Am	ae	Ah	aa	ba	ta	tee	th	ja	ha	kh	da

Arabic Char	ذ	ر	ز	س	ش	ص	ض	ط	ظ	ع	غ	ف	ق
Transliteration	dh	Ra	za	se	sh	sa	de	to	zha	ay	gh	fa	ka

Arabic Char	ك	ل	م	ن	ه	و	ؤ	ي	ى	ئ
Transliteration	ke	la	ma	na	he	wa	wl	ya	ee	al

Symbol	0	1	2	3	4	5	6	7	8	9	@	:	"
Transliteration	n0	n1	n2	n3	n4	n5	n6	n7	n8	n9	atr	col	dbq

Symbol	,	؛	،	?	!	.	()	/	\	=	-	_
Transliteration	com	com	com	qts	exc	dot	bro	brc	fsl	bsl	equ	hyp	usc

Symbol	#	%	Blank space
Transliteration	Scr	Per	Sp

Diacritics	◌ُ	◌َ	◌ِ	◌ِ	~	◌ُ	◌ِ	◌ِ
Transliteration	D	H	K	F	X	B	N	Z

Appendix B. Statistical Framework GUI Guide

In our experiments, the first step is preparing the required settings such as data labels, dictionary, datasets' separation. The GUI shown in Figure 6.2 is developed to prepare the experimental environment including several modules such as:

- Organizing the characters' labels and the relation between them: models for basic shapes without delayed strokes and others with dots.
- Modules for the training of the basic models and rules of relating them with delayed strokes to get the appropriate character model.
- Modules for detecting, separating, and connecting the main strokes and the secondary strokes.
- Modules for recognizing samples with or without delayed strokes.
- Since our method requires handling the delayed strokes and labeling the main shapes we implemented a module for annotating the data.

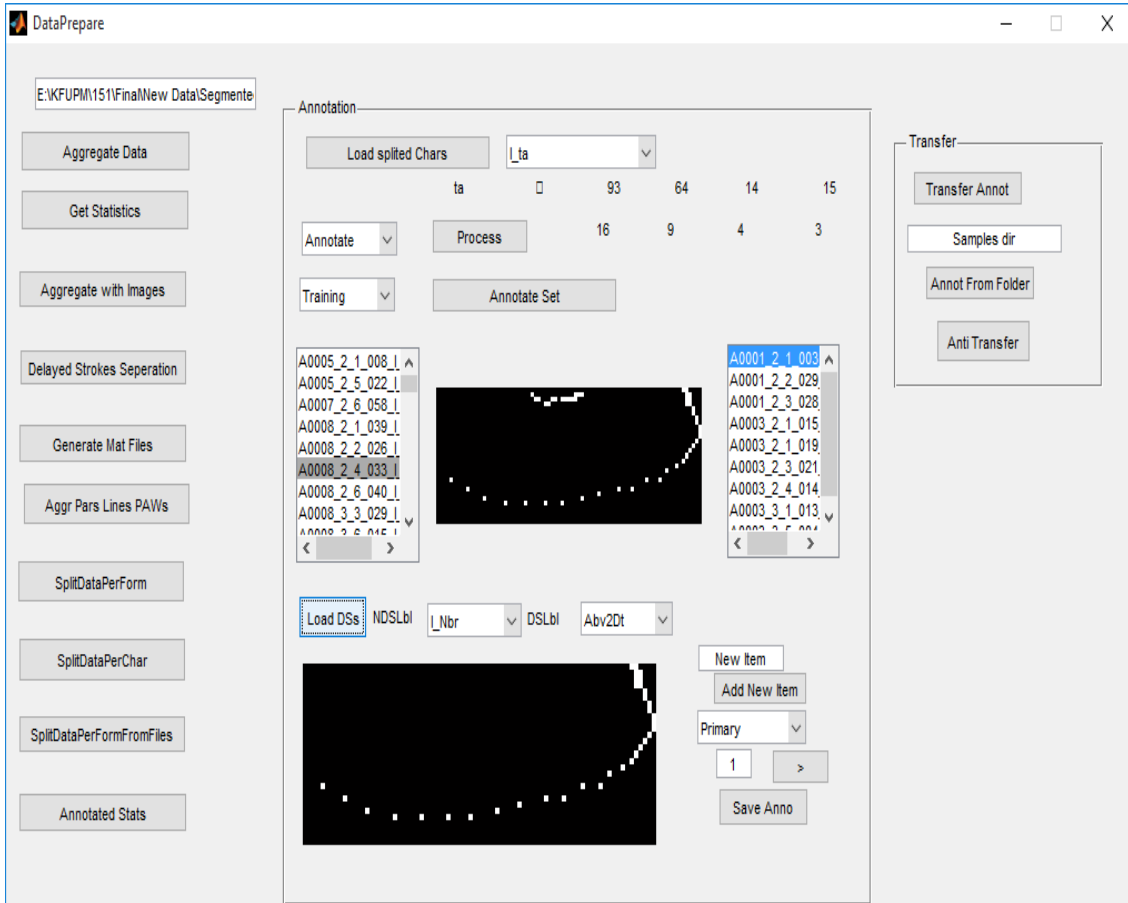


Figure 6.2 Preparing GUI.

The details of the outcome of this task are organized in excel and then converted to mat files for experimental purposes. Samples of the setting files are shown in the following tables. Table 6.3 shows mappings relate characters, models, and secondary objects. Table 6.4 shows mappings relate delayed strokes models and secondary objects.

Table 6.3 Mappings Relate Characters, Models, and Secondary Objects.

ClssLbl	LtrLbl	DSsLbl
I_aa	I_aa	NDS
I_aa	I_ae	AbvHmz
I_Nbr	I_ba	Blw1Dt
I_Nbr	I_ta	Abv2Dt
I_Nbr	I_th	Abv3Dt
M_Nbr	M_na	Abv1Dt
M_Nbr	M_ya	Blw2Dt
M_Nbr	M_al	AbvHmz

Table 6.4 Mappings Relate Delayed Strokes Models and Secondary Objects.

DSsLbl	DSsCnt	DSsPos	SubDSs
NDS	0	0	NDS
Abv3Dt	1	1	Mrg3Dts
Abv3Dt	2	1	Mrg2Dts,Dot
Abv3Dt	3	1	Dot,Dot,Dot

Then, we designed a GUI shown in Figure 6.3 that can be used to manage the proposed classifier by examining different parameters of the experiments. This GUI allows the following operations.

1. Getting Data

Choosing the training and testing data from the prepared data folder. They contain training and testing samples for the same classes.

2. PreProcessing

Showing the available preprocessing methods and selecting the needed methods and their order. The implemented operations are Simplification, Smoothing, Interpolation and Resampling.

In the Preprocessing panel,

- a. Highlight a method in the available methods list then click the button (>) to select it.
- b. Highlight a method in the selected methods list then click the button (delete) to de-select it.
- c. Click the button PreProcessing to apply the selected methods with the same listed order on both training and testing which produces mat formatted files of preprocessed data.

3. Feature extraction:

In the Statistical Features panel,

- a. Select the features category from the available combo list and the corresponding features will be listed in the available list
- b. Highlight a feature in the available features list and statistics from the statistics list

- then click the button (>) to select them.
- c. Highlight a feature in the selected features list then click (delete) to de-select it.
 - d. Click the button Feature Extraction to extract the selected features with the same listed order on both training and testing samples generate mat formatted files of the extracted features data.
 - Statistical features describing the raw data: select Stroke length or geometrical features X and Y and choose the required statistics.
 - The sliding window length and the overlapping
 - To use the original feature value, choose length=1, overlapping=0, and the statistics=max, min, or mean.
 - To use the whole stroke, we choose length=0.
4. DBN initialization. In the Classification Initialization panel,
- a. Set the number of hidden nodes (NofHSs) and the number of Gaussian Mixture models (NofGMs).
 - b. Click the button “Initialization” to initialize the DBN classifier by the stated parameters and format the training and testing data to generate mat files that are formatted according to the DBN settings.
 - c. Classifier initialization
(Number of observation nodes, Slicing time T and inference engine, Features Length)
5. Training
- In the Classification Training panel, set the maximum number of training iterations.
6. Testing Results

The classification results for each class and for all classes. All of the above experiment parameters are shown and can be saved in both .mat, and .xls formats to be used in the analysis process.

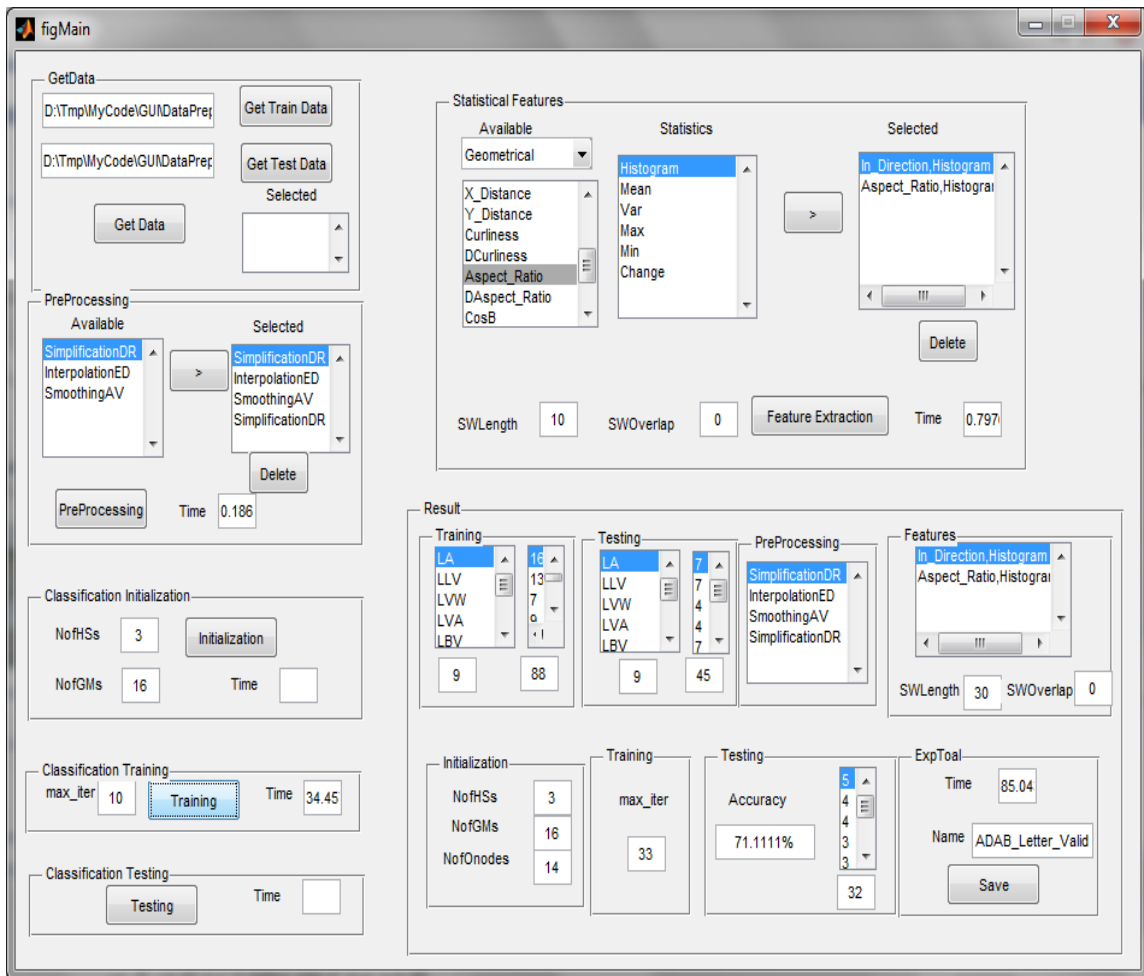


Figure 6.3 A gui for Statistical Character Recognizer.

Appendix C. Statistical Feature Extraction

We have the raw features: {'In_Direction', 'Out_Direction', 'Delta_Direction', 'Tanget', 'DTanget', 'Angle', 'DAngle', 'CosA', 'SinA', 'DCosA', 'DSinA', 'X', 'Y', 'RX', 'RY', 'DX', 'DY', 'E_Distance', 'X_Distance', 'Y_Distance', 'Curliness', 'DCurliness', 'Aspect_Ratio', 'DAspect_Ratio', 'CurvatureA', 'CurvatureB', 'DeltaAccelaration', 'Presure', 'DeltaPresure'};

And the considered statistics are: Statistics={'Histogram', 'Mean', 'Mode', 'Max', 'Min', 'Var', 'Change'};

The resulted statistical features are shown in Table 6.5.

Table 6.5 List of extracted statistical features.

In_Direction,Histogram	DAngle,Min	RX,Mean	Y_Distance,Var
In_Direction,Mean	DAngle,Var	RX,Mode	Y_Distance,Change
In_Direction,Mode	DAngle,Change	RX,Max	Curliness,Histogram
In_Direction,Max	CosA,Histogram	RX,Min	Curliness,Mean
In_Direction,Min	CosA,Mean	RX,Var	Curliness,Mode
In_Direction,Var	CosA,Mode	RX,Change	Curliness,Max
In_Direction,Change	CosA,Max	RY,Histogram	Curliness,Min
Out_Direction,Histogram	CosA,Min	RY,Mean	Curliness,Var
Out_Direction,Mean	CosA,Var	RY,Mode	Curliness,Change
Out_Direction,Mode	CosA,Change	RY,Max	DCurliness,Histogram
Out_Direction,Max	SinA,Histogram	RY,Min	DCurliness,Mean
Out_Direction,Min	SinA,Mean	RY,Var	DCurliness,Mode
Out_Direction,Var	SinA,Mode	RY,Change	DCurliness,Max

Out_Direction,Change	SinA,Max	DX,Histogram	DCurliness,Min
Delta_Direction,Histogram	SinA,Min	DX,Mean	DCurliness,Var
Delta_Direction,Mean	SinA,Var	DX,Mode	DCurliness,Change
Delta_Direction,Mode	SinA,Change	DX,Max	Aspect_Ratio,Histogram
Delta_Direction,Max	DCosA,Histogram	DX,Min	Aspect_Ratio,Mean
Delta_Direction,Min	DCosA,Mean	DX,Var	Aspect_Ratio,Mode
Delta_Direction,Var	DCosA,Mode	DX,Change	Aspect_Ratio,Max
Delta_Direction,Change	DCosA,Max	DY,Histogram	Aspect_Ratio,Min
Tanget,Histogram	DCosA,Min	DY,Mean	Aspect_Ratio,Var
Tanget,Mean	DCosA,Var	DY,Mode	Aspect_Ratio,Change
Tanget,Mode	DCosA,Change	DY,Max	DAspect_Ratio,Histogram
Tanget,Max	DSinA,Histogram	DY,Min	DAspect_Ratio,Mean
Tanget,Min	DSinA,Mean	DY,Var	DAspect_Ratio,Mode
Tanget,Var	DSinA,Mode	DY,Change	DAspect_Ratio,Max
Tanget,Change	DSinA,Max	E_Distance,Histogram	DAspect_Ratio,Min
DTanget,Histogram	DSinA,Min	E_Distance,Mean	DAspect_Ratio,Var
DTanget,Mean	DSinA,Var	E_Distance,Mode	DAspect_Ratio,Change
DTanget,Mode	DSinA,Change	E_Distance,Max	CurvatureA,Histogram
DTanget,Max	X,Histogram	E_Distance,Min	CurvatureA,Mean
DTanget,Min	X,Mean	E_Distance,Var	CurvatureA,Mode
DTanget,Var	X,Mode	E_Distance,Change	CurvatureA,Max
DTanget,Change	X,Max	X_Distance,Histogram	CurvatureA,Min
Angle,Histogram	X,Min	X_Distance,Mean	CurvatureA,Var
Angle,Mean	X,Var	X_Distance,Mode	CurvatureA,Change
Angle,Mode	X,Change	X_Distance,Max	CurvatureB,Histogram
Angle,Max	Y,Histogram	X_Distance,Min	CurvatureB,Mean
Angle,Min	Y,Mean	X_Distance,Var	CurvatureB,Mode

Angle,Var	Y,Mode	X_Distance,Change	CurvatureB,Max
Angle,Change	Y,Max	Y_Distance,Histogram	CurvatureB,Min
DAngle,Histogram	Y,Min	Y_Distance,Mean	CurvatureB,Var
DAngle,Mean	Y,Var	Y_Distance,Mode	CurvatureB,Change
DAngle,Mode	Y,Change	Y_Distance,Max	DCurvatureA,Histogram
DAngle,Max	RX,Histogram	Y_Distance,Min	DCurvatureA,Mean
DCurvatureA,Mode	DCurvatureB,Min	DCurvatureB,Mean	DCurvatureA,Var
DCurvatureA,Max	DCurvatureB,Var	DCurvatureB,Mode	DCurvatureA,Change
DCurvatureA,Min	DCurvatureB,Change	DCurvatureB,Max	DCurvatureB,Histogram

Appendix D. Character Recognition Detailed Results

Since the collected text is a natural text, the samples of the characters are not uniformly disturbed and some characters have few number of samples for some characters. Moreover, some samples are discarded due to errors in the manual segmentation. For this reason, and because we use a statistical classifier (the statistical classifiers are sensitive to the size of the used data and its distribution), some classes are not used in some experiments. A threshold value of the minimum number of samples for the classes to be considered is stated. We enforce some minimum threshold on the minimum number of samples in training and validation sets. The considered classes are based on the number of its data samples. The used models are those having predefined threshold values of samples in the training, and validation sets, where the threshold can be set to “inf” to refer to selecting all available samples. The experiments are named as KHATT_Ann_Scope_StrokeType_TrMin_VaMin_TrMax_VaMax.

Where Scope can be {All: All forms; I:Isolated Form; B:Begining Form; M:Middle Form; E:End Form}. Stroke Type: {Primary; Secondary}

1. Basic Shapes Classification

In these experiments, the goal is to recognize the basic shapes of the characters, hence, the main strokes are used and the delayed strokes are not considered.

1.1. Balanced Evaluation

All_100_30_100_30:

For classes having at least 100 training samples and 30 validation samples, we used 100 training samples and 30 testing samples (All_100_30_100_30). This setting is used to compare the local features with the statistical features as shown in Table 6.6

Table 6.6. Some results on annotated Basic Shapes classifier: All_100_30_100_30.

# Classes	# Train Samples	# Test Samples	Rec Rate	Top3 RR	Features
28	2800	840	32%	52%	Local writing direction Cos; Sin
28	2800	840	54%	83%	Mean(Cos); Mean(Sin) from a sliding segment window

There are 28 basic shape classes each has 100 training samples and 30 validation samples in the validation set resulting in a training set of 2800 samples and 840 samples in the validation set, all are uniformly distributed. The statistical features show results better than the local ones and hence they are used in the other experiments. The following table (Table 6.7) shows the confusion matrix using the statistical features. Where, Id: Class order in the experiment, Lbl: Positional label, #C: number of correctly recognized samples, #T: number of test samples, RR%: recognition rate percent.

Table 6.7 Basic Shapes Confusion Matrix: All_100_30_100_30.

Id	Lbl	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	#C	#T	RR%
1	B_Nbr	5	1	1	0	0	0	0	1	0	1	1	0	0	3	0	2	6	0	3	1	0	2	0	1	0	1	0	1	5	30	17%
2	B_ay	0	25	0	0	0	0	0	0	0	0	2	0	0	0	0	1	0	0	1	0	0	0	0	0	1	0	0	0	25	30	83%
3	B_fa	0	0	13	0	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	7	1	3	2	1	0	0	0	13	30	43%
4	B_ha	0	0	0	15	2	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	2	2	7	0	0	0	15	30	50%
5	B_he	0	0	0	0	22	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	7	0	0	22	30	73%
6	B_la	0	0	0	0	0	18	0	0	0	0	1	0	4	0	0	1	0	0	0	2	0	0	0	0	0	0	4	0	18	30	60%
7	B_ma	0	0	3	0	0	1	18	0	0	0	2	0	0	0	0	0	0	0	0	0	4	0	0	2	0	0	0	0	18	30	60%
8	E_Nbr	0	0	0	0	0	0	0	16	0	0	1	0	4	0	0	3	6	0	0	0	0	0	0	0	0	0	0	0	16	30	53%
9	E_aa	0	0	0	0	1	0	0	0	16	0	0	0	3	0	0	0	0	0	0	0	0	0	4	0	0	0	6	0	16	30	53%
10	E_da	0	2	0	0	0	1	0	3	0	12	0	0	1	2	0	2	1	0	0	0	0	6	0	0	0	0	0	12	30	40%	

11	E_ee	0	0	0	0	0	0	0	12	0	0	11	0	0	0	0	2	2	0	0	0	0	0	0	1	0	2	0	0	0	11	30	37%
12	E_he	0	0	0	0	0	0	1	0	0	0	0	27	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	27	30	90%	
13	E_la	0	0	0	0	0	0	1	0	0	0	0	24	0	0	0	0	0	0	0	0	0	0	0	0	0	5	0	24	30	80%		
14	E_ra	0	1	0	0	0	2	0	2	0	1	1	0	2	8	0	1	2	0	1	9	0	0	0	0	0	0	0	0	8	30	27%	
15	E_wa	0	0	1	0	0	0	0	0	0	1	0	0	0	10	0	0	0	0	0	13	0	1	4	0	0	0	0	10	30	33%		
16	I_Na	0	0	0	0	0	0	5	0	0	2	0	0	0	8	11	0	0	0	0	0	0	1	0	3	0	0	8	30	27%			
17	I_Nbr	0	1	0	0	0	0	3	0	0	2	0	0	0	5	19	0	0	0	0	0	0	0	0	0	0	0	0	19	30	63%		
18	I_aa	0	0	0	0	0	3	0	0	1	0	0	0	1	0	0	0	0	22	0	0	0	0	0	0	0	3	0	22	30	73%		
19	I_da	0	0	0	2	0	0	1	0	0	0	0	0	0	2	1	0	19	0	0	0	0	0	3	2	0	0	19	30	63%			
20	I_ra	0	0	0	0	0	0	0	0	1	0	0	0	1	1	0	0	1	1	24	0	1	0	0	0	0	0	24	30	80%			
21	I_wa	0	0	4	0	0	0	1	0	1	0	0	0	0	10	0	0	0	1	0	12	0	1	0	0	0	0	12	30	40%			
22	M_Nbr	2	0	0	0	0	1	0	3	0	2	1	0	0	0	0	6	0	0	0	0	10	0	1	0	1	0	3	10	30	33%		
23	M_ay	0	0	2	0	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	13	6	2	2	0	0	13	30	43%			
24	M_fa	0	0	1	0	0	0	0	0	1	0	0	0	0	5	0	0	0	0	0	0	6	15	0	2	0	0	15	30	50%			
25	M_ha	0	2	0	8	0	0	0	1	0	0	0	0	0	0	0	0	5	0	1	0	0	0	11	2	0	0	11	30	37%			
26	M_he	0	0	0	0	5	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	1	1	0	21	0	0	21	30	70%		
27	M_la	0	0	0	0	0	1	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	23	0	23	30	77%			
28	M_se	1	1	0	0	0	0	2	2	0	2	0	0	0	0	0	4	0	0	0	0	1	0	0	0	0	17	17	30	57%			
	Total																													454	840	54%	

When analyzing the recognition results, the confusion sources can be categorized as:

1. Segmentation-based intra-error recognition accounts for 119 errors (i.e. about 30% of the total errors).
2. Segmentation-based inter-error recognition accounts for about 68 errors (i.e.,17% of the total errors).
3. Errors caused from writing distortion account for about 122 errors (i.e.,31% of the total errors).

- **Experiment 2. All_70_15_70_15**

Another balanced experiment is performed for classes having at least 70 training samples and 15 validation samples, consider only 70 training samples and 15 validation samples (35 classes, 2450 samples for training and 1050 for validation, Recognition rate:53%, shown in Table 6.8).

Table 6.8 Basic Shapes Confusion Matrix: All_70_15_70_15.

Id	Lbl	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	#C	#T	RR		
1	B_Nbr	2	0	1	0	0	0	1	0	2	1	0	1	1	0	0	0	0	0	1	3	0	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0	2	15	13%	
2	B_ay	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	12	15	80%	
3	B_fa	0	0	5	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	1	0	0	0	5	0	2	1	0	0	0	0	0	0	5	15	33%		
4	B_ha	0	0	0	10	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	1	0	0	0	1	0	10	15	67%		
5	B_he	0	0	0	0	13	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	13	15	87%	
6	B_la	0	0	0	0	0	1	0	0	0	0	0	0	0	0	3	0	2	0	1	0	0	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	1	15	7%	
7	B_ma	0	0	1	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	1	1	0	0	6	0	1	4	15	27%		
8	B_sa	0	0	0	0	4	0	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	10	15	67%	
9	B_se	0	0	2	1	0	0	1	0	6	0	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	2	6	15	40%
10	E_Nbr	0	0	0	0	0	0	0	0	5	0	0	0	0	0	3	0	0	0	2	1	0	0	3	1	0	0	0	0	0	0	0	0	0	0	0	0	5	15	33%	
11	E_aa	0	0	0	1	0	0	0	0	0	10	0	0	2	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	10	15	67%	
12	E_da	0	0	0	0	0	0	0	0	1	0	9	0	1	2	0	1	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	9	15	60%	
13	E_ee	0	0	0	0	0	0	0	0	1	0	0	8	0	0	0	0	0	2	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	8	15	53%	
14	E_he	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	15	100%	
15	E_la	0	0	0	0	0	0	0	0	0	0	0	0	0	14	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	14	15	93%		
16	E_ma	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	15	100%	
17	E_ra	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	3	0	0	1	1	9	0	0	0	0	0	0	0	0	0	0	0	1	15	7%		
18	E_wa	0	0	2	0	1	0	0	0	0	0	0	0	0	0	0	0	7	0	0	0	0	0	0	0	2	0	0	1	0	2	0	0	0	0	7	15	47%			
19	I_Na	0	0	0	0	0	0	0	0	2	0	0	2	0	0	0	0	0	3	5	0	0	1	2	0	0	0	0	0	0	0	0	0	0	0	0	3	15	20%		
20	I_Nbr	0	0	0	0	0	0	0	0	2	0	0	1	0	0	0	0	0	1	10	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	10	15	67%		
21	I_aa	0	0	0	0	0	1	0	0	0	0	0	0	0	6	0	0	0	0	0	0	6	0	0	2	0	0	0	0	0	0	0	0	0	0	0	6	15	40%		
22	I_da	0	0	0	1	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	12	0	0	1	0	0	0	0	0	0	0	0	0	0	12	15	80%		
23	I_ee	0	0	0	0	0	0	0	0	1	0	0	2	0	0	0	0	0	1	0	0	0	9	2	0	0	0	0	0	0	0	0	0	0	0	0	9	15	60%		
24	I_la	0	0	0	0	0	1	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	0	0	0	0	12	15	80%		

1	B_Nbr	21	1	4	0	0	4	2	14	0	5	2	3	0	14	0	30	51	1	18	8	0	12	1	3	0	7	1	1	21	203	10%				
2	B_ay	1	79	1	0	0	0	1	0	0	0	4	0	0	0	0	3	0	3	0	0	0	0	0	1	0	0	0	79	93	85%					
3	B_fa	1	0	40	0	0	0	0	0	0	2	0	0	4	0	6	0	0	0	2	0	17	1	4	11	0	1	0	1	40	90	44%				
4	B_ha	0	0	5	35	4	0	1	0	0	0	0	0	0	1	0	0	0	0	0	0	2	0	7	4	21	1	0	0	35	81	43%				
5	B_he	0	0	0	2	30	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	30	37	81%					
6	B_la	2	0	1	0	2	108	0	1	0	1	1	0	17	22	1	2	0	5	0	3	1	0	1	0	0	0	32	0	108	200	54%				
7	B_ma	2	3	22	0	0	0	50	0	0	4	4	0	1	0	4	0	1	0	0	0	9	0	1	3	3	3	0	1	50	111	45%				
8	E_Nbr	0	0	0	0	0	1	0	26	0	0	5	0	8	0	0	12	28	0	0	0	0	4	1	1	0	0	0	0	26	86	30%				
9	E_aa	0	0	0	0	7	0	0	0	166	0	1	7	21	0	0	0	0	6	0	0	0	0	6	0	0	0	28	0	166	242	69%				
10	E_da	1	1	0	0	0	4	0	2	0	22	0	0	0	3	0	2	4	0	0	1	0	8	0	0	0	0	4	0	22	52	42%				
11	E_ee	0	1	0	0	0	3	0	16	1	0	44	0	1	0	0	45	18	0	1	0	0	0	1	0	1	3	0	0	44	135	33%				
12	E_he	0	0	0	0	1	0	0	0	0	0	0	95	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	95	96	99%				
13	E_la	0	0	0	0	0	0	0	1	0	0	0	0	31	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	31	35	89%			
14	E_ra	3	4	0	0	0	9	0	0	0	4	0	0	0	58	2	6	22	0	5	36	5	1	0	0	0	0	1	0	58	156	37%				
15	E_wa	0	0	11	0	1	0	0	0	0	7	0	0	5	0	57	0	0	0	1	0	26	0	0	6	0	2	0	0	57	116	49%				
16	I_Na	1	0	1	0	0	0	0	0	0	2	0	0	0	0	20	13	0	0	2	0	0	0	0	0	0	0	5	0	0	20	44	45%			
17	I_Nbr	2	0	0	0	0	1	0	0	0	0	2	0	0	1	0	13	28	0	0	0	0	0	0	0	0	0	1	0	0	28	48	58%			
18	I_aa	0	0	0	0	1	19	0	0	1	0	0	0	15	5	0	4	0	207	0	2	1	0	0	0	0	0	26	0	207	281	74%				
19	I_da	0	1	1	1	0	0	0	0	0	0	0	0	0	0	4	4	0	31	1	3	0	0	0	4	0	0	0	31	50	62%					
20	I_ra	1	0	0	0	0	0	0	0	0	1	0	0	0	13	0	0	4	1	2	40	1	0	0	0	0	1	0	0	40	64	63%				
21	I_wa	1	0	14	0	0	0	0	1	0	10	0	0	4	0	44	0	1	0	2	0	44	0	1	3	1	0	0	0	44	126	35%				
22	M_Nbr	6	0	1	0	0	1	1	20	1	21	0	4	0	3	0	4	34	0	0	2	0	65	0	4	0	11	0	6	65	184	35%				
23	M_ay	0	0	4	0	0	0	0	0	0	1	0	0	2	0	8	0	1	0	0	0	0	0	33	17	0	3	0	0	33	69	48%				
24	M_fa	0	0	5	0	0	0	0	0	0	0	0	0	0	0	4	0	1	0	0	0	1	0	13	46	0	1	0	0	46	71	65%				
25	M_ha	1	2	0	7	1	0	0	0	0	0	0	0	0	1	0	1	0	6	0	3	0	0	0	26	0	0	0	26	48	54%					
26	M_he	0	0	2	0	8	0	0	0	0	2	0	2	0	0	0	1	0	0	0	0	0	2	3	1	30	0	0	30	51	59%					
27	M_la	0	0	0	0	0	3	0	0	0	1	0	0	9	1	1	0	0	0	0	0	0	0	1	0	0	0	62	0	62	78	79%				
28	M_se	2	0	0	0	0	0	0	0	0	2	0	0	0	0	0	6	0	0	0	0	4	0	0	0	1	0	18	18	33	55%					
																																			1512880	53%

1. Segmentation-based intra-error recognition accounts for 414 errors (i.e. about 30% of the total errors).

2. Segmentation-based inter-error recognition accounts for about 159 errors (i.e.,12% of the total errors).
3. Errors caused from writing distortion account for about 467 errors (i.e.,34% of the total errors).

- All_70_15_inf_inf

For classes having at least 70 training samples and 15 testing samples, consider all training samples and all testing samples (35 classes 12560 training samples, 3028 validation, Table 6.10).

Table 6.10 Basic Shapes Confusion Matrix: All_70_15_inf_inf.

Lbl	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35				
1 B_Nbr	13	0	3	0	0	3	0	0	9	13	0	2	4	3	0	0	11	0	31	65	1	3	1	2	10	0	20	0	4	0	2	3	0	0	0	13	203	6%	
2 B_ay	1	75	1	0	0	0	0	0	0	0	0	3	0	0	1	0	0	0	3	0	1	4	0	0	0	0	0	0	2	2	0	0	0	0	75	93	81%		
3 B_fa	1	0	35	2	0	0	0	0	3	0	1	0	0	3	0	0	9	0	1	0	0	1	0	0	13	1	4	15	0	0	0	0	1	35	90	39%			
4 B_ha	0	0	2	20	2	0	1	21	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	2	0	8	2	20	1	0	0	1	0	20	81	25%		
5 B_he	0	0	0	5	25	0	0	4	0	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	25	37	68%		
6 B_la	2	0	0	0	0	41	0	0	0	0	1	0	0	7	0	11	2	3	0	3	1	0	86	2	0	0	3	0	0	0	38	0	0	41	200	21%			
7 B_ma	2	0	17	1	2	0	21	0	1	3	0	2	5	0	2	4	0	2	0	1	0	0	0	0	0	9	0	0	7	2	1	0	27	0	2	21	111	19%	
8 B_sa	0	0	0	0	1	0	0	25	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	25	26	96%		
9 B_se	1	2	1	3	0	0	0	0	11	0	0	1	0	0	0	0	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	5	11	26	42%	
10 E_Nbr	0	0	0	1	0	0	0	0	35	0	0	2	0	4	0	0	0	17	9	0	0	5	5	0	0	4	1	1	0	0	2	0	0	0	35	86	41%		
11 E_aa	0	0	0	0	3	0	0	7	0	0	163	0	0	17	10	0	0	0	0	6	1	0	0	0	0	0	9	0	0	0	26	0	0	163	242	67%			
12 E_da	2	1	0	0	0	2	0	0	5	0	21	0	0	0	0	0	1	2	4	0	0	0	1	1	0	6	1	1	0	0	4	0	0	21	52	40%			
13 E_ee	0	1	0	1	1	1	0	0	19	1	0	34	0	0	0	0	0	49	3	0	0	19	0	0	0	0	1	0	2	3	0	0	0	34	135	25%			
14 E_he	0	0	0	0	2	0	0	0	0	0	0	0	0	86	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	86	96	90%		
15 E_la	0	0	0	0	0	0	0	0	0	0	0	0	0	23	0	0	0	0	0	0	0	0	0	5	0	0	0	0	0	0	0	0	0	7	0	0	23	35	66%
16 E_ma	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	15	100%		
17 E_ra	0	5	0	0	0	2	0	0	1	0	2	0	0	0	0	40	0	10	19	0	1	3	21	45	6	0	0	0	0	0	1	0	0	40	156	26%			
18 E_wa	0	0	12	0	1	0	0	0	2	0	4	0	0	4	0	0	61	0	1	0	0	0	0	0	25	0	0	5	0	1	0	0	0	61	116	53%			

7	B_la	2	0	1	0	1	0	37	0	0	0	0	0	0	0	0	0	1	0	22	0	22	2	0	0	3	0	3	0	0	0	82	0	2	0	0	2	0	0	1	19	0	0	0	0	37	200	19%
8	B_ma	2	0	7	0	1	0	0	19	0	1	2	0	0	2	3	0	0	6	0	1	0	0	0	0	0	0	0	1	0	0	2	7	0	13	0	1	10	1	0	0	32	0	0	0	19	111	17%
9	B_sa	0	0	0	1	1	0	0	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	22	26	85%	
10	B_se	1	1	0	1	0	0	0	0	12	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	1	0	0	6	1	12	26	46%
11	E_Nbr	1	0	0	0	0	0	0	0	38	0	0	0	4	0	7	0	0	0	14	14	0	1	0	4	0	0	1	0	0	0	1	0	0	1	1	0	0	0	0	0	0	0	0	38	86	44%	
12	E_aa	0	0	0	0	2	0	0	7	0	0	160	0	0	0	18	9	0	0	0	0	0	4	0	1	0	0	0	2	0	0	0	1	9	0	0	0	29	0	0	0	0	0	160	242	66%		
13	E_ay	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	11	100%		
14	E_da	1	1	0	0	0	0	2	0	0	2	0	0	12	0	0	2	0	5	0	3	3	0	0	0	0	1	1	0	1	1	1	1	0	2	0	0	3	0	0	1	0	12	52	23%			
15	E_ee	0	0	0	0	1	0	1	0	0	23	0	0	0	40	0	1	0	0	0	34	11	0	2	1	13	1	0	0	0	0	0	0	0	4	0	3	0	0	0	0	0	0	40	135	30%		
16	E_he	0	0	0	0	0	0	0	0	0	0	0	0	0	92	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	92	96	96%	
17	E_la	0	0	0	0	0	0	0	0	1	0	0	0	0	0	28	0	0	0	0	0	0	0	0	0	0	1	0	0	0	2	0	0	0	3	0	0	0	0	0	0	0	0	28	35	80%		
18	E_ma	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	15	15	100%	
19	E_ra	2	0	0	0	0	2	4	0	0	1	0	0	1	7	0	1	0	44	1	6	33	0	0	2	3	0	1	4	0	43	1	0	0	0	0	0	0	0	0	0	0	0	44	156	28%		
20	E_wa	0	0	0	0	0	0	0	0	1	0	5	3	0	0	0	0	2	61	0	0	0	0	0	0	0	0	0	5	0	25	0	1	12	0	0	0	0	0	0	0	1	61	116	53%			
21	I_Na	0	0	0	0	0	6	0	0	0	1	0	0	0	4	0	0	1	0	0	19	10	0	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	44	43%		
22	I_Nbr	0	0	0	0	0	0	0	0	5	0	0	0	2	0	0	0	1	0	19	19	0	0	0	1	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	19	48	40%		
23	I_aa	0	1	0	0	0	0	2	0	0	0	1	7	0	0	9	0	8	0	1	0	215	0	0	0	0	1	22	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	215	281	77%	
24	I_com	0	0	0	0	0	0	0	3	0	0	0	0	0	0	1	0	0	0	0	0	0	5	0	0	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	12	42%		
25	I_da	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	4	7	0	0	28	0	1	0	0	1	4	0	0	0	4	0	0	0	0	0	0	0	0	0	0	28	50	56%			
26	I_ee	0	0	0	0	0	0	0	0	2	0	0	0	8	0	0	0	0	3	5	0	0	0	6	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	6	25	24%		
27	I_he	0	0	0	0	2	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	0	5	0	0	0	1	0	0	0	0	0	0	0	0	0	4	0	0	5	16	31%			
28	I_hh	0	4	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	1	0	0	0	0	5	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	5	14	36%			
29	I_la	0	0	0	0	0	0	1	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	17	76%			
30	I_ma	0	0	1	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	14	79%			
31	I_ra	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	12	0	0	6	1	0	0	0	0	0	0	41	1	0	0	0	0	0	0	0	0	0	0	0	0	0	41	64	64%			
32	I_wa	0	0	2	0	0	0	0	0	5	0	2	2	0	0	0	2	35	0	1	0	0	1	6	0	0	4	0	60	0	0	6	0	0	0	0	0	0	0	0	0	0	60	126	48%			
33	M_Nbr	9	0	0	0	0	20	0	1	0	1	26	1	0	13	1	4	0	0	6	0	2	26	0	0	0	2	1	0	0	0	2	0	53	1	3	0	0	0	0	0	12	0	53	184	29%		
34	M_ay	0	0	1	1	0	0	0	0	0	0	2	0	0	0	0	0	6	0	0	0	0	0	0	1	0	0	0	1	0	0	0	0	0	1	0	34	16	1	1	1	0	0	0	4	34	69	49%
35	M_fa	0	0	1	0	0	0	0	0	0	0	2	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	17	40	0	0	0	0	2	0	6	40	0	0	0	6	40	71	56%			
36	M_ha	1	0	0	8	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	3	0	0	1	0	1	0	3	0	0	1	26	0	0	1	0	1	0	1	26	48	54%					
37	M_he	0	0	0	2	7	0	0	0	1	2	0	0	0	1	2	0	0	0	1	0	0	0	1	0	7	0	0	0	0	0	0	1	1	0	18	0	1	1	0	5	18	51	35%				
38	M_la	0	0	0	0	0	0	1	0	0	0	2	1	0	0	20	0	1	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0	0	50	0	0	0	0	0	0	0	50	78	64%			
39	M_ma	0	1	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	2	0	13	0	2	0	13	21	62%							
40	M_sa	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	9	3	18	17%			
41	M_se	2	0	0	0	0	2	0	0	2	0	0	0	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	4	0	0	0	1	0	0	0	17	0	17	0	17	33	52%					

Table 6.13 Best results of positional classifiers on annotated balanced samples.

# Classes	# Train Samples	# Test Samples	Rec Rate	Top3 RR	Position
6	600	180	86%	100%	Isolated
7	700	210	81%	96%	Begin
7	700	210	73%	91%	Middle
8	800	240	81%	97%	End

Testing using the positional based classifiers results higher accuracy as shown in the following confusion matrices: (I-position, 13 characters, 423 samples, RR :88%, Table 6.14), (B-Position, 15 characters, 770 samples, RR :83%, Table 6.15), (M-Position, 16 characters, 510 samples, RR :82%, Table 6.16), (E-Position, 18 characters, 862 samples, RR:94% Table 6.17)

Table 6.14 Confusion matrix of I-position characters classification.

		1	2	3	4	5	6	7	8	9	10	11	12	13			
1	I_aa	83	0	0	0	0	0	0	3	0	0	1	0	0	83	87	95%
2	I_ae	0	5	0	0	0	0	0	0	0	0	0	1	0	5	6	83%
3	I_ah	0	0	7	0	0	0	0	0	0	0	0	0	0	7	7	100%
4	I_ba	0	0	0	17	0	0	0	0	0	0	0	0	0	17	17	100%
5	I_da	0	0	0	0	39	0	0	5	0	0	1	0	0	39	45	87%
6	I_dh	0	0	0	0	0	13	1	0	0	0	0	0	2	13	16	81%
7	I_na	0	0	0	0	0	0	33	0	0	1	0	3	1	33	38	87%
8	I_ra	0	0	0	0	2	0	0	66	0	0	3	0	0	66	71	93%
9	I_ta	0	0	0	0	0	0	3	0	18	2	0	0	0	18	23	78%
10	I_th	0	0	0	0	0	0	0	0	3	0	0	0	0	0	3	0%
11	I_wa	0	0	0	0	5	0	0	12	0	0	83	0	0	83	100	83%
12	I_za	0	0	0	0	0	0	3	0	0	0	0	7	0	7	10	70%
13	I_wl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
		83	5	7	17	46	13	40	86	21	3	88	11	3	371	423	88%

Table 6.15 Confusion matrix of B-position characters classification.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15			
1	B_ay	60	0	0	0	0	0	0	0	0	1	4	0	0	0	0	60	65	92%
2	B_ba	0	58	0	0	0	0	1	0	0	0	0	0	0	0	3	58	62	94%
3	B_fa	0	0	26	17	0	0	0	0	3	0	0	3	0	0	0	26	49	53%
4	B_gh	0	0	0	4	0	0	0	0	0	0	0	1	0	0	0	4	5	80%
5	B_ha	3	0	0	0	31	5	0	0	0	0	1	0	0	0	0	31	40	78%
6	B_he	0	0	0	0	13	19	0	0	0	0	5	0	0	0	0	19	37	51%
7	B_ja	0	0	0	0	0	0	25	0	0	0	0	0	0	0	0	25	25	100%
8	B_ka	0	0	2	2	0	0	0	17	0	0	0	0	0	0	0	17	21	81%
9	B_kh	0	0	1	1	0	0	0	0	16	0	0	0	0	0	0	16	18	89%
10	B_la	1	0	0	0	1	4	0	0	0	194	10	0	0	0	0	194	210	92%
11	B_ma	8	0	0	0	22	2	0	0	0	2	72	0	0	0	0	72	106	68%
12	B_na	0	0	0	1	0	0	0	0	0	0	0	35	0	0	0	35	36	97%
13	B_ta	0	0	0	0	0	0	0	0	0	0	0	2	23	0	0	23	25	92%
14	B_th	0	1	0	2	0	0	0	0	0	0	0	1	1	4	0	4	9	44%
15	B_ya	0	6	0	0	0	0	1	0	0	0	0	0	0	0	55	55	62	89%
		72	65	29	27	67	30	27	17	19	197	92	42	24	4	58	639	770	83%

Table 6.16 Confusion matrix of M-position characters classification.

id	Lbl	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	#C	#T	RR
1	M_ay	51	0	0	0	2	0	0	0	0	1	0	3	0	0	0	0	51	57	89%
2	M_ba	0	38	0	0	0	0	2	0	0	0	0	0	0	0	0	1	38	41	93%
3	M_fa	0	0	15	4	0	0	0	1	0	0	0	0	1	0	0	0	15	21	71%
4	M_gh	0	0	3	4	0	0	0	0	0	0	0	0	0	0	0	0	4	7	57%
5	M_ha	1	0	0	0	15	1	0	0	0	0	0	0	0	0	0	0	15	17	88%
6	M_he	8	0	0	0	4	35	0	0	0	0	0	0	0	0	0	0	35	47	74%
7	M_ja	0	0	0	0	0	0	10	0	0	0	0	0	0	0	0	0	10	10	100%
8	M_ka	0	0	11	1	0	0	0	34	0	0	0	0	0	0	0	0	34	46	74%
9	M_kh	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	6	6	100%

10	M_la	0	0	0	0	0	0	0	0	0	59	0	0	0	0	0	59	59	100%	
11	M_na	0	0	1	1	0	0	0	0	1	0	38	0	2	1	0	38	44	86%	
12	M_se	0	0	0	0	1	0	0	0	0	0	23	0	0	0	0	23	24	96%	
13	M_sh	0	0	0	0	0	0	0	0	0	2	0	3	0	3	0	3	8	38%	
14	M_ta	0	0	0	1	0	0	1	0	0	7	0	2	27	1	0	27	39	69%	
15	M_th	0	0	0	2	0	0	1	0	0	1	0	1	0	3	0	3	8	38%	
16	M_ya	0	14	0	0	0	0	2	1	0	0	0	0	0	1	58	58	76	76%	
	Sum	60	52	30	13	22	36	15	37	7	60	48	26	9	28	8	59	419	510	82%

Table 6.17 Confusion matrix of E-position characters classification.

		1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18			
1	E_aa	252	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	252	252	100%
2	E_ae	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	1	100%
3	E_ah	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	2	100%
4	E_ba	0	0	0	9	0	0	0	0	0	0	0	0	0	0	0	0	0	0	9	9	100%
5	E_da	1	0	0	0	38	0	0	0	3	0	0	0	0	0	0	0	0	0	38	42	90%
6	E_dh	0	0	0	0	0	14	0	0	0	2	0	0	0	0	0	0	0	0	14	16	88%
7	E_ee	3	0	0	0	0	0	26	0	0	0	0	0	0	0	0	0	0	0	26	29	90%
8	E_he	4	0	0	0	0	0	1	50	0	0	0	0	0	0	0	0	0	0	50	55	91%
9	E_la	3	0	0	0	0	0	0	0	39	0	0	0	0	0	0	0	0	0	39	42	93%
10	E_na	0	0	0	0	0	0	0	0	0	48	0	0	0	0	0	0	0	0	48	48	100%
11	E_ra	1	0	0	0	8	0	1	0	0	0	128	0	0	0	0	0	0	0	128	138	93%
12	E_ta	0	0	0	0	0	0	0	0	0	2	0	4	0	0	0	0	0	0	4	6	67%
13	E_tee	0	0	0	0	0	0	0	0	0	2	0	3	56	0	0	0	0	0	56	61	92%
14	E_th	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	1	1	100%
15	E_wa	0	0	0	0	0	0	0	0	0	0	5	0	0	0	82	0	0	0	82	87	94%
16	E_wl	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	2	2	100%
17	E_ya	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0	50	0	50	53	94%
18	E_za	0	0	0	0	0	0	0	0	0	6	0	0	0	0	0	0	0	12	12	18	67%
		264	1	2	12	46	14	28	50	42	60	133	7	56	1	82	2	50	12	814	862	94%

3. Delayed Strokes Classification

In the classification of the delayed strokes, simpler classification methods are used. We use different classifiers with several features. Some of the obtained results are shown in the following table (Table 6.18).

Table 6.18 Results on delayed strokes classification.

# Classes	# Train Samples	# Test Samples	Rec Rate	Features	Classifier
3	60	15	66.7	In_Direction,Histogram	Bnet
3	60	15	80	In_Direction,Histogram	KNN
5	2271	515	87	X,Change;Y,Change	KNN
5	2271	515	89	X,Change;Y,Change	Bnet
5	2271	515	97	X,Change; Y,Change, Aspect Ratio	KNN

Appendix E. Cursive Text Recognition Detailed Results

Here are some of the details of the experimental work of cursive text recognition.

When developing the proposed methods, we used a set of segmented PAWs (Parts of Arabic Words) which is generated from ADAB database in [41]. This dataset is prepared by segmenting at the character level and all delayed strokes are removed manually. The available dataset contains 15,854 letter samples and 513 PAW samples. Samples of this dataset are shown in Figure 6.4. Some of the achieved results with different statistical features are shown in Table 6.19.

Table 6.19 Results of using some statistical features for cursive text recognition on ADAB database.

Features	SB	HHMM	HTK
Average of Freeman Code	40.4%	57.3%	60.6%
Averages of Sine and Cosine	46.1%	55.8%	60.3%
Average of Relative Position	44.4%	48.7%	65.8%
Average of Curvature	45%	42.2%	63.7%
Average of Curliness	48.2%	47.6%	62%
Histogram of Freeman Code + Variance of Freeman Code.	51.5%	55.3%	63.7%
Histogram of Freeman Code+ average of Curliness + averages of Sine and Cosine	61.8%	63.7%	68.8%
Histogram of Freeman Code	55%	62%	59.1%
Mode of Freeman Code	46.1%	53.4%	56.3%
Variance of Freeman Code	24.4%	45.8%	58.9%
Average of Tangent	41%	33.2%	40.4%
Histogram of Freeman Code + Average of Freeman Code	66.2%	51.3%	64.9%

Due to the difficulty of examining all possible extracted statistical features, we used PCA method for feature dimension reduction. We used different dimensions and some of the achieved results are shown in Table 6.20 to compare the different classification methods.

Table 6.20 Results of using PCA statistical features for cursive text recognition on ADAB database.

Features	SB	HHMM	HTK
PCA (10)	58.3%	52.1%	61.8%
PCA (15)	<u>65.7%</u>	<u>66.6%</u>	79.7%
PCA (20)	56.3%	50.1%	<u>81.1%</u>

Table 6.21 shows some recognition results on Online-KHATT text lines using HTK with adopting PCA statistical features. Where N is the number of characters, D is the number of deletions, S is the number of substitutions, I is the number of insertions, A is the accuracy rate, and C is the correctness rate.

Table 6.21 Results of using PCA statistical features for cursive text recognition on Online-KHATT Lines.

Features	N	D	S	I	A%	C%
PCA (10)	2656	531	646	483	37.5	55.68524
PCA (15)	2656	472	555	467	43.75	61.33283
PCA (20)	2656	431	525	443	47.32681	64.00602
PCA (25)	2656	481	566	431	44.35241	60.57982

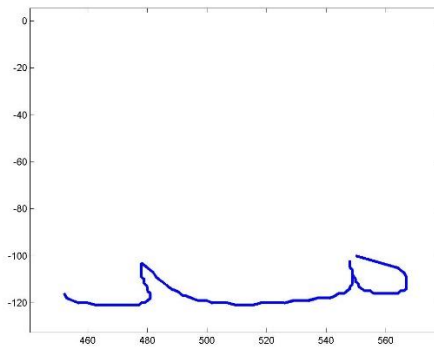
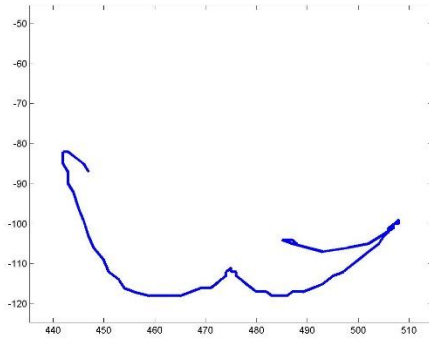
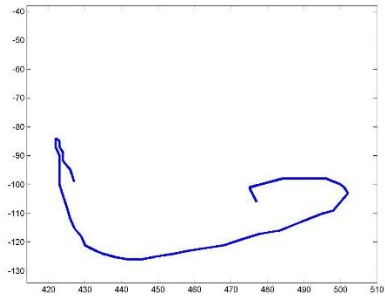


Figure 6.4 Samples of ADAB PAWs.

References

- [1] A. Sharma, "Online handwritten Gurmukhi character recognition," Thapar University, Punjab, India., 2009.
- [2] M. T. Parvez and S. A. Mahmoud, "Offline Arabic Handwritten Text Recognition: A Survey," *ACM Comput. Surv.*, vol. 45, no. 2, pp. 23:1–23:35, Mar. 2013.
- [3] B. Al-Badr and S. Mahmoud, "Survey and bibliography of Arabic optical text recognition," *Signal Processing*, vol. 41, no. 1, pp. 49–77, Jan. 1995.
- [4] N. Y. Habash, "Introduction to Arabic natural language processing," *Synth. Lect. Hum. Lang. Technol.*, vol. 3, no. 1, pp. 1–187, 2010.
- [5] S. A. Azeem, M. El Meseery, and H. Ahmed, "Online Arabic Handwritten Digits Recognition," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2012, pp. 135–140.
- [6] R. I. Elanwar, M. Rashwan, and S. Mashali, "Simultaneous segmentation and recognition of Arabic characters in an unconstrained on-line cursive handwritten document," in *Proceedings of world academy of science, engineering and technology, International conference on Machine learning and Pattern Recognition MLPR2007*, 2007, vol. 23, pp. 288–291.
- [7] R. A. Haraty and C. Ghaddar, "Arabic text recognition," *Int. Arab J. Inf. Technol.*, vol. 1, no. 2, pp. 156–163, 2004.
- [8] T. T. Kuklinski, "Components of handprint style variability," in *Proceedings of Seventh International Conference of Pattern Recognition*, 1984, pp. 924–926.
- [9] A. M. Wing, "Variability in handwritten characters," *Visible Lang.*, vol. 13, no. 3, pp. 283–298, 1979.
- [10] M. A. Abuzaraida, A. M. Zeki, and A. M. Zeki, "Problems of writing on digital surfaces in online handwriting recognition systems," in *2013 5th International Conference on Information and Communication Technology for the Muslim World (ICT4M)*, 2013, pp. 1–5.
- [11] M. Al-Ammar, R. Al-Majed, and H. Aboalsamh, "Online handwriting recognition for the Arabic letter set," *Recent Res. Commun. IT*, pp. 42–49, 2011.

- [12] S. Impedovo, "More than twenty years of advancements on Frontiers in handwriting recognition," *Pattern Recognit.*, vol. 47, no. 3, pp. 916–928, 2014.
- [13] L. Likforman-Sulem, "Recent Approaches in Handwriting Recognition with Markovian Modelling and Recurrent Neural Networks," in *Recent Advances of Neural Network Models and Applications*, Springer, 2014, pp. 261–267.
- [14] A. N. Azmi, D. Nasien, and S. M. Shamsuddin, "A review on handwritten character and numeral recognition for Roman, Arabic, Chinese and Indian scripts," *Int. J. Adv. Stud. Comput. Sci. Eng.*, vol. 2, no. 4, pp. 1–8, 2013.
- [15] U. Saeed, "Automatic Recognition of Handwritten Arabic Text: A Survey," *Life Sci. J.*, vol. 11, no. 3s, 2014.
- [16] M. A. Abuzaraida, A. M. Zeki, and A. M. Zeki, "Recognition Techniques for Online Arabic Handwriting Recognition Systems," in *International Conference on Advanced Computer Science Applications and Technologies (ACSAT)*, 2012, pp. 518–523.
- [17] M. A. Abuzaraida, A. M. Zeki, and A. M. Zeki, "Segmentation techniques for online Arabic handwriting recognition: a survey," in *Proceeding of the 3rd International Conference on Information and Communication Technology for the Moslem World (ICT4M)*, 2010, pp. 37–40.
- [18] M. A. Abuzaraida, A. M. Zeki, and A. M. Zeki, "Feature extraction techniques of online handwriting arabic text recognition," in *5th International Conference on Information and Communication Technology for the Muslim World (ICT4M)*, 2013, pp. 1–7.
- [19] N. Tagougui, M. Kherallah, and A. M. Alimi, "Online Arabic handwriting recognition: a survey," *Int. J. Doc. Anal. Recognit.*, vol. 16, no. 3, pp. 209–226, 2012.
- [20] C. C. Tappert, C. Y. Suen, and T. Wakahara, "The state of the art in online handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 8, pp. 787–808, 1990.
- [21] J. Kim and B.-K. Sin, "Online Handwriting Recognition," *Handb. Doc. Image Process. Recognit.*, pp. 887–915, 2014.
- [22] L. M. Lorigo and V. Govindaraju, "Offline Arabic handwriting recognition: a survey," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 5, pp. 712–724, 2006.

- [23] M. Kherallah, A. Elbaati, H. E. Abed, and A. M. Alimi, "The on/off (LMCA) dual Arabic handwriting database," in *11th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2008.
- [24] M. Kherallah, L. Haddad, A. M. Alimi, and A. Mitiche, "On-line handwritten digit recognition based on trajectory and velocity modeling," *Pattern Recognit. Lett.*, vol. 29, no. 5, pp. 580–594, 2008.
- [25] M. Kherallah, F. Bouri, and A. M. Alimi, "On-line Arabic handwriting recognition system based on visual encoding and genetic algorithm," *Eng. Appl. Artif. Intell.*, vol. 22, no. 1, pp. 153–170, 2009.
- [26] H. El Abed, M. Kherallah, V. Märgner, and A. M. Alimi, "On-line Arabic handwriting recognition competition," *Int. J. Doc. Anal. Recognit.*, vol. 14, no. 1, pp. 15–23, 2011.
- [27] H. El Abed, V. Margner, M. Kherallah, and A. M. Alimi, "Icdar 2009 online arabic handwriting recognition competition," in *10th International Conference on Document Analysis and Recognition, ICDAR'09.*, 2009, pp. 1388–1392.
- [28] H. Ahmed and S. A. Azeem, "On-line Arabic handwriting recognition system based on HMM," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2011, pp. 1324–1328.
- [29] S. Abdelazeem and H. M. Eraqi, "On-line Arabic handwritten personal names recognition system based on HMM," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2011, pp. 1304–1308.
- [30] H. Boubaker, A. Chaabouni, M. Kherallah, A. M. Alimi, and H. El Abed, "Fuzzy segmentation and graphemes modeling for online Arabic handwriting recognition," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2010, pp. 695–700.
- [31] H. Boubaker, A. Chaabouni, M. Ben Halima, A. El Baati, and H. El Abed, "Arabic diacritics detection and fuzzy representation for segmented handwriting graphemes modeling," in *6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, 2014, pp. 71–76.
- [32] H. Boubaker, A. El Baati, M. Kherallah, A. M. Alimi, and H. Elabed, "Online Arabic handwriting modeling system based on the graphemes segmentation," in *20th International Conference on Pattern Recognition (ICPR)*, 2010, pp. 2061–2064.
- [33] H. Boubaker, N. Tagougui, H. El Abed, M. Kherallah, and A. M. Alimi, "Graphemes Segmentation for Arabic Online Handwriting Modeling," *J. Inf. Process. Syst.*, vol. 10, no. 4, pp. 503–522, 2014.

- [34] N. Tagougui, H. Boubaker, M. Kherallah, and A. M. Alimi, "A hybrid MLPNN/HMM recognition system for online Arabic Handwritten script," in *World Congress on Computer and Information Technology (WCCIT)*, 2013, pp. 1–6.
- [35] N. Tagougui, H. Boubaker, M. Kherallah, and A. M. Alimi, "A Hybrid NN/HMM Modeling Technique for Online Arabic Handwriting Recognition," *Int. J. Comput. Linguist. Res.*, vol. 4, no. 3, pp. 107–118, 2013.
- [36] S. A. Azeem and H. Ahmed, "Combining online and offline systems for Arabic handwriting recognition," in *21st International Conference on Pattern Recognition (ICPR)*, 2012, pp. 3725–3728.
- [37] I. Hosny, S. Abdou, and A. Fahmy, "Using advanced hidden markov models for online Arabic handwriting recognition," in *First Asian Conference on Pattern Recognition (ACPR)*, 2011, pp. 565–569.
- [38] H. Al-Barhamtoshy, S. Abdou, and F. A. Al-Wajih, "A Toolkit for Teaching Arabic Handwriting," *Int. J. Comput. Appl.*, vol. 49, no. 23, pp. 17–23, 2012.
- [39] I. Abdelaziz, S. Abdou, and H. Al-Barhamtoshy, "A large vocabulary system for Arabic online handwriting recognition," *Pattern Anal. Appl.*, pp. 1–13, 2015.
- [40] R. Saabni and J. El-Sana, "Comprehensive synthetic Arabic database for on/off-line script recognition research," *Int. J. Doc. Anal. Recognit.*, vol. 16, no. 3, pp. 285–294, 2013.
- [41] G. Kour and R. Saabne, "Fast classification of handwritten on-line Arabic characters," in *6th International Conference of Soft Computing and Pattern Recognition (SoCPaR)*, 2014, pp. 312–318.
- [42] R. I. Elanwar, M. Rashwan, and S. Mashali, "OHASD: the first on-line Arabic sentence database handwritten on tablet PC," in *Proceedings of World Academy of Science, Engineering and Technology (WASET), International conference on International Conference on Signal and Image Processing ICSIP*, 2010, vol. 69, pp. 910–915.
- [43] U.-V. Marti and H. Bunke, "The IAM-database: an English sentence database for offline handwriting recognition," *Int. J. Doc. Anal. Recognit.*, vol. 5, no. 1, pp. 39–46, 2002.
- [44] M. Liwicki and H. Bunke, "IAM-OnDB-an on-line English sentence database acquired from handwritten text on a whiteboard," in *Eighth International Conference on Document Analysis and Recognition*, 2005, pp. 956–961.

- [45] S. Njah, H. Bezine, and A. M. Alimi, "On-line Arabic Handwriting Segmentation via Perceptual Codes: Application to MAYASTROUN database.," in *IEEE SSD*, 2011, pp. 1–5.
- [46] S. Njah, H. Bezine, and A. M. Alimi, "A fuzzy genetic system for segmentation of on-line handwriting: Application to ADAB database," in *IEEE SSCI 2011: Symposium Series on Computational Intelligence - GEFS 2011: 2011 IEEE 5th International Workshop on Genetic and Evolutionary Fuzzy Systems*, 2011, pp. 95–102.
- [47] S. Njah, M. Ltaief, H. Bezine, and A. M. Alimi, "The PerTOHS Theory for f or On- On - Line Handwriting Segmentation," *IJCSI Int. J. Comput. Sci. Issues*, vol. 9, no. 5, pp. 142–151, 2012.
- [48] S. Njah, B. Ben Nouma, H. Bezine, and A. M. Alimi, "MAYASTROUN: A Multilanguage Handwriting Database," in *2012 International Conference on Frontiers in Handwriting Recognition*, 2012, pp. 308–312.
- [49] S. Abdou, W. Fakhr, I. Hosny, and F. Alwajeih, "A general purpose large scale Arabic Online Handwriting Corpus," in *The Egyptian Society of Language Engineering Conference*, 2010.
- [50] I. Abdelaziz and S. Abdou, "AltecOnDB: A Large-Vocabulary Arabic Online Handwriting Recognition Database," *arXiv preprint arXiv:1412.7626*. 2014.
- [51] M. A. Abuzaraida, A. M. Zeki, and A. M. Zeki, "Online database on Quranic hadwriten words," *J. Theor. Appl. Inf. Technol.*, vol. 62, no. 2, pp. 485–492, 2014.
- [52] R. Saabni and J. El-Sana, "Efficient generation of comprehensive database for online arabic script recognition," in *10th International Conference on Document Analysis and Recognition, ICDAR'09.*, 2009, pp. 1231–1235.
- [53] H. Boubaker, A. Elbaati, N. Tagougui, H. El Abed, M. Kherallah, and A. M. Alimi, "Online Arabic databases and applications," in *Guide to OCR for Arabic Scripts*, Springer, 2012, pp. 541–557.
- [54] K. Daifallah, N. Zarka, and H. Jamous, "Recognition-based segmentation algorithm for on-line arabic handwriting," in *10th International Conference on Document Analysis and Recognition, ICDAR'09*, 2009, pp. 886–890.
- [55] D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartogr. Int. J. Geogr. Inf. Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973.

- [56] F. Biadisy, J. El-Sana, and N. Y. Habash, "Online arabic handwriting recognition using hidden markov models," in *Proceedings of the 10th International Workshop on Frontiers of Handwriting and Recognition*, 2006, pp. 85–90.
- [57] F. Biadisy, R. Saabni, and J. El-Sana, "Segmentation-free online arabic handwriting recognition," *Int. J. Pattern Recognit. Artif. Intell.*, vol. 25, no. 07, pp. 1009–1033, 2011.
- [58] G. Kour and R. Saabne, "Real-time segmentation of on-line handwritten arabic script," in *14th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2014, pp. 417–422.
- [59] M. A. Abuzaraida, A. M. Zeki, and A. M. Zeki, "Online Recognition System For Handwritten Arabic Digits," in *ICIT15: The 7th International Conference on Information Technology*, 2015, pp. 45–49.
- [60] A. Ramzi and A. Zahary, "Online Arabic handwritten character recognition using online-offline feature extraction and back-propagation neural network," in *1st International Conference on Advanced Technologies for Signal and Image Processing (ATSIP)*, 2014, pp. 350–355.
- [61] A. M. Alimi and O. A. Ghorbel, "The analysis of error in an on-line recognition system of Arabic handwritten characters," in *Third International Conference on Document Analysis and Recognition (ICDAR '95)*, 1995, pp. 890–893.
- [62] M. S. Baghshah, S. B. Shouraki, and S. Kasaei, "A novel fuzzy classifier using fuzzy LVQ to recognize online persian handwriting," in *The 2nd International Conference on Information and Communication Technologies, ICTTA '06.*, 2006, vol. 1, pp. 1878–1883.
- [63] B. Alsallakh and H. Safadi, "Arapen: an arabic online handwriting recognition system," in *The 2nd International Conference on Information and Communication Technologies, ICTTA '06.*, 2006, pp. 1844–1849.
- [64] S. Izadi and C. Y. Suen, "Online Writer-Independent Character Recognition Using a Novel Relational Context Representation," in *Seventh International Conference on Machine Learning and Applications, ICMLA '08.*, 2008, pp. 867–870.
- [65] M. E. Mustafa and H. A. Abd Alshafy, "Characters' boundaries based segmentation for online Arabic handwriting," in *International Conference on Computing, Electrical and Electronics Engineering (ICCEEE).*, 2013, pp. 306–310.
- [66] H. A. Abd Alshafy and M. E. Mustafa, "HMM based approach for Online Arabic Handwriting recognition," in *14th International Conference on Intelligent Systems Design and Applications (ISDA).*, 2014, pp. 211–215.

- [67] N. Mezghani, A. Mitiche, and M. Cheriet, "On-line recognition of handwritten arabic characters using a kohonen neural network," in *Proceedings Eighth International Workshop on Frontiers in Handwriting Recognition*, 2002, pp. 490–495.
- [68] N. Mezghani, A. Mitiche, and M. Cheriet, "Bayes classification of online arabic characters by gibbs modeling of class conditional densities," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 30, no. 7, pp. 1121–1131, 2008.
- [69] S. M. Ismail and S. N. H. S. Abdullah, "Online arabic handwritten character recognition based on a rule based approach," *J. Comput. Sci.*, vol. 8, no. 11, pp. 1859–1868, 2012.
- [70] S. M. Ismail and S. N. H. S. Abdullah, "Geometrical-Matrix Feature Extraction for on-Line Handwritten Characters Recognition," *J. Theor. Appl. Inf. Technol.*, vol. 49, no. 1, 2013.
- [71] M. Kherallah, L. Hadded, A. Mitiche, and A. M. Alimi, "On-Line Recognition Of Handwritten Digits Based On Trajectory And Velocity Modelling," *Int. J. Pattern Recognit. Lett.*, vol. 29, no. 5, pp. 580–594, 2008.
- [72] S. Izadi, M. Haji, and C. Y. Suen, "A new segmentation algorithm for online handwritten word recognition in Persian script," in *Proc. Eleventh International Conf. Frontiers in Handwriting Recognition (CFHR 2008)*, 2008, pp. 598–603.
- [73] B. Q. Huang, Y. B. Zhang, and M.-T. Kechadi, "Preprocessing techniques for online handwriting recognition," in *Intelligent Text Categorization and Clustering*, Springer, 2009, pp. 25–45.
- [74] I. Khodadad, M. Sid-Ahmed, and E. Abdel-Raheem, "Online Arabic/Persian character recognition using neural network classifier and DCT features," in *IEEE 54th International Midwest Symposium on Circuits and Systems (MWSCAS)*, 2011, pp. 1–4.
- [75] N. Mezghani, M. Cheriet, and A. Mitiche, "Combination of pruned kohonen maps for on-line arabic characters recognition," in *Seventh International Conference on Document Analysis and Recognition*, 2003, pp. 900–904.
- [76] N. Mezghani, A. Mitiche, and M. Cheriet, "On-line character recognition using histograms of features and an associative memory," in *IEEE International Conference on Acoustics, Speech, and Signal Processing, ICASSP-04*, 2004, pp. 841–844.
- [77] R. Saabni and J. El-Sana, "Hierarchical on-line arabic handwriting recognition," in *10th International Conference on Document Analysis and Recognition, ICDAR'09*, 2009, pp. 867–871.

- [78] H. Boubaker, M. Kherallah, and A. M. Alimi, "New algorithm of straight or curved baseline detection for short arabic handwritten writing," in *10th International Conference on Document Analysis and Recognition, ICDAR'09.*, 2009, pp. 778–782.
- [79] M. A. H. Omer and S. L. Ma, "Online Arabic handwriting character recognition using matching algorithm," in *The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, 2010, vol. 2, pp. 259–262.
- [80] A. T. Al-Taani, "An efficient feature extraction algorithm for the recognition of handwritten arabic digits," *Int. J. Comput. Intell.*, vol. 2, no. 2, pp. 107–111, 2005.
- [81] R. I. Elanwar, M. Rashwan, and S. Mashali, "On-Line Arabic Handwriting Text Line Detection Using Dynamic Programming," in *International Conference on Computer Mathematics and Natural Computing (ICCMNC)*, 2011, vol. 74, pp. 588–593.
- [82] M. I. Razzak, M. Sher, and S. A. Hussain, "Locally baseline detection for online Arabic script based languages character recognition," *Int. J. Phys. Sci.*, vol. 5, no. 7, pp. 955–959, 2010.
- [83] G. Al-Habian and K. Assaleh, "Online Arabic handwriting recognition using continuous Gaussian mixture HMMs," in *International Conference on Intelligent and Advanced Systems, ICIAS*, 2007, pp. 1183–1186.
- [84] T. J. Klassen and M. I. Heywood, "Towards the on-line recognition of arabic characters," in *International Joint Conference on Neural Networks IJCNN*, 2002, vol. 2, pp. 1900–1905.
- [85] N. Mezghani, A. Mitiche, and M. Cheriet, "A new representation of shape and its use for high performance in online Arabic character recognition by an associative memory," *Int. J. Doc. Anal. Recognit.*, vol. 7, no. 4, pp. 201–210, 2005.
- [86] N. Mezghani, A. Mitiche, and M. Cheriet, "A new representation of character shape and its use in on-line character recognition by a self organizing map," in *International Conference on Image Processing, ICIP'04.*, 2004, vol. 3, pp. 2123–2126.
- [87] A. M. Alimi, "An evolutionary neuro-fuzzy approach to recognize on-line Arabic handwriting," in *Proceedings of the Fourth International Conference on Document Analysis and Recognition*, 1997, vol. 1, pp. 382–386.
- [88] M. Harouni, D. Mohamad, M. S. M. Rahim, S. M. Halawani, and M. Afzali, "Handwritten Arabic Character Recognition Based on Minimal Geometric Features," *Int. J. Mach. Learn. Comput.*, vol. 2, no. 5, pp. 578–582, 2012.

- [89] M. S. El-Wakil and A. A. Shoukry, "On-line recognition of handwritten isolated Arabic characters," *Pattern Recognit.*, vol. 22, no. 2, pp. 97–105, 1989.
- [90] V. Ghods, E. Kabir, and F. Razzazi, "Fusion of HMM Classifiers, Based on x, y and (x, y) Signals, for the Recognition of Online Farsi Handwriting: a Large Lexicon Approach," *Arab. J. Sci. Eng.*, vol. 39, no. 3, pp. 1713–1723, 2014.
- [91] H. M. Eraqi and S. A. Azeem, "An On-line Arabic Handwriting Recognition System: Based on a New On-line Graphemes Segmentation Technique," in *International Conference on Document Analysis and Recognition (ICDAR)*, 2011, pp. 409–413.
- [92] A. T. Al-Taani and S. Al-Haj, "Recognition of on-line arabic handwritten characters using structural features," *J. Pattern Recognit. Res.*, vol. 1, pp. 23–37, 2010.
- [93] B. Alijla and K. Kwaik, "Oiahcr: online isolated arabic handwritten character recognition using neural network.," *Int. Arab J. Inf. Technol.*, vol. 9, no. 4, pp. 343–351, 2012.
- [94] J. Sternby, J. Morwing, J. Andersson, and C. Friberg, "On-line Arabic handwriting recognition with templates," *Pattern Recognit.*, vol. 42, no. 12, pp. 3278–3286, 2009.
- [95] R. I. Elanwar, M. Rashwan, and S. Mashali, "Unconstrained arabic online handwritten words segmentation using new hmm state design," *Int. Sch. Sci. Res. Innov.*, vol. 6, no. 4, pp. 1189–1197, 2012.
- [96] V. Ghods, E. Kabir, and F. Razzazi, "Effect of delayed strokes on the recognition of online Farsi handwriting," *Pattern Recognit. Lett.*, vol. 34, no. 5, pp. 486–491, 2013.
- [97] K. Addakiri and M. Bahaj, "On-line handwritten arabic character recognition using artificial neural network," *Int. J. Comput. Appl.*, vol. 55, no. 13, pp. 42–46, 2012.
- [98] R. G. Casey and E. Lecolinet, "A survey of methods and strategies in character segmentation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 18, no. 7, pp. 690–706, 1996.
- [99] S. Al-Emami and M. Usher, "On-line recognition of handwritten Arabic characters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 12, no. 7, pp. 704–710, 1990.
- [100] H. Boubaker, M. Kherallah, and A. M. Alimi, "New strategy for the on-line handwriting modelling," in *Ninth International Conference on Document Analysis and Recognition, ICDAR*, 2007, vol. 2, pp. 1233–1247.

- [101] S. Njah, H. Bezine, and A. M. Alimi, "A new encoding system: Application to on-line Arabic handwriting," in *International Conference on Frontiers in Handwriting Recognition (ICFHR)*, 2010, pp. 451–456.
- [102] M. Ltaief, S. Njah, H. Bezine, and A. M. Alimi, "Genetic Algorithms for Perceptual Codes Extraction," *J. Intell. Learn. Syst. Appl.*, vol. 4, no. 04, pp. 255–265, 2012.
- [103] M. Y. Potrus, U. K. Ngah, and H. A. M. Sakim, "An effective segmentation method for single stroke online cursive Arabic words," in *International Conference on Computer Applications and Industrial Electronics (ICCAIE)*, 2010, pp. 217–221.
- [104] M. Harouni, D. Mohamad, and A. Rasouli, "Deductive method for recognition of on-line handwritten Persian/Arabic characters," in *The 2nd International Conference on Computer and Automation Engineering (ICCAE)*, 2010, vol. 5, pp. 791–795.
- [105] A. T. Al-Taani and H. Maen, "Recognition of on-line handwritten Arabic digits using structural features and transition network," *Informatica*, vol. 32, pp. 275–281, 2008.
- [106] H. Bezine, A. M. Alimi, and N. Sherkat, "Generation and analysis of handwriting script with the beta-elliptic model," in *Ninth International Workshop on Frontiers in Handwriting Recognition, IWFHR-9*, 2004, pp. 515–520.
- [107] M. Kherallah, N. Rokbani, and A. M. Alimi, "Global recognition of the Arabic words by genetic algorithm and visual encoding," in *17th IMACS World Congress Scientific Computing, Applied Mathematics and Simulation (IMACS'2005)*, 2005.
- [108] M. Kherallah, L. Haddad, and A. Alimi, "A new Approach for Online Arabic Handwriting Recognition," in *Proceedings of the Second International Conference on Arabic Language Resources and Tools*, 2009, pp. 22–23.
- [109] M. Ltaief, H. Bezine, and A. M. Alimi, "A Neuro-beta-Elliptic Model for Handwriting Generation Movements," in *2012 International Conference on Frontiers in Handwriting Recognition*, 2012, pp. 803–808.
- [110] A. Chaabouni, H. Boubaker, M. Kherallah, A. M. Alimi, and H. El Abed, "Multi-fractal modeling for on-line text-independent writer identification," in *11th International Conference on Document Analysis and Recognition (ICDAR2011)*, 2011, pp. 623–627.
- [111] L. Haddad, T. M. Hamdani, and A. M. Alimi, "OHRS-MEWA: On-line Handwriting Recognition System with Multi-Environment Writer Adaptation," in

14th International Conference on Frontiers in Handwriting Recognition (ICFHR), 2014, pp. 335–340.

- [112] M. I. Razzak, F. Anwar, S. A. Husain, A. Belaid, and M. Sher, “HMM and fuzzy logic: A hybrid approach for online Urdu script-based languages’ character recognition,” *Knowledge-Based Syst.*, vol. 23, no. 8, pp. 914–923, 2010.
- [113] B. Jouini, M. Kherallah, and A. M. Alimi, “A new approach for on-line visual encoding and recognition of handwriting script by using neural network system,” in *Artificial neural nets and genetic algorithms*, 2003, pp. 161–167.
- [114] T. S. El-Sheikh and S. G. El-Taweel, “Real-time Arabic handwritten character recognition,” in *Third International Conference on Image Processing and its Applications.*, 1989, pp. 212–216.
- [115] K. Assaleh, T. Shanableh, and H. Hajjaj, “Recognition of handwritten Arabic alphabet via hand motion tracking,” *J. Franklin Inst.*, vol. 346, no. 2, pp. 175–189, 2009.
- [116] A. Biem, “Minimum classification error training for online handwriting recognition,” *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 28, no. 7, pp. 1041–1051, 2006.
- [117] M. Kherallah, S. Njah, A. M. Alimi, and N. Derbel, “Recognition of on-line handwritten digits by neural networks using circular and beta approaches,” in *IEEE International Conference on Systems, Man and Cybernetics SMC’02*, 2002, vol. 2, pp. 164–169.
- [118] S. A. Azeem and H. Ahmed, “Recognition of Segmented Online Arabic Handwritten Characters of the ADAB Database,” in *10th International Conference on Machine Learning and Applications and Workshops (ICMLA).*, 2011, vol. 1, pp. 204–207.
- [119] H. Bezine and A. M. Alimi, “Development of an Arabic Handwriting Learning Educational System,” *IJSEA Int. J. Softw. Eng. Appl.*, vol. 4, no. 2, pp. 33–49, 2013.
- [120] M. Hammadi, H. Bezine, S. Njah, and A. M. Alimi, “Towards an educational tool for Arabic handwriting learning,” in *2012 International Conference on Education and e-Learning Innovations (ICEELI)*, 2012, pp. 1–6.
- [121] H. Al-Barhamtoshy, S. Abdou, and M. Rashwan, “Mobile Technology for Illiterate Education,” *Life Sci. J.*, vol. 11, no. 9, pp. 242–248, 2014.
- [122] M. Hamdani, H. El Abed, M. Kherallah, and A. M. Alimi, “Combining multiple HMMs using on-line and off-line features for off-line Arabic handwriting

recognition,” in *10th International Conference on Document Analysis and Recognition, ICDAR'09.*, 2009, pp. 201–205.

- [123] E. Grosicki and H. El Abed, “ICDAR 2009 handwriting recognition competition,” in *10th International Conference on Document Analysis and Recognition, ICDAR'09.*, 2009, pp. 1398–1402.
- [124] F. Parwej, “The State of the Art Recognize in Arabic Script through Combination of Online and Offline,” *Int. J. Comput. Sci. Telecommun.*, vol. 4, no. 3, pp. 60–66, 2013.
- [125] S. A. Mahmoud, H. Luqman, B. M. Al-Helali, G. BinMakhashen, and M. T. Parvez, “Online-KHATT: An Open-Vocabulary Arabic Online Text Database,” 2016.
- [126] K. P. Murphy, “Dynamic bayesian networks: representation, inference and learning,” California, Berkeley, 2002.
- [127] M. Parvez and S. A. Mahmoud, “Arabic handwriting recognition using structural and syntactic pattern attributes,” *Pattern Recognit.*, vol. 46, no. 1, pp. 141–154, 2013.
- [128] S. Young, D. Kershaw, J. Odell, D. Ollason, V. Valtchev, and P. Woodland, *The HTK Book V2. 2*. Entropic Ltd., Jan, 1999.
- [129] T. Dean and K. Kanazawa, “A model for reasoning about persistence and causation,” *Comput. Intell.*, vol. 5, no. 2, pp. 142–150, 1989.
- [130] NICI, “The NICI stroke-based recognizer of on-line handwriting,” 2015. [Online]. Available: <http://www.ai.rug.nl/~lambert/recog/nici-stroke-based-recognizer.html>.
- [131] S. Fine, Y. Singer, and N. Tishby, “The hierarchical hidden Markov model: Analysis and applications,” *Mach. Learn.*, vol. 32, no. 1, pp. 41–62, 1998.
- [132] G. Zweig and S. Russell, “Speech recognition with dynamic Bayesian networks,” in *Proceedings of the fifteenth national/tenth conference on Artificial intelligence/Innovative applications of artificial intelligence*, 1998, pp. 173–180.

Vitae

Name :Baligh Mohammed Ahmed Al-Helali

Nationality :YEMEN

Date of Birth :1/1/1981

Email :baleegh.helali@gmail.com

Address :KFUPM-Students Housing

Education :BSc degree in Computer and Mathematics, University of Ibb, Yemen, 2003.MSc degree in Applied Mathematics, University of Taiz, Yemen, 2010.

PUBLICATIONS:

- Rashad A. Al-Jawfi, Baligh M. Al-Helali, Adil M. Ahmed, Fractal Image Compression Using Modified Operator (IFS). Journal of Advances in Mathematics, 5(1), pp. 549-561, 2013.
- Rashad A. Al-Jawfi, Baligh M. Al-Helali, Adil M. Ahmed, Fractal Image Compression Using Self-Organizing Mapping. Applied Mathematics, 05(12), pp. 1810-1819, 2014.
- Baligh M. Al-Helali and Sabri A. Mahmoud, ““*Arabic Online Handwriting Recognition (AOHR): A Survey*,” submitted.

- Baligh M. Al-Helali and Sabri A. Mahmoud, “*A Statistical Framework for Online Arabic Character Recognition*,” Cybernetics, submitted.
- Mahmoud, Sabri A., Hamzah Luqman, Baligh M. Al-Helali, Galal BinMakhashen, and Mohammad Tanvir Parvez. “Online-KHATT: An Open-Vocabulary Arabic Online Text Database,” submitted.
- Mohammad Tanvir Parvez, Hamzah Luqman, Baligh Al-Helali, Sabri A. Mahmoud, “*ICFHR2016 Competition on Arabic Online Text Recognition using Online-KHATT Database*”, the 15th International Conference on Frontiers in Handwriting Recognition, 23-26, October, Shenzhen, China, accepted.
- Baligh M. Al-Helali, Hamzah Luqman, and Sabri A. Mahmoud, “*Extension of Arabic online text database Online-KHATT*,” submitted.
- Patent: Baligh M. Al-Helali and Sabri A. Mahmoud, “*A Statistical Framework for Online Arabic Character Recognition*”, submitted.