

**PATTERN PRIMITIVE BASED MALAYALAM
HANDWRITTEN CHARACTER RECOGNITION
STUDIES FOR REAL-TIME APPLICATIONS**

A Thesis Submitted by

BAIJU K.B.

Under the Guidance of

Dr. LAJISH V.L.

In Partial Fulfillment of the Requirements for the Degree of

**DOCTOR OF PHILOSOPHY IN COMPUTER
SCIENCE**

Under the Faculty of Science



DEPARTMENT OF COMPUTER SCIENCE

**UNIVERSITY OF CALICUT
KERALA, INDIA - 673635**

JUNE 2023



**DEPARTMENT OF COMPUTER SCIENCE
UNIVERSITY OF CALICUT**

Dr. Lajish V.L.
Associate Professor

Calicut University (P.O)
Kerala, India-673635

Certificate

This is to certify that the thesis entitled **PATTERN PRIMITIVE
BASED MALAYALAM HANDWRITTEN CHARACTER
RECOGNITION STUDIES FOR REAL-TIME APPLICATIONS**
is a report of the original work carried out by Mr. BAIJU K.B. under my
supervision and guidance in the Department of Computer Science,
University of Calicut, Kerala and that no part thereof has been presented
for the award of any other degree.

Dr. Lajish V.L.
(Research Supervisor)

Calicut University
June 2023

Declaration

I hereby declare that the work presented in this thesis is based on the original work done by me under the supervision of Dr. Lajish V.L., Associate Professor, Department of Computer Science, University of Calicut, Kerala and that no part thereof has been presented for the award of any other degree.

Calicut University

June 2023

Baiju K.B.

Acknowledgements

Every thesis is an outcome of the contributions of numerous individuals through their rigorous efforts to motivate a researcher in momentum. First of all, I wish to express my sincere appreciation to my Research Supervisor, Dr. Lajish V.L., Associate Professor, Department of Computer Science, University of Calicut, who initiated the spark to do research and convincingly guided me even when the road got tough. Without his persistent help, the goal of this thesis would not have been realized.

I express my gratitude to Ms. Manjula K.A., Associate Professor and Head, Department of Computer Science, University of Calicut, for providing me the facilities and support in the Department.

I extend my sincere gratitude to Dr. A.G. Ramakrishnan, Director, MILE Labs, IISc, Bengaluru, for supporting me with the right motive to progress through the studies in the meantime. I heartfully acknowledge Dr. P. Nagabhushan, Vice Chancellor, Vignan's Foundation for Science, Technology and Research (deemed university), Andra Pradesh (Former Director, IIIT Hyderabad) for mentoring me and showing me the right direction during my studies.

I sincerely appreciate my colleagues, Dr. Vivek P., Dr. Benson C.C., Dr. Sandesh E.P.A., Ms. Reshma P.K., Ms. Sabna T.S., Ms. Sabeerath K., Ms. Habeebath K.P., Mr. Anoop K., and Ms. Manjary P. Ganagan for their co-operation throughout the studies. I would like to acknowledge the editing support provided by Mr. Manoj John, HSS Teacher, GHSS Kolery at Wayanad. I want to pay my special regards to

the faculties of Department of Computer Science and principal at NMSM Govt. College, Kalpetta for their extensive support. I also acknowledge the support provided by the faculty members and office staff of the Department of Computer Science, University of Calicut.

My deep and sincere gratitude to my spouse, Dr. C. Namitha. my children, Daksha Niyatha and Goutham Niyath, for their continuous and unparalleled love. I am grateful to all the family members for their valuable support throughout the research.

Baiju K.B

*Dedicated to my Late Grand Parents
Who Taught me the Value of Education*

Sri. K.P. Kunhiraman
&
Smt. Radha K.P.

ABSTRACT

The communication between people in society is partially replaced with machines as one part of the process for a few decades. As technology dwelt faster changes, more and more devices came out for simplifying the task of human-machine interaction, majorly computers and mobile devices. Nowadays, ways for human-machine coalescence are extensive in speech and handwriting termed under Natural Language Processing. For handwriting recognition, *off-line* and *online* are the two methods to simplify the task of data input in various styles followed in different scripts. Online Handwritten Character Recognition (OHCR) involves the automatic conversion of handwriting consisting of temporal information that are converted into letter codes usable in computers.

The thesis analyses existing techniques in Online Handwritten Character Recognition (OHCR) and investigates the potential use of a pattern primitive based scheme, for real-time applications in Malayalam. The study starts with an analysis of Malayalam characters and their linguistic formulations. The representation of the sounds of a language by written symbols is also a part of the study. All the vowels, consonants, and other units in the Malayalam language are detailed with the various rules, concerning the formation of conjuncts within the orthographical perspective. The study contributes a database of online handwritten characters constructed using two devices. The first device records handwritten strokes in a traditional way, where paper and digital pen with a sensor is used for data acquisition. In the second device, the touch screen capability of current technology is utilized in acquiring data with a touch screen laptop. As a part of the study, a

native database named Calicut University Online Handwriting Data Base (CU-OHDB) is created containing 30800 samples of online Malayalam handwritten characters written by twenty writers, where 13200 samples are obtained using a pen and paper method, and the other 17600 samples recorded using a touchscreen-enabled laptop. The preprocessing of raw data is necessary due to the variations in writing style and writing imperfection caused by the writer and also the device. The methods involved in the preprocessing phases are detailed in the thesis with the corresponding algorithms. For normalization, min-max normalization is used. Smoothing is done using moving average filter and resampling is carried out using equidistant resampling with spline interpolation for missing points.

The thesis addresses handwriting recognition as a pattern recognition problem. Since online handwritings are temporal sequences, they are also considered as patterns. Studies tuned in such directions reveal so many aspects of Malayalam characters. An extensive study is conducted to identify the structural and directional properties of every character. Segmentation (Lexical analysis) of Malayalam characters is conducted to identify unique patterns in them. From the analysis, it is found that, certain pattern segments, namely pattern primitives, are frequent in Malayalam characters. It is also found that, there are 26 pattern primitives to represent all the 44 single stroke vowel and consonant characters in Malayalam. These points are manually marked on every character to form a character reference set. As a part of the work, a label to every primitive is assigned, and every character is arranged as a sequence of pattern primitive labels.

The segmentation of character samples is performed in the work using three algorithms, namely, Ramer Douglas Peucker (RDP) algorithm, Eight Direction Freeman Code (EDFC) and, a combined approach of these two algorithms. The Ramer Douglas Peucker (RDP) algorithm, which is used for polygonal approximations of curves in numerous pattern recognition problems to reduce a character pattern into a finite number of points, is used for segmentation of the online handwritten character samples. Even with the variability in the writing style of the samples, the technique performed well with an average segmentation rate of 75.07%.

The role of direction in the recognition process of a character is dealt within the segmentation part of the thesis using the well-known Eight Direction Freeman Code (EDFC). The segmentation scheme using Eight Direction Freeman Code (EDFC) proposed in the study also reported an accuracy of 74.68%. A combined approach of RDP and EDFC with certain filtering over directions and redundant points achieved a segmentation rate of 91.27% through visual comparisons. An automated scheme for measuring segmentation accuracy is also implemented in the study and obtained 79.13% accuracy for the combined approach. The comparison of the three segmentation schemes suggested that the combined approach is the best method, and the segmentation of pattern primitives is performed using the combined approach. The pattern primitives of the characters obtained from segmentation using the combined approach are extensively studied in the remaining sections of the thesis.

In the next phase, an in-depth study of various features of every pattern primitive is deployed in visual and structural formulations. The

research focuses on features, mainly on the direction aspects and morphology of the characters. Preprocessing of pattern primitives is also described in the thesis for linear types. Pattern primitives with curvature values near zero are redrawn as a smooth line using Bresenham's line drawing algorithm, and angle corrections are made by measuring slope values. The study shows that the maximum number of pattern primitives in the selected single stroke character set of Malayalam is eight and the minimum number of pattern primitives is two. Recognition experiments based on pattern primitives are conducted in a class-modular approach, where clusters are formed based on the number of pattern primitives present in the characters.

For every cluster, the discriminating features like cusps, linearity, reduced direction code, pattern primitive length, the direction of a pattern primitive, and intersections in various pattern primitives are identified. An intra-class recognition using the extracted features displayed higher accuracies in the classification experiments using Support Vector Machine (SVM) classifier for the 44 single stroke online handwritten samples of Malayalam characters. The accuracy thus obtained ranges from 81.67% to 100% in various clusters that emphasizes the suitability of pattern primitive approach for the recognition of online handwritten Malayalam characters.

The predictive model for real-time recognition of Malayalam handwritten characters, based on pattern primitives, is presented in the thesis with prime focus. Two approaches are proposed in the model. In the first approach, Deterministic Finite Automata (DFA) based OHCR, using pattern primitive string representation of characters, is described. In this approach, the character will be recognized only after finishing the

writing of the entire character. Even though the experiments showed an average accuracy of 65.75%, the method uses a holistic approach, which causes considerable delay in recognition. Hence an alternate approach is also proposed. In this approach, a real-time prediction model for OHCR is used. The transition sequences of pattern primitives that constitute each character are analyzed in the predictive model. These pattern primitive transition information, obtained from the characters, are utilized for constructing a tree structure for the possible predictions of every character in a real-time environment. The average Reduction in Writing Time (RWT) is the significant advantage of the predictive model to be incorporated in Malayalam OHCR. The study shows that most of the Malayalam characters start writing in the upward direction. It can also be seen that fifty percentage of the characters start with a unique pattern primitive. The most frequent pattern primitive transition is also identified in the study . Experiments are also conducted to verify the effectiveness of the model using online handwritten character samples taken from CU-OHDB dataset and obtained an average prediction accuracy of 77.63%. The prediction part of the system is quick and straightforward to redefine existing systems that recognize the handwritten characters, only after completing the writing process. The model is useful in building a fast handwritten character recognition system to be realized in real-time environments.

A Keyword Spotting (KWS) technique is implemented as an application of handwriting recognition in the final section of the thesis. The well-known pattern matching technique, Dynamic Time Warping (DTW) is used to compare the words in the experiments. An average performance score of 78.31% over a regular DTW search obtained in

the experiment suggests that, the method can effectively be used for Keyword Spotting (KWS) for Malayalam online handwritten documents in a real-time environment.

Contents

List of Figures	xviii
List of Tables	xxii
List of Abbreviations	xxvi
1. Introduction	1-8
1.1. Background	1
1.2. Motivation	3
1.3. Outline of the Thesis Organization	5
2. Review of Previous Works	9-38
2.1. Introduction	9
2.2. Review of various Segmentation Techniques used in Pattern Recognition applicable to Handwritten Characters	10
2.3. Review of Online Handwritten Character Recognition Studies Focusing on Segmentation	21
2.4. Review of the Studies related to Real-Time Recognition of Handwritten Characters	28
2.5. Review of Keyword Spotting (KWS) in Handwritten Documents	32
2.5.1. <i>Review of Keyword Spotting in Handwritten Document Images</i>	32
2.5.2. <i>Review of Keyword Spotting in Online Handwritten Documents</i>	36
2.6. Conclusion	37
3. Malayalam Online Handwritten Character Database Creation and their Pre-processing	39-57
3.1. Introduction	39
3.2. Orthography and Categorization of Malayalam Characters	40
3.2.1. <i>Vowels and Dependent Signs in Malayalam</i>	40
3.2.2. <i>Special Symbols in Malayalam (Anusvaram, Visargam, Chandrakkala)</i>	42
3.2.3. <i>Diphthongs in Malayalam</i>	42
3.2.4. <i>Malayalam Consonant Classes</i>	43
3.2.5. <i>Malayalam Chillu / Chillaksharam</i>	44

3.2.6. <i>Formation of Various Conjunct Classes in Malayalam</i>	45
3.3. Database Creation for Online Handwritten Characters in Malayalam	47
3.3.1. <i>Devices used for Data Acquisition</i>	48
3.3.2. <i>Setting up of Data Capturing Environment</i>	49
3.3.3. <i>Labeling of Samples</i>	51
3.3.4. <i>Number of Samples in the Database</i>	51
3.4. Preprocessing Techniques applied to the Online Handwritten Character Samples	52
3.4.1. <i>Min-max Normalization</i>	53
3.4.2. <i>Smoothing</i>	54
3.4.3. <i>Resampling</i>	55
3.5. Conclusion	57
4. Analysis of Pattern Primitives and Segmentation of Online Malayalam Handwritten Characters using Ramer Douglas Peucker Algorithm and Eight Direction Freeman Code	59-113
4.1. Introduction	59
4.2. Pattern Primitives in Malayalam Characters	61
4.3. Pattern Primitives, Labeling of Characters and Creation of Reference Character Set	63
4.3.1. <i>Representation of Malayalam Vowel Characters using Pattern Primitives</i>	67
4.3.2. <i>Representation of Malayalam Consonant Characters using Pattern Primitives</i>	69
4.3.3. <i>Creation of Reference Set of Malayalam Characters based on Pattern Primitives</i>	74
4.4. Frequency of Occurrence of Pattern Primitives in Malayalam Characters	79
4.4.1. <i>Frequency of Occurrence of Pattern Primitives</i>	79
4.5. Pattern Primitive based Character Segmentation Algorithms	90
4.5.1. <i>Character Segmentation based on Ramer Douglas Peucker Algorithm</i>	91
4.5.2. <i>Character Segmentation based on Eight Direction Freeman Code Algorithm</i>	95

4.5.3. <i>A Combined Approach of RDP and EDFC Algorithm for Segmentation</i>	103
4.5.4. <i>Automated Method for Determining Segmentation Accuracies</i>	107
4.6. Conclusion	112
5. Recognition of Malayalam Online Handwritten Characters based on Pattern Primitives: A Class Modular Approach	114-161
5.1. Introduction	111
5.2. Preprocessing of Pattern Primitives	116
5.2.1. <i>Categorization of Pattern Primitives into Straight-line and Arc Segments using Curvature Values</i>	116
5.2.2. <i>Smoothing of Straight-line Pattern Primitives using Bresenham's Line Drawing Algorithm</i>	119
5.2.3. <i>Skew Correction of Horizontal and Vertical Straight-line Pattern Primitives</i>	120
5.3. Feature Extraction from Pattern Primitives	123
5.3.1. <i>Minimized Chain Code (Reduced Direction Code) Features</i>	124
5.3.2. <i>Cusp Identification in Pattern Primitives</i>	126
5.3.3. <i>Length of the Pattern Primitives</i>	128
5.3.4. <i>Direction Category Code Features (DCCF) of Pattern Primitives</i>	129
5.3.5. <i>Intersection (Crossing) of Pattern Primitives</i>	131
5.4. Grouping of Malayalam Handwritten Characters based on the Features of Pattern Primitives	133
5.4.1. <i>Pattern Primitive Features of Characters in Cluster A</i>	135
5.4.2. <i>Pattern Primitive Features of Characters in Cluster B</i>	136
5.4.3. <i>Pattern Primitive Features of Characters in Cluster C</i>	138
5.4.4. <i>Pattern Primitive Features of Characters in Cluster D</i>	140
5.4.5. <i>Pattern Primitive Features of Characters in Cluster E</i>	142
5.4.6. <i>Pattern Primitive Features of Characters in Cluster F</i>	143
5.4.7. <i>Pattern Primitive Features of Characters in Cluster G</i>	145
5.5. Experimental Setup	145

5.6. Analysis of the OHCR Results	148
5.7. Conclusion	160
6. A Predictive Model for Real-Time Recognition of Malayalam Handwritten Characters based on Pattern Primitives	162-213
6.1. Introduction	162
6.2. Pattern Primitive String (PPS) Representation of Malayalam Online Handwritten Characters	165
6.3. Pattern Primitive based OHCR using DFA	167
6.3.1. <i>Design of the Deterministic Finite Automata based on Pattern Primitives Present in the Characters</i>	169
6.3.2. <i>Implementation of the DFA based OHCR</i>	170
6.4. The Predictive Model for Real-Time Recognition of Malayalam Handwritten Characters based on Pattern Primitives	177
6.4.1. <i>Pattern Primitive Transition Sequences in the Malayalam Online Handwritten Characters</i>	177
6.5. Tree Representation of the Predictive Model for Real-Time OHCR Implementation	188
6.5.1. <i>Tree Representation of the Predictive Model based on the First Pattern Primitive b as Root Node</i>	188
6.5.2. <i>Tree Representation of the Predictive Model based on the First Pattern Primitive p as Root Node</i>	190
6.5.3. <i>Tree Representation of the Predictive Model based on the First Pattern Primitive c as Root Node</i>	191
6.5.4. <i>Tree Representation of the Predictive Model based on the First Pattern Primitive y as Root Node</i>	192
6.5.5. <i>Tree Representation of the Predictive Model based on the First Pattern Primitive g as Root Node</i>	193
6.5.6. <i>Tree Representation of the Predictive Model based on the First Pattern Primitive h as Root Node</i>	194
6.5.7. <i>Tree Representation of the Predictive Model based on the First Pattern Primitive a as Root Node</i>	195
6.6. Reduction in Writing Time (RWT) for the Malayalam Online Handwritten Characters	199
6.6.1. <i>Reduction in the Number of Classes in Character Prediction</i>	199
6.6.2. <i>Reduction in Writing Time (RWT)</i>	202

6.6.3. <i>Implementation of the Character Prediction Model for Real-Time OHCR</i>	205
6.7. Conclusion	211
7. Keyword Spotting in Online Handwritten Malayalam Documents using Dynamic Time Warping	214-230
7.1. Introduction	214
7.2. Creation of Keyword Dataset and Preprocessing	217
7.2.1. <i>Malayalam Online Handwritten Keyword Dataset Creation</i>	217
7.2.2. <i>Preprocessing Methods Applied</i>	220
7.3. Feature Extraction	224
7.4. Experimental Setup	227
7.5. Result Analysis	229
7.6. Conclusion	229
8. Conclusions and Recommendations	231-238
8.1. Conclusions	231
8.2. Contributions of the Thesis	236
8.3. Recommendations	237

List of Figures

Figure No	Title	Page No
1.1	Thesis organization	8
3.1	E-writemate pen and memory unit	48
3.2	User interface for touchscreen-enabled data acquisition	49
3.3	A sample page used for single stroke data acquisition of handwritten characters	50
3.4 (a)	Character അ <a> in raw form	54
3.4 (b)	Character അ <a> in normalized form	54
3.5	Shape variation of Malayalam character അ <a> after applying moving average filter of various filter lengths	55
3.6	Original, re-sampled, and interpolated points in a Malayalam character sample റ <ra>	56
3.7 (a)-(b)	Original and re-sampled set of Malayalam characters അ <a>, ഇ <i>, and ഉ <u>	57
4.1	Common pattern segments in Malayalam	60
4.2 (a)-(g)	Segments of the Malayalam character അ <a>	61
4.3	Unique segments of the Malayalam character അ <a>	62
4.4	Segments of the Malayalam character ഘ <gha>	62
4.5	Unique segments of the Malayalam character ഘ <gha>	62
4.6	Frequency of occurrence of pattern primitives in the Malayalam handwritten characters	90
4.7	The output obtained after applying RDP algorithm on the character അ <a> for various ϵ values	93
4.8	Directions and codes in Eight Direction Freeman Code algorithm	96
4.9	Plot of direction codes and segmentation points marked by the EDFC segmentation algorithm for the Malayalam character യ <ya>	101

4.10	Segmentation points obtained for the Malayalam characters അ <a>, ഴ <ga>, യ <ya>, ര <ra>, ല <la>, and ഹ <ha> using the combined approach for segmentation	105
5.1	Block diagram of the proposed pattern primitive based OHCR system	115
5.2	Average curvature and angle of contingence of an arc	117
5.3 (a)	Slope values	121
5.3 (b)	Slope values and directions	122
5.4	Direction Code of the first pattern primitive of the Malayalam character അ <a>	124
5.5	Cusp in a pattern primitive present in the Malayalam characters അ <a>, ആ <a: >, ദ <da>, ഏ <e: > and ഒ <o >	127
5.6	Length of pattern primitives (blue dots), segmentation points (red dots) for the Malayalam handwritten character samples of അ <a>, സ <sa>, ഹ <ha>	128
5.7	Intersection points obtained for the Malayalam characters അ <a>, ത <ta>, and ഏ <e>	133
5.8 (a)-(c)	Direction code features of second pattern primitive for the Malayalam characters റ <ra>, ഭ <bha> and ദ <da> respectively	136
5.8 (d)	Cusp features of second pattern primitive for the Malayalam characters ദ <da>	133
5.9	Normalized confusion matrix for cluster A	149
5.10	Normalized confusion matrix for cluster B	150
5.11	Normalized confusion matrix for cluster C(Up)	151
5.12	Normalized confusion matrix for cluster C(Down)	152
5.13	Normalized confusion matrix for cluster D(Up)	153
5.14	Normalized confusion matrix for cluster D(Down)	154
5.15	Normalized confusion matrix for cluster E	155
5.16	Normalized confusion matrix for cluster F(L)	156
5.17	Normalized confusion matrix for cluster F(NL)	157
5.18	Normalized confusion matrix for cluster G	158

5.19	Average recognition accuracies for each cluster	160
6.1	Proposed model for real-time recognition of Malayalam handwritten characters based on pattern primitives	164
6.2	DFA which accepts all strings starting with 'ab'	169
6.3	DFA of the pattern primitive string 'abab' representing the Malayalam online handwritten character റ <na>	170
6.4	A sample DFA representing the Malayalam online handwritten character ച <cha>	172
6.5	The possible transitions from first to second pattern primitives in Malayalam handwritten characters	184
6.6	Heat map showing the frequency of pattern primitive transitions obtained for the Malayalam online handwritten characters	187
6.7	Tree representation of the predictive model based on the first pattern primitive <i>b</i> as root node	189
6.8	Tree representation of the predictive model based on the first pattern primitive <i>p</i> as root node	190
6.9	Tree representation of the predictive model based on the first pattern primitive <i>c</i> as root node	191
6.10	Tree representation of the predictive model based on the first pattern primitive <i>y</i> as root node	192
6.11	Tree representation of the predictive model based on the first pattern primitive <i>g</i> as root node	193
6.12	Tree representation of the predictive model based on the first pattern primitive <i>h</i> as root node	194
6.13	Tree representation of the predictive model based on the first pattern primitive <i>a</i> as root node	196
6.13(a)	Tree representation of the predictive model based on the first pattern primitive <i>a</i> as root node	197
6.13(b)	Tree representation of the predictive model based on the first pattern primitive <i>a</i> as root node	198
7.1	Block diagram of the KWS system	216
7.2	A sample Malayalam online handwritten document written using E-writemate device	218

7.3	Samples from the word dataset	220
7.4	Stages of preprocessing	221
7.5(a)	Malayalam online handwritten word അജണ്ട /ajaṅṅa/ before pre-processing	223
7.5(b)	Malayalam online handwritten word അജണ്ട /ajaṅṅa/ after preprocessing	223
7.6(a)	Direction of a stroke point in the Malayalam character ത <ta>	225
7.6(b)	Height of a stroke point in the Malayalam character ത <ta>	225
7.6(c)	Curvature of a stroke point in the Malayalam character ത <ta>	226
7.7	Precision and Recall values obtained in the Keyword Spotting experiments	229

List of Tables

Table No	Title	Page No
2.1	Summary of the studies related to segmentation of two dimensional curves	15
2.2	Summary of studies related to segmentation of online handwritten characters in various scripts	20
2.3	Summary of OHCR studies specific to segmentation	27
2.4	Summary of major studies in real-time recognition of online handwritten characters	30
3.1	Vowels and Dependent signs in Malayalam	41
3.2	Usage and sign of <i>Anusvaram</i> , <i>Visargam</i> and <i>Chandrakkala</i>	42
3.3	Malayalam Diphthongs	43
3.4	Malayalam Consonant <i>Varga</i> Classification	43
3.5	Malayalam Consonants other than <i>Varga</i>	44
3.6	Malayalam <i>Chillu</i> Consonants	45
3.7	Rules for the formation of compound letters in Malayalam	46
3.8	Number of samples from various methods and devices	52
4.1	Patterns primitives present in the Malayalam characters	63
4.2	Representation of Malayalam vowel characters using pattern primitives	68
4.3	Representation of Malayalam consonant characters using pattern primitives	69
4.4	Reference set of Malayalam characters marked based on pattern primitives	75
4.5	Frequency of occurrence of pattern primitives in Malayalam characters	81
4.6	Pattern primitives in the descending order of frequency of occurrence	89
4.7	Segmentation accuracy of the RDP algorithm	94

4.8	Conditions and corresponding directions in EDFC	96
4.9	Shapes in Malayalam characters and corresponding direction codes	99
4.10	Segmentation accuracy of Eight Direction Freeman Code based segmentation algorithm	101
4.11	Segmentation accuracy of the combined approach for segmentation	106
4.12	Direction category code string representation of the Malayalam vowel characters	108
4.13	Segmentation accuracy of the combined approach for segmentation through the automated method	111
4.14	Average segmentation rate of the Combined approach	112
5.1	Average pattern primitive lengths of the character samples അ <a> , സ <sa> , ഹ <ha>	129
5.2	Direction Category Code Features (DCCF) of pattern primitives	130
5.3	Character clusters based on the number of pattern primitives	134
5.4	Pattern primitive features of the characters in cluster A	136
5.5	Pattern primitive features of the characters in cluster B	137
5.6	Pattern primitive features of the characters in cluster C(Up)	139
5.7	Pattern primitive features of the characters in cluster C(Down)	140
5.8(a)	Pattern primitive features of the characters i4 cluster D(Up)	141
5.8(b)	Pattern primitive features of the characters in cluster D(Down)	141
5.9	Pattern primitive features of the characters in cluster E	142
5.10	Pattern primitive features of the characters in cluster F(L)	144
5.11	Pattern primitive features of the characters in cluster F(NL)	144
5.12	Pattern primitive features of the characters in cluster G	145
5.13	Number of features in various clusters	147

5.14	Performance scores for cluster A	149
5.15	Performance scores for cluster B	150
5.16	Performance scores for cluster C(Up)	151
5.17	Performance scores for cluster C(Down)	152
5.18	Performance scores for cluster D(Up)	153
5.19	Performance scores for cluster D(Down)	154
5.20	Performance scores for cluster E	155
5.21	Performance scores for cluster F(L)	156
5.22	Performance scores for cluster F(NL)	157
5.23	Performance scores for cluster G	154
5.24	Average recognition accuracies for each cluster	159
6.1	Modified labels of pattern primitives	165
6.2	Pattern Primitive String (PPS) representation of Malayalam handwritten characters	166
6.3	Features of the pattern primitives present in the Malayalam vowel characters	173
6.4	Recognition accuracies obtained for the OHCR experiments using Malayalam vowel characters based on DFA	176
6.5	Occurrence of a pattern primitive after a selected pattern primitive obtained for the Malayalam characters	179
6.6	Transition of first to second pattern primitives and the corresponding characters	185
6.7	Reduction in the number of classes in character prediction	200
6.8	Reduction in Writing Time (RWT) for the Malayalam characters	203
6.9	Features of the nine pattern primitives constituting the Malayalam vowel characters	205

6.10	Prediction accuracies obtained for the Malayalam vowel characters	211
7.1	Details of the Malayalam online handwritten keyword dataset	220
7.2	List of features identified for the Malayalam online handwritten words for KWS experiments	224

List of Abbreviations

ADP	: Accurate Dominant Points
BDLSTM	: Bi-Directional Long-Short Term Memory
CUOHDB	: Calicut University Online Handwriting Data Base
CV	: Consonant-Vowel
DFA	: Deterministic Finite Automata
DTW	: Dynamic Time Warping
DWT	: Daubechies Wavelet Transform
ED	: Euclidian Distance
EDFC	: Eight Direction Freeman Code
HCI	: Human Computer Interaction
HMM	: Hidden Markov Model
KNN	: K-Nearest Neighbor
KWS	: Key Word Spotting
LPF	: Low Pass Filter
NLP	: Natural Language Processing
NN	: Neural Networks
OCR	: Offline Character Recognition
OHCR	: Online Handwritten Character Recognition
RDP	: Ramer Douglas Peucker
SLP	: Single Layer Perceptron
SPR	: Syntactic Pattern Recognition
SVM	: Support Vector Machine

1.1. Background

Communication is an innate part of existence and the core of the human social system. For the human being, it is not just for meeting basic needs; instead, a social and cognitive manifestation of human psychology. The current era is endowed with advanced technologies that have fastened the growth of communication in every field. A remarkable contribution in this field is Human-Computer Interactions (HCI) that simulates machines as humanoids in all aspects. Speech and handwriting have been the major players in HCI for years, as they are the ineluctable part of Natural Language Processing (NLP). Due to the variance and disparities in languages over continents challenged the growth of HCI in speech and handwriting. The tremendous technological growth paved the way for sophisticated devices and techniques to overcome the dilemma in HCI.

Handwriting recognition has been a potential research area in human-machine communication for decades. Compared to other communication modes, handwriting recognition is computationally less expensive and potentially more accurate; especially in ambient noises. In Handwritten Character Recognition (HCR), the handwritten data is converted into machine-readable text, using specialized hardware and software. The two approaches in HCR, offline and online, are also embarked as a result of technological progress. In offline HCR, the handwritten images are scanned and recognized using suitable pattern recognition techniques. With the

advancement of technology, there introduced a way for acquiring handwriting, using digitizers and storing them as stroke series. These spatio-temporal handwriting sequences, also called digital ink, considerably reduce storage. In online HCR (OHCR), recognition is performed on these digital inks.

Over the past few decades, the research in OHCR has been tremendous, and it achieved promising results globally. The ease of access in touch screen displays diversified the studies in OHCR to realize faster recognition. Systems with real-time recognition capabilities are also evolved from extensive research to cope up with rapidly changing technology. But in India, text input in local Indic languages is a complex problem. On the other hand, the use of handwriting is widely entrenched in the domestic environment, government, and business ecosystems and forms the basis for recordkeeping and communication. In this context, technology for OHCR in Indic languages and scripts can play a significant role in promoting IT in these languages, especially in Malayalam.

Among the 22 official languages and ten scripts in India, Malayalam is primarily used in Kerala and the union territories of Lakshadweep and Mahe. Malayalam consists of 11 vowels, 36 consonants, 5 *chillus*, 2 diphthongs, and 15 modifier symbols. It is considered as the complex one to be entered into a computer, as it involves multiple keystrokes per character. The large alphabet size of the Malayalam script, two-dimensional structure of the characters, inter-class similarity among some letters, writing style issues, language, regional specific usages, etc., are significant obstacles in recognizing Malayalam characters. The potential use of online handwriting recognition in

Malayalam scripts is a well-studied problem. However, studies on recognition schemes based on structural features suitable for real-time applications are comparatively lesser in Malayalam. Hence, an in-depth study is to be initiated for developing OHCR systems in Malayalam catering to the essentials of real-time applications.

1.2. Motivation

Research in OHCR for Indic scripts has been gaining momentum for the last few years. There is a higher demand for simplifying the communication between computing devices and scripts used by various linguistic communities in India. But, the structure of the scripts and various writing styles pose challenges in materializing this aim. To fulfill hassle-free communication between computing devices and scripts, we require customized techniques for OHCR.

The prime motive of this study is to address Malayalam online handwritten character recognition as a Syntactic Pattern Recognition (SPR) problem, by considering their structural and directional properties. Since a standard database is necessary to conduct various experiments involving data, this study proposes a Malayalam online handwritten character database to accommodate different writing styles. Moreover, familiarizing the major preprocessing techniques for these characters is also a part of the proposed study.

The direction and structure of Malayalam online handwritten characters have a decisive role in recognizing them, as in the case of other Indic scripts. The visual similarity of various Malayalam characters is also highly relevant

in identifying them. Studies that describe the structural and directional features of character segments are comparatively lesser in Malayalam OHCR. An extensive analysis of these properties and understanding the smallest identifiable units, termed pattern primitives in Malayalam online handwritten characters, are also a part of the study. The analysis of pattern primitives and representation of characters using these pattern primitives are proposed in the study with the prime focus. A study of the frequency of occurrence of pattern primitives in every character is also proposed in the work.

Segmentation of pattern primitives from Malayalam characters using various algorithms is proposed in the study. The performance of pattern primitives in recognizing Malayalam characters is to be analyzed with existing methods. The study suggests a class-modular approach in character recognition to effectively address the advantage of pattern primitives by clustering the characters based on the number of pattern primitives present in each character.

Faster recognition has been an indispensable demand for computing devices over the years, which is further exacerbated with the onset of touchscreen interfaces. A predictive model for the real-time recognition of Malayalam handwritten characters is proposed in the study. The pattern primitives extracted from the characters and their transition sequences are the major components of the proposed model. The study also focusses on the most frequent first pattern primitive, its writing direction, and most frequent pattern primitive transition present in various Malayalam online handwritten characters. A Deterministic Finite Automata (DFA) based recognition scheme, using Pattern Primitive String (PPS) representation of the Malayalam

handwritten characters, is also proposed as a part of the work. In the real-time prediction model, the pattern primitive transition sequences are to be analyzed in detail and based on this, tree representations for the real-time predictions of the characters are also proposed. Finally, as a real-time application, a Keyword Spotting (KWS) system, that can be effectively used to retrieve online handwritten Malayalam documents, is also proposed.

1.3. Outline of the Thesis Organization

The outline of the thesis and the organization of various chapters are described as follows. The necessary information, concerning the previous studies in online recognition of handwritten characters, is described in chapter 2. Various segmentation techniques, recognition schemes using pattern primitives, and real-time recognition of online handwritten characters are also reviewed in this chapter. A review of keyword spotting in handwritten documents is also given in this chapter.

Chapter 3 describes the orthographical categorization of the Malayalam script, within the constraints for the formation rules of vowels and consonants, in detail. The orthography of various writing units specific to Vowels, *Anusvaram*, *Visargam*, *Chandrakkala*, Diphthongs, *Chillu*, and Consonants is also detailed. An online handwritten character database for Malayalam from various handwritings is also created as part of the study in this chapter. The database named, CU-OHDB (Calicut University Online Handwriting Data Base) consists of 30800 samples of 44 single stroke Malayalam online handwritten characters (8 vowels and 36 consonants) that are properly labeled and categorized. Various preprocessing techniques applied to the dataset are also described in this chapter.

In chapter 4, a detailed study is conducted to identify the unique segments present in the 44 selected characters, and a set of such segments, known as pattern primitives, is obtained. There are 26 pattern primitives for these 44 Malayalam characters. Every pattern primitive is assigned a unique label for identification purposes, and every character is represented using these labels. The frequency of occurrence of various pattern primitives present in these characters, is also analyzed in detail. A reference set for the 44 Malayalam characters, corresponding to the pattern primitives, is also procured. The segmentation scheme for partitioning the characters into pattern primitives is proposed in this chapter. In the first method, the well-known Ramer Douglas Peucker (RDP) algorithm to approximate planar curves, is used for extracting segmentation points in a character. The second method explains segmentation using the directional properties of the character. For this purpose, the Eight Direction Freeman Code (EDFC) algorithm is used. As a better method for getting more précised segmentation points, a segmentation scheme by combining RDP and EDFC algorithms is implemented. The combined approach using RDP and EDFC algorithms to segment pattern primitives is novel and unique for OHCR, with a profound understanding of the Malayalam language and its writing units. The accuracy of the segmentation schemes are obtained through visual comparisons with the reference character set. An automated method for obtaining the accuracy of segmentation points is also implemented and compared with the visual comparison method.

In chapter 5, a handwritten character recognition method, incorporating pattern primitives is described. A class-modular approach is proposed, in which the characters are clustered and features are extracted to differentiate the characters inside these clusters. The projected classification technique is

Support Vector Machine (SVM), and it is implemented inside every cluster. The recognition accuracy of the class-modular approach proves that the pattern primitives are suitable candidates for Malayalam OHCR systems.

The predictive model for real-time recognition of Malayalam handwritten characters, based on pattern primitives, is presented in chapter 6. Two approaches are proposed in the model. In the first approach, Deterministic Finite Automata (DFA) based OHCR, using Pattern Primitive String (PPS) representation of characters, is described. In the next approach, a real-time prediction model for OHCR is described. The transition sequences of pattern primitives that constitute each character are analyzed in the predictive model. These pattern primitive transition information, obtained from the characters, are utilized for constructing a tree structure for the possible predictions of every character in a real-time environment. The average Reduction in Writing Time (RWT) is the significant advantage of the predictive model to be incorporated in Malayalam OHCR. Experiments are also conducted to verify the effectiveness of the model using online handwritten character samples taken from CU-OHDB dataset. The prediction part of the system is quick and straightforward to redefine existing systems that recognize the handwritten characters, only after completing the writing process. The model is useful in building a fast handwritten character recognition system to be realized in real-time environments.

In chapter 7, a Keyword Spotting (KWS) system for online handwritten Malayalam documents is implemented. This study is the first of its kind in Keyword Spotting (KWS) for Malayalam online handwritten documents to

the best of our knowledge. The method is adaptable in realizing a Keyword Spotting (KWS) system for online handwritten Malayalam documents.

The findings and contributions of the thesis, followed by future research directions, are summarized in chapter 8. The figure 1.1 shows the entire outline of the chapter organization in the thesis.

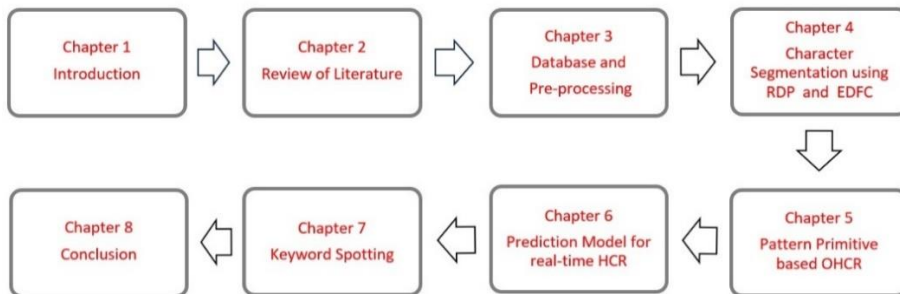


Fig 1.1 Thesis organization

Review of Previous Works

2.1. Introduction

Pattern recognition studies address the machine understanding of various patterns and how they are differentiated from one another. Despite years of research, a general-purpose recognizer for patterns remains an elusive goal. Handwritten Character Recognition (HCR), one of the most studied and demanded pattern recognition problems, is diverse in all its aspects, and that includes OCR (Offline Handwriting Character Recognition) and OHCR (Online Handwriting Character Recognition).

Reviewing past studies is essential in developing more diversified approaches in any research. In this chapter, a review of various literature on pattern recognition studies of online handwritten character recognition is conducted. The study primarily focuses on the segmentation of online handwritten characters and their recognition. Various segmentation techniques applied to two dimensional patterns and their applicability in online handwritten characters are also reviewed in this chapter. Studies that focus on segmentation based approaches and recognition of online handwritten characters based on pattern primitives (unique sub-character units) are also explored from previous studies. Studies on real-time recognition of handwritten characters are also reviewed to adopt suitable techniques for online handwritten character recognition. Existing studies on Keyword Spotting (KWS) techniques applied to handwritten document images and online handwritten documents are also reviewed in this chapter.

The remaining part of the chapter is organized as follows. Section 2.2 reviews the studies on various segmentation techniques used in pattern recognition, majorly focusing on handwritten characters. Section 2.3 of the chapter reviews various studies with a prime focus on syntactic pattern recognition approaches. A review of real-time studies in handwriting recognition specific to OHCR is given in section 2.4. In section 2.5, various studies on Keyword Spotting (KWS) in handwritten documents are reviewed. The review is concluded in section 2.6.

2.2. Review of various Segmentation Techniques used in Pattern Recognition Applicable to Handwritten Characters

A means for converting handwriting to machine-readable text has gained great research attention from past centuries. With the invention of the digital computer and modern technologies, scientific studies in this area became more exigent. A milestone on the move was the invention of a handwritten digit recognizer by IBM in 1966 [1].

Charles. C. Tappert *et al.* describe the distinction between online and offline recognition in a vivid manner in an intensive study [2]. Moreover, their analysis of various studies shows that online recognition takes advantage of offline recognition, mainly due to the difference in data acquisition. They further explore that online devices capture the dynamic and temporal information of the writing, like the number of strokes in writing, the writing order of the strokes, the direction of each stroke, and the speed at which the data is recording. The major challenges in recognizing online handwriting listed in the study are the similarity of variable stroke classes, cursive writing styles, and the speed at which writing occurs.

Due to the difficulty in analyzing patterns like handwriting, there is a strong requirement for reducing complex patterns into apparently simpler ones without losing their essence. There are many investigations done in this direction to address the pattern reduction problems, especially for handwriting, where a lot of ambiguity exists. One of the early attempts of Eden *et al.* demonstrated that a set of primitives with a set of rules could produce handwritings that are indistinguishable from humane [3][4]. Their studies had essential implications in handwriting recognition. A method to describe complicated two dimensional patterns into one dimensional patterns is evolved, and also a way to distill the essential information from intricate patterns in a more straight forward way. As a result, segmentation techniques were extensively used in handwriting recognition. Segmentation of the strokes is considered as a preprocessing technique in the studies by Charles. C. Tappert *et al.*, and it is also differentiated into internal segmentation and external segmentation [2]. Segmentation inside the stroke is called internal segmentation, and the isolation of various writing units into words and characters before recognition is termed external segmentation. They enlighten the fact that internal segmentation also requires certain recognition.

Most of the works reported in early literature extensively give comprehensive coverage on image segmentation techniques, and these methods also need to be reviewed. Nikhil R Pal and Shankar K Pal reviewed the methods used in image segmentation in greater depth [5]. Their study observes that not a single method exists, which is performing better for all images. One of the works studied by them gives a clear division between clustering and segmentation, as clustering is a grouping done in measurement space. In contrast, segmentation is done in the spatial domain. Standard

segmentation methods structured under their study are thresholding, relaxation, Markov Random Fields, surface-based segmentation, and edge detection. In edge detection, the edges of the image are identified, and the image is segmented. Edge detection has a crucial role in pattern recognition, especially in handwriting recognition.

In the work of Herbert Freeman, arbitrary geometrical configurations are encoded using direction code in a simple way [6]. The study explores the fundamental aspects of chain coding using various minimal waveforms. The rotation, curve length, inverse, path reversal, closure, loops, intersections, and area are found simply using operations on chain code sequences of the curve. The study reveals that the freeman chain coding is not only a simple encoding scheme to be employed for arbitrary geometrical configurations but also a better tool to perform various transformations on them. Moreover, the insights into the segmentation scheme give many ideas regarding the division of a curve. Consequently, the large size of segments leads to a minimum number of segments. Still, it increases complexity to represent them using various mathematical functions, whereas the lesser sized segments create more of such in number, and a straight line approximation may suffice. For this reason, equal segmentation of the curve is employed as a simple mean but not efficient. Another approach illustrated is the segmentation based on the properties of a curve, which is relatively efficient.

In another work by Herbert Freeman and Jeremy M Glass, a quantization scheme for the curve is implemented using linear segments and direction code [7]. The method illustrated in the work is grid-intersect quantization, where a grid mesh is superimposed with the curve. The points on the mesh where the

curve intersects are marked, and the nodes close to these points are marked as curve points. Therefore, each curve point has eight neighbor points in the grid, and the next curve point in the sequence must meet any of these neighboring points. After all, these points are coded using direction code, and the work illustrates a procedure to reconstruct the curve from the direction code thus obtained. The study gives insights into the importance of chain codes in the quantization process of curves.

In a work, URS Ramer implemented an iterative procedure for the polygonal approximation of planar curves [8]. The procedure is to divide a subset into two subsets. Here, the division is based on the criteria of maximum distance and approximated straight line. Experiments in this work integrate the fact that the division of point of maximum distance from the approximating straight - line results in a low number of vertices and very little extra computation. For closed curves, the starting and ending points in the approximated polygon are the same as that of the original curve. Further, the work compared chain code and found that the performance is comparable in polygon approximations, but the processing time is considerably shorter than chain coding.

A study on the number of points required to represent a curve without losing the essential properties is conducted by David H Douglas and Thomas K Peucker on digitized lines and caricatures referred to as the Ramer Douglas Peucker algorithm [9]. The work demonstrates a method where the first point is marked as an anchor, and the last point as a floatingpoint, and these two points define a straight line segment. The intervening points along the curved line are examined to find the greatest perpendicular distance between it and

the straight line defined by the anchor and floater. The point is retained in the work by measuring the tolerance distance measure. Various measures of tolerance and a corresponding number of points are also listed in the work. The study forwards an idea about reducing the number of issues with various tolerances and is crucial in the segmentation of open curves.

In another work by Herbert Freeman, a detailed study of computer processing of line-drawing images is conducted [10]. The work emphasizes chain coding aspects of the images and explains how these codes can effectively extract features. Further, the study describes detailed procedures for rotating, expanding, and smoothing line structures, and a correlation technique for establishing the degree of similarity is also detailed. It is also observed that the chain coding scheme is established from the grid intersect quantization technique popular in that period. A method for detecting sharp corners in a chain coded plane curve is described by Herbert Freeman and Larry S Davis [11]. A chain code is scanned by moving the variable line segment, and the angular differences between successive segment positions are used to measure local curvature along the chain. It follows that a distinction can be made between a straight-line segment and curvature after the linear scanning. The study infers those three incremental curvature regions characterize a corner in a chain coded curve, and a possible measure for the prominence of a corner is the product of the lengths of the uniform chain sections and the angle of discontinuity.

The shape is described using critical points in another work of Herbert Freeman [12]. The work extracts the boundary line of the object, and shape descriptors are found from the boundary line. The corners are described as

points of discontinuity as one shape descriptor followed by inflection points, curvature maxima, intersections, and points of tangency. In this study, curves are divided into segments and use relatively simple features to characterize them. This study also transforms the shape into chain codes, and a line segment scan is used for detecting discontinuity points. The methods illustrated in the work are useful for both closed and open curves.

Li De Wu extended his studies by analyzing various studies of Herbert Freeman and describes a procedure to identify lines from chain codes by pointing out the three criteria to be met by chain coded lines [13]. Accordingly, they are 1) The presence of two basic directions, which can differ by modulo eight or unity 2) One of these values occur singly. 3) Successive occurrences of the principal directions occurring alone are to be uniformly spaced. A mathematical analysis of the chain code of a line is performed to satisfy the above criterion simply throughout the study. A summary of the studies related to the segmentation of two-dimensional curves is shown in table 2.1.

Table 2.1 Summary of the studies related to segmentation of two dimensional curves

Sl.No	Authors	Year	Work	Method
1	Herbert Freeman [6]	1961	Operation on arbitrary geometrical shapes	Eight Direction Freeman Code
2	Herbert Freeman and Jeremy M Glass [7]	1969	Quantization of 2-d curves	Linear segments and Eight Direction Freeman Code
3	URS Ramer [8]	1971	Approximation of 2-d curves	Ramer Douglas Peucker Algorithm

Table 2.1 Summary of the studies related to segmentation of two dimensional curves (cont.)

Sl.No	Authors	Year	Work	Method
4	David H Douglas and Thomas K Peucker [9]	1973	Approximation of 2-d curves with the number of points	Ramer Douglas Peucker Algorithm
5	Herbert Freeman [10]	1974	Operation on lines	Eight Direction Freeman Code
6	Herbert Freeman and Larry S Davis [11]	1976	Sharp corners in 2-d curves	Eight Direction Freeman Code
7	Herbert Freeman [12]	1978	Critical points	Eight Direction Freeman Code
8	Li De Wu [13]	1982	Line encoding	Eight Direction Freeman Code

In the work of Samir Al Emami and Mike Usher, online recognition of handwritten Arabic characters is performed using segmentation based on the length and slope [14]. The approach followed here is considering three neighboring points as a single unit and finding the length and slope angle of the two sides of the middle point followed by a threshold value matching to keep the midpoint as a segment point. The method uses direction code after segmentation as a feature of the segment even though the slope angle is similar to a direction change. Richard G Casey and Eric Lecolinet conducted a survey on various techniques that pertain to the segmentation of characters in the offline domain [15]. Holistic and character based approaches for word segmentation and character segmentation are listed. Hidden Markov Models (HMM) are described with numerous examples and address the advantage of HMM due to the property of state-to-state transition within a character and letter-to-letter transition in a word. They are used for both character and word level recognition. Claudio D Steffano *et al.* presented a segmentation scheme for online handwriting, using Daubechies Wavelet Transform (DWT) and

curvature maximum [16]. Accordingly, they suggest a method using histogram projection and finding the most salient curvature maxima among the candidates to obtain the segmentation points. The experiments in the study achieved a segmentation rate of 65.68% for an in-house data set comprising 2000 words with an average segmentation time of 600 milliseconds. In the work of Jianying Hu *et al.*, a detailed study on handwritten online Arabic characters is performed [17]. Their research addresses character recognition at the stroke level, with each character concatenating several stroke models. This, in turn, is referred to as segments analogous to other works. The system learns the strokes through rigorous training, and no segmentation scheme about them is addressed. Even though the work is based on HMMs and a little information on segments is given about the state of the character in HMMs.

A survey on various segmentation techniques in character level and word level segmentation for online Arabic text is reported by Musthafa Ali Abuzaraida *et al.* [18]. Their study reveals that direction codes, the Douglas Peucker algorithm, slope values of the strokes, and pen movements are extensively used for Arabic character recognition. It is also reported that a maximum rate of 92% has been obtained in the works with the Douglas Peucker algorithm for both dependent and independent recognition. Houcine Boubaker *et al.* presented a scheme for the segmentation of Arabic characters based on two methods, namely crisp segmentation and fuzzy segmentation [19]. The methods illustrated as crisp segmentation are based on particular points obtained by tangent detection for valleys and extreme points for angular points. Consequently, the disadvantage of crisp segmentation is listed as local segmentation and introduced a new scheme fuzzy segmentation based on a confidence score on a particular point illustrated in their work.

Detection of character boundaries and segmenting the character into segments based on directional features are presented in their work by Mohamed Elhafiz Mustafa and Hozeifa Adam Abd Alshafy for Arabic character segmentation [20]. They listed directional features as a combination of cosine and sine values for each x and y value of the online stroke sequence. They concluded their work by stating the importance of segmentation in simplifying the recognition of online handwriting. Mustafa Ali Abuzaraida and Salem Meftah Jebriel identified suitable reduction values for the online handwritten digits for the Douglas Peucker algorithm [21]. As the reduction value increases, the distortion in shape occurs considerably. They demonstrated that a 100% reduction factor for the handwritten digits would be dissipated as only a few points could not be identified as a digit. Their study also shows that a reduction in the number of points fastens the processing of the entire system.

Paul L Rosin assessed the behavior of various polygonal approximation algorithms based on the breakpoint stability index, monotonicity index, and consistency index [22]. It is evident from the work that Ramer and Douglas Peucker algorithms are excellent in attaining all the standard indexes specified. Dileep K Prasad *et al.* explores the idea of three dominant point extraction methods, namely Ramer-Douglas-Peucker, Masood, and Carmona, and illustrates a nonparametric method for the techniques [23]. A modified approach for all the algorithms is presented in the work without comparing the performance. A system with a segmentation approach is mentioned by Hesham M. Eraqi and Sherif Abdel Azeem for online Arabic handwriting recognition [24]. The segmentation in the work is based on the horizontal angle and segmentation junctions.

Consequently, the characters are built from different segments based on construction rules. The importance of direction in recognition is coined by A. Alper Atici and Fatos T. Yarman-Vural using a heuristic algorithm [25]. Contours in both directions were plotted, and segmentation is again ruled out in the work based on local minima and local maxima. Segmentation using the RDP algorithm is demonstrated in the work by Hesham M Eraqi and Sherif Abdelazeem for Arabic scripts [26]. Critical points are extracted from the total segmentation points obtained from RDP based on a set of rules pertaining to Arabic script. The method efficiently addressed the segmentation using the above method and attained a segmentation accuracy of 91.27% compared to the other methods.

Stroke segmentation using predefined rules and reference sets is described by Nilanjana B Bhattacharya and Umapada Pal for Indian script Bengla [27]. They developed a ground truth set for a segmentation set of possible points as ground truth and achieved a segmentation rate of 97.89 % compared to the reference set. A rule set based approach for Indian script segmentation similar to phoneme segmentation does odd thinking for the researchers to equalize handwriting recognition similar to speech recognition. The work also makes a remarkable impression that directional features are essential for obtaining expected accuracies, as they obtained a recognition accuracy of 97.68% using directional features. A method to segment handwritten Malayalam characters is presented as a combination of curvature weighted sampling and equidistant sampling in the work of Prabhu Teja S and Anoop M Namboodiri [28]. A recent work by Sukhdeep Sing *et al.* extracts dominant points in the handwritten characters using the RDP algorithm [29]. Further, the method reduces the number of points determined by the RDP algorithm for angular

and direction changes. Based on the above modifications in the feature extraction phase, the work attained a remarkable recognition rate in their work. The importance of direction in extracting the dominant points has been addressed in detail throughout the work. It is clear from the literature on the segmentation of online handwriting, direction and point reduction play a crucial role in segmenting the series into fragments. The study also observed that the segmentation of a character stroke series makes it easy for the recognition systems to classify them in a divide and conquer way. Most of the studies in segmentation used either Freeman Direction Code or the Ramer Douglas Peucker algorithm. This observation directly associates a close connection between segmentation and the techniques of Ramer Douglas Peucker and the Eight Direction Freeman Code. A summary of the major works related to the segmentation of online handwritten characters and their recognition rate is given in table 2.2.

Table 2.2 Summary of studies related to segmentation of online handwritten characters in various scripts

Sl.No	Authors	Year	Work	Method	Accuracy (%)
1	Samir Al Emami and Mike Usher [14]	1990	Segmentation of Online Arabic Handwritten Characters	Eight Direction Freeman Code	-
2	Claudio D Steffano <i>et al.</i> [16]	2007	Segmentation of Online English Handwritten Characters	Daubechies Wavelet Transform	65.68
4	Musthafa Ali Abuzaraida and Ahmad M Zeiki [18]	2010	Segmentation of Online Arabic Handwritten Characters	Ramer Douglas Peucker Algorithm	92%

Table 2.2 Summary of studies related to segmentation of online handwritten characters in various scripts (cont.)

Sl.No	Authors	Year	Work	Method	Accuracy (%)
5	Hesham M. Eraqi and SherifAbdela zeem [26]	2011	Segmentation of Online Arabic Handwritten Characters	Ramer Douglas Peucker Algorithm	91.27
6	Nilanjana B Bhattacharya and Umapada Pal [27]	2012	Segmentation of Online Bengla Handwritten Characters	Rule Set	97.89
7	Prabhu Teja S and Anoop M Namboodiri [28]	2013	Recognition of Online Malayalam Handwritten Characters with Segmentation	Dynamic Time Warping for recognition	94.55
8	Mustafa Ali Abuzaraida and Salem Meftah Jebriel [21]	2015	Segmentation of Online Handwritten Digits	Ramer Douglas Peucker Algorithm	100
9	Sukhdeep Sing <i>et al.</i> [29]	2017	Recognition of Online Gurumukhi Handwritten Characters with Segmentation	Hidden Markov Model for recognition	98.27

2.3. Review of Online Handwritten Character Recognition Studies Focusing on Segmentation

OHCR, being a pattern recognition problem, has so many approaches in previous studies. In this section, a review of past studies pertaining to segmentation approaches in OHCR is conducted. One of the earlier works by K S Fu and P H Swain reports the concept of describing patterns in terms of primitive elements, sub-elements, and their relationships [30]. It is also evident from their studies that a syntactic model of pattern analysis is initiated

by determining the set of primitives in terms of the data of interest. Some methods for determining the primitive elements are illustrated as the system itself learns the primitive elements from training samples based on a sample point and a threshold value.

In another work by Herbert Freeman, the description of shape is mentioned as the key element in pattern recognition [12]. Freeman explains that the shape of a two-dimensional object is conveyed by the curving of the boundary line, and the curving may be regarded as a concatenation of arcs with varying instantaneous radii of curvature. The work facilitates shape description by segmenting the boundary at critical points, namely corners, points of inflection, and curvature maxima.

The method of subdividing a trajectory into metastrokes and recognition using dynamic information for online handwriting is described in the work of Shelya A Guberman and V V Rozentsveig [31]. Bellegarda *et al.* identify five categories of online handwriting recognition techniques: (i) primitive decomposition, (ii) motor models, (iii) elastic matching, (iv) stochastic models, and (v) neural networks where primitive decomposition identifies sub-strokes that form common building blocks for characters such as loops, dots, crossovers, arcs, ascenders, and descenders and also need segmentation of strokes [32]. An intermediate cursive words description language constructed of elements in the form of metastrokes wherein each of the metastrokes is a member of a metric space, the metric space being expressible as a matrix of the likelihood of matching input metastrokes and predefined metastrokes forming a vocabulary is explained by S.A. Guberman *et al.* [33]. Yoshiyuki Yamashata *et al.* extracted sub-patterns from Kanji characters

using directional features [34]. The characters are matched for similarity using structured segment matching, and they observed a higher recognition rate using this.

Curve matching using freeman code is demonstrated by Bo Yu *et al.* with corner detection and curve direction as major characteristics [35]. The work implicates the importance of chain code in the structural matching of curves, which are analogous to pattern sequences of online characters.

The issues in Online Handwriting Recognition are deployed by Vikas Kumar with techniques available for the representation of handwriting and its recognition [36]. In this work, online patterns are represented structurally, statistically or statistical-structural methods and classified using coarse or fine classification. The author describes pattern primitives as the representation of the patterns in a structural way. Mohammad Badiul Islam *et al.* proposed a method for recognizing handwritten Bengali characters using pattern primitives and string comparison with matching scores [37]. Numerical stroke codes represent curves and line slopes in the work followed by a matching matrix for every segment and reference segment. A matching score of 100 is treated as a similar segment, and a maximum score near 100 is treated as the most similar segment from the reference set for an input segment.

Experiments conducted by Samir Al-Emami and Mike Usher to recognize online Arabic characters identified four groups of shapes in the characters, namely beginning shapes, central shapes, termination shapes, and stand-alone shapes [14]. They achieved higher recognition accuracy for the 390 Arabic online handwritten characters using decision trees. Xiaolin Li and Dit Yan Yeung performed the recognition experiments using dominant points and

directions and achieved an accuracy of 91% [38]. Eight direction features are used to extract direction information in a segment, and the dominant points are identified using local extrema of curvature. Template matching using a decision tree was proposed by Scott D. Connell and Anil K. Jain for English alphabets and numerals and achieved an accuracy of 86.9% [39]. A sub-stroke based recognition for online handwriting recognition is proposed by Karteek Alahariet *al.* using Fischer Discriminant Analysis for English alphabets and digits [40]. The system used the alignment of strokes, modeling of strokes, and discriminating potential of strokes in sequence and achieved promising results for recognizing the character using HMM classifier.

Ahmad T and Al Taani reported a method using primary and secondary primitives for the recognition of digits [41]. The key idea in the work is to form digit patterns by naming primitives excavated from the samples and combining them in various sequences incomparable with existing samples. The above work achieved greater recognition accuracy with lesser computation involving grammar identification, and it was independent of upward and downward drawings. Fadi Biadsi *et al.* applied HMM for online Arabic handwritten characters and achieved better results for writer dependent and writer independent experiments [42]. The features were local angle features, super segment features, and loop features, and modeled the system using discrete HMM and word-part dictionary. Aleksei Ustimov *et al.* describe a less complex approach by converting characters into graphs of segmented points [43]. The graph is converted into codes with an automaton of 36 states with 36 X 36 transitions and achieved a recognition rate of 71.94% in lesser time. Houcine Boubaker *et al.* segmented Arabic characters into sub-components and represented every segment using Fourier

descriptors and geometrical features [44]. The method suggested a way to reproduce handwritten characters by decomposing them as segments and extracting features.

A primitive based approach for handwritten Bengali alpha-numeric characters is presented by Abhijith Dutta and Santanu Chaudhury [45]. They identified the junctions where primitives meet using local minima and local maxima. The features are extracted using the Gaussian kernel for various parameter values. They achieved an average recognition accuracy of 70% for handwritten characters using the above method. Priyanka Das *et al.* excavated pattern primitives from Bengali handwritings in a faster and simple way [46]. Straight lines and other curved primitives are detected from the character, and a matching score is obtained by comparing it with existing reference shapes; an optimal score is obtained to identify the existence of a particular shape, and finally, the primitives are converted to Freeman code.

A naive approach has been implemented in the work of Ujjwal Bhattacharya *et al.* for Bengali online handwritten characters [47]. Freeman direction code and center of gravity are used as the features, and the system achieved an average recognition rate of 83.61%. It is also evident from the work that directional information is crucial in classifying Bangla Characters.

A novel online recognition system for Devanagari script with primitives and fuzzy directional features is detailed by Lajish VL and Sunil Kopperappu [48]. They referred to a primitive set of 14 for the Devanagari script and compared it with the test data, and achieved better results than directional features. The method used second-order statistics for representing primitives as features, and it is pointed out that the comparison is computationally

effective. An accuracy of 82.75% is achieved by the method using fuzzy directional features. A segmentation-based approach for online Bengali handwriting is described in the work of Niranjana Bhattacharya and Umapada Pal [27]. Segmentation is performed by identifying busy zones, and accuracy is obtained by testing against a ground truth set with ideal segmentation points. The work reported an accuracy of 97.89% for segmentation and 97.68% for recognizing the strokes using directional features and Support Vector Machine (SVM). A rule-based segmentation is also described in the work by converting online sequences to offline images for experimenting on pixel levels. Directional string formation and recognition using the Levenshtein distance matrix are depicted in S Dutta Chowdhury *et al.* for online Bangala handwriting [49]. In their work, direction strings are reduced to a single occurrence from repeated similar codes. The edit distance between the unknown samples and known samples results in a metric through the Levenshtein distance method. The metric upon analysis displays the most similar character with minimum edit distance. The exploited method is simple, and the redundant samples are also eliminated from the known group for faster recognition. Recent work by Sukhdeep Singh *et al.* demonstrated recognition of online handwritten samples from the UNIPEN dataset using dominant points [29]. These dominant points are extracted using the Ramer Douglas Peucker algorithm with various distance measures and prepare the feature vector based on key points and curve directions between key points of the stroke.

A summary of the major studies reported in this section which includes the Syntactic Pattern Recognition approach in OHCR, is shown in table 2.3.

Table 2.3 Summary of OHCR studies specific to segmentation

Sl.No	Authors	Year	Language	Features	Classifier	Sample Size	Accuracy (%)
1	Samir Al-emami and Mike Usher [50]	1990	Arabic	Structure analysis with direction	Decision Trees	390	100
2	Dit-Yan Yeung and Xiaolin Li [38]	1996	English	Dominant points	Dynamic Programming with Elastic Matching	1860	91
3	Anil K jain and Scott D Connel [39]	2001	English	Template based	Decision Trees	17928	86.9
4	KarteekA lahari <i>et al.</i> [40]	2005	English	Fischer Discriminant Analysis	HMM	1200	96
5	Ahmad T and Al-Taani [41]	2005	Arabic digits	Feature strings	Finite Automata	10000	95
6	Fadi Biadsy <i>et al.</i> [42]	2006	Arabic	Letter shape models	HMM	6220	94.40
7	Ujjwal Bhattacharya <i>et al.</i> [47]	2007	Bangla	Direcion code	MLP	7043	83.61
8	Aleksei Ustimove <i>t al.</i> [43]	2010	Turkish	Constructive learning	Automata	12000	71.94

Table 2.3 Summary of OHCR studies specific to segmentation (cont.)

Sl.No	Authors	Year	Language	Features	Classifier	Sample Size	Accuracy (%)
9	Lajish VL and Sunil Koppurapu [48]	2010	Devanagari	Fuzzy directional features	Viterbi	545	82.75
10	Niranjana Bhattacharya and Umapada Pal [27]	2012	Bangla	Rule set	SVM	5448	97.68
11	S Dutta Chowdhury <i>et al.</i> [49]	2013	Telugu	Feature strings	Levenshtein distance	45217	87.10
12	Sukhdeep Singh <i>et al.</i> [29]	2017	Gurumukhi	Dominant points using Ramer Douglas Peucker algorithm	SVM	39200	97.29

2.4. Review of the Studies related to Real-Time Recognition of Handwritten Characters

There is a strong demand for recognizing handwritings in a simple way with the fast development of computer systems. Online handwriting recognition is a superior choice coined by most researchers but brought in a time element between writing and recognition. Many studies addressed the solution to eliminating this gap with a novice term inculcated as real-time recognition. Charles C Tappert *et al.* says that online handwriting is also known as real-time or dynamic recognition [2]. But the delay caused by the devices made a difference between online and real-time recognition. The need for a real-time

recognition system became an essential component with the technological advancement in touch input devices. A means to transform the data faster is possible through real-time recognition. Hence, real-time recognition research has a few decades of experiments apart from online handwriting recognition, which was initiated in the 1960s.

T. S. El-Sheikh and S. G. El-Taweel demonstrated a real-time recognition system for Arabic characters in one of their earlier works [51]. They describe the shape of the Arabic character, which depends on the position in the word, and divide them into four disjoint sets with various features like aspect ratio, number of minima and maxima, and the relative distance between the first and last points. They achieved a recognition rate of 98% with a recognition time of 0.3 seconds. Statistical methods are detailed in the work of Krishna S Nathan *et al.* for unconstrained handwriting [52]. They used HMM in modeling characters and words with fast and detailed match recognition. The system reported better accuracies in unconstrained and writer-independent modes with quicker recognition time ranging from 0.4 to 0.48 seconds. The requirements of a real-time handwriting recognition system are reported in the work by Bartosz Paszkowski *et al.* [53]. They identified a higher accuracy rate as the essential requirement for such a system with invariance on geometric transformations. The system must work on low computing machines and receive data from pointing devices or finger touch. They must immediately display the recognized unit on the screen for the fixed set of classes. Experiments conducted in the work used Single Layer Perceptron (SLP) as a classifier with a recognition rate above 97% with an average recognition time of 8 milliseconds, which is fair for real-time recognition systems.

Alberto Beltran and Sonia Mendoza described a real-time handwriting recognizer for mobile devices [54]. The work details the importance of directional features in the recognition of handwritten characters. They implemented the system with a computationally less expensive procedure for English alphabets and achieved higher recognition accuracies acceptable to real-time environments.

Da Han Wang *et al.* proposed a real-time recognition approach for sentence-based Chinese handwriting input [55]. With a dynamically maintained segmentation–recognition candidate lattice, the system integrates multiple contexts, including character classification, linguistic, and geometric context. For every new stroke, dynamic text line segmentation and character over-segmentation are performed to locate the position of the stroke in text lines and updated. Candidate characters are generated and recognized to assign candidate classes, linguistic context, and geometric context involving the newly created candidate characters. The candidate lattice is updated while the writing process continues. After the threshold time exceeds, the system finalizes the search to interpret the character sequence produced in the entire process and recognize it as the sentence.

Qi Song and Zehua Gao described a real-time handwriting system for mobile devices [56]. Pyramid Histogram of Oriented Gradients are used in work as features and achieve faster recognitions than other methods with a recognition speed of 20 milliseconds per digit.

A real-time recognition system for digits is reported using deep learning for embedded systems by Vinh Dinh Nguyen *et al.* [57]. The proposed digit recognition system used NVIDIA’s Jetson Tx1 Developer Kit and Caffe

framework in a real-time environment. The above multiple digit recognition system's accuracies were 98.23%, trained, and tested on messily handwritten numbers but the results of testing as reported on multiple handwritten digits on a physical whiteboard or paper (real-time) is lower due to camera conditions and writing angles. A summary of the major studies in real-time recognition of online handwritten characters is listed in table 2.4.

Table 2.4 Summary of major studies in real-time recognition of online handwritten characters

Sl. No	Authors	Year	Language	Features	Classifier	Sample Size	Accuracy (%)
1	S.G.El-Taweel and T.S.El-Sheikh [51]	1990	Arabic	Position, shape	Hierarchical classifiers	700	98.00
2	Krishna S Nathan <i>et al.</i> [52]	1995	English	x,y coordinates	HMM	21000	81.10
3	Alberto Beltran and Sonia Mendoza [54]	2011	Spanish	Directional features	Rough classification	1000	98.40
4	Da Han Wang <i>et al.</i> [55]	2012	Chinese	Directional features	MQDF	1,082,220	94.09
5	Bartosz Paszkowski <i>et al.</i> [53]	2007	English	Shape features	SLP	15431	97.83
6	Vinh Dinh Nguyen <i>et al.</i> [57]	2013	Digits	Contours	Deep Learning	10094	98.23

2.5. Review of Keyword Spotting (KWS) in Handwritten Documents

The growth of various technologies in acquiring data, like digital pens and touch screen displays, depicted an enormous amount of data as handwritten document images and online handwritten documents. The storage requirement for handwritten document images is higher compared to online handwritten documents. Keyword Spotting (KWS) is a technique to extract portions of data with specific words from a massive amount of data, either document images or online handwritten documents.

A keyword spotting system is a combination of keyword indexing and retrieval techniques. Indexing is performed for fast retrieval and avoids perplexity in retrievals. Most of the methods are earmarked to spot keywords relevant to handwritten document images, and limited experiments are proposed in keyword spotting in online handwritten documents. In this section, prominent keyword spotting techniques available in the literature for handwritten document images and online handwritten documents are extensively addressed. It is found that the studies about keyword spotting in online handwritten documents are limited in literature, especially for Indic scripts.

2.5.1. Review of Keyword Spotting in Handwritten Document Images

Most of the ancient studies augmented the technology for handwritten document images using normal text matching schemes. But the poor recognition accuracy of these technologies restricted their popularity in word indexing and spotting. An earlier work by Manmatha *et al.* proposed an alternative scheme for indexing handwritten text images [58]. Each page of the document is segmented into words, and the images of the words are then

matched against each other to create equivalence classes. Here each class contained instances of similar words. The Euclidian Distance measures are used to compare the similarity of instances in a class and also out of class instances. They achieved minimum errors for inter-class instances and maximum distances for out-of-class instances. In another work by Manmatah *et al.* describes two methods for word matching in handwritten documents [59]. The first method is based on Euclidian distance measures, and the second method is based on the algorithm developed by Scott and Longuet Higgins, where words are matched by affine transformations. The study reveals that the performance of the algorithm depends on the variability of handwriting. Affine transformations performed well in comparison to Euclidian distance in poor handwriting in their studies. A comprehensive survey on various keyword indexing techniques is conducted by Angelos P. Giotis *et al.* including word retrieval from handwritten documents [60]. A thorough analysis of keyword spotting methods in various historical and modern handwritten documents is presented in a lucid manner. They also agree that the OCR systems are the least efficient in addressing keyword spotting due to the low recognition rate observed in various experiments. The survey also briefs various assessment measures, standard datasets, and performance of the state-of-the-art techniques in keyword spotting. Rashad Ahmed *et al.* surveyed word spotting techniques from handwritten documents [61]. A comparison of word spotting and word recognition is detailed in the work. Extensive coverage of various features pertained to document image retrieval, including gradient, structural, and concavity features, profile-based features, a bag of features, shape coding, and contour, is analyzed in various experiments. Srihari *et al.* make use of binary Gradient, Structural, and

concavity features to spot words in handwritten documents [62]. Text lines are segmented into connected components and used correlation distance for matching two GSC binary feature vectors in the work.

Bai *et al.* describe a word shape coding based approach for keyword spotting in handwritten documents [63]. A total of seven features were extracted from each word. These features are descenders, ascenders, eastward concavity, westward concavity, horizontal line, intersection, and holes. Each word was represented by a sequence of codes based on the position and type of features. DTW technique is used for word matching and achieves higher precision and recall rates.

An Urdu word recognition system was built by Sagheer *et al.* [64]. They segmented the documents into connected components, and by combining neighboring components, generated words using a sliding window. Then, profile and gradient features are extracted for each generated word. SVM is used as the classifier. The experiments are performed on 40 pages obtained from the CENPARMI Urdu Database. The work reported lesser rates compared to other methods. Al Aghbari and Brook proposed an algorithm for retrieving Arabic historical documents [65]. They combined several structural and statistical features. The structural features used are upper, lower profiles, and projection profiles. The statistical features are the ratio between punctuation and main connected parts. Neural networks are used to classify extracted features into word classes. Can and Duygulu proposed a line-based representation for historical document matching, where word images are designated as a set of contour segments [66]. These are approximated to lines using the polygonal approximation Douglas Peucker algorithm. Each line is described in this work using the position, the length, and the orientation for

faster feature computations. Andreas Fischer *et al.* describe a word spotting system based on the character Hidden Markov Model in one of their studies [67]. It reveals that arbitrary keywords are easily spotted in an efficient lexicon-free approach without pre-segmenting text words. It is shown that the proposed learning-based system outperforms a standard template matching method for multi-writer data on the IAM offline database as well as for single-writer data on historical data sets. A keyword spotting method using two approaches is described in the study of David Fernandez, which consists of a hierarchical process and characteristic loci features [68]. Experiments show that the system performed well in comparison to other existing methods. Shingo Uchihashi and Lynn Wilcox exploited the capabilities of chi-square values in determining the index terms [69]. An automated scheme for indexing is described with dynamic programming for matching similar text. A dictionary of possible index terms is created in the work by clustering groups of ink strokes corresponding to words

Keyword Spotting is also addressed in the Indic script, especially in Devanagari and Bengla. Sharada B and Sushma S. N demonstrated an approach for keyword spotting in handwritten Devanagari documents [70]. Using graph editing methods for word matching, they achieved an accuracy of 88.22%. Kartik Dutta *et al.* proposed a framework for keyword spotting in large-scale handwritten word images using deep neural networks for Devanagari, Bangla, and Telugu scripts. They achieved 96.15% accuracy for Bangla scrips [71]. Sharada B *et al.* made a comparison of the performance of keyword spotting system using Euclidean distance and cosine similarity metrics for Devanagari scripts [72]. The average matching accuracy obtained for Manhattan distance is 84% in their work. A system for keyword spotting

of the Tamil language is proposed by Sigappiet *et al.* [73]. The features are projection profile, lower and upper word profile, and background-to-ink transitions features extracted from each segmented word. This approach makes use of HMM to characterize strokes' variations of handwritten characters.

2.5.2. Review of Keyword Spotting in Online Handwritten Documents

Keyword spotting in online handwritten documents is extensively reported in CJK scripts than Indic scripts. Heng Zhang *et al.* proposed a keyword spotting method for Chinese online handwritten character documents to directly estimate the posterior probability (also called confidence) of candidate characters based on the N-best paths from the candidates segmentation-recognition lattice and achieved an accuracy of 91.74% [74]. Heng Zhang *et al.* describes an online handwritten keyword spotting system for Chinese script by using a one-vs-all strategy trained prototype classifier and a support vector machine (SVM) classifier for similarity scoring and achieved an accuracy of 90.88% [75]. A lattice-based method for keyword spotting in online Chinese handwriting using a character lattice based method by generating candidate lattice of N-best list achieved a recall rate of 94.50% using string matching algorithm [76].

An indexing scheme for online handwritten documents is proposed by Sebastian Pena Saldarriaga and Mohamed Cheriet using a word confusion matrix [77]. Alignment for mutually exclusive words and the posterior probability of each word is done using a word confusion matrix. While querying the index for a given keyword using their technique, the complexity $O(\log n)$ is obtained compared to usual keyword spotting algorithms, which run in $O(n)$. Emmanuel Indermuhl *et al.* proposed a new approach for mode

detection (text/non-text categorization) that uses Bi-Directional Long-Short Term Memory (BDLSTM) Neural Networks (NN) [78]. They experimented with the potential of BDLSTM over traditional methods for mode detection, which are usually based on stroke classification. The proposed system is trainable and does not rely on user-defined heuristics.

C. V. Jawahar *et al.* describes a word retrieval system for Indic scripts using a synthesis approach [79]. The language-specific features are also considered in their work to address the problems in writing variations for ink domain transformations. They elaborated on the characteristics of Indian scripts in detail, and three features, namely curvature, tangent, and height, are considered for word matching with DTW. Anil K Jain and Anoop M Namboothiri experimented to index and retrieve documents in online handwritten English words [80]. They achieved higher values of precision and recall rates in retrieving documents. A DTW string matching is performed with four major features to compare a keyword against indexed word groups with higher accuracies. They observed that the error rate depends on the similarity of words in clusters.

2.6. Conclusion

A brief review of previous research in OHCR specific to segmentation-based approaches has been conducted in this chapter. A review of OHCR studies focusing on Syntactic Pattern Recognition is performed. The chapter summarizes the major studies reported in this direction. Studies on real-time recognition of handwritten characters, including Keyword Spotting (KWS) in handwritten document images and online handwritten documents, are also reviewed.

Even though the structure of the characters present in various languages is different, the studies considering OHCR as a syntactic pattern recognition problem are limited in Indian context. Hence, a remarkable contribution to OHCR in Indic languages is possible by initiating studies in this direction. In the following chapters, the online handwritten character recognition problem is described in the light of Syntactic Pattern Recognition (SPR) specifically for real-time applications.

Malayalam Online Handwritten Character Database Creation and their Preprocessing

3.1. Introduction

Language is a system of communication or arbitrary vocal sounds through which human beings communicate and interact in their everyday lives. At present, the number of languages in the world number in thousands [81]. In Indian context, a variety of languages exist in various parts of the country. Among the 22 official languages in India, Malayalam is a language spoken in the state of Kerala and the union territories of Lakshadweep and Puducherry. It is also a scheduled language in India, nearly spoken by 35 million people worldwide. Malayalam is a Dravidian language - a family of languages primarily used in India's southern parts - the scripts are derived from the Brahmi script of the 5th century BC. The Malayalam language was declared as a classical language by the Government of India in 2013. The Malayalam language is agglutinative, with separate units of different meaning in a single word [82]. The writing direction of the Malayalam script is from left to right. It is also alphasyllabary, i.e., the writing system is partially alphabetic and partially syllable-based [83]. There are thirteen vowels(including two diphthongs), thirty-six consonants, and five *chillus* in Malayalam. It also consists of vowel modifiers and special symbols, including *anusvaram*, *visargam*, and *chandrakkala*.

This chapter describes the orthographical categorization of Malayalam characters and the creation of an online handwritten database for Malayalam

characters. A detailed description of preprocessing methods applied to the OHCR dataset for further processing, is also presented in the chapter.

3.2. Orthography and Categorization of Malayalam Characters

Orthography - the art of writing words with the proper letters - is an obligatory part of linguistics. It is also the arrangement of various sounds of a language by written or printed symbols [84]. These symbols are known as graphemes- the smallest unit of a writing system of any given language. A grapheme may or may not correspond to a single phoneme of the spoken language [85]. When the alphabet of a language is introduced, the graphemes are termed as characters. A character is the unit of information that roughly corresponds to a grapheme, or symbol, such as, the alphabet in the written form of a natural language [86]. The following section explains the orthographical representations of different writing units in Malayalam.

3.2.1. Vowels and Dependent Signs in Malayalam

The major writing units of Malayalam are vowels and consonants. The other conjunct units are formed by combining consonants and vowels. There are eleven vowels in Malayalam, comprising of six short and five long vowels. Usually, an independent vowel occurs as the first letter for a vowel prefixed word. The dependent vowel transforms the consonants to form valid Consonant-Vowel (CV) combinations. The list of independent vowel characters and dependent vowel signs, namely *diacritics* of the Malayalam script, is displayed in Table 3.1

Table 3.1 Vowels and Dependent signs in Malayalam

Vowel List		Independent Vowel	Dependent Vowel Sign
Short	a	അ <a>	-
	i	ഇ <i>	ി
	u	ഉ <u>	ു
	e	എ <e>	െ
	o	ഒ <o>	ൊ
	r	ഋ <r>	ൂ
Long	a:	ആ <a:>	ാ
	i:	ഈ <i:>	ീ
	u:	ഊ <u:>	ൂ
	e:	ഈ <e:>	േ
	o:	ഓ <o:>	ോ

From the table, it is observed that the dependent vowel signs of <e> and <e:> occur at the left part of a consonant letter, and the vowel signs <i>, <a:>, <i:> are occurring at the right part of a consonant. The vowel signs <o> and <o:> are unique in the group with two parts: the first part attached to the left of a consonant, and the second part is placed at the right. The vowel signs <u>, <u:>, <r> occur at the right side of the consonant.

3.2.2. Special Symbols in Malayalam (Anusvaram, Visargam, Chandrakkala)

A separate group, usually considered with vowels is *anusvaram*, which denotes the nasalization of the preceding vowel. In Malayalam, *anusvaram* is represented as $\text{◌}^\text{◌}$ /m/, and it merely represents a consonant /m/ after a vowel, though this /m/ may be analogized to another nasal consonant. A *visargam* ($\text{◌}^\text{◌}$ /h/) represents a consonant /h/ after a vowel, and it is transliterated as *h*. It is never followed by an inherent vowel or another vowel, as in the case of *anusvaram*. *Chandrakkala* ($\text{◌}^\text{◌}$ /ə/) is a *diacritic* constrained to the ending part of a word. It is attached to a consonant letter and treated as a semi-vowel, which is an allophone of the vowel /u/. The signs and the usage of special characters, *anusvaram*, and *chandrakkala*, are listed in table 3.2.

Table 3.2 Usage and sign of Anusvaram, Visargam and Chandrakkala

Character	Independent	Dependent	
		Sign	Example
<i>Anusvaram</i>	അം <am>	$\text{◌}^\text{◌}$ /m/	ഓം
<i>Visargam</i>	-	$\text{◌}^\text{◌}$ /h/	ഃ
<i>Chandrakkala</i>	-	$\text{◌}^\text{◌}$ /ə/	ഌ

3.2.3. Diphthongs in Malayalam

A combination of two short vowels within the same syllable is termed as a diphthong. Even though diphthongs are vowels, they are categorically separated in Malayalam. The diphthongal articulations in Malayalam are ഐ <ai> and ഔ <au>. The diphthong, /ai/ is named a falling diphthong, which

starts with a central vowel and terminates in a semi-vowel. The diphthong /au/ is a closing diphthong as it begins with an open-element and ends in a closed element. Table 3.3 describes Malayalam diphthongs with their signs.

Table 3.3 Malayalam Diphthongs

Diphthong	Independent	Dependent	
		Sign	Example
ai	ഈ <ai>	ഈ	ഈ
au	ഔ <au>	ഔ	ഔ

3.2.4. Malayalam Consonant Classes

Malayalam consonants are grouped, based on their articulation of classes. Malayalam has five sets of *varga* consonants and eleven other *non-varga* consonants. The *varga* set comprises of *velar*, *palatal* (*Postalveolar*), *retroflex*, *dental*, *labial*. Five characters are included in each of these sets. Based on its phonetic property, the *varga* set is classified as voiceless and voiced. These two classes are then subdivided into aspirated, unaspirated, and nasal. Malayalam *varga* consonants and subdivisions are given in table 3.4, followed by consonants other than *varga* in table 3.5.

Table 3.4 Malayalam Consonant Varga Classification

Consonant Classes	Voiceless		Voiced		
	Unaspirated	Aspirated	Unaspirated	Aspirated	Nasal
Velar	ക <ka>	ഖ <kha>	ഗ <ga>	ഘ <gha>	ങ <ṅa>
Palatal	ച <ca>	ഛ <cha>	ജ <ja>	ഝ <jha>	ഞ <ña>

Table 3.4 Malayalam Consonant Varga Classification (cont.)

Consonant Classes	Voiceless		Voiced		
	Unaspirated	Aspirated	Unaspirated	Aspirated	Nasal
Retroflex	ട <ṭa>	ഠ <ṭha>	ഡ <ḍa>	ഢ <ḍha>	ണ <ṇa>
Dental	ത <ta>	ഥ <tha>	ദ <da>	ധ <dha>	ന <na>
Labial	പ <pa>	ഫ <pha>	ബ <ba>	ഭ <bha>	മ <ma>

Table 3.5 Malayalam Consonants other than Varga

S.No	1	2	3	4	5	6	7	8	9	10	11
Consonant	യ <ya>	ര <ra>	ല <la>	വ <va>	ശ <śa>	ഷ <ṣa>	സ <sa>	ഹ <ha>	ള <ḷa>	ഴ <ḷa>	റ <ra>

3.2.5. Malayalam Chillu (Chillaksharam)

Chillu (*chillaksharam*) represents a pure consonant that independently exists without the help of a *virama*. *Chillu* is considered a special consonant letter that is never followed by an inherent vowel. In Unicode Text Format (UTF) representation, *chillu* letters are treated as independent characters and encoded automatically. Table 3.6 lists all the five *chillu* consonants in Malayalam.

Table 3.6 Malayalam Chillu Consonants

S.No	Chillu	Base letter
1	ൻ	ന <na>
2	ര	ര <ra>
3	ൽ	ല <la>
4	ൾ	ള <la>
5	ണ	ണ <ṇa>

3.2.6. Formation of Various Conjunct Classes in Malayalam

Conjunct classes are formed by combining more than one consonant in Malayalam. They are usually formed either by combining the same consonants or different consonants. But, all the consonant combinations are not valid compounds in Malayalam. For example, aspirated consonants typically do not double in Malayalam. The formation of compound letters in Malayalam is categorized based on the following two factors.

1. Position and the order of the component letters,
2. The presence of special symbols.

In the first category, *i.e.*, based on the position and order of component letters, seven rules are created to form Malayalam compound letters, as described in table 3.7.

Table 3.7 Rules for the formation of compound letters in Malayalam

S.No	Method	Example
1	Side by side writing of component letters	തത (ത + ത), (<ta>+<ta>)
2	One component letter on top of the other	പ്പ (പ + പ), (<pa>+ <pa>)
3	The first component fully on the left and the second partially to the right	ന്ന (ന + ന), (<na>+<na>)
4	First component partially on the left and the second attached to the right	ന്നു (ന്ന + ള), (<na><ta>)
5	The second component is partially written below to the first one	സു (സ + സ), (<sa>+ <sa>)
6	Component letters attached opposite to their pronunciation order.	ന്ന (ന + ന), (<na>+ <na>) read as (ന + ന), (<na>+ <na>)
7	More than two components joined together to form a compound letter either fully or partially	ഡ്യ (ദ + ധ + യ), (<da>+ <dha>+ <ya>)

In the second category, the conjunct classes are formed using special symbols in three ways.

1. Compound letters formed by the arbitrary joining of the component letters.

Example:

ക (ക <ka> + ക <ka>), നു (നു <na>+ ള <pa>).

2. Compound letters formed by combining different approximants with the consonant. The approximants (Consonant diacritics) in Malayalam letters are, യ <ya>, ര <ra>, ല <la> and ള <va>.

Examples for compound letters using approximants:

പ <pa>+ യ <ya>→പ്യ, ക <ka>+ ല <la> →കു.

3. Letters formed using the doubling symbol.

Example:

ച <ca> + ച <ca> → ച്ച, വ <va> + വ <va> → വ്വ.

A detailed description on orthography and categorization of Malayalam Characters is given in this section. From the section, various characters present in Malayalam script are identified. The rules governing the formation of various conjunct letters are also identified. In the following section, creation of a Malayalam online handwritten character database is presented.

3.3. Database Creation for Online Handwritten Characters in Malayalam

The database is a major part of any system which is based on training models. In handwritten character recognition studies, the potential of addressing variations in writing styles, is an essential factor, that improves the performance of the algorithm. The accuracy of a system for handwriting is measured upon the variability and the features of different calligraphic styles. Offline handwritten character databases like MNIST, IAM, CASIA, KHATT, *etc.*, are popular in foreign languages. For Indic scripts, certain databases are also available for offline handwritten characters. Due to the finite tenure of OHCR research in the Indian context, the databases for online handwritings in Indic scripts are still limited, especially in Malayalam. In this context, a unique database of online Malayalam handwritings is necessary to conduct various experiments in the study. In this section, the procedure for creating the proposed OHCR database in Malayalam is included, with a detailed description of devices and data collection methods used

3.3.1. Devices used for Data Acquisition

Two devices are used for data recording in the proposed work, namely, E-writemate and touchscreen-enabled laptop. A brief description of the devices used in data acquisition is given below.

(a) E-writemate

E-writemate is a device specifically used for recording the online handwritten data. The device consists of a memory unit and an electronic pen. The technology used in the device is compatible enough to record natural handwriting on plain paper. Figure 3.1 shows the components of the E-writemate device. A plain paper is attached to the memory unit, and writing on the paper is carried out using an electronic pen. The memory unit can store up to two hundred A4 pages of handwritten data stored in the form of (x, y) coordinates of the neighboring points [87]. The recorded data is available in the form of a text file. In this text file, every row with three values correspond to a point in a stroke. The first value in a row indicates the pen down and pen up information. A '1' indicates pen down, and '0' indicates pen up for the corresponding x and y values. This information is used to separate each stroke from the text file for further processing.



Fig. 3.1 E-writemate pen and memory unit

(b) Touchscreen-enabled laptop

To acquire online handwritten data, a touchscreen-enabled laptop is used. A user interface for data acquisition is also developed using MATLAB software, where the writer should write using finger or stylus. The software automatically stores a text file for every stroke. Each row of the text file corresponds to the point of the stroke with time information. The first two values in a row of the text file represent the x and y values of a point in the stroke. The last value in the row represents the time to plot the point on the screen. Figure 3.2 shows the user interface part designed for handwritten data capture.

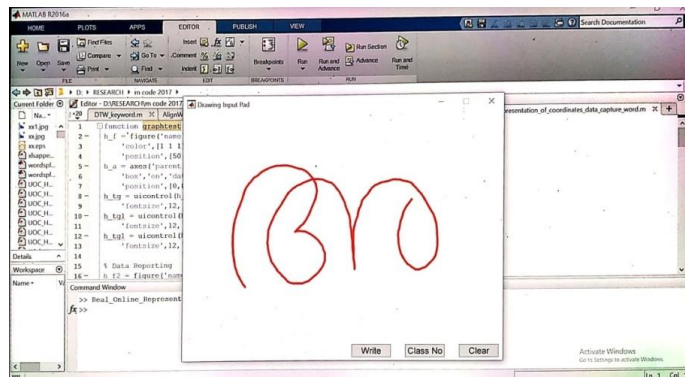


Fig. 3.2 User interface for touchscreen-enabled data acquisition

3.3.2. Setting up of Data Capturing Environment

For the creation of the database, single stroke writing is employed. In the case of single stroke writing, the character units are written in an isolated form, separately, for each character at fixed positions with moderate speed. For E-writemate, the plain paper is printed with a sample of the character, and thirteen empty boxes are printed near the sample character. From the recorded

3.3.3. Labeling of Samples

There exist two types of recognition schemes based on writer dependency, namely, writer dependent, and writer independent. To train and test writer-dependent models, the information regarding the writer is essential. A unique labeling scheme is proposed, in this study, to differentiate various samples of a character, written by different writers.

The labeling procedure of the acquired data samples is as follows. It is a combination of writer, character number, and sample number. For e.g., *W001C001S001* is the first sample of the first character written by writer1. The labeling scheme allows 999 writers to write 999 samples of every 999 characters. The samples from E-writemate and the touchscreen-enabled laptop are easily differentiable, as E-writemate does not contain time information.

3.3.4. Number of Samples in the Database

Fifty writers are involved in the database creation. All these writers are graduates within the age group of 20 to 40. The handwriting is performed in a moderate speed. The database includes samples of eight vowel and thirty six consonant characters in Malayalam. The samples are selected, labeled, as per the unique labeling scheme explained in the previous section, and stored as a text file. Table 3.8 lists the details of the samples in the database. The database is named CU-OHDB (Calicut University Online Handwriting Data Base), containing 30800 samples of 44 single stroke characters stored as text files.

Table 3.8 Number of samples from various methods and devices

Device	Writers	Characters	Samples per Character	Total Samples
E-writemate	30	44	10	13200
Touchscreen enabled laptop	20	44	20	17600
Total				30800

3.4. Preprocessing Techniques Applied to the Online Handwritten Character Samples

Preprocessing is an essential component in data processing systems with natural interfaces. Online handwritten strokes mostly contain writing imperfections like jitters, hooks, and duplicated points [88]. The collected handwritten data samples are preprocessed to reduce them. The major preprocessing techniques, applied in the proposed work, are normalization, smoothening, and resampling.

The stroke series are normalized to the range of [0, 1] using min-max normalization, which is one of the most suitable normalization methods used for online handwritten character data [89]. The normalized data is then smoothed using a moving average filter, which is a low-pass filter with filter coefficients equal to the reciprocals of the span [90]. The resampling technique used in the work is equidistant resampling, which is found reliable in most of the works in OHCR [91]. A detailed description of min-max normalization, smoothing using moving average filter, and equidistant resampling is given in the following section.

3.4.1. Min-Max Normalization

Normalization techniques are used to reduce the within-class variations. Normalization is generally performed to fix a variable range of values into a fixed range. The variability in the handwriting of various writers can be minimized, through the normalization process to an extent. Also, the normalized points always conserve the shape of the character. Fixing the x and y values to a particular range is essential to perform better feature extraction, especially, for the handwritten samples of similar characters. The normalization method which is used in the proposed work is min-max normalization [92]. The technique is often known as feature scaling, where the values of a numeric range of data features are reduced to a scale between 0 and 1. The original coordinates (x,y) of the handwritten characters are normalized to the range of $(0,1)$ using equation 3.1, where each of the (x,y) values are divided by the difference of maximum and minimum values of x and y .

$$X = \frac{x - \min(x)}{\max(x) - \min(x)} \quad Y = \frac{y - \min(y)}{\max(y) - \min(y)} \quad (\text{Eq. 3.1})$$

Figure 3.4 (a)-(b) represents the samples of Malayalam character $\text{അ} \langle a \rangle$ written by two different writers and their normalized form.

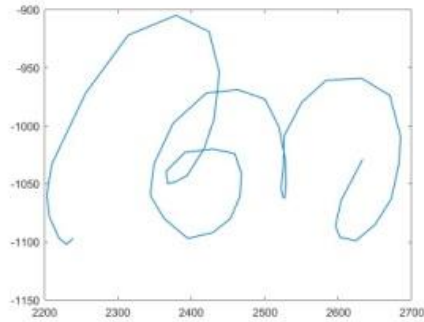


Fig. 3.4(a) Character അ <a> in raw form

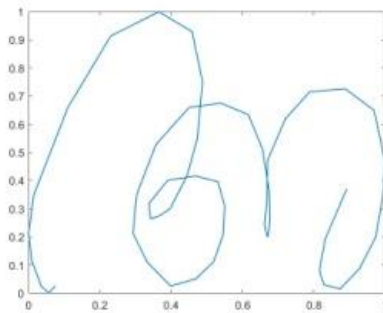


Fig. 3.4(b) Character അ <a> in normalized form

3.4.2. Smoothing

Smoothing is the process of removing jitters, hooks, and redundant points present in the handwritten data. From the literature, it is evident, that, usage of moving average filter, preserves the original shape of the character in the case of Indic scripts [93][94]. Since most of the Malayalam characters contain smooth edges, a moving average filter, is used in this work for smoothening. The primary factor, deciding the smoothness of shape in the moving average filter technique, is filter length, and it is dependent on the data[95]. As the filter length increases, the smoothness of the shape also increases. Equation

3.2 describes a moving average filter where $N+1$ is the filter length, and M is the sliding window.

$$M = \frac{x[n] + x[n-1] + x[n-2] + \dots + x[n-N]}{N+1} \quad (\text{Eq. 3.2})$$

It is observed that as the filter length increases beyond five, the shape of the characters change drastically. Hence, in the experiments, the filter length is fixed as five, to preserve the original shape and smoothness of all the characters. The sample outputs obtained for the Malayalam character അ <a>, after applying a moving average filter with various filter lengths, are shown in figure 3.5.

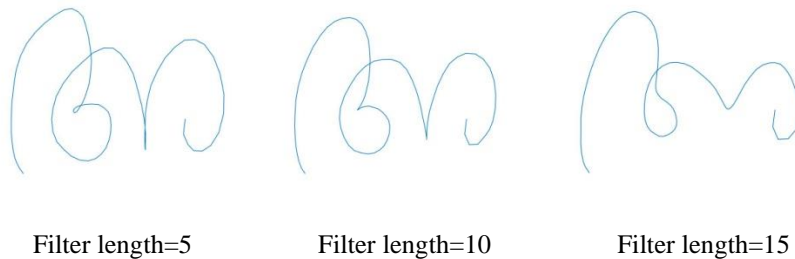


Fig. 3.5 Shape variation of Malayalam character അ <a> after applying moving average filter of various filter lengths

3.4.3. Resampling

From the experiments, it is found that the writing speed profoundly affects the number of points in the stroke series. The stroke sequence obtained from a slow writer has a higher number of points than a fast writer for the same character. Hence, it is mandatory to resample the points to overcome the assimilations caused by the writing speed. Resampling is the process of

arranging the data such that, the distance between adjacent points is approximately equal. Besides removing the variation, this step substantially reduces the number of stroke points to the desired value.

The method used for resampling is equidistant resampling, where the distance between two points is fixed to a value based on the required resampling points [96]. The points are resampled with equal length, and missing points are interpolated. A parametric spline approximation, which is effectively used in various OHCR studies, is used to interpolate missing points [97]. Figure 3.6 illustrates the interpolation and resampling of points between original points for the Malayalam character റ <ra>.

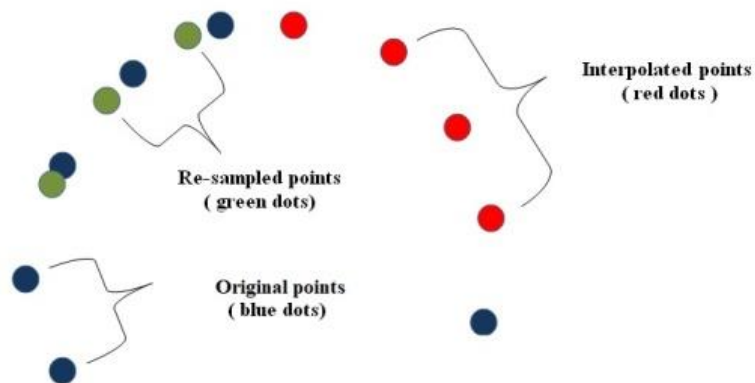
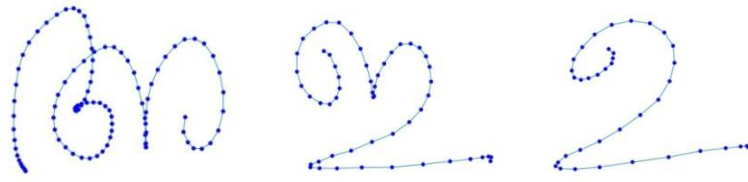
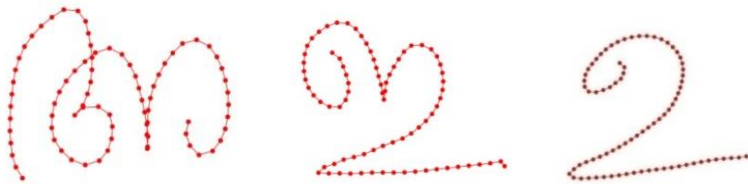


Fig. 3.6 Original, resampled, and interpolated points in a Malayalam character sample റ <ra>

Original and resampled set of the Malayalam characters അ <a>, ഇ <i>, ഉ <u> are displayed in figure 3.7(a)-(b). From the visual inspection of the final number of points, it is evident, that, the general shape of the character is preserved even after resampling.



(a) Original



(b) Resampled

Fig. 3.7 (a)-(b) Original and resampled set of Malayalam characters അ <a>, ഇ <i>, and ഉ <u>

3.5. Conclusion

The characteristics of the Malayalam language, orthographical formulations of various Malayalam characters, and the database creation procedures are explained in the first part of this chapter. The rules governing the formation of various conjuncts and the classes of aspirated and unaspirated consonants are also demonstrated.

The importance of a benchmark online handwritten character database is addressed in the study. The database named CU-OHDB, consisting of 30800 samples of 44 Malayalam online handwritten characters, is developed as part of the study. The devices and methods for data acquisition are also familiarized

in this chapter. A unique labeling scheme is also proposed as part of the study to distinguish different samples written by various writers.

The various preprocessing methods used in the proposed work are explained in the second part of this chapter. The preprocessing techniques, namely, min-max normalization, moving average filter based smoothing, and equidistant resampling, are implemented using the online handwritten character samples in the database. Based on the inspection, it is found that, the implemented methods perform well and can be used effectively for further experimental purposes.

Analysis of Pattern Primitives and Segmentation of Online Malayalam Handwritten Characters Using Ramer Douglas Peucker Algorithm and Eight Direction Freeman Code

4.1. Introduction

Online Handwritten Character Recognition (OHCR) has been dealt with in various studies where the recognition is immediately followed by writing [2][98]. The devices are vital in such recognitions as they must be capable of recording the temporal or dynamic sequences of strokes. This temporal information consists of writing directions, the order of strokes, and the speed of handwriting. It depicts, that, the shape of any character in a language, is only a visual component, and the underlying concept is a set of points. These points are represented in a mathematical form as $(x_1, y_1), (x_2, y_2), \dots, (x_N, y_N)$. The sets of points corresponding to the characters are the major entities in the online handwriting dataset to conduct various research activities.

The segmentation approach for OHCR is primarily focused in this chapter. The segmentation approach to handwriting recognition has been studied in numerous works using various techniques [43][99][100][101]. Here, a pattern recognition scheme to segment Malayalam online handwritten characters into sub-pattern units, namely pattern primitives is proposed.

The structural similarity of Malayalam characters is thought-provoking to be considered for segmentation. It is also observed that, most of the characters

in the Malayalam script contain similar geometrical patterns, as shown in figure 4.1. These standard segments are used to discriminate various character classes in handwriting recognition. A comparison of various patterns and finding the most similar pattern has a crucial role in character recognition.



Fig. 4.1 Common pattern segments in Malayalam

The first part of this chapter analyses the structural aspects of the characters in the Malayalam script to identify various pattern primitives. The second part emphasizes on various segmentation schemes to extract pattern primitives from the Malayalam online handwritten characters. Section 4.2 discusses the pattern primitives identified in the Malayalam character set. The formation of the handwritten character reference set marked with pattern primitives and labeling sequences for every Malayalam character is presented in section 4.3. The frequency of occurrences of various pattern primitives in the Malayalam online handwritten characters is described in section 4.4. A detailed reporting of various algorithms used for pattern primitive segmentation and the experiments conducted to verify the efficiency of these algorithms are described in section 4.5. An automated method for measuring segmentation accuracy is also described in this section. The concluding remarks of the chapter are outlined in section 4.6.

4.2. Pattern Primitives in Malayalam Characters

From visual inspection, it is clear that, the Malayalam characters, when divided into segments, consist of similar patterns common to most of them. The order of arrangement and direction of these segments are highly informative for the recognition purpose. There are generally two different categories of segments identified in Malayalam characters -arc segments and straight-line segments. These segments are determined based on direction changes, writing pattern styles, and similarity among patterns present in the characters. As an example, for the Malayalam character അ<a>, there are seven segments as shown in figure 4.2(a)-(g). From the figure, it can be observed that, there are three similar segments, (a), (c), (g), and four distinct segments, (b), (d), (e), and (f).

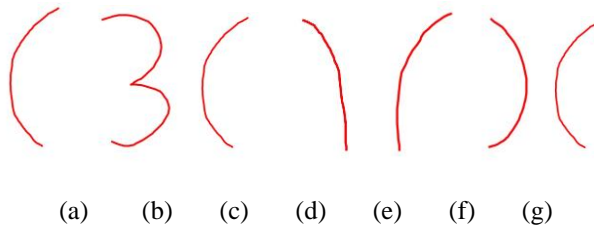


Fig. 4.2 (a)-(g) Segments of the Malayalam character അ <a>

A minimized representation of the character അ<a> is possible because, there exist redundant segments. After eliminating these redundant segments, the final set of unique segments for the character അ <a> is arranged as shown in figure 4.3.

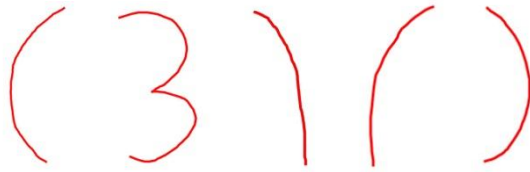


Fig. 4.3 Unique segments of the Malayalam character അ <a>

The unique segments identified in the letter അ <a> are all arc segments. This is one of the most relevant features of these types of characters. Consider another Malayalam character ഘ <gha>, as shown in figure 4.4. From the figure, it can be seen that, it includes a combination of arc and straight-line segments.



Fig.4.4 Segments of the Malayalam character ഘ <gha>

The character ഘ <gha> also has redundant segments. The unique segments obtained after removing these redundant segments from the character ഘ <gha> are shown in figure 4.5.







Fig. 4.5 Unique segments of the Malayalam character ഘ <gha>













In this context, an analysis of the shape and writing direction of the segments, common to the 44 Malayalam characters, is conducted as part of the work. From the analysis, 26 unique segments are identified, which are enough to represent the 44 Malayalam characters. If the direction is avoided, the number of segments will again be reduced to 18. The direction of pattern primitives is found to be most relevant in identifying visually similar patterns. Hence, it is decided to consider 26 unique segments for further processing. These 26 unique segments are known as pattern primitives in the proposed work. The following section describes these pattern primitives identified for the 44 selected Malayalam online handwritten characters.











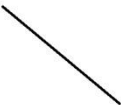



4.3. Pattern Primitives, Labeling of Characters and Creation of Reference Character Set















The 26 pattern primitives present in the 44 Malayalam characters with their direction, type, and label are listed in table 4.1. The label uniquely identifies these pattern primitives.





Table 4.1 Patterns primitives present in the Malayalam characters

S.No	Pattern Primitives	Direction	Type	Label
1			Arc	A1
2			Arc	A2

S.No	Pattern Primitives	Direction	Type	Label
3			Arc	B1
4			Arc	B2
5			Arc	C1
6			Arc	C2
7			Arc	D1
8			Arc	D2

S.No	Pattern Primitives	Direction	Type	Label
9			Arc	E1
10			Arc	F2
11			Linear	G
12			Linear	H1
13			Linear	H2
14			Linear	H3
15			Arc	I1

S.No	Pattern Primitives	Direction	Type	Label
16			Arc	I2
17			Arc	I3
18			Arc	J1
19			Arc	J2
20			Arc	K
21			Arc	L1
22			Arc	L2

S.No	Pattern Primitives	Direction	Type	Label
23		↑	Arc	M1
24		↓	Arc	M2
25		↑	Arc	N1
26		↓	Arc	N2

The following sections describe the representation of each of the 44 Malayalam handwritten characters (8 vowels and 36 consonants) using pattern primitives.

4.3.1. Representation of Malayalam Vowel Characters using Pattern Primitives

There are eight vowel characters in Malayalam. Representation of these characters using pattern primitives is given in table 4.2. From the table, it is evident that, the character അ <a> has eight pattern primitives. Characters ആ <a: >, എ <e> and ഐ <e: > have seven pattern primitives. The character ഇ <i> has six pattern primitives. Characters ഉ <u> and ഊ <r> have four pattern primitives, and the character ഓ <o> has three pattern primitives. The pattern

primitive sequences corresponding to each character in the table can be effectively used to identify the Malayalam vowel characters.

Table 4.2 Representation of Malayalam vowel characters using pattern primitives

S.No	Character	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Label sequence
1	അ <a>	(3	()	(A1-II-A1- C2-D1-B2- A1
2	ആ <a:̣>	(3	()	(/	A1-II-A1- C2-D1-B2- A1-K
3	ഈ <i:̣>)	(/	—			B2-A1-C2- D1-K-G
4	ഊ <u>)	(/	—					B2-A1-K-G
5	ഋ <ru>)	S)	(B1-J1-J2-2
6	എ <e>	()	—	—	—	()		A1-B2-G- H1-H2-A1- B2
7	ഐ <e:̣>	()	—	—	—	(3		A1-B2-G- H1-H2-A1- I1

Table 4.2 Representation of Malayalam vowel characters using pattern primitives (cont.)

S.No	Character	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Label sequence
8	൭ <o>									B2-A1-II

4.3.2. Representation of Malayalam Consonant Characters using Pattern Primitives

There are 36 consonant characters in Malayalam. Representation of these characters using pattern primitives is given in table 4.3

Table 4.3 Representation of Malayalam consonant characters using pattern primitives

S.No	Character	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Label Sequence
1	ക <ka>									A1-B2-M1- N2-A1-B2
2	ഖ <kha>									B2-A1-C2- G-H1
3	ഗ <ga>									A2-L1-B2

Table 4.3 Representation of Malayalam consonant characters using pattern primitives

(cont.)

S.No	Character	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Label Sequence
4	ഘ <gha>	()	—			—	—		A1-B2-G- C1-D2-G-H1
5	ണ <ña>)	(3				B2-A1-C2- D1-I1
6	ച <ca>	(/	—	—					A1-K-G-H1
7	ഛ <cha>	(/	—	()	(A1-K-G-A1- B2-A1
8	ജ <ja>)	(/	/)	(B2-A1-C2- D1-K-L1-B2- A1
9	ജ്ഞ <jha>	()	(\)	l)		A1-B2-A1- L2-E1-F2-B1
10	ഞ <ña>)	()	()		B2-A1-C2- D1-B2-A1- B2

Table 4.3 Representation of Malayalam consonant characters using pattern primitives

(cont.)

S.No	Character	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Label Sequence
11	ᵛ <ta>									B1-J1-A1
12	ᵛ <tha>									A2-B1-A2
13	ᵛ <da>									A1-L2-E1- F2-B1
14	ᵛ <dha>									A1-L2-E1- F2-B1-A2
15	ᵛ <na>									B2-A1-C2- D1-C2-D1- B2
16	ᵛ <ta>									A1-B2-A1- B2
17	ᵛ <tha>									H2-G-C1-D2

Table 4.3 Representation of Malayalam consonant characters using pattern primitives

(cont.)

S.No	Character	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Label Sequence
18	ദ <da>	(3							A1-I1
19	ഢ <dha>	())					A2-E1-F2-B1
20	ന <na>	()	()					A1-B2-A1-B2
21	പ <pa>	()	—	—					A1-B2-G-H1
22	ഫ <pha>	()	—		(A1-B2-G-C1-D2
23	ബ <ba>)	(()	—	—		B2-A1-C2-D1-B2-G-H1
24	ഭ <bha>	(3							A1-I3

Table 4.3 Representation of Malayalam consonant characters using pattern primitives

(cont.)

































S.No	Character	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Label Sequence
25	മ <ma>									A1-B2-G-H3
26	യ <ya>									A1-B2-A1- L2-B1
27	ര <ra>									A1-B2-A1
28	ല <la>									G-C1-D2-G- H1
29	വ <va>									A1-B2-G-H1
30	ശ <sa>									A2- L1- B2- A1
31	ഛ <sa>									A1-B2-G- H1-N2-M1- C2

Table 4.3 Representation of Malayalam consonant characters using pattern primitives

(cont.)

S.No	Character	Seg1	Seg2	Seg3	Seg4	Seg5	Seg6	Seg7	Seg8	Label Sequence
32	ഔ <sa>									A1-C2-D1- L2-B1
33	ഹ <ha>				—					A1-B2-G- A1-B2
34	ഐ <la>					—				B2-A1-I2-G
35	ര <ra>									M2-N1-J1
36	റ <ra>									A1-B2

4.3.3. Creation of Reference Set of Malayalam Characters based on Pattern Primitives

As a part of this study, a reference set of Malayalam handwritten characters is created from the CU-OHDB dataset. Each character in the reference set is manually marked based on pattern primitives, as shown in table 4.4. This

character reference set is used to compute the segmentation accuracy in the proposed experiments.

Table 4.4 Reference set of Malayalam characters marked based on pattern primitives

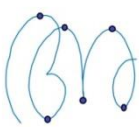

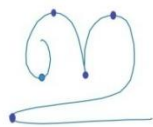
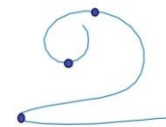

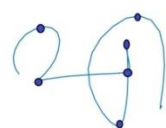
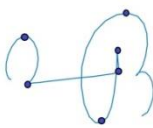

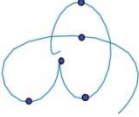
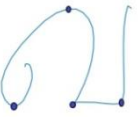



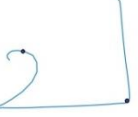

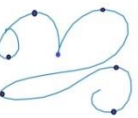
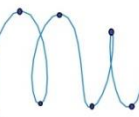
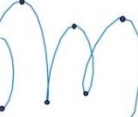
S.No	Character	Character marked with pattern primitives	S.No	Character	Character marked with pattern primitives
1	അ <a>		2	ആ <a:>	
3	ഇ <i>		4	ഉ <u>	
5	ഈ <i:>		6	ഊ <e>	
7	ഋ <e:>		8	ഌ <o>	

Table 4.4 Reference set of Malayalam characters marked based on pattern primitives
(cont.)

S.No	Character	Character marked with pattern primitives	S.No	Character	Character marked with pattern primitives
9	ക <ka>		10	ഖ <kha>	
11	ഗ <ga>		12	ഘ <gha>	
13	ങ <ṅa>		14	ച <ca>	
15	ഛ <cha>		16	ജ <ja>	
17	ജ്ഞ <jha>		18	ഞ <ña>	

**Table 4.4 Reference set of Malayalam characters marked based on pattern primitives
(cont.)**



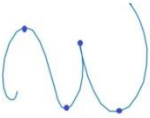
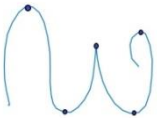
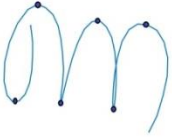




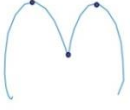
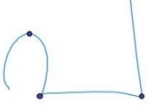
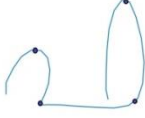
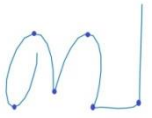




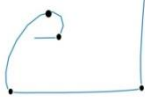

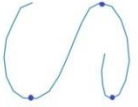
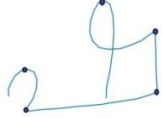
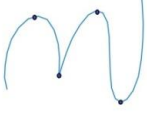



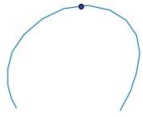
S.No	Character	Character marked with pattern primitives	S.No	Character	Character marked with pattern primitives
19	സ <ta>		20	ഠ <tha>	
21	ഠ <da>		22	ഠഠ <dha>	
23	ഠഠ <na>		24	ഠ <ta>	
25	ഠ <tha>		26	ഠ <da>	
27	ഠ <dha>		28	ഠ <na>	

Table 4.4 Reference set of Malayalam characters marked based on pattern primitives

(cont.)

S.No	Character	Character marked with pattern primitives	S.No	Character	Character marked with pattern primitives
29	പ <pa>		30	ഫ <pha>	
31	ബ <ba>		32	ഭ <bha>	
33	മാ <ma>		34	യാ <ya>	
35	രാ <ra>		36	ല <la>	
37	വാ <va>		38	ശ <sa>	

**Table 4.4 Reference set of Malayalam characters marked based on pattern primitives
(cont.)**

S.No	Character	Character marked with pattern primitives	S.No	Character	Character marked with pattern primitives
39	ഷ <ṣa>		40	സ <sa>	
41	ഹ <ha>		42	ഈ <ā>	
43	ഘ <gha>		44	ഠ <ṭa>	

4.4. Frequency of Occurrence of Pattern Primitives in Malayalam Characters

An analysis of the frequency of occurrences of various pattern primitives present in the Malayalam online handwritten characters is given in this session.

4.4.1. Frequency of Occurrence of Pattern Primitives

A detailed analysis of the occurrence of pattern primitives in Malayalam characters reveals that, certain pattern primitives are frequently occurring in

Malayalam characters. The position in which these pattern primitives occur also varies from character to character. This information can be considered vital, that can be incorporated while implementing an HCR system. Table 4.5 shows the frequency of occurrence of pattern primitives identified for the 44 Malayalam handwritten characters.

Table 4.5 Frequency of occurrence of pattern primitives in Malayalam characters

Character	Pattern primitives and corresponding frequency of occurrence																										
	A1	A2	B1	B2	C1	C2	D1	D2	E1	F2	G	H1	H2	H3	I1	I2	I3	J1	J2	K	L1	L2	M1	M2	N1	N2	
	(())							-				3	3	3	S	}	}	-	-	(())	
൧	3			1		1	1								1												
൨	3			1		1	1								1					1							
൩	1			1		1	1				1									1							
൪	1			1							1									1							
൫		1	1															1	1								
൬	2			2							1	1	1														

Table 4.5 Frequency of occurrence of pattern primitives in Malayalam characters (cont.)

Character	Pattern primitives and corresponding frequency of occurrence																									
	A1	A2	B1	B2	C1	C2	D1	D2	E1	F2	G	H1	H2	H3	I1	I2	I3	J1	J2	K	L1	L2	M1	M2	N1	N2
	(())							-			\	3	3	3	S	}	/	-	-	(())
എക	2			1							1	1	1		1											
എറ	1			1											1											
എക	2			2																			1			1
എറ	1			1		1					1	1														
എട		1		1																		1				
എല	1			1	1				1		2	1														

Table 4.5 Frequency of occurrence of pattern primitives in Malayalam characters (cont.)

Character	Pattern primitives and corresponding frequency of occurrence																									
	A1	A2	B1	B2	C1	C2	D1	D2	E1	F2	G	H1	H2	H3	I1	I2	I3	J1	J2	K	L1	L2	M1	M2	N1	N2
	(()))		-			\	3	3	3	S	}	/	-	-	(())
<cha>	1			1		1	1								1											
<ca>	1									1	1									1						
<cha>	3			1							1										1					
<ja>	2			2		1	1														1	1				
<jha>	2		1	1					1	1													1			
<ña>	2			3		1	1																			

Table 4.5 Frequency of occurrence of pattern primitives in Malayalam characters (cont.)

Character	Pattern primitives and corresponding frequency of occurrence																										
	A1	A2	B1	B2	C1	C2	D1	D2	E1	F2	G	H1	H2	H3	I1	I2	I3	J1	J2	K	L1	L2	M1	M2	N1	N2	
	(()))		-			\	3	3	3	S	}	/	-	-	(())	
^s <fa>	1		1															1									
^o <ha>		2	1																								
^e <fa>	1		1						1	1												1					
^{us} <pha>	1	1	1						1	1												1					
^m <pa>	1			2		2	2																				

Table 4.5 Frequency of occurrence of pattern primitives in Malayalam characters (cont.)

Character	Pattern primitives and corresponding frequency of occurrence																									
	A1	A2	B1	B2	C1	C2	D1	D2	E1	F2	G	H1	H2	H3	I1	I2	I3	J1	J2	K	L1	L2	M1	M2	N1	N2
	(()))		-			\	3	3	3	S	}	/	-	-	(())
<ta>	2			2																						
<tha>					1			1			1		1													
<da>	1														1											
<dha>		1	1						1	1																
<na>	2			2																						

Table 4.5 Frequency of occurrence of pattern primitives in Malayalam characters (cont.)

Character	Pattern primitives and corresponding frequency of occurrence																									
	A1	A2	B1	B2	C1	C2	D1	D2	E1	F2	G	H1	H2	H3	I1	I2	I3	J1	J2	K	L1	L2	M1	M2	N1	N2
	(()))		-			\	3	3	3	S	}	/	-	-	(())
<pa>	1			1							1	1														
<pha>	1			1	1			1			1															
<ba>	1			2		1	1				1	1														
<bha>	1																1									
<ma>	1			1							1			1												

Table 4.5 Frequency of occurrence of pattern primitives in Malayalam characters (cont.)

Character	Pattern primitives and corresponding frequency of occurrence																									
	A1	A2	B1	B2	C1	C2	D1	D2	E1	F2	G	H1	H2	H3	I1	I2	I3	J1	J2	K	L1	L2	M1	M2	N1	N2
	(()))		-				3	3	3	S	}		-	-))
∧ _a ∅	2		1	1																		1				
∧ _a e	2			1																						
∧ _a e					1			1			2	1														
∧ _a e	1			1							1	1														
∧ _a s	1	1		1																		1				
∧ _a B	1			1		1					1	1												1		1

Table 4.5 Frequency of occurrence of pattern primitives in Malayalam characters (cont.)

Character	Pattern primitives and corresponding frequency of occurrence																										
	A1	A2	B1	B2	C1	C2	D1	D2	E1	F2	G	H1	H2	H3	I1	I2	I3	J1	J2	K	L1	L2	M1	M2	N1	N2	
	(())							-				3	3	3	S	}	/	-	-					
⟨sa⟩	1		1			1	1																1				
⟨ha⟩	2			2							1																
⟨la⟩	1			1							1					1											
⟨ra⟩																			1						1	1	
⟨ra⟩	1			1																							

The pattern primitives in the descending order of frequency of occurrence are also displayed in table 4.6. From the table, it is evident that, the pattern primitives A1, B2, and G are the most frequently occurring pattern primitives in Malayalam characters. The pattern primitives H3, I2, I3, J2, M2, and N1, occur only once.

Table 4.6 Pattern primitives in the descending order of frequency of occurrence

Primitives	A1	B2	G	C2	D1	H1	B1	A2	I1	K	L2	C1	D2
Frequency	54	41	20	12	10	10	9	7	6	6	5	4	4
Primitives	E1	F2	H2	J1	L1	M1	N2	H3	I2	I3	J2	M2	N1
Frequency	4	4	3	3	3	2	2	1	1	1	1	1	1

A bar diagram showing the frequency of occurrences of pattern primitives in the Malayalam handwritten characters is given in figure 4.6.

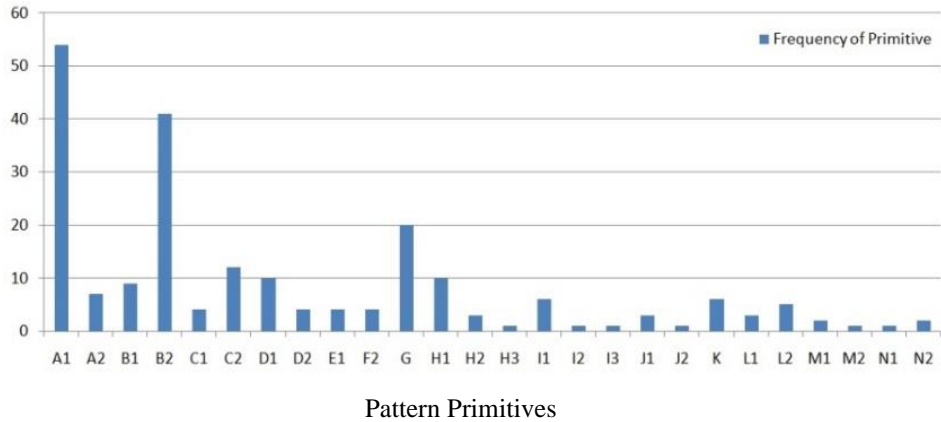


Fig. 4.6 Frequency of occurrence of pattern primitives in the Malayalam handwritten characters

The following sections describe the algorithms used for segmentation of Malayalam online handwritten characters based on pattern primitives.

4.5. Pattern Primitive based Character Segmentation Algorithms

Pattern primitive based character segmentation algorithms that are going to be discussed in this section, marks the segmentation points on the character samples as per the character reference set, described in section 4.3. Three algorithms are described and implemented in this work, namely, the Ramer Douglas Peucker (RDP) algorithm, the Eight Direction Freeman Code (EDFC) algorithm, and a combination of RDP and EDFC algorithms [6][9]. It is observed from previous studies, that, these algorithms are giving promising results for these types of problems.

4.5.1. Character Segmentation based on Ramer Douglas Peucker

Algorithm

Segmentation of characters based on Ramer Douglas Peucker (RDP) algorithm is described in this section. RDP algorithm is extensively used to approximate curves in numerous studies [8]. The objective of using the RDP algorithm is to segment the sequence of points representing a character into a minimum number of points by preserving its shape. The following procedure illustrates the RDP algorithm, which is used for the experimental purpose.

Procedure 4.1 Implementation of Ramer Douglas Peucker Algorithm

Input: PointList, The set of points describing the character

Epsilon, The value for Tolerance

Output: ResultList, The set of reduced points

Procedure DouglasPeucker(PointList, Epsilon)

maximum_distance = 0 // *Find the point with the maximum distance*

index = 0

N= length(PointList)

For all other points do

 Let d = Perpendicular distance between PointList[i] and Line with first
 and last point

 if d>maximum_distance then

 index = i

 maximum_distance = d

 end

 // *If max distance is greater than epsilon, recursively simplify*

 if maximum_distance> epsilon then

 recResults1[] = DouglasPeucker(PointList[1...index], epsilon)

 recResults2[] = DouglasPeucker(PointList[index...N], epsilon)

```
ResultList[] = {recResults1[1...length(recResults1)-1],
recResults2[1...length(recResults2)]}
else
ResultList[] = {PointList[1], PointList[N]}
Return ResultList[]
```

End

The procedure is as follows. A character is recursively divided by measuring the distance between locations. The first and last points in the character are kept as starting and ending points for segmentation. A line is drawn from the starting point to the ending point. A point on the character, which is maximum distant from this line, is also found. This is the furthest point on the character from the line segment between the starting and ending points and retained as a marked point. The point must be kept if the point furthest from the line segment is higher than ε (the value for tolerance) from the approximation. The procedure is repeated for the two resultant segments obtained from the selected point on the character, *i.e.*, a segment with the starting point and furthest point obtained and another segment with the most distant point and the endpoint. The recursive execution with marked points based on ε continues, until all the points are considered at least once. A new set of points is generated for the character where the number of points is minimum, and these are the points marked as kept by the procedure. These marked points are further used for segmenting the characters.

For conducting experiments, 150 samples of the 44 Malayalam online handwritten characters from the CU-OHDB dataset that are acquired using the E-writemate device are selected. All these samples are preprocessed using the methods as described in chapter 3. Before

proceeding with segmentation, an ideal tolerance(ϵ) value needs to be decided for the RDP algorithm. For this purpose, some random samples of every character are selected and segmented using RDP algorithm. From the results, it is observed that, the tolerance(ϵ), with the value 0.25, gives the number of points as expected for most of the character samples. Hence, the tolerance(ϵ) value is fixed as 0.25 for all the character samples. Various segmentation points obtained for the samples of the Malayalam character അ <a> with ϵ values 0.00, 0.05, 0.15, 0.25, 0.35, and 0.50 are shown in figure 4.7.

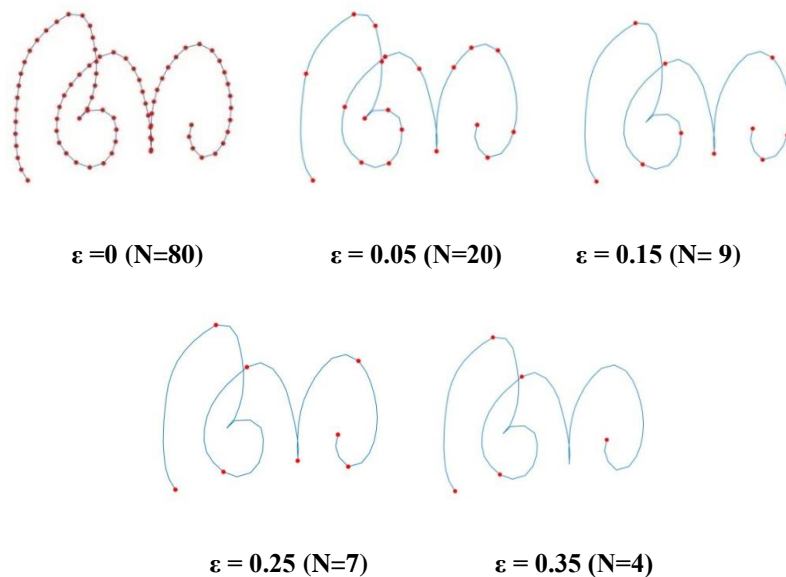


Fig. 4.7 The output obtained after applying the RDP algorithm on the character അ <a> for various ϵ values

As a next step, each of the character samples are segmented using the RDP algorithm with a tolerance value of 0.25 and the segmentation points obtained for these samples are manually compared against the

character reference set. Accuracy of the segmentation algorithm for each character is obtained by dividing the number of samples correctly segmented as per the reference set by the total number of samples of the same character. The segmentation accuracy of the RDP algorithm obtained for the 44 Malayalam characters is listed in table 4.7. The time complexity of the algorithm is $O(n)$.

Table 4.7 Segmentation accuracy of the RDP algorithm

S. No	Character	Accuracy (%)	S. No	Character	Accuracy (%)
1	അ <a>	64.00	23	ണ <ṇa>	88.66
2	ആ <a:>	60.66	24	ത <ta>	85.33
3	ഇ <i>	70.00	25	ഥ <tha>	74.66
4	ഉ <u>	68.66	26	ദ <da>	76.00
5	ഋ <ṛ>	60.66	27	ധ <dha>	92.00
6	എ <e>	77.33	28	ന <na>	92.00
7	ഏ <e:>	80.66	29	പ <pa>	80.66
8	ഒ <o>	90.00	30	ഫ <pha>	78.00
9	ക <ka>	64.66	31	ബ <ba>	80.66
10	ഖ <kha>	70.00	32	ഭ <bha>	88.00
11	ഗ <ga>	92.00	33	മ <ma>	77.33
12	ഘ <gha>	60.66	34	യ <ya>	84.00
13	ങ <kha>	88.00	35	ര <ra>	92.00
14	ച <ca>	86.66	36	ല <la>	73.33
15	ഛ <cha>	76.00	37	വ <va>	82.66
16	ജ <ja>	70.00	38	ശ <śa>	90.00
17	ഝ <jha>	73.33	39	ഷ <śa>	70.66
18	ഞ <ña>	80.00	40	സ <sa>	92.66
19	ട <ṭa>	80.00	41	ഹ <ha>	90.00

Table 4.7 Segmentation accuracy of the RDP algorithm (cont.)

S. No	Character	Accuracy (%)	S. No	Character	Accuracy (%)
20	o <ṭha>	50.00	42	ḡ <ḷa>	71.33
21	ω <ḍa>	78.00	43	ḡ <ḷa>	82.00
22	ωḡ <ḍha>	78.00	44	o <ṛa>	94.66
Average Segmentation Accuracy = 75.07 %					

It can be seen that the average segmentation accuracy of the RDP algorithm is 75.07%. The algorithm produced better results for certain characters, especially the characters, ṣṇ <ṇa>, ṭ <ta>, ṇ <na>, ω <dha>, and ṣṇ <ṇā>. A detailed analysis of the characters with lower segmentation accuracy depicts the fact that, the directional aspects of the character also need to be considered for better results. In this context, the Eight Direction Freeman Code (EDFC) algorithm is proposed to address directional aspects for the segmentation of characters, which is described in the following section.

4.5.2. Character Segmentation based on Eight Direction Freeman Code Algorithm

Eight Direction Freeman Code (EDFC) is a popular method to represent curves using eight directions [102]. Each of the points has a direction, as shown in figure 4.8. In the online handwriting recognition process, the direction has a crucial role. The advantage of direction in the recognition of online handwriting is addressed in many studies[6]. In this method, the segmentation is carried out using directional information of every point of the temporal sequence of the character. For the segmentation, the character stroke series is transformed into a

chain code based on EDFC. The difference between neighboring points corresponding to a character is compared, and the difference in x and y values decides the direction of a point. The conditions which determine the directions in the EDFC algorithm are listed in table 4.8.

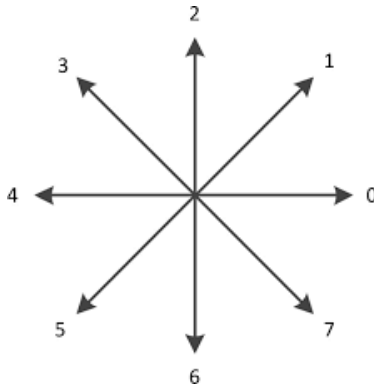





Fig. 4.8 Directions and codes in Eight Direction Freeman Code algorithm

Table 4.8 Conditions and corresponding directions in EDFC

Conditions	Code	Direction
$x > 0$ and $y = 0$	0	→
$x > 0$ and $y > 0$	1	↗
$x = 0$ and $y > 0$	2	↑
$x < 0$ and $y > 0$	3	↖
$x < 0$ and $y = 0$	4	←

Table 4.8 Conditions and corresponding directions in EDFC (cont.)

Conditions	Code	Direction
$x < 0$ and $y < 0$	5	
$x = 0$ and $y < 0$	6	
$x > 0$ and $y < 0$	7	

The algorithm to find the Eight Direction Freeman Code of the online handwritten sequence of (x, y) points representing the characters are described in procedure 4.2.

Procedure 4.2 Implementation of Eight Direction Freeman Code algorithm

Input: PointList, The set of points describing the character

Output: ResultList, The direction code of the points

Procedure Freeman_Eight_Direction_Code(PointList)

```

x=PointList(x) // Extract x-values
y=PointList(y) // Extract y-values
x=leftshift(x)-x; // Find the difference
y=leftshift(y)-y;
for i=1 to size(x)do
if(x>0 && y==0)
    ResultList(i)=0;
elseif(x>0 && y>0)
    ResultList(i)=1;
elseif(x==0 && y>0)
    ResultList(i)=2;
elseif(x<0 && y>0)
    ResultList(i)=3;

```

```

elseif(x<0 && y==0)
ResultList(i)=4;
elseif(x<0 &&y<0)
ResultList(i)=5;
elseif(x==0 && y<0)
ResultLis (i)=6;
elseif(x>0 && y<0)
ResultList(i)=7;
end







```

End

Initially, the direction codes for every character are computed using the EDFC algorithm. The chain code thus obtained is refined for directional imperfections by modifying the varying directional code between two different code sequences in a character. For example, in the direction code 1110777, the direction code 0 will be changed to 1 or 7 to obtain the correct direction.

The segmentation of the character in correspondence with the pattern primitive is performed based on the direction code. A moving direction code windowing technique is used to implement the character segmentation to its pattern primitive sequences. In this technique, a direction code sequence containing four digits is used as a code window. For example, the code window of 1177 represents the change from directions 1 to 7. Some sample shapes and corresponding code windows used for the experimental purposes are shown in table 4.9.

Table 4.9 Shapes in Malayalam characters and corresponding direction codes

Shapes	Code window
	1177 or 3355
	7711 or 5533
	6600 or 4422
	0022 or 6644
	7711 or 5533
	1177 or 3355

The algorithm for finding the segmentation points using the Eight Direction Freeman Code based segmentation is illustrated in procedure 4.3.

Procedure 4.3 Implementation of Eight Direction Freeman Code based segmentation algorithm

Input: PointList, The set of points describing the character

Output: ResultList, The set of reduced points

Procedure Direction_Code_Segmentation(PointList)

direct=**Freeman_Eight_Direction_Code**(PointList)// direction code

 for i=1 to size(direct,2)-2 do

 if((direct(i) ≠ direct(i+1))&& (direct(i+1) ≠ direct(i+2)))

 direct(i+1)=direct(i);

```

        end
    end
    // Load Code window representing shapes / direction changes
    CodeWindow={'1177','3355','7711','5533','6600','4422','0022','6644',...}
    for i=1 to size(direct) do
        // Find locations where code window matches
        ResultList=Find(direct,CodeWindow)
    end
End

```

The procedure is as follows. The direction chain code of the character is obtained using the Eight Direction Freeman Code algorithm, as discussed in procedure 4.2. After getting the direction chain codes of the character, the segmentation algorithm first removes the varying direction codes in between a sequence of similar direction codes from the direction chain code. The corresponding code windows representing the direction changes are compared with the direction chain codes of the character. If the code window matches the sequence in the character chain code, the position is marked. After all the windows have passed through the chain code, the segmentation algorithm returns a collection of indices corresponding to the marked points.

For experimental purposes, the same 150 preprocessed samples of the 44 Malayalam online handwritten characters taken from the CU-OHDB dataset to conduct experiments with the Ramer Douglas Peucker segmentation algorithm in the previous section are used. The segmentation algorithm is applied to the samples of every character and the segmentation points are obtained. The segmentation points, obtained for these samples, are manually compared against the character reference

set. Accuracy of the segmentation algorithm for each character is obtained by dividing the number of samples correctly segmented as per the reference set, by the total number of samples of the same character. Figure 4.9 shows the direction codes and the segmentation points marked using the Eight Direction Freeman Code algorithm, for a sample of the Malayalam character ω <ya>. It is clear from the figure that the algorithm exactly marks the segmentation points as per the character reference set.

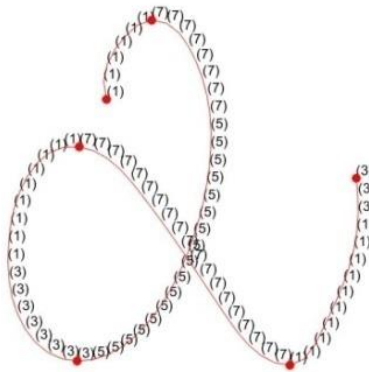


Fig.4.9 Plot of direction codes and segmentation points marked by EDFC segmentation algorithm for the Malayalam character ω <ya>

The segmentation accuracy of the EDFC based segmentation algorithm obtained for the 44 Malayalam characters is listed in table 4.10. The time complexity of the algorithm is $O(n)$.

Table 4.10 Segmentation accuracy of Eight Direction Freeman Code based segmentation algorithm

S. No	Character	Accuracy (%)	S. No	Character	Accuracy (%)
1	അ <a>	68.00	23	ണ <na>	84.00
2	ആ <a:>	60.00	24	ത <ta>	90.00

Table 4.10 Segmentation accuracy of Eight Direction Freeman Code based segmentation algorithm (cont.)

S. No	Character	Accuracy (%)	S. No	Character	Accuracy (%)
3	ഇ <i>	73.33	25	ഥ <tha>	65.33
4	ഉ <u>	72.00	26	ദ <da>	82.66
5	ഋ <r>	40.66	27	ധ <dha>	77.33
6	എ <e>	71.33	28	ന <na>	80.00
7	ഏ <e:>	66.00	29	പ <pa>	70.66
8	ഒ <o>	86.00	30	ഫ <pha>	75.33
9	ക <ka>	60.66	31	ബ <ba>	74.66
10	ഖ <kha>	65.33	32	ഭ <bha>	67.33
11	ഗ <ga>	80.00	33	മ <ma>	76.00
12	ഘ <gha>	50.00	34	യ <ya>	76.00
13	ങ <kha>	81.33	35	ര <ra>	91.33
14	ച <ca>	80.00	36	ല <la>	65.33
15	ഛ <cha>	72.00	37	വ <va>	76.66
16	ജ <ja>	75.33	38	ശ <śa>	86.00
17	ഝ <jha>	76.66	39	ഷ <śa>	56.00
18	ഞ <ñā>	84.66	40	സ <sa>	84.66
19	ട <ṭa>	86.00	41	ഹ <ha>	84.66
20	ഠ <ṭha>	72.00	42	ള <la>	61.33
21	ഡ <ḍa>	81.33	43	ഴ <ḷa>	84.00
22	ഢ <ḍha>	77.33	44	റ <ṛa>	96.66
Average Segmentation Accuracy = 74.68 %					

The average segmentation accuracy is obtained as 74.68% for the Eight Direction Freeman Code based segmentation algorithm. From the table, it can be seen that, the performance of the algorithm is more promising for the characters with more number of arc type pattern

primitives than, for the characters, which consist of straight-line pattern primitives. Hence a better approach is needed to segment the characters more precisely by utilizing the benefits of both of the proposed segmentation algorithms. In this context, a combined approach for segmentation by incorporating RDP and EDFC algorithms is proposed as described in the following section.

4.5.3. A Combined Approach of RDP and EDFC Algorithm for Segmentation

It is observed that the segmentation accuracy of the RDP algorithm is reduced due to excess segmentation points. Similarly, for the EDFC segmentation algorithm, the direction change at sharp points, mostly straight-line pattern primitives, are not precisely marked by the algorithm. To tackle these imperfections in marking, a fine-tuning over both of these segmentation methods are performed to obtain a combined approach. In the combined approach, the segmentation points from RDP and EDFC algorithms are effectively used to form a new set of segmentation points. In this new set, the nearest points and points marked, based on unreferred directions are avoided. The implementation of the algorithm used in the combined approach for segmentation is described in procedure 4.4.

Procedure 4.4 Implementation of combined approach of RDP and EDFC based segmentation algorithms

Input: PointList, The set of points describing the character

Output: CombinedList, The set of reduced points

Procedure Combined_Approach(PointList)

 // Call Procedure 4.1 (RDP) with $\epsilon = 0.25$

 CombinedList1=**DouglasPeucker**(PointList, ϵ)

```

// Call Procedure 4.3
CominedList2=Direction_Code_Segmentation(PointList)
// Combine the lists and sort in ascending x
CombinedList=CombinedList1+CombinedList2
Unique(CombinedList) // Make the list unique
For i=1 to size(CombinedList)
    If (CombinedList(i+1)-CombinedList(i) <5)
        Remove CombinedList(i) // Delete two closer points
    end
    Direction=Eight_direction_Freeman_code(PointList)
    If(Direction(CombinedList(1), CombinedList(2),..... CombinedList(end))
    = unreferred direction)
        Delete(CombinedList(Item)) //Item is the unreferred direction
    end
    Return(CombinedList); // Return the final set of segmentation points.
End

```

The combined approach for segmentation is as follows. Initially, the segmentation points obtained from RDP and EDFC algorithms for every character sample are joined to form a new set of points. It is observed that, the joined set of points contain redundant or nearby points marked by these algorithms. These points are to be eliminated. It is found by experiments that, these nearby points are usually occurring within five points. Hence, if the joined set of points have duplicate or nearby segmentation points, make them unique by removing any of the points as per the following criteria. If the two segmentation points obtained are five points apart, then keep both of them, else, remove one of the points from the set of points. It is also observed from the samples segmented, using the RDP algorithm, that, the points have some directions that are

not a part of the code window used in the EDFC algorithm. The direction codes of such points are identified and eliminated in the algorithm.

For experimental purposes, the same 150 preprocessed samples of the 44 characters taken from the CU-OHDB dataset to conduct experiments with the RDP segmentation algorithm and EDFC algorithm based segmentation are used. Segmentation is performed on every sample of the character using the combined approach. The set of points, thus obtained from the experiments, contain the segmentation points corresponding to pattern primitives. The segmentation accuracy is found through a manual comparison between the resultant set of points and the character reference set for every character. The segmentation points obtained for the Malayalam online handwritten characters, അ <a>, ട <ga>, യ <ya>, ര <ra>, ല <la> and ഹ <ha> based on the combined approach are shown in figure 4.10.

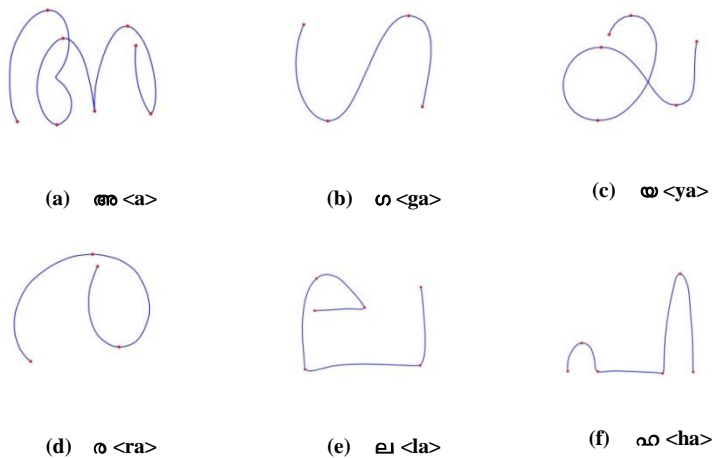


Fig. 4.10. Segmentation points obtained for the Malayalam characters അ <a>, ട <ga>, യ <ya>, ര <ra>, ല <la>, and ഹ <ha> using the combined approach for segmentation

The accuracy of the combined approach for segmentation obtained for the 44 Malayalam characters is listed in table 4.11. The average segmentation accuracy of the algorithm is obtained as 91.27%. The time complexity of the algorithm is estimated as $O(n)$. From the table, it is clear that, the performance of the algorithm is better compared to the other two methods.

Table 4.11 Segmentation accuracy of the combined approach for segmentation

S. No	Character	Accuracy (%)	S. No	Character	Accuracy (%)
1	അ <a>	92.00	23	ണ <ṇa>	100.00
2	ആ <a:>	92.00	24	ത <ta>	100.00
3	ഇ <i>	90.66	25	ഥ <tha>	84.66
4	ഉ <u>	92.66	26	ദ <da>	96.00
5	ഋ <ṛ>	96.66	27	ധ <dha>	100.00
6	എ <e>	82.66	28	ന <na>	100.00
7	ഏ <e:>	80.00	29	പ <pa>	84.66
8	ഒ <o>	94.00	30	ഫ <pha>	86.66
9	ക <ka>	73.33	31	ബ <ba>	86.00
10	ഖ <kha>	84.66	32	ഭ <bha>	100.00
11	ഗ <ga>	100.00	33	മ <ma>	86.00
12	ഘ <gha>	88.00	34	യ <ya>	92.00
13	ങ <kha>	92.00	35	ര <ra>	100.00
14	ച <ca>	85.33	36	ല <la>	78.66
15	ഛ <cha>	88.66	37	വ <va>	86.00
16	ജ <ja>	90.00	38	ശ <śa>	96.00
17	ഝ <jha>	96.00	39	ഷ <śa>	81.33
18	ഞ <ña>	94.00	40	സ <sa>	100.00
19	ട <ta>	92.00	41	ഹ <ha>	96.00
20	ഠ <ṭha>	80.66	42	ള <ḷa>	94.00

Table 4.11 Segmentation accuracy of the combined approach for segmentation (cont.)

S. No	Character	Accuracy (%)	S. No	Character	Accuracy (%)
21	ഓ <ḍa>	96.00	43	ഘ <ḷa>	92.66
22	ഔ <ḍha>	94.00	44	ഊ <ṛa>	100.00
Average Segmentation Accuracy = 91.27 %					

4.5.4. Automated Method for Determining Segmentation Accuracies

The segmentation accuracy is found in the above discussed methods by manually comparing the segments obtained from the experiments with the reference set. But when the sample size increases, it is difficult to visualize all the pattern primitive segments and compare them with the reference set. Hence, an automated method for finding the segmentation accuracy is proposed. For this purpose, the segments are represented using various direction category codes using eight direction freeman code. If the direction codes, obtained for a segment, are majorly in the upward direction; and so the segment is considered as an up segment and coded as 1. If the direction codes are majorly in the downward direction for a segment, it is regarded as a down segment and coded as 0. Similarly, linear segments in the horizontal direction are coded as 2. The direction category code string representation of all 44 characters in the reference set is manually generated as per the criteria mentioned. The direction category code string representation of the vowel characters in Malayalam is shown in table 4.12.

Table 4.12 Direction category code string representation of the Malayalam vowel characters

Character	Segment count	Direction category code string representation
അ <a>	7	1 0 1 0 1 0 1
ആ <a:>	8	1 0 1 0 1 0 1 0
ഇ <i>	6	0 1 0 1 0 2
ഉ <u>	4	0 1 0 2
ഋ <r>	4	1 0 1 0
എ <e>	7	1 0 2 1 0 1 0
ഐ <e:>	7	1 0 2 1 0 1 0
ഓ <o>	3	0 1 0

Procedure 4.5 describes the representation of segments into corresponding direction category codes.

Procedure 4.5 Implementation of the algorithm to represent a segment into corresponding direction category code

Input: PointList, The set of points describing the segments of a character

Output: SegmentCode, The direction category code of the segment

Procedure SegmentCoding(PointList)

```

str=Freeman_Eight_Direction_Code(PointList) // direction code by procedure 4.2
ts1=No of occurrence of '1' in str;           // upward directions
ts2=No of occurrence of '2' in str;           // upward directions
ts3=No of occurrence of '3' in str;           // upward directions
ts4=No of occurrence of '0' in str;           // horizontal directions
ts5=No of occurrence of '4' in str;           // horizontal directions
total=length(str);                             // length of direction string
upmajor=ts1+ts2+ts3;                           // count of upward directions

```

```

updwminor=ts4+ts5;           // count of horizontal directions
ts6=No of occurrence of '5' in str; // downward directions
ts7=No of occurrence of '6' in str; // downward directions
ts8=No of occurrence of '7' in str; // downward directions
dwmajor=ts6+ts7+ts8;       // count of downward directions
up=upmajor/(upmajor+dwmajor); // majority vote of up directions
down=dwmajor/(upmajor+dwmajor); // majority vote of downward direction
if(up>down)
    rd=1;                     // upward directions
elseif(down>up)
    rd=0;                     // downward directions
elseif(updwminor/total>0.50)
    rd=2;                     // predicted as x-axis parallel
end
SegmentCode=rd;
Return(SegmentCode)         // return the code corresponding to the segment
End

```

For determining the segmentation accuracy through the automated method, the segments of the character samples obtained from the combined approach are used. As a first step, they are counted to test whether the number of segments is the same as the number of segments in the reference character set for that character. If the number of segments differs from the count of segments in the reference character set, the character sample is considered as not segmented as per the reference character set. If the number of segments is the same, then the direction category code of every segment of the character sample is found using procedure 4.5. The codes thus obtained are compared with the reference category code of the character. If they are the same, the character is

considered as correctly segmented. The accuracy of each character class is obtained by dividing the number of correctly segmented samples by the total number of samples of the same character. The implementation of the algorithm to estimate the segmentation accuracy through the automated method is described in procedure 4.6.

Procedure 4.6 Implementation of the algorithm to measure segmentation accuracy through automated method

Input: Chstring, The array of direction category code strings of characters

Refstring, The array of codes strings of the reference character set

Output: AvgSegAccuracy, The Average Segmentation Accuracy

Procedure SegAccuracy(Chstring,Refstring)

for character class **i** = 1 to 44 // loop 1

Initialise Chmatch(1..N) as 0// where N is the number of samples

NotSegmented = 0 // Count of segments not as per reference set count

for Character samples j= 1 to N // loop 2

Chstring=**SegmentCoding**(Character)// Find code using Procedure 4.5

If size(Chstring) = size(Refstring) then

If Chstring matches with Refstring of the same character class **then**

Chmatch=Chmatch+1 // match count incremented

EndIf

else

NotSegmented = NotSegmented +1

end of for // loop 2

Chmatch=Chmatch-NotSegmented// difference segment count

ClassAccuracy(j)=Chmatch/ N// Segmentation accuracy

end of for // loop 1

AvgSegAccuracy=ClassAccuracy/44 // Average Accuracy

End

The segmentation accuracies for every character obtained for the combined approach measured through the automated method are presented in table 4.13. The average segmentation accuracy of the combined approach through the automated method is 79.13 %. Table 4.14 compares the average segmentation accuracy of the combined approach through manual comparison and automated method. The segmentation accuracy obtained through manual comparison is higher for all the characters compared to the automated method. The higher segmentation accuracy obtained for the combined approach through manual comparison shows that, the method is suitable for the segmentation of online handwritten Malayalam characters into pattern primitives, in the proposed work.

Table 4.13 Segmentation accuracy of the combined approach for segmentation through the automated method

S. No	Character	Accuracy (%)	S. No	Character	Accuracy (%)
1	അ <a>	78.66	23	ണ <ṇa>	94.00
2	ആ <a:>	73.33	24	ത <ta>	96.00
3	ഇ <i>	70.00	25	ഥ <tha>	76.66
4	ഉ <u>	78.66	26	ദ <da>	66..88
5	ഋ <ṛ>	78.00	27	ഌ <dha>	93.33
6	എ <e>	64.00	28	ന <na>	94.00
7	ഏ <e:>	62.00	29	പ <pa>	80.66
8	ഒ <o>	80.00	30	ഫ <pha>	80.00
9	ക <ka>	62.66	31	ബ <ba>	81.33
10	ഖ <kha>	76.66	32	ഭ <bha>	91.33
11	ഗ <ga>	84.00	33	മ <ma>	78.66

Table 4.13 Segmentation accuracy of the combined approach for segmentation through the automated method (cont.)

S. No	Character	Accuracy (%)	S. No	Character	Accuracy (%)
12	ഘ <gha>	76.00	34	യ <ya>	76.66
13	ഖ <kha>	78.66	35	ര <ra>	94.00
14	ച <ca>	76.00	36	ല <la>	72.66
15	ശ <cha>	76.00	37	വ <va>	77.33
16	ജ <ja>	72.66	38	ശ <śa>	87.33
17	ജ <jha>	82.00	39	ഷ <śa>	75.33
18	ണ <ña>	86.00	40	സ <sa>	94.00
19	ട <ṭa>	84.00	41	ഹ <ha>	86.00
20	ഠ <ṭha>	74.00	42	ള <la>	78.66
21	ഡ <ḍa>	91.33	43	ഴ <ḷa>	80.66
22	ഢ <ḍha>	92.66	44	റ <ra>	96.00
Average Segmentation Accuracy = 79.13 %					

Table 4.14 Average segmentation rate of the Combined approach

Average Segmentation Rate (%)	
Comparison	Automated Method
91.27	79.13

4.6. Conclusion

The pattern primitives in the Malayalam handwritten characters are identified in this chapter. An extensive study of the pattern primitives and the segmentation of the characters into pattern primitives are also carried out as part of the work. A total number of 26 pattern primitives are identified to represent the 44 Malayalam online handwritten characters. The reference set of these characters are marked with the

pattern primitives, and their labeled sequences are also obtained in this chapter. The analysis of the frequency of occurrences of various pattern primitives present in the Malayalam characters show that, certain pattern primitives are frequently occurring in them.

This study proposes an approach, that performs the segmentation of Malayalam online handwritten characters as per the character reference set marked with pattern primitives. The ability of the Ramer Douglas Peucker (RDP) algorithm to reduce a curve into a finite number of points is utilized in the segmentation experiments. Even with the variability in the writing style of the samples, the algorithm performed well with an average segmentation rate of 75.07%. Segmentation using the EDFC algorithm also obtained a better accuracy of 74.68%. The proposed combined approach, based on RDP and EDFC algorithms, achieved a segmentation rate of 91.27% through the visual comparison method. An automated method to compute the segmentation accuracy is also implemented in the chapter. The automated method for estimating segmentation accuracy shows an accuracy of 79.13%. So, the higher accuracy of the combined approach shows that the method is efficient for segmenting Malayalam online handwritten characters into pattern primitives.

**Recognition of Malayalam Online Handwritten
Characters based on Pattern
Primitives: A Class Modular Approach**

5.1. Introduction

In pattern recognition, machine understanding of natural handwriting has been an in-depth study for centuries. The technological upgrades resulted in several novel methods to achieve better results in handwriting character recognition. Most of the Handwriting Character Recognition (HCR) studies were focused on improving recognition accuracy by considering various features and experimenting with different classifiers. The shape of handwriting is addressed in many studies on Indic scripts for offline HCR, where, the images of handwritten characters are considered [103]. The research in OHCR in Indic scripts, mainly focused on the statistical features than shape dependent features [98]. For recognition of the characters, the features of the entire character are considered in these studies. The methods to address HCR as a Syntactic Pattern Recognition (SPR) problem based on pattern primitives are very few in Indic scripts. The extraction of pattern primitives is considered for Bangla offline character recognition with chain code features by Priyanka Das *et al.* [46]. Lajish V.L. and Sunil Kumar Kppparapu attempted to address primitive based recognition of online handwritten strokes for Devanagari scripts and achieved higher accuracies [48]. The pattern primitives, being the smallest recognizable

unit found in most of the characters, are highly useful in recognizing a character, especially for OHCR. Techniques that use pattern primitives and their features for recognizing Malayalam online handwritten characters are still in infancy.

In this chapter, a detailed analysis of various features specific to the pattern primitives, identified for Malayalam handwritten characters in the previous chapters is described. Using these features, experiments are conducted on online handwritten character samples taken from the CU-OHDB dataset to verify the effectiveness of pattern primitives in Malayalam OHCR. A class modular approach is employed in the recognition experiments using Support Vector Machine (SVM) classifier. Support Vector Machines are extensively used in the classification experiments for OHCR researches, and they are also found to be effective in Indian scripts [104][105][106]. A block diagram of the proposed OHCR system is displayed in figure 5.1.

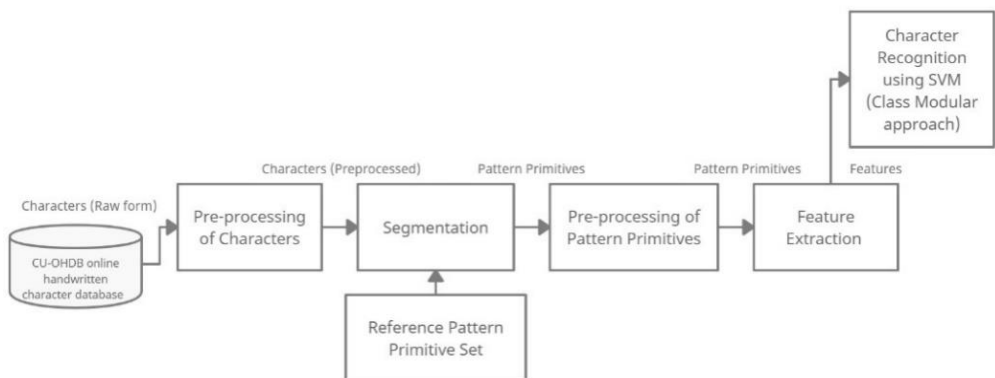


Fig. 5.1 Block diagram of the proposed pattern primitive based OHCR system

The rest of the chapter is divided as follows. In section 5.2, various methods used for the preprocessing of pattern primitives are discussed.

A detailed description of various pattern primitive features extraction techniques applied to the samples in the work is given in section 5.3. The description of intra-cluster pattern primitive features to discriminate the characters within the clusters is detailed in section 5.4. The experiments conducted in every cluster to verify the effectiveness of pattern primitives in OHCR using SVM are explained in section 5.5, followed by the experimental analysis in section 5.6. The conclusion of the chapter is given in section 5.7.

5.2. Preprocessing of Pattern Primitives

It is found that the pattern primitive segments obtained from the segmentation experiments are also susceptible to shape deformations and slope variations. Hence the pattern primitives need to be preprocessed before feature extraction to obtain better recognition rates. The following sections describe various preprocessing methods applied to the pattern primitives in the work.

5.2.1. Categorization of Pattern Primitives into Straight-line and Arc Segments using Curvature Values

According to Xiaolin Li *et al.*, online handwritten characters contain two types of segments upon segmentation, namely arc and straight-line segments [107]. It can be observed, that, the pattern primitives obtained through the combined approach of Ramer Douglas Peucker algorithm and Eight Direction Freeman Code for segmentation discussed in chapter 4 also include straight-line and arc segments.

The curvature value gives the average bending of a segment. Since the curvature values of straight-line segments are approximately zero,

they are easily separable from arc segments. Hence the curvature values can be effectively used for categorizing pattern primitives into straight-line and arc segments.

The curvature of a pattern primitive segment is the best information to differentiate straight-line and arc segments. The arc segment shown in figure 5.2 contains some sample points to describe the mathematical basis of curvature. If the arc length is measured from a fixed point, A to P is s , and A to Q is $s + \delta s$ (i.e., $PQ = \delta s$). The ratio of $\delta\psi$ and δs is called the *average bending* or *average curvature* where $\delta\psi$ is the angle of contingence and δs , the arc length (Equation 5.1). When the limits of the average curvature tend to zero, the *average curvature* is called the *curvature* at the point P.

$$K = \lim_{\Delta s \rightarrow 0} \frac{\Delta\psi}{\Delta s} = \frac{d\psi}{ds} \quad (\text{Eq. 5.1})$$

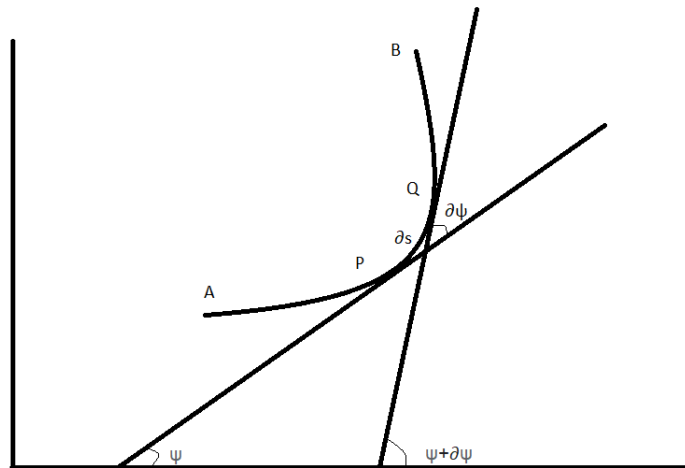


Fig. 5.2 Average curvature and angle of contingence of an arc

The curvature of a curve $y=f(x)$ at any point P on it is the instantaneous rate of change of the angle of inclination ψ of the tangent at P with respect to the arc length. In a non-uniform curve like pattern primitive, the curvature value is approximated to some threshold for facilitating the decidability of the curvature.

In another method, the curvature of a pattern primitive is defined through the *osculating circle*, which is the circle that best approximates the pattern primitive at a point. The curvature is the reciprocal of the radius of curvature of the *osculating circle*. Procedure 5.1 describes the algorithm used to find the radius of curvature of the *osculating circle* and, thereby the curvature of the pattern primitives for the experimental purposes.

Procedure 5.1 Implementation of the algorithm to find the curvature of the pattern primitive using osculating circle based method

Input: PointList, The set of points describing the pattern primitive

Output: C, The value of the curvature

Procedure Curvature (PointList)

$x=PointList(x)$, $y=PointList(y)$;

\bar{x} , \bar{y} // Find the mean of x and y ,

$X = x - \bar{x}$, $Y = y - \bar{y}$ // Get the difference.

$\partial x = \overline{X^2}$, $\partial y = \overline{Y^2}$ // Get the variance of x and y .

Find the solutions a , b using least mean square.

$$a = \frac{X}{(X^2 - \partial x + Y^2 - \partial y)/2}$$

$$b = \frac{Y}{(X^2 - \partial x + Y^2 - \partial y)/2}$$

Calculate the radius r of the osculating circle.

$$r = \sqrt{\partial x + \partial y + a^2 + b^2}$$

Find the radius of curvature $C = \frac{1}{r}$

End

The algorithm explained in procedure 5.1 is used to find the curvature values of the pattern primitives using osculating circle based methods in the recognition experiments. The following section describes the smoothing of straight-line pattern primitives using Bresenham's line drawing algorithm.

5.2.2. Smoothing of Straight-line Pattern Primitives using Bresenham's Line Drawing Algorithm

It is found that the straight-line pattern primitives obtained from segmentation experiments are not always smooth lines as expected. It is mainly due to the over smoothing of the straight-line segments happened, as a result of moving average filter, performed on the online handwritten character samples as described in chapter 3, section 3.4.2. To retain the smoothness of such straight-line pattern primitives, it is proposed to use line drawing algorithms to redraw them. In this work, Bresenham's line drawing algorithm illustrated in procedure 5.2 is used to preserve the linearity of straight-line pattern primitives by redrawing them.

Procedure 5.2 Implementation of Bresenham's line drawing algorithm

Input: PointList ; The list of points

Output: {X,Y} ;The set of new x and y values

Procedure Bresenham (PointList)

 dx= X₂-X₁ // X₁,Y₁ and X₂,Y₂ are start and points of the PointList

 dy=Y₂-Y₁

 p=2*dy-dx // Decision parameter

```

X= X1, Y= Y1 // initial value of x and y
plot the point (X, Y)
for i=1 to size (PointList)
    if p>= 0
        X=X+1
        plot the point (X, Y)
        p=p+2*dy-2*dx
    else
        Y=Y+1
        p=p+2*dy
        plot the point (X, Y)
    end of for

```

End

Most of the straight-line pattern primitive segments obtained from segmentation experiments are either oriented in horizontal or vertical directions. But it is found from visual observations that these primitives are skewed to a certain degree. Since most of the pattern primitive features proposed in the study depend on angular and directional variations, it is mandatory to eliminate skew variations from the straight-line pattern primitives. The method proposed to correct the skew of the horizontal and vertical pattern primitives using slope correction method, is described in the following section.

5.2.3. Skew Correction of Horizontal and Vertical Straight-line Pattern Primitives

The slope of a line or gradient is a number defining the direction of the line and steepness[108]. The slope is determined by computing the ratio between two distinct points between the vertical transition and the horizontal change in a line. The ratio is sometimes represented as a

quotient (rise over run), giving the same number on the same line for every two separate points.

A falling line has a negative rise. The slope's absolute value measures a line's steepness, incline, or grade, as shown in figure 5.3(a)-(b). A slope with a higher absolute value indicates a steeper line. A decreasing or increasing slope in a line also differentiates them as a line drawn downwards or upwards. Whether the line is increasing, decreasing, horizontal, or vertical is determined as per the conditions below.

- a) A line is a growing line if it goes up from left to right, *i.e.*, positive slope ($m > 0$).
- b) If the line falls from left to right, the line is called a decreasing line. In that case, the slope is negative ($m < 0$).
- c) The slope is zero if a line is horizontal.
- d) The slope is undefined if a line is vertical.

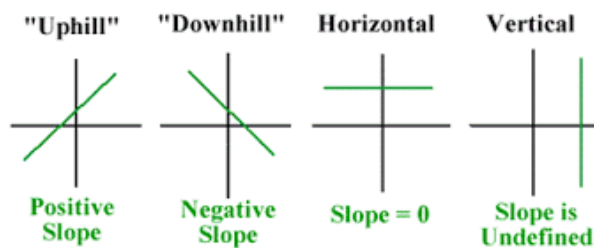


Fig. 5.3(a) Slope values

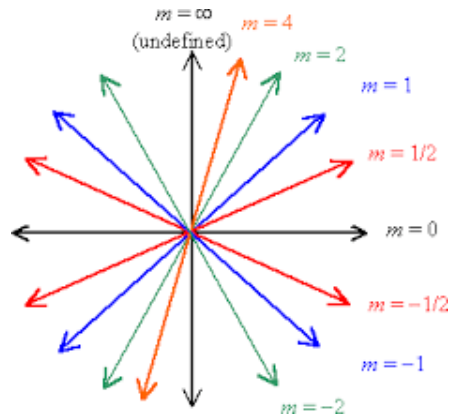


Fig. 5.3(b) Slope values and directions

The slope of a line is estimated using the differences, gradients, and mean in the current experiment. The gradient of a function $F(x,y)$ is given in equation 5.2, where ∇F is the gradient of the coordinate values of x and y over the linear segment and $\frac{\partial F}{\partial x}, \frac{\partial F}{\partial y}$, the partial derivatives.

$$\nabla F = \frac{\partial F}{\partial x} \vec{i} + \frac{\partial F}{\partial y} \vec{j} \quad (\text{Eq. 5.2})$$

Now slope is estimated as the mean of the gradient of y as in equation 5.3, where ∂x stands for difference, F is a gradient, and S is the slope.

$$\partial x = \overline{x - \bar{x}} \quad , \quad F = \frac{\partial y}{\partial x} \quad , \quad S = \bar{F} \quad (\text{Eq. 5.3})$$

A threshold value is fixed to distinguish horizontal lines from others. If the value of the slope is below the threshold value, then the line is adjusted to be a horizontal line by making all y values zeros. If the value of the slope is higher than the threshold value, then the line is adjusted

as a vertical line by making all x values zeros. In all other cases, the lines are kept with the same angle of inclination with the x -axis or y -axis.

Features are the essential components used to represent data and the feature selection is highly dependent on them. A detailed analysis of structural and directional aspects of the pattern primitives are conducted to select the appropriate features. The following section describes various features of pattern primitives identified for the experimental purposes.

5.3. Feature Extraction from Pattern Primitives

There are many features considered for developing the Malayalam OHCR systems [104]. But all these features are not applicable for the pattern primitives, which is either an arc or a straight-line segment. The characters are analyzed specific to the shape, and the pattern primitives that constitute each character are also identified. The majority of the pattern primitives identified has basic geometrical shapes including straight-line segments and arcs varying in direction, size, and curvature. Hence, features are determined based on these geometrical shapes. The direction of pattern primitives (down/up), cusp in the pattern primitives, and crossings (intersections) present in the Malayalam characters are identified as features. Certain additional features like minimized chain code, length, and curvature values of pattern primitives are also identified as features.

The following section describes various feature extraction techniques applied to pattern primitives for the conduct of OHCR experiments.

5.3.1. Minimized Chain Code (Reduced Direction Code) Features

The direction of handwriting is vital in identifying a character in most of the online handwriting recognition systems. In this work, the direction code of the pattern primitive is obtained using the Eight Direction Freema Code (EDFC) algorithm, described in chapter 4. It can be seen, that, the direction code obtained for a character's pattern primitive is a sequence of redundant code numbers. For example, the direction code for a pattern primitive shown in figure 5.4 is 333333333111111111. The directions obtained are reduced into a minimum code by restricting the same direction code continuously repeating more than three times into a single code. The code expressed in minimum form (31 for the pattern primitive shown in figure 5.4) without redundancy is a piece of information about the direction of the segment, which is called minimized chain code representation.

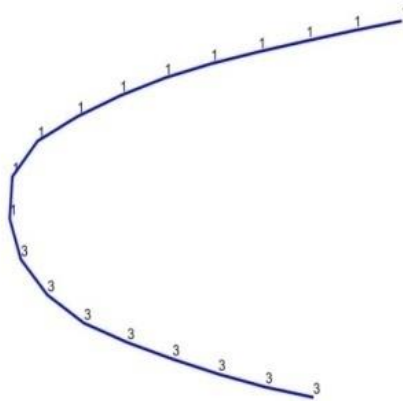


Fig. 5.4 Direction Code of the first pattern primitive of the Malayalam character

അ <a>

The following section describes cusp identification from pattern primitives.

5.3.2. Cusp Identification in Pattern Primitives

The sequence representing online handwriting is time-varying. Hence, the movement of points is an alternate meaning given to the online handwriting representation of characters. These movements by the points generate variations in the path identified as bumps, cusps, crossing, *etc.* A cusp is a point on a curve, where a moving point on the curve must start to move in the opposite direction. The presence of a cusp point in a handwritten character easily differentiates them from other characters. Cusps are common in most of the characters in Malayalam. The position and occurrence of the cusp are different in various characters. Sharp cusps are observable in the pattern primitives of the Malayalam characters അ <a> ,ആ <a:> ,ദ <da> ,ഏ <e:>, and ഒ <o>. In the characters അ <a>, ആ <a:> and ദ <da> ,the cusp is visible in the second primitive. For the character ഏ <e:>, the cusp is visible in the seventh primitive, and for the character ഒ <o>, the cusp is visible in the third primitive. The pattern primitive with a cusp visible in the Malayalam characters അ <a> ,ആ <a:> ,ദ <da> ,ഏ <e:>and ഒ <o> is displayed in figure 5.5. The dot marked in the primitive is the cusp point.

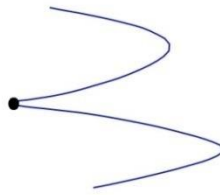


Fig. 5.5 Cusp in a pattern primitive present in the Malayalam characters അ

<a> ,ആ <a:> ,ദ <da> ,ഏ <e:>and ഒ <o>

A sharp turn in a curve known as a cusp is identifiable by measuring the curvature between three consecutive points. Let (x_1, y_1) , (x_2, y_2) , and (x_3, y_3) be three such adjacent points in a curve. The curvature of a circle drawn through them is simply four times the area of the triangle formed by the three points divided by the product of its three sides. Procedure 5.4 illustrates the algorithm to identify cusps in the pattern primitives.

Procedure 5.4 Implementation of the algorithm for the detection of cusps in pattern primitives

Input: data ; The 2-d array containing x,y values representing pattern primitives

Output: $cusp_x(i)$, $cusp_y(i)$; The cusp coordinates

Procedure Cusp(data)

for all points do

ax=x₁, bx=x₂, cx=x₃;

ay=y₁, by=y₂, cy=y₃;

$$K = \frac{2 * |((bx - ax) * (cy - ay) - (cx - ax) * (by - ay))|}{\sqrt{((bx - ax)^2 + (by - ay)^2) * ((cx - ax)^2 + (cy - ay)^2) * ((cx - bx)^2 + (cy - by)^2)}}$$

if (K > threshold) // threshold is a numeric value

cusp_x=x₂; cusp_y=y₂; // cusp_x, cusp_y are cusp coordinates

end of for

End

Using the above algorithm, the cusp features present in the pattern primitives of the character are extracted and used for further experiments. The following section describes the method for determining the length of a pattern primitive.

5.3.3. Length of the Pattern Primitives

The number of points per pattern primitive segment, called the length of pattern primitive, or segment length, is a strong candidate as a feature. As the entire character sequence is resampled to a fixed size in the preprocessing phase, the expected number of points present in a pattern primitive is almost similar for different samples of the same character. The figure 5.6 shows the samples of the characters അ <a>, സ <sa>, and ഹ <ha> with the length of the pattern primitives (blue dots) and the segmentation points (red dots). Table 5.1 lists the average number of points per pattern primitive representing the Malayalam online handwritten character samples of അ <a>, സ <sa>, and ഹ <ha> taken from the CU-OHDB dataset.

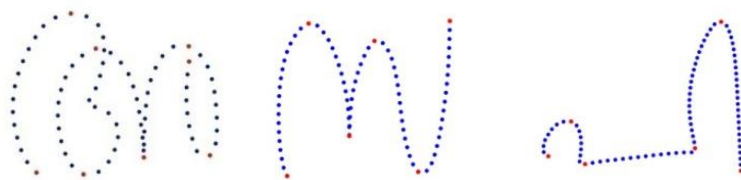


Fig. 5.6 Length of pattern primitives (blue dots), segmentation points (red dots) for the Malayalam handwritten character samples of അ- <a>, സ <sa>, ഹ <ha>

Table 5.1 Average pattern primitive lengths of the character samples അ <a> , സ <sa> , ഹ <ha>

Character	Average length of pattern primitives						
	Seg 1	Seg 2	Seg 3	Seg 4	Seg 5	Seg 6	Seg 7
അ <a>	14	16	11	10	10	9	9
സ <sa>	18	14	11	16	19	-	-
ഹ <ha>	8	8	15	22	26	-	-

Pattern primitive length features are extracted from the pattern primitives of the handwritten character samples and used for further experiments. The following section describes the method used to determine the writing direction of a pattern primitive.

5.3.4. Direction Category Code Features (DCCF) of Pattern Primitives

Most of the characters in Malayalam start writing with an upward direction and ends in a downward direction. The information regarding the start and end direction of an online handwritten character available with the first and last pattern primitives, can be effectively used in the recognition of handwriting. To extract the Direction Category Code Features (DCCF) of a pattern primitive, the direction codes of the pattern primitive are obtained using Eight Direction Freeman Code algorithm at first. If majority of the directions present in the direction codes are in upward direction, i.e., 1,2,3 then, the pattern primitive is considered as written in upward direction and the direction category code is assigned as 1. If majority of the directions present in the direction codes are in

downward direction, i.e., 5,6,7 then, the pattern primitive is considered as written in downward direction and the direction category code is assigned as 0. If majority of the directions present in the direction codes are in horizontal direction, i.e., 0,4 then, the pattern primitive is considered as written in horizontal direction and the direction category code is assigned as 2. The procedure described in section 4.4.4 of chapter 4 is used to extract the DCC features from the pattern primitives.

It is observed that, each of the 44 selected characters can be represented using the DCCF. The DCCF of pattern primitives computed for Malayalam vowel online handwritten character samples, taken from CU-OHDB data set, are listed in table 5.2.

Table 5.2 Direction Category Code Features (DCCF) of pattern primitives

S.No	Character	Direction Category Code Features (DCCF) corresponding to pattern primitives (1 to 8)							
		1	2	3	4	5	6	7	8
1	അ <a>	1	0	1	0	1	0	1	
2	ആ <a:>	1	0	1	0	1	0	1	0
3	ഇ <i>	0	1	0	1	0	2		
4	ഉ <u>	0	1	0	2				
5	ഋ <ṛ>	1	0	1	0				
6	എ <e>	1	0	2	1	0	1	0	
7	ഐ <e:>	1	0	2	1	0	1	0	
8	ഒ <o>	0	1	0					

5.3.5. Intersection (Crossing) of Pattern Primitives

In the case of Malayalam handwritten characters, it can be seen that, most of the pattern primitives intersect with other pattern primitives. This information is highly beneficial in differentiating the characters. The number of such intersections and their positions also varies from character to character. A common point shared by two pattern primitive segments is generally known as the point of intersection. The following section describes the method used to determine the intersection points in the character samples.

Given two pattern primitives, P_1 and P_2 , with end points (x_1, y_1) and (x_2, y_2) for P_1 and (x_3, y_3) and (x_4, y_4) for P_2 . In this case, four equations with four unknowns are to be solved, to identify t_1 , t_2 , x_0 , and y_0 where, (x_0, y_0) is the intersection of P_1 and P_2 .

t_1 is the distance from the starting point of P_1 to the intersection relative to the length of P_1

t_2 is the distance from the starting point of P_2 to the intersection relative to the length of P_2 .

Accordingly, four equalities are framed as shown in equation 5.4.

$$\begin{aligned}(x_2 - x_1) * t_1 &= x_0 - x_1 \\(x_4 - x_3) * t_2 &= x_0 - x_3 && \text{(Eq. 5.4)} \\(y_2 - y_1) * t_1 &= y_0 - y_1 \\(y_4 - y_3) * t_2 &= y_0 - y_3\end{aligned}$$

These equations are represented in matrix form as shown in equation 5.5.

$$\begin{vmatrix} (x_2 - x_1) & 0 & -1 & 0 \\ 0 & (x_4 - x_3) & -1 & 0 \\ (y_2 - y_1) & 0 & 0 & -1 \\ 0 & 0 & (y_4 - y_3) & 1 \end{vmatrix} * \begin{vmatrix} t_1 \\ t_2 \\ 0 \\ 0 \end{vmatrix} = \begin{vmatrix} x_0 - x_1 \\ x_0 - x_3 \\ y_0 - y_1 \\ y_0 - y_3 \end{vmatrix} \quad (\text{Eq. 5.5})$$

After solving the equations for t_1 and t_2 , it can be determined that whether P_1 and P_2 intersect by verifying the following conditions.

If $0 \leq t_1 < 1$ and $0 \leq t_2 < 1$, then the two pattern primitives P1 and P2 intersect at (x_0, y_0) .

Using this method, presence of intersection points is identified from various online handwritten character samples. If there is an intersection point in the pattern primitive, the feature is represented using 1, else it is represented using 0. The intersection points obtained for the characters a , t , and e are shown in figure 5.7.

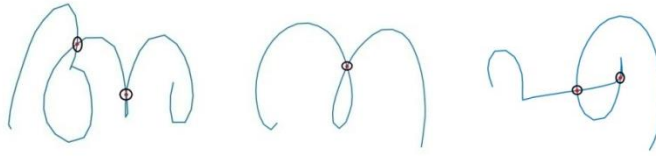


Fig. 5.7 Intersection points obtained for the Malayalam characters

അ <a>, ത <ta>, and എ <e>

A detailed study of various pattern primitive features, described in previous sections applied to differentiate handwritten character samples, is described in the following section.

5.4. Grouping of Malayalam Handwritten Characters based on the characteristics of Pattern Primitives

A class modular approach is proposed in the work to cluster and further recognize the characters. For this purpose, the number of pattern primitives present in each character is identified as a basic feature to form various clusters. It is identified that the number of pattern primitives present in Malayalam online handwritten characters ranges from two to eight. Hence, seven character clusters are formed corresponding to the number of pattern primitives. The Malayalam characters coming under various clusters formed, based on the number of pattern primitives present in the character, are shown in table 5.3.

Table 5.3 Character clusters based on the number of pattern primitives

Cluster	Number of pattern primitives	Characters
A	2	റ <ra>, ള <bha>, റ <da>
B	3	ഒ <o>, ഗ <ga>, ട <ta>, ര <ra>, റ <tha>, ഴ <la>
C	4	ഉ <u>, ള <ga>, ച <ca>, ത <ta>, ഫ <tha>, ധ <dha>, ന <na>, പ <pa>, മ <ma>, വ <va>, ശ <sa>, ള <la>
D	5	ഖ <kha>, ണ <kha>, ഡ <da>, ഫ <pha>, ല <la>, റ <sa>, ഹ <ha>, യ <ya>
E	6	ഇ <i>, ക <ka>, ച <cha>, ഡ <dha>
F	7	അ <a>, എ <e>, ഏ <e:>, ഘ <gha>, ഞ <jha>, ണ <ña>, ണ <na>, ബ <ba>, ഷ <sa>
G	8	ആ <a:>, ജ <ja>

From table 5.3, it can be seen that, the clusters A, B, C, D, E, F, and G have 3, 6, 12, 8, 4, 9, and 2 numbers of distinct characters, respectively. The pattern primitive features of the characters included in each of the character clusters are studied in detail. Based on the studies, the distinct pattern primitive features with respect to each character cluster are identified and used further in recognition experiments.

The following sections describe the features of the pattern primitives, which are capable of distinguishing the characters present in each cluster.

5.4.1. Pattern Primitive Features of Characters in Cluster A

Cluster A has three characters, namely \circ <ra>, $\text{\textcircled{a}}$ <bha>, and $\text{\textcircled{d}}$ <da>, consisting of two pattern primitives. The first pattern primitive for all these characters is visually similar, and hence it is not considered as a feature for recognition. The features of the second pattern primitives of these characters are distinguishable in nature and hence considered for recognition purposes.

The minimized direction code feature extracted from the second pattern primitive is identified as a feature for distinguishing \circ <ra> from $\text{\textcircled{a}}$ <bha> and $\text{\textcircled{d}}$ <da>. The minimized direction code of the second pattern primitive for \circ <ra> is '75' or '7'. For the other two, it is '7575' or '757'. It may also be noted that, the characters, $\text{\textcircled{a}}$ <bha> and $\text{\textcircled{d}}$ <da> are not distinguishable using these minimized direction code features alone.

It is visible that, the character $\text{\textcircled{d}}$ <da> has a sharp cusp in the second primitive, whereas it is not present in $\text{\textcircled{a}}$ <bha>. Hence, cusp in the second pattern primitive is taken as an additional feature to differentiate them. The number of direction changes in the second primitive is also different for \circ <ra> and the other two characters, $\text{\textcircled{a}}$ <bha> and $\text{\textcircled{d}}$ <da>. For \circ <ra>, the number of direction changes is identified as two, and it is more than two for $\text{\textcircled{a}}$ <bha> and $\text{\textcircled{d}}$ <da>. Figure 5.8 (a)-(d) shows the visual representations of directional features and cusps present in the selected character samples under cluster A.

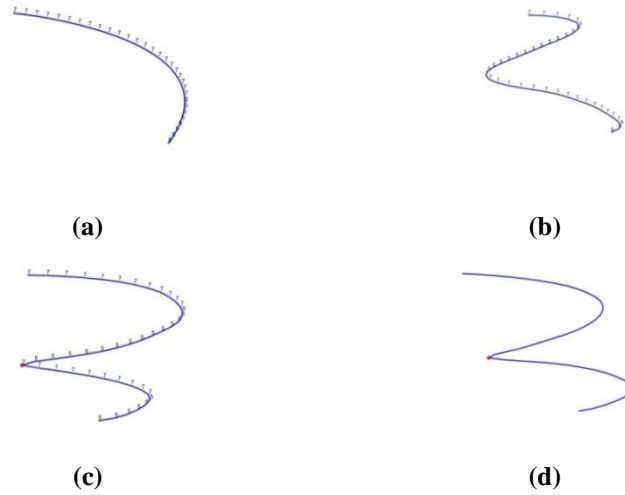


Fig. 5.8 (a)-(c) Direction code features of second pattern primitive for the Malayalam characters \circ <ra>, e <bha> and e <da> respectively and Fig. 5.8 (d) Cusp features of second pattern primitive for the Malayalam characters e <da>

The features considered for classifying the characters representing cluster A are listed in Table 5.4.

Table 5.4 Pattern primitive features of the characters in cluster A

Sl. No	Features	Pattern primitive	Related characters
1	Cusp	Second	e <da>
2	Minimized direction code	Second	\circ <ra>, e <bha>
3	Number of direction changes	Second	\circ <ra>, e <bha>, e <da>

5.4.2. Pattern Primitive Features of Characters in Cluster B

Characters with three numbers of pattern primitives are included in cluster B. The characters in cluster B are e <o>, o <ga>, s <ta>, o <ra>, \circ <tha>, and e <la>. It is observed that, a cusp found in the third pattern primitive is a unique feature of the character e <o> and hence, it is

considered for classification purpose. For the character \circ <ra>, the direction of the first pattern primitive is considered as a unique attribute that differentiates it from other characters in the cluster. It is the only character written in an upward direction compared to the other members in this cluster. The minimized direction codes of the character ς <ta> majorly represent the downward directions, unlike the other three members. The minimized direction codes specific to the second pattern primitive of the character ς <ta> are computed as ‘575’, in which 5 and 7 are downward directions. It is a unique feature found only in the character ς <ta>. For differentiating the character, ς <ga>, from other members, the minimized direction code of it’s first pattern primitive (57) is considered. The specialty of \circ <tha> is the crossing made by the first pattern primitive with the last pattern primitive. In \wp <la>, a crossing by the third pattern primitive and the second pattern primitive is unique. The pattern primitive features of the characters present in cluster B used for classification experiments are listed in table 5.5.

Table 5.5 Pattern primitive features of the characters in cluster B

Sl. No	Features	Pattern primitive	Related characters
1	Cusp	Third	\circ <o>
2	Pattern primitive up/down	First	\circ <ra>
3	Minimized direction code	Second	ς <ta>
4	Minimized direction code	First	ς <ga>
5	Crossing	Final	\circ <tha>, \wp <la>
6	Minimized direction code	Third	\circ <tha>, \wp <la>

5.4.3. Pattern Primitive Features of Characters in Cluster C

The characters with three pattern primitives are included in cluster C. The cluster includes twelve characters namely, \underline{u} <u>, \underline{r} <r>, \underline{ca} <ca>, \underline{ta} <ta>, \underline{tha} <tha>, \underline{dha} <dha>, \underline{na} <na>, \underline{pa} <pa>, \underline{ma} <ma>, \underline{va} <va>, \underline{sa} <sa>, and \underline{la} <la>. The cluster is then subdivided into two based on the direction of the first pattern primitive. The first pattern primitives of the characters \underline{r} <r>, \underline{ca} <ca>, \underline{ta} <ta>, \underline{na} <na>, \underline{pa} <pa>, \underline{ma} <ma> and, \underline{va} <va> are starting in the upward direction, and the cluster is named as C(Up). Similarly, the first pattern primitives of the characters \underline{u} <u>, \underline{tha} <tha>, \underline{dha} <dha>, \underline{sa} <sa>, and \underline{la} <la> are starting in a downward direction, and the cluster is named as C(Down).

As a next step, specific features are identified for the characters in cluster C(Up) for ease of classification. A crossing in the third pattern primitive is identified as a unique feature of both the characters \underline{r} <r> and \underline{ta} <ta>. For discriminating the characters, \underline{pa} <pa> from \underline{va} <va>, the lengths of the first, second, and third pattern primitive are considered. It can be observed, that the third pattern primitive is always a straight-line for the characters \underline{pa} <pa>, \underline{ca} <ca>, and \underline{va} <va>. Hence it is also included in the feature set. The straight-line with a slope in the fourth pattern primitive of the character \underline{ma} <ma> is identified as a unique feature to differentiate it from the other characters in the cluster. Table 5.6 lists the various pattern primitive features of the characters in cluster C(Up).

Table 5.6 Pattern primitive features of the characters in cluster C(Up)

Sl. No	Features	Pattern primitives	Related character
1	Crossing	Third	ᱚ <ᱦ>, ᱟ <ta>, ᱠ <na>
2	Pattern primitive length	Third	ᱚ <ᱦ>, ᱟ <ta>, ᱡ <pa>, ᱣ <va>
3	Pattern primitive length	Second	ᱡ <pa>, ᱣ <va>
3	Straight-line pattern primitive	Third	ᱡ <pa>, ᱣ <va>, ᱤ <ca>, ᱥ <ma>
4	Straight-line pattern primitive with slope	Fourth	ᱥ <ma>
5	Direction code, Pattern primitive length	First	ᱤ <ca>, ᱡ <pa>

In the cluster C(Down), the character ᱢ <tha> can be easily distinguished from other characters, based on the presence of straight-line in the first pattern primitive and the downward direction of the last pattern primitive. A cusp identified in the third pattern primitive is capable of easily classifying the character ᱚ <ᱦa>.

The minimized direction code and the length of the fourth pattern primitive are identified as unique features for the character ᱟ <dha>. The direction code of the first pattern primitive is considered as a feature to differentiate ᱚ <u> and ᱚ <ᱦa> from other characters in the cluster. The pattern primitive features of the characters in cluster C(Down) are displayed in table 5.7.

Table 5.7 Pattern primitive features of the characters in cluster C(Down)

Sl. No	Features	Pattern primitive	Related character
1	Straight-line pattern primitive	First	᳚ <tha>
2	Direction	Last	᳚ <tha>
3	Cusp	Third	᳚ <la>
4	Minimized direction code, Pattern primitive length	Last	᳚ <dha>, ᳚ <śa>
5	Minimized direction code	First	᳚ <u>, ᳚ <la>

5.4.4. Pattern Primitive Features of Characters in Cluster D

The cluster D is formed by grouping the characters which consist of five numbers of pattern primitives. The cluster consists of the characters, ᳚ <kha>, ᳚ <kha>, ᳚ <ḍa>, ᳚ <pha>, ᳚ <la>, ᳚ <sa>, ᳚ <ha>, and ᳚ <ya>. The cluster is then subdivided into two, based on the direction of the final pattern primitive. The final pattern primitives of the characters, ᳚ <kha>, ᳚ <ḍa>, ᳚ <la>, ᳚ <sa>, and ᳚ <ya> are in upward directions, and the cluster is named D(Up). Similarly, the final pattern primitives of the characters ᳚ <kha>, ᳚ <pha>, and ᳚ <ha> are in the downward direction, and the cluster is named, D(Down).

A crossing in the fourth pattern primitive is identified as a unique feature of ᳚ <ya> in cluster D(Up). The minimized direction code of the second and fourth pattern primitive is the major feature that differentiates the characters ᳚ <sa> and ᳚ <ḍa>. The minimized direction code of the third primitive of ᳚ <la> is found to be unique in this cluster. It is also found that the character ᳚ <kha> starts in the downward direction. Hence, the direction of the first pattern primitive is considered as the feature to identify the character ᳚ <kha> in the cluster. The summary of

the selected features to identify the characters in group D(Up) is given in table 5.8(a).

Table 5.8(a) Pattern primitive features of the characters in cluster D(Up)

Sl. No	Features	Pattern primitive	Related character
1	Starting direction	First	ഖ <kha>
2	Crossing	Fourth	യ <ya>
3	Minimized direction code	Second	സ <sa> , റ <ḍa>
4	Minimized direction code	Fourth	സ <sa> , റ <ḍa>
5	Minimized direction code	Third	ല <la>

In the cluster D(Down), it is found that, the character ഘ <kha> starts in the downward direction with a cusp in the fifth pattern primitive. Hence the features, starting direction, and cusp in the fifth pattern primitive are selected for further processing. The straight-line nature and length of the third pattern primitive along with the minimized direction code of the fourth pattern primitive are identified to differentiate the characters, ഘ <pha> and ഞ <ha>. Table 5.8(b) lists the pattern primitive features of the characters present in cluster D(Down).

Table 5.8(b) Pattern primitive features of the characters in cluster D(Down)

Sl. No	Features	Pattern primitive	Related character
1	Starting direction	First	ഖ <kha>
2	Cusp	Fifth	ഖ <kha>
3	Pattern primitive length	Third	ഘ <pha> , ഞ <ha>
4	Minimized direction code	Fifth	ഘ <pha> , ഞ <ha>

5.4.5. Pattern Primitive Features of Characters in Cluster E

The cluster E is formed by grouping the characters which consists of six numbers of pattern primitives. The members in the cluster are ഇ <i>, ഐ <ka>, ച <cha>, and ധ <dha>. It is found that, the character ഇ <i> starts in the downward direction, and there are no other characters in the cluster that starts in the downward direction. Hence, the starting direction of the first pattern primitive is selected as a feature to recognize the character ഇ <i>. The length of the third pattern primitive of the character ഐ <ka> is shorter than the other characters. Also, the minimized direction code of the fourth primitive is found to be unique for the character ഐ <ka>. These two features are considered to differentiate the character ഐ <ka> from others. The minimized direction code of the sixth pattern primitive is a feature distinguishing the character ധ <dha>. The straight-line feature of the third pattern primitive differentiates the character ച <cha> from others in the cluster. The direction of the last primitive is upward for the character ച <cha> unlike other characters. Hence, these features are selected to identify the character ച <cha>. The features selected for classifying the characters in cluster E are listed in table 5.9.

Table 5.9 Pattern primitive features of the characters in cluster E

Sl. No	Features	Pattern primitive	Related character
1	Starting direction	First	ഇ <i>
2	Pattern primitive length	Third	ഐ <ka>
3	Minimized direction code	Fourth	ഐ <ka>

Table 5.9 Pattern primitive features of the characters in cluster E (cont.)

Sl. No	Features	Pattern primitive	Related character
4	Minimized direction code	Sixth	റു <ḍha>
5	Straight-line pattern primitive	Third	ച <cha>

5.4.6. Pattern Primitive Features of Characters in Cluster F

Cluster F includes nine characters with seven numbers of pattern primitives. The characters in the cluster are അ <a>, എ <e>, ഏ <e: >, ഘ <gha>, ത <jha>, ഞ <ña>, ണ <ṇa>, ബ <ba>, and ഷ <ṣa>. The cluster is then subdivided into two, based on the presence of straight-line pattern primitive in the third position. The first cluster, F(L), is formed by including the characters എ <e>, ഏ <e: >, ഘ <gha>, and ഷ <ṣa>, which consists of a straight-line pattern primitive in the third position. The second cluster, F(NL), contains the characters അ <a>, ത <jha>, ഞ <ña>, ണ <ṇa>, and ബ <ba>, with an arc in the third pattern primitive.

In the F(L) cluster, the characters എ <e> and ഏ <e: > possess similar features, except a cusp, present in the seventh primitive. Hence, the cusp in the seventh pattern primitive is identified as a feature for recognizing the character ഏ <e: >. In the case of the character ഘ <gha>, the upward direction of the final pattern primitive is found to be unique in this cluster. The minimized direction code of the sixth pattern primitive is identified as a unique feature to distinguish the character ഷ <ṣa> from others. The list of pattern primitive features for the F(L) cluster characters are given in table 5.10.

Table 5.10 Pattern primitive features of the characters in cluster F(L)

Sl. No	Features	Pattern primitives	Related character
1	Straight-line pattern primitive	Third	എ <e>, ഏ <e: >, ഘ <gha>, ഷ <ṣa>
2	Cusp	Seventh	ഏ <e: >
3	Direction of pattern primitive	Last	ഘ <gha>
4	Minimized direction code	Sixth	ഷ <ṣa>

The direction of starting pattern primitive is upward for the characters അ <a> and ഓ <jha> present in the F(NL) cluster. Additionally, a cusp in the second pattern primitive is identified as a unique feature for the character അ <a>. For the characters ഞ <ña>, ണ <ṇa>, and ബ <ba>, the direction of starting pattern primitive is downward with a crossing in the sixth pattern primitive exclusively for the character ഞ <ña>. The occurrence of a straight-line pattern primitive at the sixth position is considered as a unique feature for the character ബ <ba>. The features selected for differentiating the characters in cluster F(NL) are detailed in table 5.11.

Table 5.11 Pattern primitive features of the characters in cluster F(NL)

Sl. No	Features	Pattern Primitives	Related character
1	Starting direction	First	അ <a>, ഓ <jha>, ഞ <ña>, ണ <ṇa>, ബ <ba>
2	Cusp	Second	അ <a>
3	Crossing	Sixth	ഞ <ña>
4	Straight-line pattern primitive	Sixth	ബ <ba>
5	Minimized direction code	Third	ഓ <jha>

5.4.7. Pattern Primitive Features of Characters in Cluster G

The cluster G is the smallest cluster which consists of only two characters അ <a:> and ജ <ja> with eight number of pattern primitives. The starting directions of these characters are considered for the classification purposes. The character അ <a:> starts in the upward direction, whereas the character ജ <ja> starts in the downward direction. In addition to that, a cusp present in the second pattern primitive is identified as unique for the character അ <a:>. Table 5.12 lists the features identified for the cluster G characters.

Table 5.12 Pattern primitive features of the characters in cluster G

Sl. No	Features	Pattern primitives	Related character
1	Starting direction	First	അ <a:>, ജ <ja>
2	Cusp	Second	അ <a:>

The following section describes the experiments conducted for the cluster-wise classification of the online handwritten character samples.

5.5. Experimental Setup

In this section, the experimental framework is set to classify the characters present within each cluster (A to G), which are already formed, based on the number of pattern primitives present in each character. For this purpose, online handwritten samples of each 44 Malayalam characters taken from the CU-OHDB dataset are used. This dataset consists of eight single stroke vowel characters and thirty six consonant characters. For the experimental purpose, the 100 samples of each character which are correctly segmented based on the pattern

primitive segmentation experiments conducted using the combined approach of RDP and EDFC, are used as described in chapter 4.

As a next step, the straight-line pattern primitive segments have undergone two preprocessing phases. In the first phase, the characters having straight-line pattern primitives are smoothed using the methods described in section 5.2.1 and 5.2.2. The osculating circle based method mentioned in procedure 5.1 is implemented in MATLAB to find the curvature of the pattern primitive. The experiment is conducted on the correctly segmented pattern primitives and computed their curvature values. From the experimental results, it is observed that the pattern primitives with a straight-line nature also hold certain curvature values. Considering this, the character segments with curvature values less than 0.75 are categorized as straight-line pattern primitives. These pattern primitives are then redrawn as straight-lines using Bresenham's line drawing algorithm as described in Procedure 5.2.

In the second phase, pattern primitives identified as straight-line are adjusted for angular variations by measuring slope values as described in section 5.2.3. For this purpose, a threshold value which is fixed as 0.2, is used to distinguish horizontal lines. If the value of the slope is below the threshold value, then the line is adjusted to be a horizontal. The threshold value fixed at 4.0 is used to distinguish vertical lines. If the value of the slope is higher than the threshold value, then the line is adjusted as vertical. In all other cases, the lines are kept with the same angle of inclination. In the next stage, the specific features are extracted from the preprocessed pattern primitives of the characters under each cluster.

Table 5.13 lists the number of characters in each cluster and the corresponding number of features to be extracted for experimental purposes. The features specified for each cluster are already described in sections 5.4.1 to 5.4.7.

Table 5.13 Number of features in various clusters

Sl.No	Cluster	Number of characters	Number of features
1	A	3	3
2	B	6	6
3	C(Up)	7	7
4	C(Down)	5	6
5	D(Up)	5	5
6	D(Down)	3	4
7	E	4	5
8	F(L)	4	4
9	F(NL)	5	5
10	G	2	2

The cluster wise (class-modular) character classification experiments are conducted based on the features extracted from the pattern primitives with respect to the characters belonging to each cluster using the Support Vector Machines (SVMs). The train and test sets are prepared in the ratio of 3:2 for experimental purposes. Five-fold cross-validation is also performed to measure the classification accuracy. The following three basic numeric scores, true positive, false positive, and false negative, are computed to evaluate the performance of the system.

-
- True positive: Number of characters correctly recognized
 - False-positive: Number of characters falsely recognized
 - False-negative: Number of missed characters

The performance scores *viz.* precision, recall, and F1-score of the experiments, are also computed based on the above parameters. The precision is computed to measure how many of the characters are identified correctly. Recall measures, how many characters in the test samples are not missed in recognition. F1 score is the weighted average of precision and recall, and it is also referred to as harmonic mean, which represents the overall accuracy of the system.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}}$$

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}}$$

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}}$$

The following section describes the performance analysis of the proposed pattern primitive based OHCR system.

5.6. Analysis of the OHCR Results

In this work, the experiments are conducted in a class modular approach, i.e., classification is performed within every cluster. In cluster A, there are three character classes with three features. The performance measures of the classifier used for classifying the characters within cluster A with precision, recall, and F1-score are shown in table 5.14. The average recognition accuracy obtained for the characters in cluster

A is 96.67%. The normalized confusion matrix obtained for cluster A is shown in figure 5.9.

Table 5.14 Performance scores for cluster A

Character	Class label	Precision	Recall	F1-Score
ᳵ <bha>	32	0.77	1.00	0.87
ᳶ <da>	26	1.00	0.80	0.89
᳷ <ra>	44	1.00	0.90	0.95

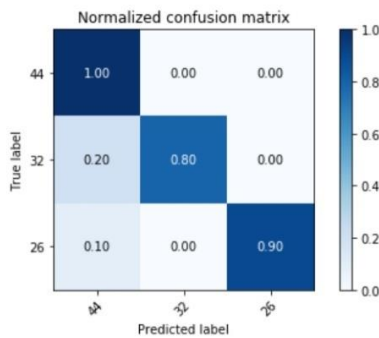


Fig. 5.9 Normalized confusion matrix for cluster A

In cluster B, there is six character classes with six features. The performance scores obtained for the classifier used to classify the characters of cluster B are shown in table 5.15. The average recognition accuracy obtained for the characters in cluster B is 98.88%. The normalized confusion matrix obtained for cluster B is shown in figure 5.10.

Table 5.15 Performance scores for cluster B

Character	Class label	Precision	Recall	F1-Score
ᳵ <o>	8	1.00	1.00	1.00
ᳶ <ga>	11	1.00	1.00	1.00
᳷ <ṭa>	19	0.95	1.00	0.98
᳸ <ṭha>	20	1.00	1.00	1.00
᳹ <ra>	35	1.00	0.95	0.97
ᳺ <la>	43	1.00	1.00	1.00

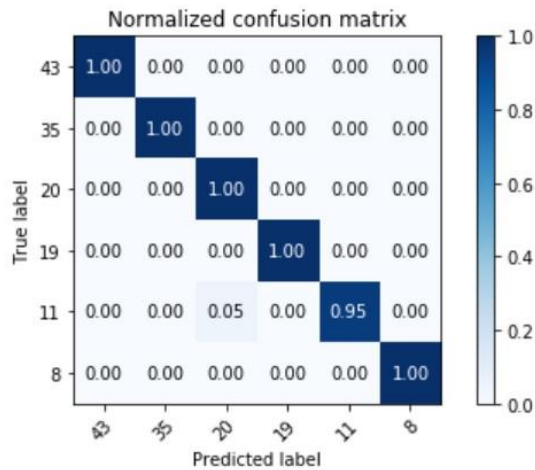


Fig. 5.10 Normalized confusion matrix for cluster B

In cluster C(U_p), there are seven characters with seven features. The performance scores obtained for the classifier used to classify the characters of cluster C(U_p) are shown in table 5.16. The average recognition accuracy obtained for the characters in cluster C(U_p) is

96.20%. The normalized confusion matrix obtained for cluster C(Up) is shown in figure 5.11.

Table 5.16 Performance scores for cluster C(Up)

Character	Class Label	Precision	Recall	F1-Score
ᱫ <r>	5	1.00	0.85	0.92
ᱠ <ca>	14	0.83	1.00	0.91
ᱠᱤ <ta>	24	1.00	1.00	1.00
ᱠᱤᱢ <na>	28	0.91	1.00	0.95
ᱠᱤᱨ <pa>	29	0.89	0.85	0.87
ᱠᱤᱣ <ma>	33	1.00	0.85	0.92
ᱠᱤᱦ <va>	37	0.95	1.00	0.98

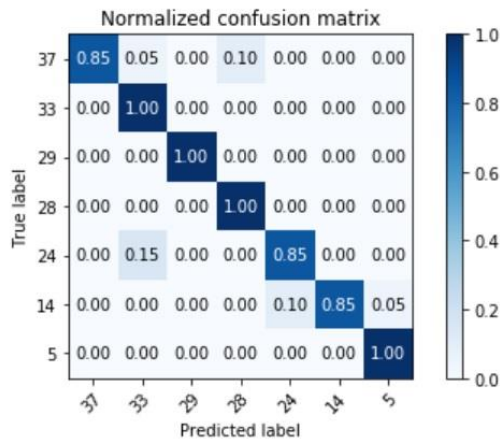


Fig. 5.11 Normalized confusion matrix for cluster C(Up)

In cluster C(Down), there are five characters with six features. The performance scores obtained for the classifier used to classify the characters of cluster C(Down) are shown in table 5.17. The average

recognition accuracy obtained for the characters in cluster C(Down) is 98.67%. The normalized confusion matrix obtained for cluster C(Down) is shown in figure 5.12.

Table 5.17 Performance scores for cluster C(Down)

Character	Class label	Precision	Recall	F1-Score
உ <u>	4	1.00	1.00	1.00
ட <tha>	25	1.00	1.00	1.00
ஓ <dha>	27	1.00	1.00	1.00
ஐ <la>	42	1.00	1.00	1.00
ஔ <śa>	38	1.00	1.00	1.00

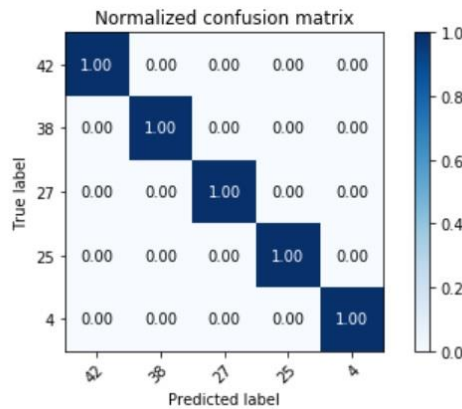


Fig. 5.12 Normalized confusion matrix for cluster C(Down)

In cluster D(Up), there are five characters with six features. The performance scores obtained for the classifier used to classify the characters of cluster D(Up) are shown in table 5.18. The average

recognition accuracy obtained for the characters in cluster D(Up) is 94.67 %. The normalized confusion matrix obtained for cluster D(Up) is shown in figure 5.13

Table 5.18 Performance scores for cluster D(Up)

Character	Class label	Precision	Recall	F1-Score
ഖ <kha>	10	1.00	1.00	1.00
ട <ḍa>	21	1.00	0.80	0.89
ല <la>	34	1.00	1.00	1.00
സ <sa>	36	1.00	1.00	1.00
യ <ya>	40	0.83	1.00	0.91

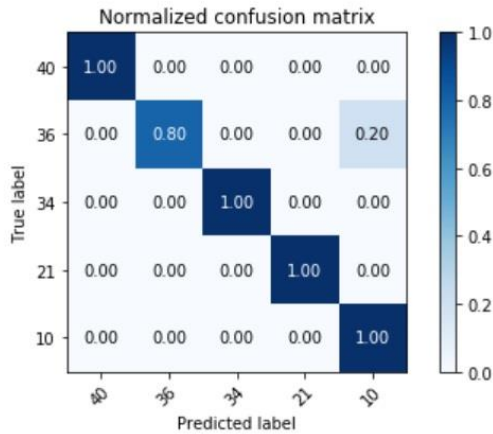


Fig. 5.13 Normalized confusion matrix for cluster D(Up)

In cluster D(Down), there are three characters with four features. The performance scores obtained for the classifier used to classify the characters of cluster D(Down) are shown in table 5.19. The average recognition accuracy obtained for the characters in cluster D(Down) is

100%. The normalized confusion matrix obtained for cluster D(Down) is shown in figure 5.14.

Table 5.19 Performance scores for cluster D(Down)

Character	Class label	Precision	Recall	F1-Score
ਖ਼ <kha>	13	1.00	1.00	1.00
ਘ <pha>	30	1.00	1.00	1.00
ਹ <ha>	41	1.00	1.00	1.00

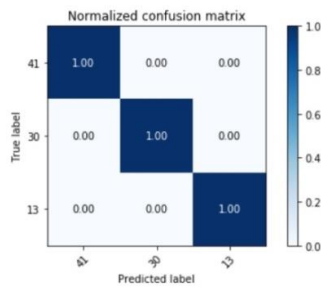


Fig. 5.14 Normalized confusion matrix for cluster D(Down)

In cluster E, there are four characters with five features. The performance scores obtained for the classifier used to classify the characters of cluster E are shown in table 5.20. The average recognition accuracy obtained for the characters in cluster E is 100%. The normalized confusion matrix obtained for cluster E is shown in figure 5.15.

Table 5.20 Performance scores for cluster E

Character	Class label	Precision	Recall	F1-Score
ഇ <i>	3	1.00	1.00	1.00
ക <ka>	9	1.00	1.00	1.00
ച <cha>	15	1.00	1.00	1.00
ഛ <ḥha>	22	1.00	1.00	1.00

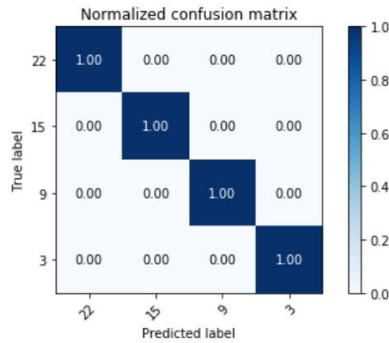


Fig. 5.15 Normalized confusion matrix for cluster E

In cluster F(L), there are four characters with four features. The performance scores obtained for the classifier used to classify the characters of cluster F(L) are shown in table 5.21. The average recognition accuracy obtained for the characters in cluster F(L) is 81.67%. The normalized confusion matrix obtained for cluster F(L) is shown in figure 5.16.

Table 5.21 Performance scores for cluster F(L)

Character	Class label	Precision	Recall	F1-Score
എ <e>	6	1.00	0.40	0.57
ഏ <e:>	7	1.00	1.00	1.00
ഈ <gha>	12	1.00	1.00	1.00
ഘ <ṣa>	39	0.62	1.00	0.77

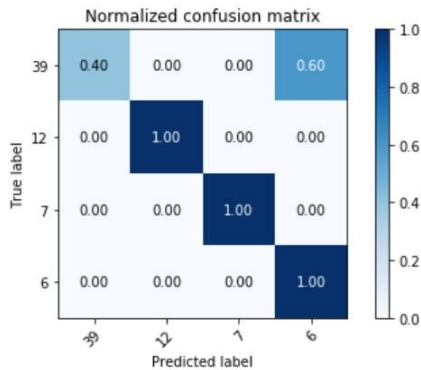


Fig. 5.16 Normalized confusion matrix for cluster F(L)

In cluster F(NL), there are five characters with five features. The performance scores obtained for the classifier used to classify the characters of cluster F(NL) are shown in table 5.22. The average recognition accuracy obtained for cluster F(NL) characters is 99.33%. The normalized confusion matrix obtained for cluster F(NL) is shown in figure 5.17.

Table 5.22 Performance scores for cluster F(NL)

Character	Class label	Precision	Recall	F1-Score
അ <a>	1	1.00	1.00	1.00
ഝ <jha>	17	0.95	1.00	0.98
ഞ <ña>	18	1.00	1.00	1.00
ണ <ṇa>	23	1.00	0.95	0.97
ബ <ba>	31	1.00	1.00	1.00

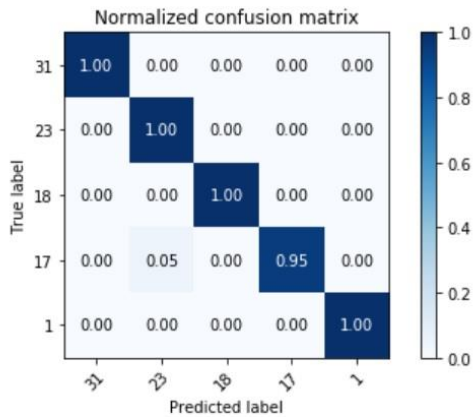


Fig.5.17 Normalized confusion matrix for cluster F(NL)

In cluster G, there are two characters with two features. The performance scores obtained for the classifier used to classify the characters of cluster G are shown in table 5.23. The average recognition accuracy obtained for the characters in cluster G is 100%. The normalized confusion matrix obtained for cluster G is shown in figure 5.18.

Table 5.23 Performance scores for cluster G

Character	Class label	Precision	Recall	F1-Score
അ <a:>	2	1.00	1.00	1.00
ജ <ja>	16	1.00	1.00	1.00

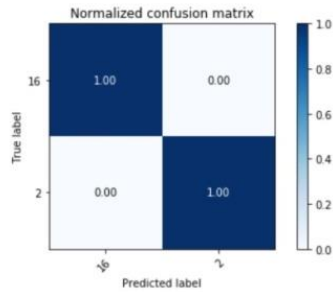


Fig. 5.18 Normalized confusion matrix for cluster G

The average recognition accuracies obtained for each character cluster are listed in table 5.24. The graphical representation of the same is shown in figure 5.19.

Table 5.24 Average recognition accuracies for each cluster

Cluster	Number of characters per cluster	Accuracy (%)
A	3	96.67
B	6	98.88
C(U)	7	96.20
C(D)	5	98.67
D(U)	5	94.67
D(D)	3	100.00
E	4	100.00
F(L)	4	81.67
F(NL)	5	99.33
G	2	100.00
	Average Accuracy	96.61

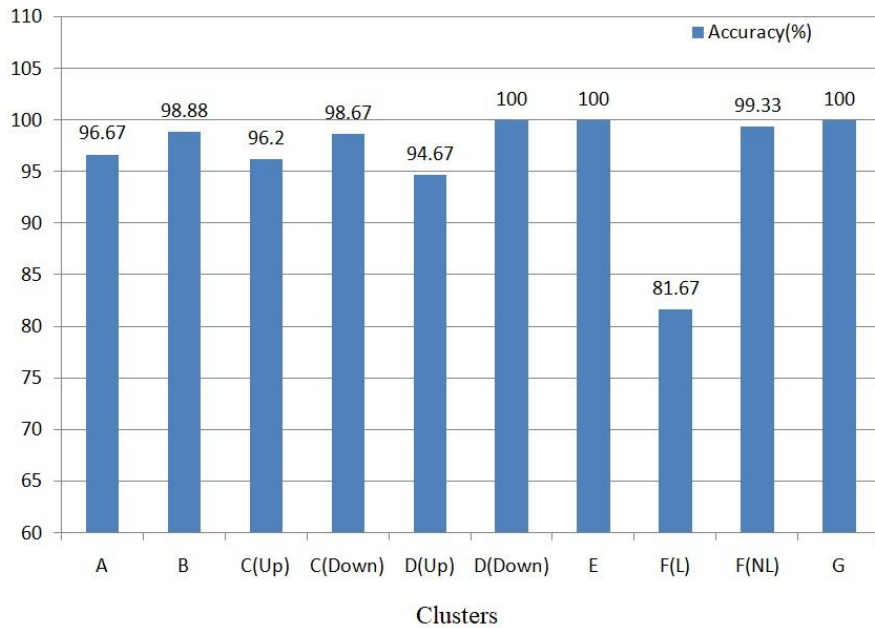


Fig. 5.19 Average recognition accuracies for each cluster

From the experimental results, it is found that, the average recognition accuracy obtained for all the clusters is 96.61%. Hence, it is evident that the features related to the pattern primitives can be effectively utilized to recognize online handwritten characters.

5.7. Conclusion

The chapter discussed a scheme to recognize Malayalam online handwritten characters based on pattern primitives. Initially, various clusters are formed based on the number of pattern primitives present in each character. A detailed description of various pattern primitive features and the algorithms used to extract these features are presented. The specific features are extracted from the pattern primitives constituting the characters representing each cluster.

The classification experiments are performed based on the pattern primitive features using the Support Vector Machine classifier. The recognition accuracies obtained from each character cluster are tabulated. The average recognition accuracy obtained for all the clusters is 96.61%. The performance of the proposed method is found to be promising, and it can be effectively utilized for the development of an OHCR system in Malayalam based on pattern primitives.

A Predictive Model for Real-Time Recognition of Malayalam Handwritten Characters based on Pattern Primitives

6.1. Introduction

In the last few decades, the rapid technological changes in input-output devices accelerated the research to simplify handwriting recognition systems. Even though online handwriting recognition systems are also known as real-time systems, the latency in data acquisition and execution time for recognition engines inculcates a time factor that obliterates the meaning of real-time behavior [2][109].

In real-time handwriting recognizers, the system must be capable of recognizing the characters within a lesser time. The invention of faster and more compact devices with touch screen interfaces made attention to new techniques to recognize handwriting faster. An online handwriting recognition method with minimum recognition time is best suited for real-time applications, especially for mobile platforms and touch interfaces. Recently, certain promising OHCR applications for mobile devices have been launched that are suited for real-time recognition of handwritten characters in multiple languages [110]. But these systems recognize the characters after completion of writing with a time delay that is negligible for human understanding. Ideally, in a real-time system, it is expected that the character should be recognized even before the completion of writing. Studies that focus on real-time recognition of handwritten characters are limited in literature, especially

for Indic scripts. In this context, it is highly relevant to conduct an extensive study emphasizing real-time aspects of online recognition of handwritten characters.

This chapter proposes a real-time model for Malayalam OHCR based on pattern primitives. It includes a Deterministic Finite Automata (DFA) based OHCR and a predictive model for the real-time recognition of Malayalam handwritten characters using pattern primitives. The proposed model is shown in figure 6.1.

The rest of the chapter is organized as follows. The representation of Malayalam online handwritten characters as pattern Primitive String (PPS) is described in section 6.2. The design and implementation of a Deterministic Finite Automata (DFA) for Malayalam OHCR is detailed in section 6.3. The predictive model for real-time recognition of Malayalam handwritten characters based on pattern primitives with their transition sequence and tree representation is presented in section 6.4. The average Reduction in Writing Time (RWT) for the characters is estimated in section 6.5, followed by the implementation of the real-time prediction model. The results are analyzed in detail, and conclusive remarks are given in section 6.6.

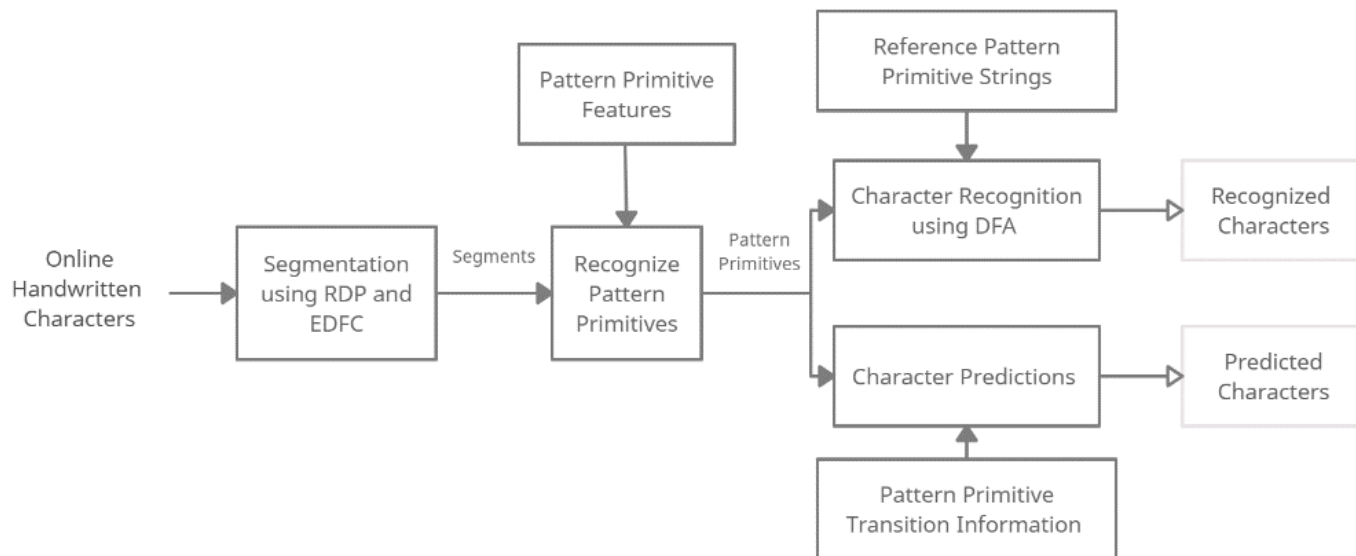


Fig. 6.1 Proposed model for real-time recognition of Malayalam handwritten characters based on pattern primitive

6.2. Pattern Primitive String (PPS) Representation of Malayalam Online Handwritten Characters

The representation of Malayalam online handwritten characters in the form of pattern primitive strings are discussed in this section. Representing characters in the form of feature strings are common in OHCR [39][111][112]. The characters represented as Pattern Primitive Strings (PPS) are easily identifiable and useful for further recognition processes. All the pattern primitives present in Malayalam characters have already been identified and labeled in chapter 4. In order to conveniently represent the characters as a string of pattern primitives, the existing labels assigned to the pattern primitives are modified, as shown in table 6.1.

Table 6.1 Modified labels of pattern primitives

Existing Label	Modified Label	Existing Label	New Label
A1	a	E1	n
B2	b	F2	o
G	c	H2	p
C2	d	J1	q
D1	e	L1	r
H1	f	M1	s
B1	g	N2	t
A2	h	H3	u
I1	i	I2	v
K	j	I3	w
L2	k	J2	x
C1	l	M2	y
D2	m	N1	z

The characters are represented as Pattern Primitive Strings (PPS) using the modified labels based on the pattern primitive sequences represented in table 4.2 and table 4.3 of chapter 4. Table 6.2 lists the PPS representations of the Malayalam characters thus obtained. These PPS representations are used effectively for constructing DFAs corresponding to each character.

Table 6.2 Pattern Primitive String (PPS) representation of Malayalam handwritten characters

S.No	Character	PPS representation	S.No	Character	PPS representation
1	അ <a>	aiadeba	23	ണ <na>	badedeb
2	ആ <a:>	aiadebaj	24	ത <ta>	abab
3	ഇ <i>	badejc	25	ഥ <tha>	pclm
4	ഉ <u>	bajc	26	ദ <da>	ai
5	ഋ <ṛ>	gqXH	27	ധ <dha>	hnog
6	എ <e>	abcfpab	28	ന <na>	abab
7	ഏ <e:>	abcfpai	29	പ <pa>	abcf
8	ഒ <o>	bai	30	ഫ <pha>	abclm
9	ക <ka>	abstab	31	ബ <ba>	badebcf
10	ഖ <kha>	badcf	32	ഭ <bha>	aw
11	ഗ <ga>	hrb	33	മ <ma>	abcu
12	ഘ <gha>	abclmcf	34	യ <ya>	abakg
13	ങ <kha>	badei	35	ര <ra>	aba

Table 6.2 Pattern Primitive String (PPS) representation of Malayalam handwritten characters (cont.)

S.No	Character	PPS representation	S.No	Character	PPS representation
14	ച <ca>	ajcf	36	ല <la>	clmcf
15	ച <cha>	ajcaba	37	വ <va>	abcf
16	ജ <ja>	badejrba	38	ശ <sa>	hrba
17	ജ <jha>	abaknog	39	ഷ <sha>	abcftsd
18	ഞ <ña>	badebab	40	സ <sa>	adekg
19	ട <ta>	gqa	41	ഹ <ha>	abcab
20	ഠ <tha>	hgh	42	ള <la>	bavc
21	ഡ <da>	aknog	43	ഴ <za>	yzq
22	ഢ <dha>	aknogh	44	റ <ra>	ab

6.3. Pattern Primitive based OHCR using DFA

This section describes the design and implementation of pattern primitive based OHCR using Deterministic Finite Automata (DFA). As a first step, the online handwritten characters are segmented as per the pattern primitive reference set using the combined approach of RDP and EDFC algorithms described in chapter 4. In the next step, the pattern primitives are recognized using their features. These pattern primitives are then used for the conduct of OHCR experiments based on DFA.

Finite State Automata or Deterministic Finite State Automata (DFA) have been attempted in various offline and online handwritten character

recognition problems for the last few decades[41][113][114]. In a Deterministic Finite State Automata (DFA), initial states are accepted to predict the final state. The Deterministic Finite Automata (DFA) is a finite-state machine that accepts or rejects a given string of symbols by running through a state sequence uniquely determined by the string[115]. The term deterministic refers to the uniqueness of the computation run. In search of the simplest models to capture finite state machines, Warren Mc Culloch , and Walter Pitts were among the first researchers to introduce a concept of finite concept automata [116][117].

DFA is a quintuple $(Q, \Sigma, \delta, q_0, F)$ where,

Q ; is the fixed non-empty finite states of the automata

Σ ; is the fixed non-empty alphabet, $Q \cap \Sigma = \{ \}$

δ ; is a transition function, $Q \times \Sigma \rightarrow Q$ (i.e., for each $q \in Q, a \in \Sigma$,

$\delta(q,a)$ is the next state when processing symbol a from state q)

q_0 ; is the initial state, $q_0 \in Q$

F ; is the set of accepting states(final state), $F \subseteq Q$

A sample DFA that accepts all strings starting with 'ab' is shown in figure 6.2, where the alphabets are $\{a,b\}$ and L (Language) is $\{ 'ab', 'abab', 'abbbb', 'ab*' \}$

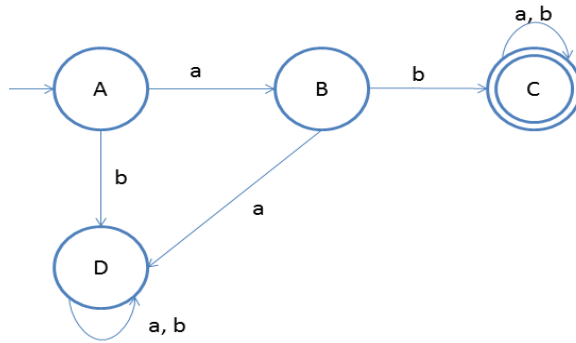


Fig. 6.2 DFA which accepts all strings starting with ‘ab’

It can be observed from figure 6.2, that after accepting the smallest string ‘ab’, whatever string comes, the DFA consider them as don’t care since the condition is already satisfied. In a DFA, all the input alphabets at every state are to be considered. So the input symbol ‘b’ on state A is rejected to state D (rejected state). On state B, if ‘a’ comes, then reject it, so this is directed to state D. On state D for input ‘a’ and ‘b’, a self-loop is made as these are don’t care states.

6.3.1. Design of the Deterministic Finite Automata based on Pattern Primitives Present in the Characters

DFAs are constructed for every Pattern Primitive String (PPS) representing each of the 44 Malayalam online handwritten characters. For example, the Pattern Primitive String (PPS) of the Malayalam online handwritten character \cap <na> is obtained as ‘abab’, and the corresponding DFA representing the character is shown in figure 6.3.

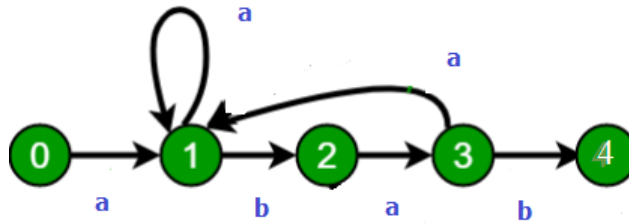


Fig. 6.3 DFA of the pattern primitive string ‘abab’ representing the Malayalam online handwritten character ഞ <na>

After obtaining the DFAs for all the characters, an unknown character represented as PPS is fed to these DFAs for recognition. If the PPS representing a character is accepted by any of the DFA, then the character is recognized as the character corresponding to that DFA. Figure 6.4 illustrates a sample DFA representing the Malayalam online handwritten character ച <cha>.

6.3.2. Implementation of the DFA based OHCR

The experiments are conducted for the recognition of Malayalam online handwritten characters, based on the DFAs generated, as described in section 6.3.1. For the experimental purpose, hundred samples of each vowel characters taken from the CU-OHDB dataset are used. These samples are then segmented, using the combined approach of RDP and EDFC algorithms. It is found that, there are thirteen distinct pattern primitives present in these vowel characters. The features identified for these thirteen distinct pattern primitives constituting the Malayalam vowel characters are listed in table 6.3. A reference feature set for each pattern primitive (present in eight vowel characters under study) is created by extracting the features specified in table 6.3. As a next step,

the character segments obtained are classified into pattern primitives, by matching the features extracted from each segment against all reference pattern primitive features.

In the next phase, a reference Pattern Primitive String (PPS) is created for each vowel character as per the representation given in section 6.2. The DFAs corresponding to each vowel character is created using these reference Pattern Primitive String (PPS). The algorithms described in procedure 6.1 and procedure 6.2 is implemented in MATLAB for the conduct of recognition experiments. The recognition experiments are conducted for the hundred samples of each of the eight vowel characters. The Pattern Primitive String (PPS) corresponding to each test sample are given as input to the reference DFAs created for each vowel character. The recognition accuracies obtained for each vowel character are tabulated in table 6.4. The average recognition accuracy obtained for all the characters is 65.75 %.

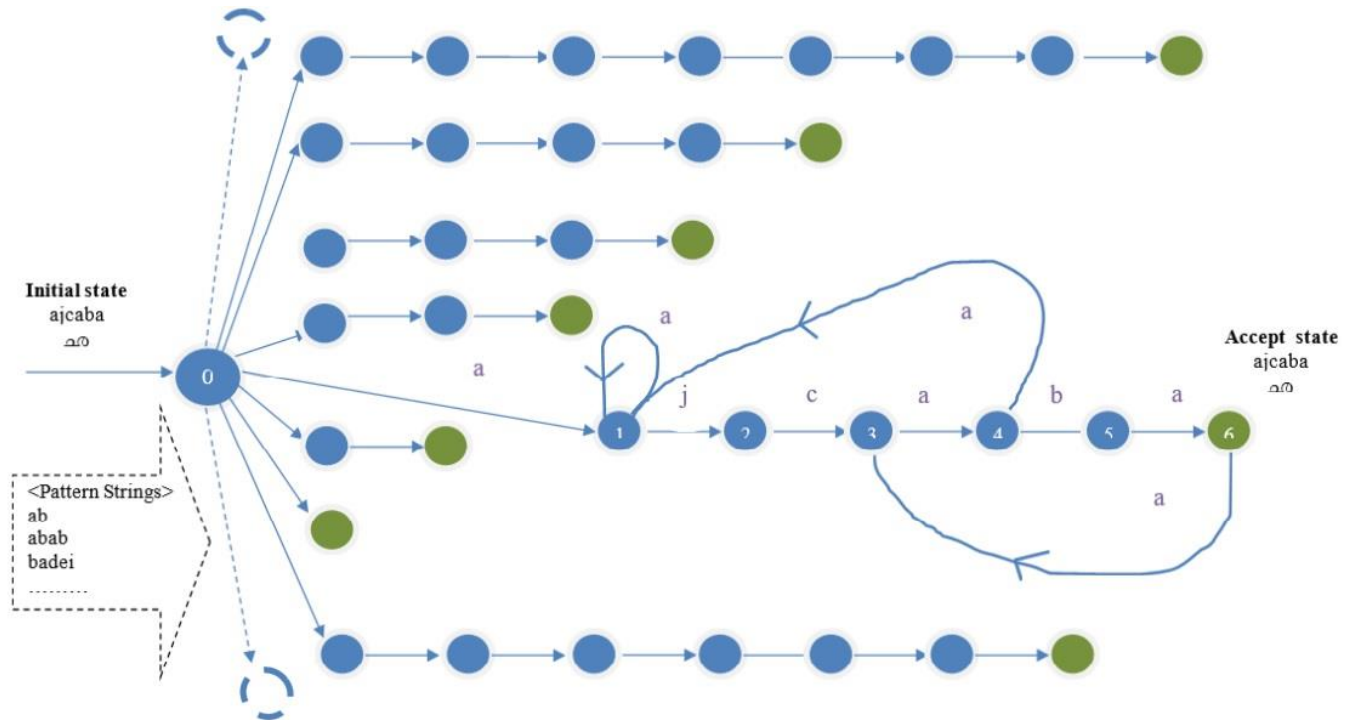


Fig. 6.4 A sample DFA representing the Malayalam online handwritten character ച <cha>

Table 6.3 Features of the pattern primitives present in the Malayalam vowel characters














Sl.No	Pattern primitive	Label	Direction	Features
1		a	Upwards	Reduced Direction Code, Writing direction, Arc / Straight-line
2		b	Downwards	Reduced Direction Code, Writing direction, Arc / Straight-line
3		c	Left to right	Reduced Direction Code, Writing direction, Arc / Straight-line
4		d	Downwards	Reduced Direction Code, Writing direction, Arc / Straight-line
5		e	Upwards	Reduced Direction Code, Writing direction, Arc / Straight-line
6		f	Upwards	Reduced Direction Code, Writing direction, Arc / Straight-line
7		g	Upwards	Reduced Direction Code, Writing direction, Arc / Straight-line

Table 6.3 Features of the pattern primitives present in the Malayalam vowel characters (cont.)

Sl.No	Pattern primitive	Label	Direction	Features
8		h	Downwards	Reduced Direction Code, Writing direction, Arc / Straight-line
9		i	Downwards	Reduced Direction Code, Writing direction, Arc / Straight-line, Cusp
10		j	Downwards	Reduced Direction Code, Writing direction, Arc / Straight-line
11		p	Downwards	Reduced Direction Code, Writing direction, Arc / Straight-line
12		q	Downwards	Reduced Direction Code, Writing direction, Arc / Straight-line
13		x	Upwards	Reduced Direction Code, Writing direction, Arc / Straight-line

Procedure 6.1 Implementation of the OHCR using DFA

Input: PointList(N), The set of points describing N characters

Output: Hit, The hit ratio of every character

Procedure OHCR_DFA()

 For i=1 to N

 Segments(i) = **Combined_Approach**(PointList(i)) *// using procedure 4.4*

 Features(i) = Extract features from Segments(i) *// using techniques in chapter 5*

 PPS(i) = Compare (Features(i), Reference (1..13)) *// Assign pattern primitive*

labels

 Hit(i)=DFAT(PPS(i), ReferencePPS) *// test DFA*

 End

End

Procedure 6.2 Implementation of the algorithm for generating DFA

Input: Pattern, The pattern primitive string to be searched

 String, The reference string of pattern primitives

Output: Hit, The state where the pattern ends if any else return '0'

Procedure DFAT (Pattern, String)

 Find Newtstate from Currentstate as

 Newtstate=Transition (Strings, States, Currentstate, Stringcount)

 Represent transitions as a table from States

 Check for similarities for the pattern primitive strings

 If found

 Hit=Position where the pattern ends

 else

 Hit=0

 Return (Hit)

End

Table 6.4 Recognition accuracies obtained for the OHCR experiments using Malayalam vowel characters based on DFA

Character	Pattern Primitive String (PPS)	Recognition Accuracy(%)
അ <a>	aiadeba	66
ആ <a:>	aiadebaj	62
ഇ <i>	badec	70
ഉ <u>	baec	74
ഋ <r̄>	bqXH	76
എ <e>	abcfpab	56
ഐ <e:>	abcfpai	52
ഓ <o>	bai	70
Average Recognition Accuracy		65.75

Even though the results are promising, the method uses a holistic approach, which considers the entire character for recognition purpose. In this approach, the character will be recognized only after finishing the writing of the entire character which causes considerable delay in recognition. To address the issues related to the time factors in real-time implementation of the OHCR systems, an alternate predictive model for the recognition of Malayalam handwritten characters is proposed.

6.4. The Predictive Model for Real-Time Recognition of Malayalam Handwritten Characters based on Pattern Primitives

The predictive model for real-time recognition of Malayalam handwritten characters based on pattern primitives is presented in this section. For this purpose, the transition sequences of pattern primitives that constitute each character are analyzed in detail. These pattern primitive transition information obtained from the characters are utilized for constructing a tree structure for the possible predictions of every character in a real-time environment. The Reduction in Writing Time (RWT) for the Malayalam online handwritten characters is computed from these tree representations. Experiments are also conducted to verify the effectiveness of the model, using online handwritten character samples taken from the CU-OHDB dataset.

6.4.1. Pattern Primitive Transition Sequences in the Malayalam Online Handwritten Characters

Pattern primitive transition sequences present in the Malayalam online handwritten characters are analyzed in this section. The sequence of occurrence of a pattern primitive after a selected pattern primitive in the formation of a handwritten character is termed as pattern primitive transition sequence. A detailed description of pattern primitives present in a character is already given in chapter 4. The pattern primitive transition sequences obtained for the 44 Malayalam online handwritten characters are displayed in table 6.5. The tick mark (✓) in the table denotes the occurrence of the pattern primitive after the selected pattern

primitive, and the cross mark (x) denotes the non-occurrence of the pattern primitive.

This information is highly useful in designing a predictive model for real-time HCR. While proceeding with the character recognition, some characters could be excluded from the process based on the possibility of the occurrence of certain pattern primitives preceding the other. This predictive approach will considerably reduce certain computations. Hence it can be effectively utilized for implementing an improved OHCR system.

Table 6.5 Occurrence of a pattern primitive after a selected pattern primitive obtained for the Malayalam characters
































Pattern primitives with labels	a	h	g	b	l	d	e	m	n	o	c	f	p	u	i	v	w	q	x	j	r	k	s	y	z	t
																										
a 	x	x	x	✓	x	✓	x	x	x	x	x	x	x	x	✓	✓	✓	x	x	✓	x	✓	x	x	x	x
h 	x	x	✓	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x
g 	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x
b 	✓	x	x	x	x	x	x	x	✓	x	✓	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	x
l 	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 6.5 Occurrence of a pattern primitive after a selected pattern primitive obtained for the Malayalam characters (cont.)

Pattern primitives with labels	a	h	g	b	l	d	e	m	n	o	c	f	p	u	i	v	w	q	x	j	r	k	s	y	z	t
d	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
e	x	x	x	✓	x	✓	x	x	x	x	x	x	x	x	✓	x	x	x	x	✓	x	✓	x	x	x	x
m	x	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
n	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
o	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 6.5 Occurrence of a pattern primitive after a selected pattern primitive obtained for the Malayalam characters (cont.)

Pattern primitives with labels	a	h	g	b	l	d	e	m	n	o	c	f	p	u	i	v	w	q	x	j	r	k	s	y	z	t
c —	✓	x	x	x	✓	x	x	x	x	x	x	✓	x	✓	x	x	x	x	x	x	x	x	x	x	x	x
f	x	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	✓
p	✓	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
u \	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
i 3	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
v 3	x	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 6.5 Occurrence of a pattern primitive after a selected pattern primitive obtained for the Malayalam characters (cont.)
































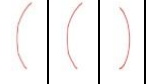






























Pattern primitives with labels		a	h	g	b	l	d	e	m	n	o	c	f	p	u	i	v	w	q	x	j	r	k	s	y	z	t
																											
w		x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
q		✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x
x		x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
j		x	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x
r		x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x

Table 6.5 Occurrence of a pattern primitive after a selected pattern primitive obtained for the Malayalam characters (cont.)

Pattern primitives with labels	a	h	g	b	l	d	e	m	n	o	c	f	p	u	i	v	w	q	x	j	r	k	s	y	z	t
																										
k 	x	x	✓	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
s 	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓
y 	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	x
z 	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	x	x	x	x	x	x	x	x	x
t 	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	✓	x	x	x

From the table 6.5, it can be observed, that the transition sequences of pattern primitives differ in various characters. It can be seen, that pattern primitive *a* is followed by maximum number of pattern primitives, and certain pattern primitives like *u* and *w* are not followed by any other pattern primitives.

It is also observed that, only seven pattern primitives are found to be occurring as the first pattern primitive in Malayalam characters, namely *a, b, c, h, g, p,* and *y*. The possible occurrence of the subsequent pattern primitives after these first primitives is also found comparatively lesser in number. It is also found that, there are only five possible subsequent pattern primitives for *a*, three possible pattern primitives subsequent to *h*, and two followers for *b*. For all the other first pattern primitives, a unique pattern primitive is followed in the second position. The possible transitions from the first pattern primitive to the second pattern primitives for the Malayalam handwritten characters are shown in figure 6.5.

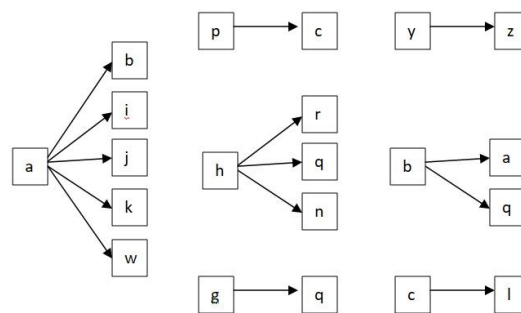


Fig. 6.5 The possible transitions from first to second pattern primitives in Malayalam handwritten characters

Considering the significance of the transitions from the first to second pattern primitives in Malayalam handwritten characters in the recognition process, the characters that belong to such transitions are identified and listed in Table 6.6.

Table 6.6. Transition of first to second pattern primitives and the corresponding characters

S.No	Transition of pattern primitives	Characters	No of characters
1	a → b	എ <e>, ഏ <e: >, ക <ka>, ഘ <gha>, ഞ <jha>, ത <ta>, ന <na>, പ <pa>, ഫ <pha>, മ <ma>, യ <ya>, ര <ra>, ള <va>, ഷ <sa>, ഹ <ha>, റ <ra>	16
2	b → a	ഇ <i>, ഉ <u>, ഒ <o>, ഖ <kha>, ണ <kha>, ജ <ja>, ണ <ña>, ണ <ña>, ബ <ba>, ള <la>	10
3	a → i	അ <a> , അ <a: >, ട <da>	3
4	a → j	ച <ca>, ച <cha> ,	2
5	a → k	ഡ <da>, ഡ <dha>	2
6	g → q	ട <ta>, റ്റ <t>	2
7	h → r	ഗ <ga>, ഴ <śa>	2
8	a → d	സ <sa>	1
9	a → w	ഭ <bha>	1
10	c → l	ല <la>	1
11	h → g	ഠ <tha>	1
12	h → n	ഡ <dha>	1
13	p → c	ഥ <tha>	1
14	y → z	ഴ <la>	1

From the table 6.6, it can be deduced that out of 44 characters, 25 characters start with the pattern primitive *a*, and 10 characters start with the pattern primitive *b* and most of the characters belong to the transition sequences *a* → *b* and *b* → *a*. The frequency of occurrence of various pattern primitive transitions present in the 44 Malayalam characters is

obtained in the form of a heat map, as shown in figure 6.6. From the figure, it can be seen that there are mainly six transitions occurring frequently. The highest frequency of transition is identified for $a \rightarrow b$, which is 23. For $b \rightarrow a$, the frequency of transition is 21 and for the transitions $b \rightarrow c$, $a \rightarrow d$, $c \rightarrow f$, and $d \rightarrow e$; it is 10. For all other transitions, the frequency ranges from 0 and 5.

While proceeding with the character recognition, some characters could be excluded from the process, based on the information obtained from the pattern primitive transition sequences in the Malayalam online handwritten characters. This will considerably reduce certain computations; hence can be effectively utilized for implementing an improved OHCR system.

	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z
a	0	23	0	10	0	0	0	0	5	3	4	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0
b	21	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	1	0	0	0	0	0	0	0
c	2	0	0	0	0	10	0	0	0	0	0	4	0	0	0	0	0	0	0	0	1	0	0	0	0	0
d	0	0	1	0	10	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
e	0	5	2	1	0	0	0	0	1	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
f	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	0	0	0	1	0	0	0	0	0	0
g	0	0	0	0	0	0	0	2	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	0
h	0	0	0	0	0	0	1	0	0	0	0	0	0	1	0	0	0	2	0	0	0	0	0	0	0	0
i	2	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
j	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
k	0	0	0	0	0	0	2	0	0	0	0	0	0	3	0	0	0	0	0	0	0	0	0	0	0	0
l	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
m	0	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
n	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0
o	0	0	0	0	0	0	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
p	2	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
q	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0
r	0	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
s	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0
t	1	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
u	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
v	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
w	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
x	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	0
y	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
z	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	0

Fig. 6.6 Heat map showing the frequency of pattern primitive transitions obtained for the Malayalam online handwritten characters

6.5. Tree Representation of the Predictive Model for Real-Time OHCR Implementation

In this section, tree representation of the predictive model proposed for the implementation of real-time OHCR is described in detail. The pattern primitives, especially, the starting primitives have a decisive role in the real-time recognition of handwritten characters. Hence, tree representations are generated by fixing the first pattern primitives as the root node. The proposed character prediction models, based on tree representation with first pattern primitives as root nodes, are explained in the following sections.

6.5.1. Tree Representation of the Predictive Model based on the First Pattern Primitive *b* as Root Node

Out of the 44 Malayalam online handwritten characters under consideration in the study, the characters namely, ഉ <u>, ഓ <o>, ഓ <kha>, ജ <ja>, ഞ <ña>, ഖ <kha>, ണ <ṇa>, ബ <ba>, ഇ <īa> and, ഇ <i> start with the pattern primitive *b*. The tree representation of the proposed predictive model for these characters, generated, based on the pattern primitive transitions with the character prediction stage, is shown in figure 6.7. From the tree representation, it can be observed that, the characters ഉ <u>, ജ <ja>, ഞ <ña>, ഖ <kha>, ണ <ṇa> and, ബ <ba> can be predicted even before completely writing all its constituent pattern primitives.

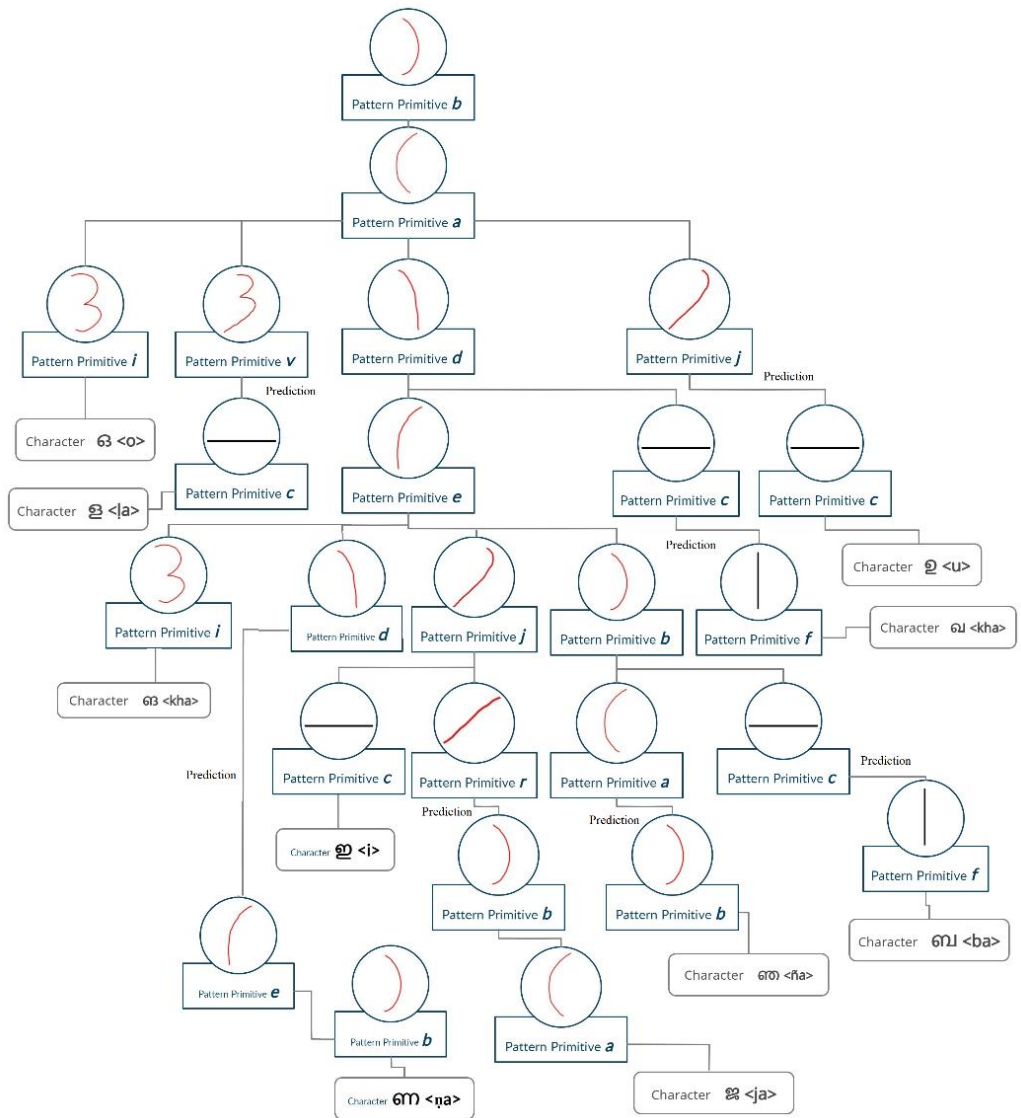


Fig. 6.7 Tree representation of the predictive model based on the first pattern primitive *b* as root node

6.5.2. Tree Representation of the Predictive Model based on the First Pattern Primitive p as Root Node

The only Malayalam character which starts with the pattern primitive p is ത <tha>. Even though the character ത <tha> has four pattern primitives, it can be easily predicted soon after the first pattern primitive is written. The tree representation of the proposed predictive model for the character, generated, based on the pattern primitive transitions with the character prediction stage is shown in figure 6.8.

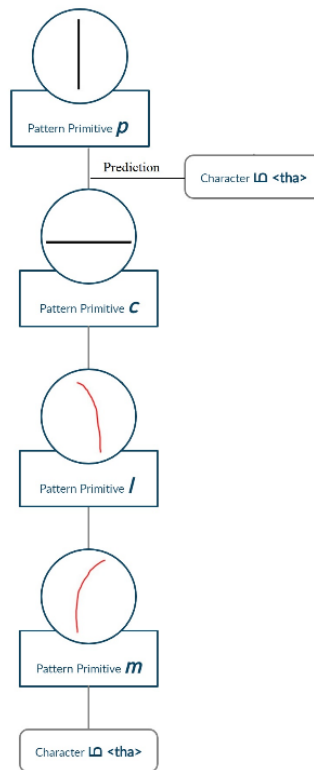


Fig. 6.8 Tree representation of the predictive model based on the first pattern Primitive p as root node

6.5.3. Tree Representation of the Predictive Model based on the First Pattern Primitive *c* as Root Node

From the analysis of the pattern primitive transitions, it is found that the character $\ominus \langle la \rangle$ is the only character that starts with the pattern primitive *c*. The character $\ominus \langle la \rangle$ has five pattern primitives. The tree representation of the proposed predictive model for the character $\ominus \langle la \rangle$, generated, based on the pattern primitive transitions with the character prediction stage is shown in figure 6.9. From the figure it is evident that, the character $\ominus \langle la \rangle$ can be predicted soon after writing the first pattern primitive.

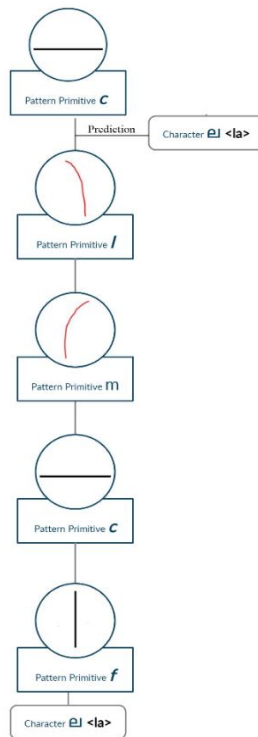


Fig. 6.9 Tree representation of the predictive model based on the first pattern Primitive *c* as root node

6.5.4. Tree Representation of the Predictive Model based on the First Pattern Primitive y as Root Node

The Malayalam online handwritten character φ <la> is identified to be starting with the pattern primitive y . The character has only three pattern primitives. The tree representation of the proposed predictive model for the character φ <la> with its character prediction stage is shown in figure 6.10. It can be noted that, the character φ <la> can be predicted even after completely writing the first pattern primitive.

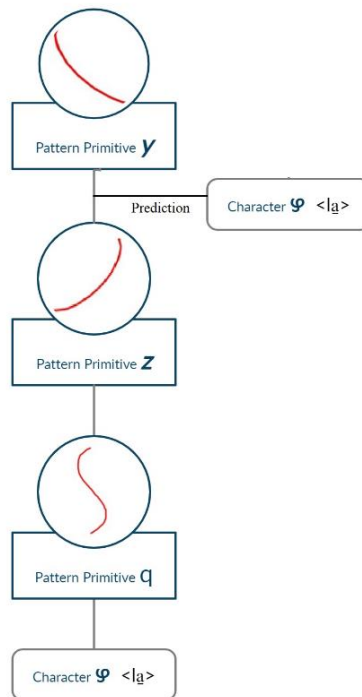


Fig. 6.10 Tree representation of the predictive model based on the first pattern Primitive y as root node

6.5.5. Tree Representation of the Predictive Model based on the First Pattern Primitive g as Root Node

The characters s <ᵗᵃ> and g <ᵗᵣ> start with the pattern primitive g . The first two pattern primitives of the characters s <ᵗᵃ> and g <ᵗᵣ> are same. Hence the third pattern primitive needs to be written completely to predict the character g <ᵗᵣ>. Figure 6.11 shows the tree representation of the pattern primitive transitions present in the characters s <ᵗᵃ> and g <ᵗᵣ> with their character prediction stages.

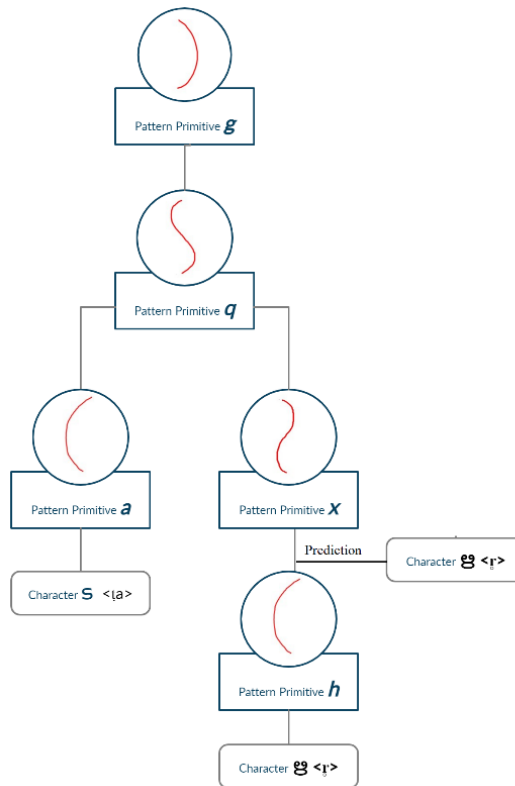


Fig. 6.11 Tree representation of the predictive model based on the first pattern Primitive g as root node

6.5.6. Tree Representation of the Predictive Model based on the First Pattern Primitive *h* as Root Node

The characters ω <ga>, \circ <ṭha>, ω <dha> and, ω <śa> start with the pattern primitive *h*. Figure 6.12 shows the tree representation of the pattern primitive transitions present in the characters ω <ga>, \circ <ṭha>, ω <dha> and, ω <śa> with the character prediction stages. From figure 6.12, it can be observed that, the character ω <dha> can be easily predicted after writing the second pattern primitive. Similarly, the character \circ <ṭha> can be predicted after writing the second pattern primitive.

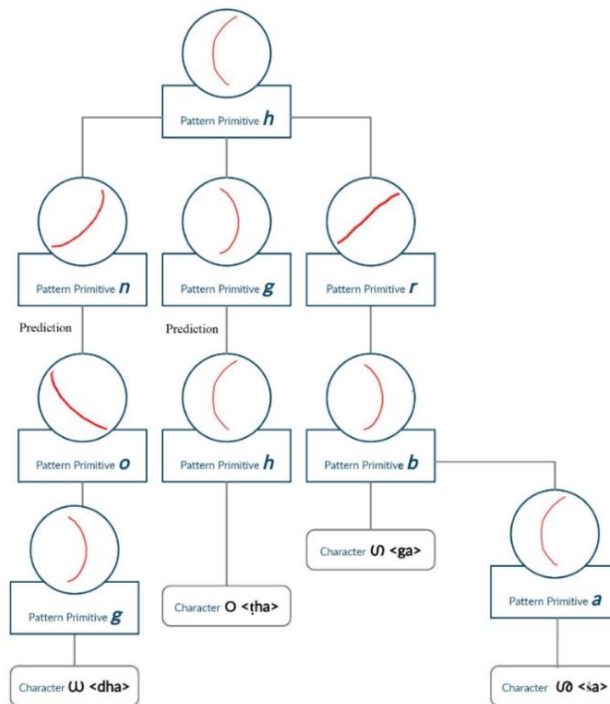


Fig. 6.12 Tree representation of the predictive model based on the first pattern Primitive *h* as root node

6.5.7. Tree Representation of the Predictive Model based on the First Pattern Primitive *a* as Root Node

There are twenty characters starting with pattern primitive *a*. They are അ <a>, ആ <a:>, എ <e>, ഏ <e:>, ക <ka>, ഘ <gha>, ച <ca>, ഛ <cha>, ത <jha>, ഡ <ḍa>, ഢ <ḍha>, ത <ta>, ദ <da>, ന <na>, പ <pa>, ഫ <pha>, ഭ <bha>, മ <ma>, യ <ya>, ര <ra>, വ <va>, ഷ <ṣa>, സ <sa>, ഹ <ha> and, റ <ra>. The tree representations of the proposed predictive model for these characters with their character prediction stages are shown in figure 6.13, 6.13(a), and 6.13(b). From the figures, it can be seen that, the characters അ <a>, ആ <a:> and, ദ <da> can be easily predicted after writing the second pattern primitive *i*. The character സ <sa> can be predicted after writing the second pattern primitive. After completing the writing process of the second pattern primitive *k*, the characters ഡ <ḍa> and ഢ <ḍha> can be predicted. Similarly, the characters ച <ca> and ഛ <cha> can be predicted after writing the second pattern primitive *j*. The characters എ <e> and, ഏ <e:> are predictable after writing their fourth pattern primitive. After writing the fifth pattern primitive, the character ഷ <ṣa> is predictable. For the character ത <jha>, prediction is possible after writing the fifth pattern primitive. The character ക <ka> can be predicted after its third primitive.

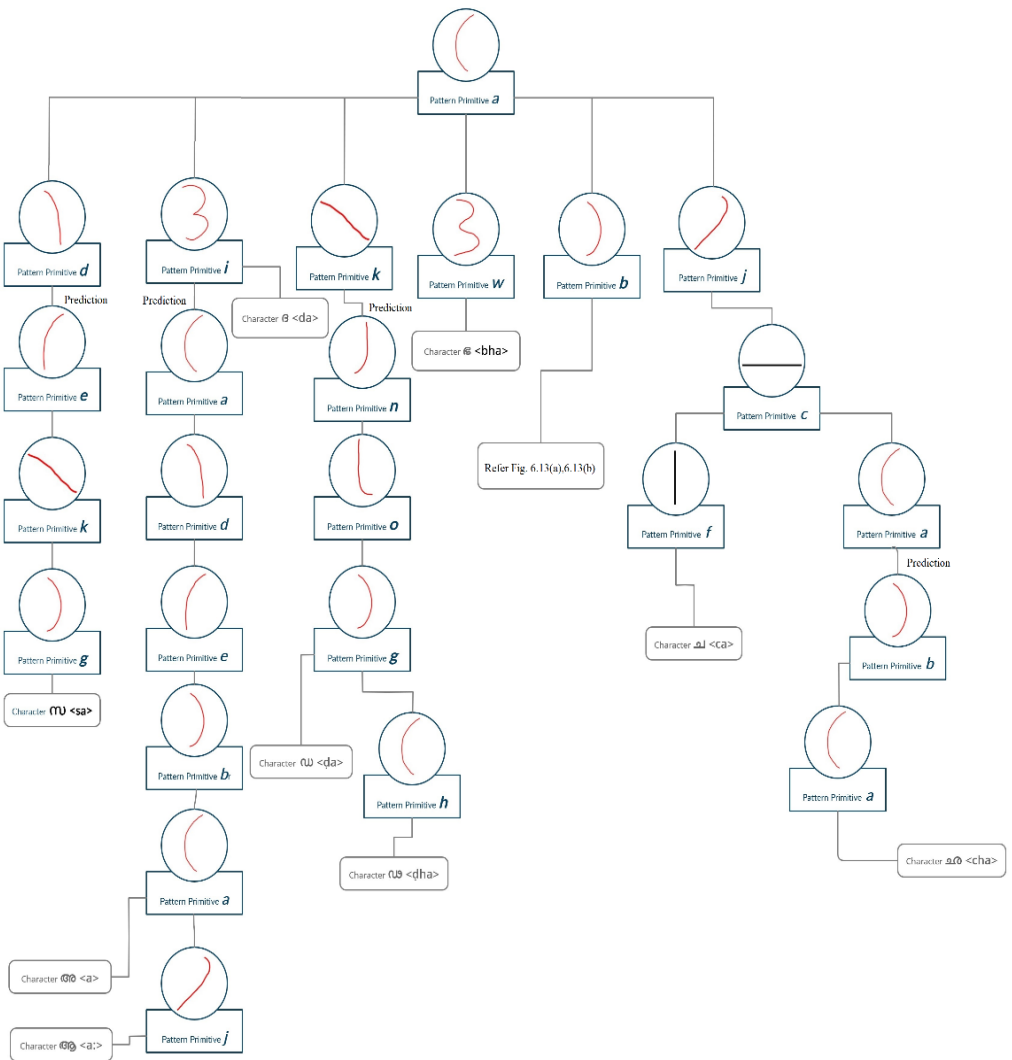


Fig. 6.13 Tree representation of the predictive model based on the first pattern primitive *a* as root node

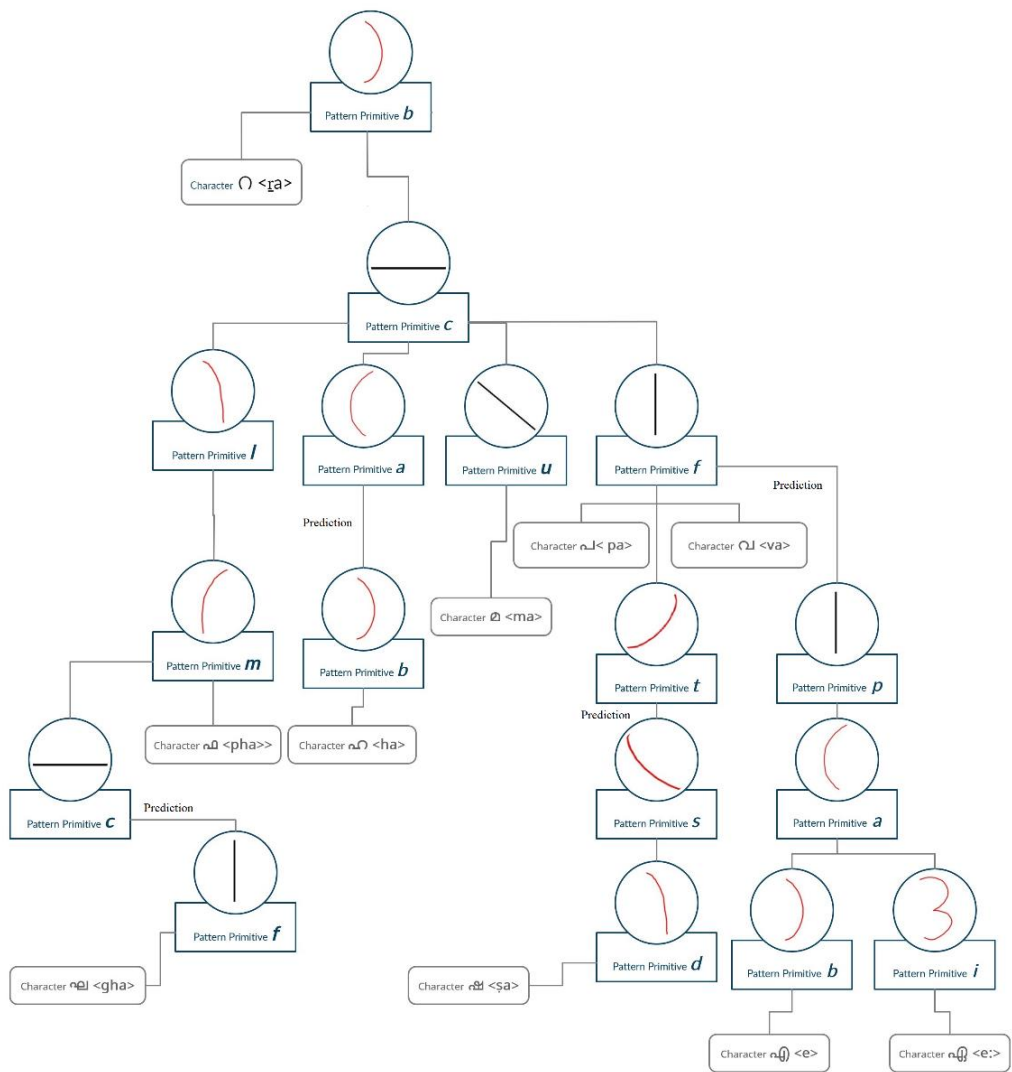


Fig. 6.13(a) Tree representation of the predictive model based on the first pattern primitive *a* as root node

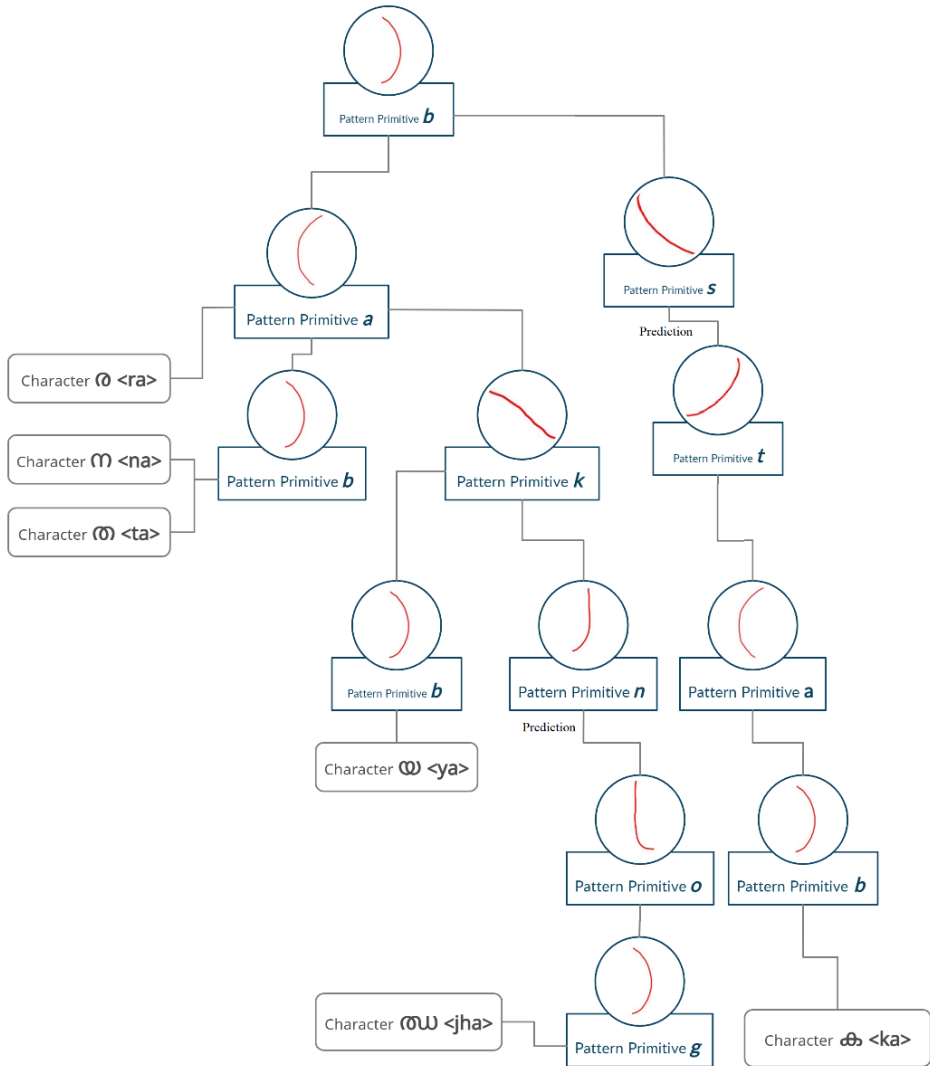


Fig. 6.13(b) Tree representation of the predictive model based on the first pattern primitive *a* as root node

The tree representation of the proposed character predictive models, incorporating the pattern primitive transitions, are effectively utilized to compute the average reduction in writing time for every character, as described in the next section.

6.6. Reduction in Writing Time (RWT) for the Malayalam Online Handwritten Characters

From the tree representations of the predictive model described in the previous section, it is evident that, most of the Malayalam characters are predictable even before completely writing them. The detailed analysis of the tree representations reveals that there exist a considerable reduction in the number of character classes to be taken into consideration for the next level of prediction. Based on the same, the reduction in writing time for each character is computed. The following section describes the process of estimating the reduction in writing time for the characters.

6.6.1. Reduction in the Number of Classes in Character Prediction

In a real-time environment, after each pattern primitive is written, the number of character classes to be involved in the next stage of prediction reduces considerably. The reduction in character classes to be considered for character prediction thus obtained is listed in table 6.7. The columns P1, P2, P3, P4, P5, P6, P7 and P8 in the table denote the pattern primitives. The cross (X) mark in the table denotes the pattern primitives that need not be considered for predicting the character. The number '1' indicates the position of the pattern primitive where the character prediction is taken place.

Table 6.7 Reduction in the number of classes in character prediction

S.No	Character	Reduction in character classes							
		P1	P2	P3	P4	P5	P6	P7	P8
1	അ <a>	24	3	2	2	2	2	2	
2	ആ <a:>	24	3	2	2	2	2	2	1
3	ഇ <i>	10	10	7	6	2	1		
4	ഉ <u>	10	10	1	X				
5	ഋ <ṛ>	2	2	1	X				
6	എ <e>	24	16	9	5	2	2	1	
7	ഏ <e:>	24	16	9	5	2	2	1	
8	ഒ <o>	10	10	1					
9	ക <ka>	24	16	1	X	X	X		
10	ഖ <kha>	10	10	7	1	X			
11	ഗ <ga>	4	2	1					
12	ഘ <gha>	24	16	9	2	2	1	X	
13	ങ <kha>	10	10	7	6	1			
14	ച <ca>	24	2	2	1				
15	ഛ <cha>	24	2	2	1	X	X		
16	ജ <ja>	10	10	7	6	2	1	X	X
17	ഝ <jha>	24	16	5	2	1	X	X	
18	ഞ <ña>	10	10	7	6	2	1	X	
19	ട <ṭa>	2	2	1					
20	ഠ <ṭha>	4	1	X					
21	ഡ <ḍa>	24	2	2	2	2			

Table 6.7 Reduction in the number of classes in character prediction (cont.)

S.No	Character	Reduction in character classes							
		P1	P2	P3	P4	P5	P6	P7	P8
22	ഘ <dha>	24	2	2	2	2	1		
23	ണ <na>	10	10	7	6	5	1	X	
24	ത <ta>	24	16	5	2				
25	ഥ <tha>	1	X	X	X				
26	ദ <da>	24	3						
27	ധ <dha>	4	1	X	X				
28	ന <na>	24	16	5	2				
29	പ <pa>	24	16	9	5				
30	ഫ <pha>	24	16	9	2	2			
31	ബ <ba>	10	10	7	6	2	1	X	
32	ഭ <bha>	24	1						
33	മ <ma>	24	16	9	1				
34	യ <ya>	24	16	5	2	1			
35	ര <ra>	24	16	5					
36	ല <la>	1	X	X	X	X			
37	വ <va>	24	16	9	5				
38	ശ <śa>	4	2	2	1				
39	ഷ <śa>	24	16	9	5	1	X	X	
40	സ <sa>	4	1	X	X	X			
41	ഹ <ha>	24	16	9	1	X			
42	ള <ḷa>	10	10	1	X				

Table 6.7 Reduction in the number of classes in character prediction (cont.)

S.No	Character	Reduction in character classes							
		P1	P2	P3	P4	P5	P6	P7	P8
43	ϕ <la>	1	X	X					
44	ο <ra>	24	16						

The information regarding the reduction in the number of classes to be considered for character prediction, shown in table 6.7, is effectively utilized for determining the reduction in writing time for every character, as described in the following section.

6.6.2. Reduction in Writing Time (RWT)

The reduction in writing time for the characters is considered as a potential information that can be utilized to build an online handwriting recognition system in a real-time environment. The Reduction in Writing Time (RWT) is computed based on the information related to the reduction in number of classes to be considered for predictions as shown in table 6.7. Table 6.8 shows the reduction in writing time obtained for various characters, where t is the average time required for writing the character. From the table, it can be observed that the maximum reduction in writing time (RWT) is obtained as $4t/5$ for the character ρ <la>.

Table 6.8 Reduction in Writing Time (RWT) for the Malayalam characters

S.No	Character	Number of pattern primitives (A)	Position of the pattern primitive from which prediction is possible (B)	Reduction in Writing Time (RWT) (A-B)*t/A
1	അ <a>	7	7	0
2	ആ <a:>	8	8	0
3	ഇ <i>	6	6	0
4	ഉ <u>	4	3	t/4
5	ഋ <ṛ>	4	3	t/4
6	എ <e>	7	7	0
7	ഏ <e:>	7	7	0
8	ഒ <o>	3	3	0
9	ക <ka>	6	3	t/2
10	ഖ <kha>	5	4	t/5
11	ഗ <ga>	3	3	0
12	ഘ <gha>	7	6	t/7
13	ങ <kha>	5	5	0
14	ച <ca>	4	4	0
15	ഛ <cha>	6	4	t/3
16	ജ <ja>	8	6	t/4
17	ഝ <jha>	7	6	t/7
18	ഞ <ña>	7	6	t/7
19	ട <ta>	3	3	0
20	ത <tha>	3	2	t/3
21	ദ <da>	5	5	0
22	ഢ <dha>	6	6	0
23	ണ <na>	7	6	t/7
24	ത <ta>	4	4	0

Table 6.8 Reduction in Writing Time (RWT) for the Malayalam characters (cont.)

S.No	Character	Number of pattern primitives (A)	Position of the pattern primitive from which prediction is possible (B)	Reduction in Writing Time (RWT) $(A-B)*t/A$
25	ത <tha>	4	1	$3t/4$
26	ദ <da>	2	2	0
27	ധ <dha>	4	2	$t/2$
28	ന <na>	4	4	0
29	പ <pa>	4	4	0
30	ഫ <pha>	5	5	0
31	ബ <ba>	7	6	$t/7$
32	ഭ <bha>	2	2	0
33	മ <ma>	4	4	0
34	യ <ya>	5	5	0
35	ര <ra>	3	3	0
36	ല <la>	5	1	$4t/5$
37	വ <va>	4	4	0
38	ശ <śa>	4	4	0
39	ഷ <śa>	7	5	$2t/7$
40	സ <sa>	5	2	$3t/5$
41	ഹ <ha>	5	4	$t/5$
42	ള <ḷa>	4	3	$t/4$
43	ഴ <ḷa>	3	1	$2t/3$
44	റ <ra>	2	2	0

The following section describes the implementation of the proposed character prediction model, which can be effectively used for real-time OHCR applications.

6.6.3. Implementation of the Character Prediction Model for Real Time OHCR

Experiments are conducted for the recognition of Malayalam online handwritten characters, based on the proposed character prediction model. For the experimental purpose, samples of each vowel characters, taken from the CU-OHDB dataset are used. In a real-time prediction model, the system lists out the possible characters after every pattern primitive, is written. In the case of Malayalam vowel characters, it is observed that, the character prediction will be taken place even after the first pattern primitive is written. There are thirteen distinct pattern primitives present in the vowel characters. The features of these pattern primitives constituting the Malayalam vowel characters considered for character prediction purposes are listed in table 6.9. A reference feature set for each of these pattern primitive (present in eight vowel characters under study) is created by extracting the features specified in table 6.9

Table 6.9 Features of the nine pattern primitives constituting the Malayalam vowel characters



Sl.No	Pattern primitive	Label	Direction	Features
1		a	Upwards	Reduced direction code, writing direction, arc / straight-line
2		b	Downwards	Reduced direction code, writing direction, arc / straight-line

Table 6.9 Features of the nine pattern primitives constituting the Malayalam vowel characters (cont.)












Sl.No	Pattern primitive	Label	Direction	Features
3		c	Left to right	Reduced direction code, writing direction, arc / straight-line
4		d	Downwards	Reduced direction code, writing direction, arc / straight-line
5		e	Upwards	Reduced direction code, writing direction, arc / straight-line
6		f	Upwards	Reduced direction code, writing direction, arc / straight-line
7		g	Upwards	Reduced direction code, writing direction, arc / straight-line
8		h	Downwards	Reduced direction code, writing direction, arc / straight-line
9		i	Downwards	Reduced direction code, writing direction, arc / straight-line, cusp

Table 6.9 Features of the nine pattern primitives constituting the Malayalam vowel characters (cont.)

Sl.No	Pattern primitive	Label	Direction	Features
10		j	Downwards	Reduced direction code, writing direction, arc / straight-line
11		p	Downwards	Reduced direction code, writing direction, arc / straight-line
12		q	Downwards	Reduced direction code, writing direction, arc / straight-line
13		x	Upwards	Reduced direction code, writing direction, arc / straight-line

The detailed procedure proposed for implementing the prediction model is described in procedure 6.3. The procedure is implemented in MATLAB for the conduct of prediction experiments. The prediction experiments are conducted using hundred samples, among each of the eight vowel characters. Each sample of the character is given as input to the algorithm. As a first step, these characters are segmented using the combined approach of RDP and EDFC algorithms, as described in chapter 4. In the next step, every segment of the character is compared with the corresponding reference pattern primitive feature set. If the segment is recognized as a pattern primitive, and the number of

predicted characters is equal to 1, then the character is considered as predicted and the next character sample can be considered as the input to the algorithm. If the segment is recognized as a pattern primitive, and the number of predicted characters is not equal to 1, then the next segment of the character is considered for recognition. The procedure is repeated until the number of predicted characters become 1. If a segment is not recognized as a pattern primitive in any of the stages, the character is considered as invalid. The procedure is repeated using every character sample to obtain the recognition accuracy. The recognition accuracies thus obtained for each vowel character are tabulated in table 6.10. From the table, it can be seen that, the average recognition accuracy obtained for the characters is 77.63 %. The time complexity of the algorithm in average case is estimated as $O(n)$. Procedure 6.3 can be extended to recognize all 44 characters by making appropriate changes based on the pattern primitive transitions present in them.

Procedure 6.3 Implementation of the prediction model for real-time HCR

Input: PointList ; The set of points describing a character

Output: Ch ; Predicted character

Procedure Predict()

Segments(i)=**Combined_Approach** (PointList) // *Getcharacter segments*
// using procedure 4.4

Ch={1,2,3,4,5,6,7,8} // Labels of eight vowel characters

// 1 for അ <a>, 2 for ആ <a:>, 3 for ഇ <i>, 4 for ഉ <u>, 5 for റ <r>, 6 for ഐ <e>,
// 7 for ഊ <e:>, 8 for ഓ <o>

P={a,b,c,d,g,i,j,q,x} // Reference pattern primitives

i=1

While number of characters in Ch>1

Read(Segments(i))

```

// First pattern primitive
If (i=1) then
    If (Segments(i) = a) then Ch= {1,2,6,7}
    Elseif (Segments(i) = b) then Ch= {3,4,8}
    Elseif (Segments(i) = g) then Ch= {5}
    Else Ch= {0}
// Second pattern primitive
If (i=2) then
    If (Segments(i) = i) then Ch= {1,2}
    Elseif (Segments(i) = b) then Ch= {6,7}
    Elseif (Segments(i) = a) then Ch= {3,4,8}
    Else Ch= {0}
// Third pattern primitive
If (i=3) then
    If (Segments(i) = a) then Ch= {1,2}
    Elseif (Segments(i) = c) then Ch= {6,7}
    If (Segments(i) = d) then Ch= {3}
    Elseif (Segments(i) = j) then Ch= {4}
    Elseif (Segments(i) = i) then Ch= {8}
    Elseif (Segments(i) = x) then Ch= {5}
    Else Ch= {0}
// Fourth pattern primitive
If (i=4) then
    If (Segments(i) = d) then Ch= {1,2}
    Elseif (Segments(i) = f) then Ch= {6,7}
    Else Ch= {0}
// Fifth pattern primitive
If (i=5) then
    If (Segments(i) = e) then Ch= {1,2}
    Elseif (Segments(i) = p) then Ch= {6,7}
    Else Ch= {0}
// Sixth pattern primitive

```

```
If (i=6) then
    If (Segments(i) = b) then Ch= {1,2}
    Elseif (Segments(i) = a) then Ch= {6,7}
    Else Ch= {0}
// Seventh pattern primitive
If (i=7) then
    If (Segments(i) = a then
        If (Segments(i+1) =Null) then
            Ch= {1}
        Else
            Ch={1,2}
        Elseif (Segments(i) = b) then Ch= {6}
        Elseif (Segments(i) = i) then Ch= {7}
        Else Ch= {0}
// Eighth pattern primitive
If (i=8) then
    If (Segments(i) = j) then Ch= {2}
    Else Ch= {0}
    i=i+1
```

End of While

End

Table 6.10 Prediction accuracies obtained for the Malayalam vowel characters

Character	Prediction Accuracy (%)
അ <a>	77
ആ <a:>	72
ഇ <i>	79
ഉ <u>	82
ഊ <ɻ>	84
എ <e>	73
ഐ <e:>	73
ഓ <o>	81
Average Prediction Accuracy 77.63 %	

The prediction accuracies show that the method is suitable for the real-time recognition of Malayalam online handwritten characters. In this approach, the character is predicted as soon as certain pattern primitives are completed. Hence, a considerable amount of time required for writing the character can be saved. This will improve the overall performance of the system. Hence, the method is adaptable in a real-time OHCR environment.

6.7. Conclusion

A predictive model for the real-time recognition of Malayalam handwritten characters, based on pattern primitives, is proposed as part of the study. In the first method, DFA based OHCR using pattern primitive is proposed. Here, the entire pattern primitives constituting each of the characters are considered for the recognition purpose. In this

approach, the character is recognized only after finishing the writing of the entire character. The recognition experiments are conducted for the eight vowel characters and achieved an accuracy of 65.75%.

An alternate predictive model for the real-time implementation of the Malayalam OHCR systems is also proposed as part of the study. It can be observed that, most of the Malayalam characters can be predicted even after completing the first pattern primitive. Considering this, the pattern primitive transition sequences present in the Malayalam online handwritten characters are analyzed and found that, the order of occurrence of pattern primitives is highly useful in designing a predictive model for real-time OHCR. It is also observed that, only seven pattern primitives have the possibility of occurrence at the starting position of the Malayalam characters. The study shows that most of the Malayalam characters start writing in the upward direction. It can also be seen that fifty percentage of the characters start with the pattern primitive 'a'. The most frequent pattern primitive transition is also identified as 'a → b'. The information about writing direction, starting pattern primitive and frequent pattern primitive transition are highly useful in designing prediction based HCR systems. In the chapter, the tree representations of character predictions based on its pattern primitive transitions, by fixing the first pattern primitives as the root node, are described. These tree representations as character predictive models are effectively utilized to compute the average reduction in writing time for every character. The Reduction in Writing Time (RWT), obtained for the characters shows that, certain characters need not be written completely to be predicted. The advantage in RWT is

highly beneficial in designing real-time HCR systems. Experiments are conducted for the recognition of eight Malayalam online handwritten vowel characters based on the proposed character prediction model and obtained an average recognition accuracy of 77.63%. The results show that the proposed model is suitable for real-time Malayalam HCR systems.

CHAPTER 7

Keyword Spotting in Online Handwritten Malayalam Documents

7.1. Introduction

Over the last few decades, advances in data capturing technologies and its extensive distribution in digital devices have created vast amount of handwritten data that need to be stored and retrieved in an efficient way. To cope with these drastic changes, notebook PCs, PDAs and, other mobile computing devices are now migrating from keyboard data input methods to simple data input methods like touch screen interfaces. This allows the data to be stored as digital handwritten documents. In this context, methods for archiving and recovering handwritten documents stored in the digital form need special attention.

The technology of earlier decades kept most historical records in document image format. The studies were also aligned to handle handwritten document image data. When stored in document image format, a text data document consumes more storage space. The recent technologies reduce storage space and speed up data retrieval by allowing text data to be stored in the form of online handwritten documents. In online handwritten documents, the data is recorded as a sequence of points. These points are the carriers of original data, making contact over the surface, either a paper or an electronic surface. These sequences of points considerably reduce the storage requirement for handwritten data. Hence, most handwritten data are now stored in the form of online handwritten documents rather than handwritten

document images. But, retrieval of these online handwritten documents is difficult from a massive collection of such documents. To simplify the task of retrieving handwritten documents, a technique called Keyword Spotting (KWS) is extensively used to extract documents that contain specific words from vast amounts of handwritten documents.

Most of the earlier studies on KWS addressed the problem of retrieval from handwritten document images than online handwritten documents. Manmatha *et al.* describes the importance of word spotting and word indexing methods in document retrieval from historical documents stored as images [59]. Stroke directions, Hough transform, Gabor filters, and shape features are extensively reported in various document image based word spotting techniques [62][118][119][120]. Keyword Spotting (KWS) techniques specific to online handwritten documents have been studied for various scripts to a limited extent [74][79][121][80]. In Indic scripts, Keyword Spotting techniques for retrieving online handwritten documents are rare, especially for the Malayalam script. In this context, a study initiated for Keyword Spotting (KWS) on Malayalam online handwritten documents is found to be highly useful in real-life applications.

In Keyword Spotting, a document is retrieved based on the input keyword present in the document. A Keyword Spotting (KWS) system is a combination of keyword indexing and retrieval techniques. In indexing, the instances of a keyword are generated from the document when it is stored. While retrieving, the keywords are matched against indexed keywords. The block diagram of a KWS system is shown in figure 7.1. It can be seen, that the input documents need to be

preprocessed. The preprocessed documents are segmented to obtain the list of reference words. The features are extracted from these reference words to obtain similar instances for indexing. These indexed words are stored and used for matching with the input keyword for document retrieval.

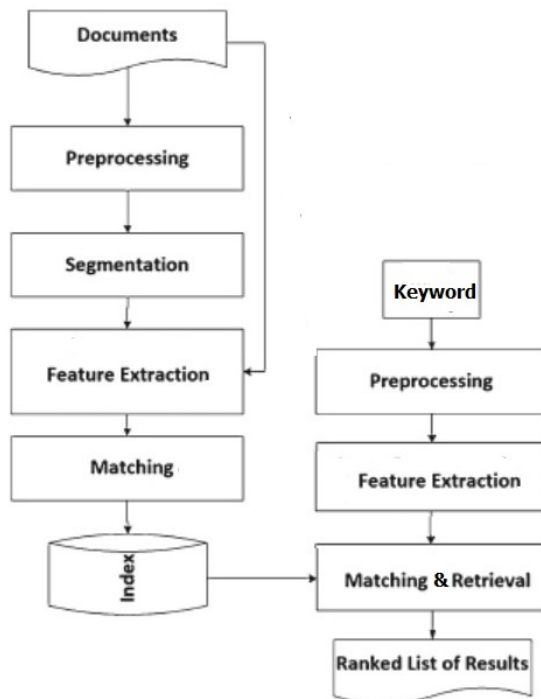


Fig 7.1 Block diagram of the KWS system

In the proposed work, a keyword dataset is prepared from online handwritten documents through word segmentation, and it is used for experimental purposes. The samples in the keyword dataset are preprocessed to remove skews, jitters, and writing attenuations and they are labeled manually. There are twenty such reference keywords and fifteen instances of each of them. In the keyword spotting experiments, the features extracted from the input keyword is matched with the

features of the reference keyword dataset using Dynamic Time Warping (DTW) algorithm. The rest of the chapter is organized as follows. Section 7.2 describes the creation of keyword dataset and preprocessing methods. Various feature extraction techniques applied to the keyword samples are described in section 7.3. Section 7.4 describes the Keyword Spotting experiments conducted, followed by the result analysis in section 7.5. Section 7.6 of the chapter concludes the work.

7.2. Creation of Keyword Dataset and Preprocessing

This section describes the creation of a reference keyword dataset generated from online handwritten documents and various preprocessing methods applied to the keyword dataset.

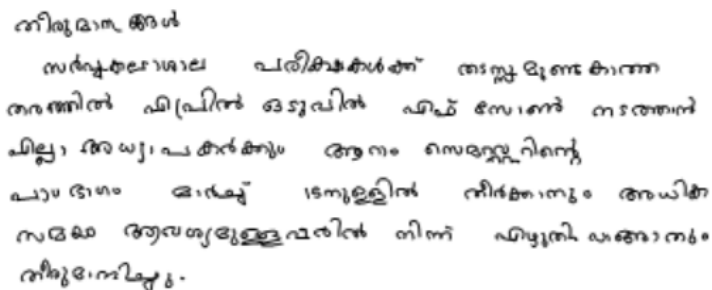
7.2.1. Malayalam Online Handwritten Keyword Dataset Creation

The role of data samples used to conduct OHCR experiments is very crucial. They must address all types of variations as expected in a real-time environment. It is evident from the literature, that the reliability of a system based on natural handwriting, improves when it handles various writing styles and writing speeds [2]. Hence, considerable attention has been paid while creating a dataset, that accommodates handwritten samples with variations in writing styles and speeds.

In the proposed work, online handwritten Minutes of Meetings of Academic Council pertaining to a Government College in Kerala are considered to acquire handwritten samples. The Academic Council is the superior body of a college that makes academic and administrative decisions. Usually, the Minutes of Meetings are kept in their original handwritten form and the machine-readable text of the same has to be

circulated among the council members in printed form. In most of the cases, it may require to search for referring to certain major decisions taken or information regarding the members present in the previous meetings.

For experimental purposes, the online handwritten version of the original Minutes of the Meetings of the Academic Council is created using E-writemate device as described in chapter 3. The handwritten keyword dataset is created by extracting selected keywords through segmentation from twenty such handwritten pages of Minutes of the Meetings written by five different writers. A sample Minutes of the Meeting written using E-writemate device is shown in figure 7.2.



നിയമസഭയിൽ
സംസ്കാരപരിഷ്കരണ കമ്മിറ്റി
അതിന്റെ ഹിസ്റ്ററിയിൽ ഹിസ്റ്ററി കമ്മിറ്റി
മിഷൻ കമ്മിറ്റി പരിഷ്കരണ കമ്മിറ്റി
പരിഷ്കരണ കമ്മിറ്റി പരിഷ്കരണ കമ്മിറ്റി
പരിഷ്കരണ കമ്മിറ്റി പരിഷ്കരണ കമ്മിറ്റി
പരിഷ്കരണ കമ്മിറ്റി പരിഷ്കരണ കമ്മിറ്റി
പരിഷ്കരണ കമ്മിറ്റി പരിഷ്കരണ കമ്മിറ്റി

Fig. 7.2 A sample Malayalam online handwritten document written using E-writemate device

Segmentation of words from online handwritten documents is more effortless than handwritten document images due to the availability of temporal information. In the proposed work, word segmentation is carried out by measuring the difference between neighboring x -values in the stroke series. A threshold value is determined based on the (x, y)

values in the stroke series, corresponding to the online handwritten documents. When the difference between neighboring x -values exceeds the threshold, the word is segmented. The word segmentation method employed in the current work is described in procedure 7.1.

Procedure 7.1: Implementation of the algorithm for word segmentation from online handwritten documents

Input: Text, The online handwritten document text file containing x and y values

Output: Words, The array of words

Procedure Word_Seperate(Text)

$j=1$

 TextX= x -value of Text

 From start to end of Text

 If (difference of (TextX (i)- TextX (i+1))> threshold)

 Words(j)=store (x,y) points up to the point which exceeds threshold

$j=j+1$

 End

Words // Contains various words in the document

End

After segmenting the words from the Minutes of the Meeting, the number of instances of every word is labeled to create the reference word dataset, which includes fifteen instances of twenty Malayalam words. Sample words taken from the word dataset are shown in figure 7.3.



Fig. 7.3 Samples from the word dataset

Table 7.1 shows the details of the own created Malayalam online handwritten word dataset. The dataset includes 1500 samples of handwritten words taken from the handwritten Minutes of the Meeting.

Table 7.1 Details of the Malayalam online handwritten keyword dataset

Number of words	Number of samples	Number of writers	Total number of words
20	15	5	1500

7.2.2. Preprocessing Methods Applied

Preprocessing is an essential phase in systems handling natural handwritings, which vary in speed, skew, and size [122]. It is evident that, most of the samples are affected by writing attenuations like skew, jitters, and size variations. Hence, preprocessing is applied to remove such noises in the samples to conduct further experiments. The preprocessing stages applied to the word samples consist of smoothing, resampling, skew correction, and normalization. Figure 7.4 shows the various stages of preprocessing methods applied to the online handwritten word samples.

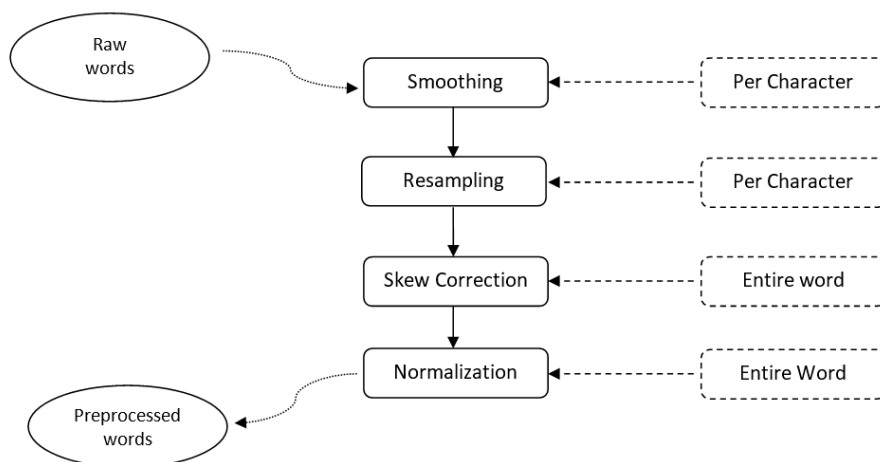


Fig. 7.4 Stages of preprocessing

(a) Smoothing

The smoothing operation is carried out in every character in the word using a moving average filter. The detailed procedure for smoothing the characters is already described in chapter 3, section 3.4.2.

(b) Resampling

The speed of handwriting and hardware limitations result in missing and crowded points in online handwritten words. To eliminate the disparity in the distribution of strokes, resampling is used. Here, every stroke in the word is resampled to 80 points to adjust missing points and make all points equidistant. A parametric spline approximation, which is effectively used in various OHCR studies, is used to interpolate missing points [97][123][124].

(c) Skew Correction

The skew of writing induces errors in the features extracted from the word. Most of the methods in previous studies employed Hough transforms, Radon transforms, and centre of gravity for skew detection and correction[125]. In the proposed experiments, the centre of gravity is used for skew correction. The method of using centre of gravity is based on the principle of rotation correction by gravity center balancing [126]. The word is divided into two parts by measuring the centroid (gravity centre). The skew angle is estimated as the inclination of the line connecting the gravity centres. Procedure 7.2 describes the centre of gravity method used for skew correction of online handwritten words in the proposed work.

Procedure 7.2: Implementation of the algorithm for skew correction of online handwritten words using the centre of gravity method

Input: RawWord, The raw word data

Output:SkewCorrectedWord, The skew corrected word

Procedure COG(RawWord)

$$X=x_1+x_2+x_3+\dots+x_n$$

$$Y=y_1+y_2+y_3+\dots+y_n$$

$$X_c=X / n$$

$$Y_c=Y / n // X_c \text{ and } Y_c \text{ are centroids of the word}$$

$$\text{Word1}=(x_1,y_1),(x_2,y_2),\dots,(X_c,Y_c)$$

$$\text{Word2}=(X_{c+1},Y_{c+1}),(X_{c+2},Y_{c+2}),\dots,(X_n,Y_n)$$

$$W1X_c=(x_1+x_2+x_3+\dots+X_c) / \text{size}(\text{Word1})$$

$$W1Y_c=(y_1+y_2+y_3+\dots+Y_c) / \text{size}(\text{Word1})$$

// $W1X_c$ and $W1Y_c$ are centroids of word part 1

$$W2X_c=(X_{c+1}+X_{c+1}+X_{c+1}+\dots+X_n) / \text{size}(\text{Word2})$$

$$W2Y_c=(Y_{c+1}+Y_{c+1}+Y_{c+1}+\dots+Y_n) / \text{size}(\text{Word2})$$

// $W2X_c$ and $W2Y_c$ are centroids of word part 2

Slope= (W2Y_c - W1Y_c)/ (W2X_c - W1X_c) // slope $m=(y_2-y_1)/(x_2-x_1)$

Theta=tan⁻¹(slope) // skew angle

$x_{new} = x*\cos(\text{Theta}) - y*\sin(\text{Theta})$; // skew correction

$y_{new} = y*\cos(\text{Theta}) + x*\sin(\text{Theta})$;

SkewCorrectedWord=(x_{new} , y_{new})// skew corrected word

End

(d) Normalization

A word level normalization is performed over the samples to standardize the size of every word. The normalization performed here is min-max normalization, as described in chapter 3, section 3.4.1.

A sample Malayalam online handwritten word അജണ്ട / ajaṅṅa / before and after various preprocessing phases is shown in figures 7.5(a), 7.5(b), respectively.



Fig. 7.5(a) Malayalam online handwritten word അജണ്ട /ajaṅṅa/ before preprocessing



Fig. 7.5(b) Malayalam online handwritten word അജണ്ട / ajaṅṅa / after preprocessing

The following section presents features that are identified for KWS on Malayalam online handwritten words and the methods to extract these features.

7.3. Feature Extraction

Identifying various features and techniques to extract these features is essential in KWS. In this study, structural features that are found to be highly relevant for differentiating the Malayalam online handwritten words are considered. Three features are considered in the work namely, direction, height and curvature of the stroke points in the word. Table 7.2 shows the list of features extracted from the words for the conduct of KWS experiments.

Table 7.2 List of features identified for the Malayalam online handwritten words for KWS experiments

Sl.No	Feature	Relevance
1	Direction of the stroke points in the word	Direction of stroke point differ in different words
2	The height of the stroke points in the word	The height of the stroke points from base line differs in various words
3	The curvature of the stroke points in the word	The angle subtended by lines joining the point and its neighbors vary

The following section describes the methods used for extracting the above features from the online handwritten Malayalam words present in the dataset.

(a) The direction of the Stroke Points in the Word

The direction of every stroke point in the handwritten word is a strong feature to differentiate one word from others. The counter clockwise angle between x-axis and the tangent to the stroke point is considered as

the stroke direction. The line joining neighboring points of the stroke point is considered as the tangent for extracting directions. A sample graphical representation of the direction of the stroke points is shown in figure 7.6 (a).

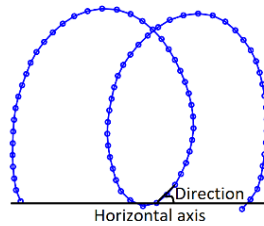


Fig. 7.6(a) Direction of a stroke point in the Malayalam character ta

(b) Height of the Stroke Points in the Word

The distance from the base of the word to the stroke points is considered as the height. This is obtained by estimating the distance of the stroke points from the base of the word. The base of the word is the horizontal line drawn through the lowest y-value in the stroke points. For every stroke point corresponding to the word, the height value is obtained. A graphical representation of the height of the stroke points in the word is shown in figure 7.6 (b).

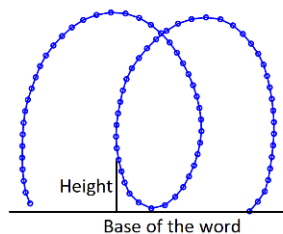


Fig. 7.6(b) Height of a stroke point in the Malayalam character ta

(c) Curvature of the Stroke Points in the Word

In the case of Malayalam online handwritten words, majority of the characters contain arc segments. Hence, the curvature of the stroke points can be considered as a feature to identify various words. The curvature of a stroke point means the angle subtended by the stroke point with neighboring points. To calculate the curvature value of a stroke point, lines are drawn from neighboring points to the stroke point. The angle between these lines is estimated to obtain the curvature value of the stroke point. Similarly, the curvature value of every stroke point is obtained. Figure 7.6 (c) shows the graphical representations of the curvature feature of stroke points in the words.

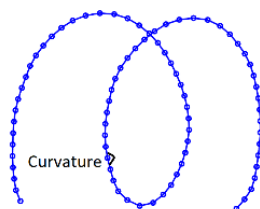


Fig. 7.6(c) Curvature of a stroke point in the Malayalam character ത <ta>

The algorithm described in procedure 7.3 is implemented in MATLAB to extract the features mentioned.

Procedure 7.3: Procedure to extract the direction, height and curvature features from the words

Input: Word, The stroke points in the word with (x, y) values

Output: Direction, height, and curvature features

Procedure DHC(Word)

// To find the direction of the stroke points //

Read from second to N-1th point of a strokes
 Get the neighbor points of the stroke point
 Estimate the line from left neighbor to the right neighbor
 Estimate the horizontal axis of the point
 Estimate the angle between the line and horizontal axis to obtain the
 direction
 Repeat the procedure up to N-1th point
 // To find the height of the stroke points //
 Estimate the base of the word
 Read from first to Nth point of a strokes
 Compute the vertical distance from base of the word to the stroke
 point to obtain height
 Repeat the procedure up to Nth point
 // To find the curvature of the stroke points //
 Read from second to N-1th point of a strokes
 Get the neighbor points of the stroke point
 Estimate the line from left neighbor to stroke point
 Estimate the line from right neighbor to stroke point
 Find the angle between these two lines to obtain the curvature
 Repeat the procedure up to N-1th point

End

7.4. Experimental Setup

This section details the experiments carried out to verify the effectiveness of the Keyword Spotting system. Dynamic Time Warping (DTW) is the proposed technique for matching the keywords. Dynamic Time Warping (DTW) is a powerful technique that has been applied to match temporal sequences [127][128]. In the proposed experiments, the samples from the word dataset discussed in section 7.2.1 are used. The experiments are setup to verify the effectiveness of a writer dependent

KWS system. Hence, separate experiments are conducted for five different writers. Ten instances of twenty selected words, written by a writer are considered as the reference word dataset for the experiments. These words are preprocessed using the techniques mentioned in section 7.2.2. Features are also extracted from these words using the procedure 7.3. After extracting the features, an optimal value for the DTW distance is computed for every reference word in the dataset.

Initially, an instance of a word is taken from the dataset and it is considered as the keyword. The keyword is preprocessed, and features are also extracted, as discussed in previous sections. Matching between the input keyword and the remaining instances of every word is performed using the DTW technique. Matches within the optimal value of the DTW distance for the keyword are identified from the experiments. The same procedure is repeated with an instance of the next word until all the words are accommodated. From these results, the precision and recall values are obtained for twenty keywords. Precision is referred to here as the percentage of correct matches out of the total matches in the comparison. Recall denotes the fraction of the number of correct matches to the total number of instances of the keywords. Similarly, the entire experiment is repeated for all the five writers using the word samples written by them. After conducting experiments for all the five writers, the average precision and recall values are calculated. From this precision and recall values, the average F1 score is also computed.

7.5. Result Analysis

The precision and recall rates obtained in the KWS experiments for the five writers are shown in figure 7.7, where the x-axis denotes the writers, and the y-axis denotes precision and recall values obtained for every writer. The average F1 score obtained for the proposed technique is 78.31%, for an average precision of 90% and an average recall rate of 69.31%.

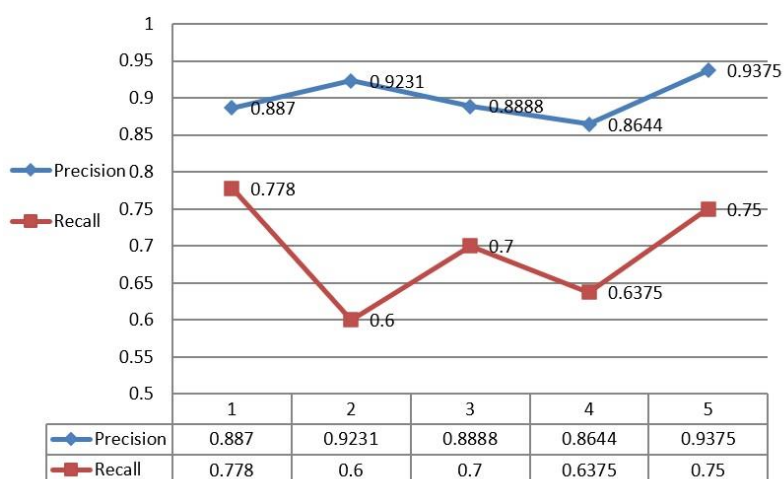


Fig. 7.7 Precision and Recall values obtained in the Keyword Spotting experiments

7.6. Conclusion

This chapter discussed the implementation of a Keyword Spotting (KWS) system for online handwritten Malayalam documents. As a part of the work, a Malayalam online handwritten keyword dataset is created from online handwritten documents. Various preprocessing methods applied to the online handwritten keywords are familiarized in the study. The techniques to extract the features, that apply to the selected online

handwritten Malayalam keywords, are identified in the work. These features are extracted using the methods described. Experiments are also conducted using these extracted features to verify the effectiveness of the system. An average performance score of 78.31% is obtained for twenty keywords in the KWS experiments. The results show that, the method can be effectively used in Malayalam Keyword Spotting systems with massive online document databases.

Conclusions and Recommendations

8.1. Conclusions

Technological growth in human-machine interaction deployed many research contributions in online handwritten character recognition (OHCR) for various scripts. In the Indian context, the research in OHCR for various Indic scripts is tremendous for the last few years. Various methods, which vary in features and classifiers, are reported in most of the studies to attain higher recognition rates. In this study, a diversified approach which is a part of Syntactic Pattern Recognition (SPR), is proposed for OHCR of Malayalam characters. The thesis mainly proposed an approach by segmenting Malayalam online handwritten characters into pattern primitives and implemented recognition schemes suitable for developing real-time applications. As a part of this study, a detailed analysis of pattern primitives in Malayalam online handwritten characters have been conducted. The directional and structural features of the characters and their pattern primitives are addressed in the research with prime focus.

The characteristics of the vowels and consonants in the Malayalam language are examined as part of the study. The various rules that pertain to the formation of conjuncts are also discussed within the orthographical perspective. The necessity of creating a benchmark dataset for Malayalam online handwritten character recognition is addressed in the study. A database of Malayalam online handwritten characters is created and used for experimental purposes. Two methods are employed in data acquisition. In the first method, handwritten

strokes are acquired while writing on a normal paper using a digital pen. In the second method, the touch screen capability of current technology is utilized in acquiring data through a laptop computer. The own created dataset named CU-OHDB contains 30800 samples of 44 Malayalam online handwritten characters written by twenty different writers properly categorized and labeled. Various preprocessing techniques, including smoothing, normalization, resampling is also proposed and implemented as part of the study.

The directional and structural properties of the Malayalam online handwritten characters are identified in the study. By analyzing these properties, the subunits in the characters called pattern primitives are also identified. The study shows that, eight vowel characters and thirty six consonant characters in Malayalam script can be represented by using these twenty six pattern primitives. A character reference set for the online handwritten Malayalam characters with marked points, corresponding to the pattern primitives, is also formed as part of the study. All the 44 characters are visually arranged and represented using the labels of the pattern primitives.

Segmentation of Malayalam online handwritten characters is performed using three algorithms: Ramer Douglas Peucker (RDP), Eight Direction Freeman Code (EDFC), and a combined approach of both algorithms. The segmentation accuracy of the algorithms is also computed by comparing the pattern primitive segments obtained with the reference character set. The performance of the Ramer Douglas Peucker(RDP) algorithm, applied for reducing a curve into a finite number of points is analyzed; and this advantage is utilized to segment

the characters into pattern primitives. Even with the variability in the writing style of the samples, the Ramer Douglas Peucker(RDP) algorithm based segmentation scheme performed well with an average segmentation rate of 78.36%. The segmentation scheme, based on the Eight Direction Freeman Code (EDFC), also had a better average segmentation accuracy of 74.55%. The combined approach based on the Ramer Douglas Peucker algorithm and Eight Direction Freeman Code attained an average segmentation accuracy of 91.12%. The accuracy obtained for the combined approach to segmentation shows that, the method can be effectively used to segment Malayalam online handwritten characters into pattern primitives. An automated method to estimate segmentation accuracy is also implemented. The automated method obtained a segmentation accuracy of 81.59% using the combined approach of RDP and EDFC.

A scheme to recognize Malayalam online handwritten characters based on pattern primitives is discussed in the thesis. Various preprocessing methods for categorizing arc and straight-line pattern primitives, smoothing, and slope correction of linear pattern primitives are implemented in the study. Initially, various clusters are formed based on the number of pattern primitives present in each character. A detailed description of various pattern primitive features and the algorithms used to extract these features are presented. The specific features are then extracted from the pattern primitives constituting the characters representing each cluster.

The classification experiments are performed based on the pattern primitive features using the Support Vector Machine (SVM) classifier.

The recognition accuracies obtained for each character cluster are tabulated. The average recognition accuracy obtained for all the clusters is 96.61%. The performance of the proposed method is found to be promising and hence it can be effectively used for the development of OHCR systems in Malayalam.

A model for the real-time recognition of Malayalam handwritten characters, based on pattern primitives, is proposed as part of the study. In the first method, the pattern primitive based OHCR system using DFA is proposed. Here, the entire pattern primitives constituting each of the characters are considered for the recognition purpose. In this approach, the character is recognized only after finishing the writing of the entire character. The recognition experiments are conducted for the eight vowel characters and achieved an average accuracy of 65.75%.

A predictive approach for the real-time recognition of Malayalam handwritten characters is also proposed as part of the model. It can be observed that, most of the Malayalam characters can be predicted even after completing the first pattern primitive. Considering this, the pattern primitive transition sequences present in the Malayalam online handwritten characters are analyzed, and found that, the order of occurrence of pattern primitives is highly useful in designing a predictive model for real-time HCR. It is also observed that, only seven pattern primitives have the possibility of occurrence at the starting position of the Malayalam characters. The study shows that most of the Malayalam characters start writing in the upward direction. It can also be seen that fifty percentage of the characters start with the pattern primitive 'a'. The most frequent pattern primitive transition is also

identified as 'a \rightarrow b'. The information about writing direction, starting pattern primitive and frequent pattern primitive transition are highly useful in designing prediction based HCR systems.

In the next stage of the prediction model, the tree representations of characters based on its pattern primitive transitions by fixing the first pattern primitives as the root node are implemented. These tree representations, which would be used as character predictive models, are effectively utilized to compute the average reduction in writing time for every character. The Reduction in Writing Time (RWT), obtained for the characters shows that, certain characters need not be written completely to be predicted. The advantage in RWT is highly beneficial in designing real-time HCR systems.

Experiments are conducted for the recognition of eight Malayalam online handwritten vowel characters based on the proposed character prediction model and obtained an average recognition accuracy of 77.63%. The results show that, the proposed model is suitable for real-time Malayalam HCR systems.

Keyword Spotting (KWS) in online handwritten documents has been attempted as an application of handwriting recognition in the thesis. Various preprocessing methods applied to the online handwritten keywords are familiarized in the study. The techniques to extract the features that apply to the selected online handwritten Malayalam keywords are identified in the study; and using these techniques, the features are extracted. Experiments are also conducted using these extracted features to verify the effectiveness of the system. An average performance score of 78.31% is obtained in the KWS experiments. The

results show that, the proposed system could be considered for Keyword Spotting (KWS) in online handwritten documents in Malayalam.

8.2. Contributions of the Thesis

The major contributions that have been reported as part of the thesis in the field of OHCR are as follows. An online handwritten character database for Malayalam has been developed as a part of the study. The database named CU-OHDB (Calicut University Online Handwriting Data Base) consists of 30800 samples of the 44 Malayalam characters written by twenty different writers, that are properly arranged and labeled. This dataset can be effectively used for the conduct of further studies in Malayalam OHCR.

The study on pattern primitives, derived from each character present in the thesis, is beneficial for implementing segmentation based OHCR in Malayalam. The representation scheme proposed for the 44 Malayalam online handwritten characters based on 26 pattern primitives is a unique contribution of the thesis.

Segmentation using the Ramer Douglas Peucker (RDP) algorithm and Eight Direction Freeman Code (EDFC) that ensure both structural and directional aspects of the characters, implemented for the recognition of Malayalam online handwritten characters in the thesis, is a notable contribution. The pattern primitive features of the characters, described in the study, can be effectively used for implementing real-time handwritten character recognition systems in Malayalam.

The information regarding transitions from one pattern primitive to the other, identified for all the Malayalam online handwritten characters,

is a remarkable outcome of the study. The frequency of occurrence of various pattern primitives and their position in a character are also identified and reported. Representation of the pattern primitive transition sequences in the form of trees and the computation of Reduction in Writing Time (RWT) based on the predictions are also unique to the thesis. The analysis done on the RWT for various characters would be useful, in understanding the effect of using pattern primitive based features, in a real-time handwriting recognition environment.

Implementation of the Keyword Spotting (KWS) system is another major contribution of this thesis, which effectively utilizes the features of online handwritten words in Malayalam. The method suggested is useful in developing Keyword Spotting (KWS) systems for retrieving specific documents from massive collection of online handwritten Malayalam documents.

8.3. Recommendations

The study made a novel attempt in online handwriting recognition for various Malayalam character units consisting of 44 characters which include vowels and consonants. There are also *conjuncts*, *chillu*, modifiers, and other symbols in Malayalam. The variability of those characters and symbols may not be reflected in the studies. As a future direction, the work can be extended to include the entire character set of Malayalam. As an initial work based on pattern primitives in Malayalam OHCR, certain selected directional and structural features are only considered. In future studies, additional features, including statistical features, need to be considered. As the recognition accuracy of the entire

system mentioned in the thesis depends entirely on the segmentation accuracy, various pattern primitive segmentation schemes may also be considered to get more précised results.

In the proposed real-time prediction model, the system makes predictions from the first pattern primitive for certain characters. This information may be used to convert the present character prediction model to a word prediction model, by incorporating a word dictionary. Also, the model may be fully implemented to test the effect of pattern primitive features for real-time recognition of handwritten characters in future studies. As a future recommendation, the proposed KWS method can be extended to include additional word level features and other word matching schemes.

The practical challenegs for incorporating the techniques in a real-time application include accomodating various writing styles, rotation / skew corrections, delayed and broken strokes etc. These are also be addressed in future studies for improving the performance of the system in a real-time environment.

Studies to achieve lesser time and space complexity may also be considered in the future with in-depth research on multi-dimensional aspects of Malayalam characters. The studies in the thesis are entirely focused on Malayalam characters and their pattern primitives, but capable of performing well in other scripts. It is also a future recommendation to identify pattern primitives in other Indian scripts and initiate studies to develop a language independent framework for real-time handwritten character recognizer in Indian scripts.

References

- [1] IBM, "Rochester Chronology," *IBM Rochester Lab* , 1966.
- [2] C. Y. Suen ,T. Wakahara C. C. Tappert, "The state of the art in online handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 8, pp. 787-808, 1990.
- [3] Eden M, "On the Formalization of Handwriting," *Structure of Language and its Mathematical Aspect*, American Mathematical Society, pp. 83-88, 1961.
- [4] Eden M, "Handwriting and Pattern Recognition," *IRE Trans. Information Theory*, pp. 160-166, 1963.
- [5] Sankar K Pal, Nikhil R Pal, "A review on image segmentation techniques," *Pattern Recognition*, pp. 1277-1294, 1993.
- [6] H Freeman, "On the Encoding of arbitrary geometric configurations," *IRE Transaction on Electronic Computers*, pp. 260-268, 1961.
- [7] Jeremy M Glass, H Freeman, "On the quantization of line drawing data," *IEEE Transactions on Information Systems and Cybernetics*, pp. 70-78, 1969.
- [8] URS Ramer, "An iterative procedure for the polygonal approximation of plane curves," *Computer Graphics and Image Processing*, pp. 244-256, 1972.
- [9] Thomas K Peucker, David H Douglas, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *The Canadian Cartographer*, pp. 112-122, 1973.
- [10] Herbert Freeman, "Computer processing of line-drawing images," *Computing Surveys*, pp. 58-97, 1974.

- [11] Larry S. Davis, Herbert Freeman, "A corner finding algorithm for chain coded curves," *IEEE Transaction on computers*, pp. 297-303, 1977.
- [12] Herbert Freeman, "Shape description via the use of critical points," *Pattern Recognition*, pp. 159-166, 1978.
- [13] Li-De Wu, "On the chain code of a line," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 347-353, 1982.
- [14] Mike Usher, Samir Al-Emami, "On-Line Recognition of Handwritten Arabic Characters," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 704-710, 1990.
- [15] Eric Lecolinet, Richard G. Casey, "A survey of methods and strategies in character segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 690-706, 1996.
- [16] Ciro D'elia, Alessandra Scotto Di Freca, Angelo Marcelli Claudio De Stefano, "Incorporating A Wavelet Transform Into A Saliency-Based Method For Online Handwriting Segmentation," *International Journal of Pattern Recognition and Artificial Intelligence*, pp. 43-59, 2007.
- [17] Sok Gek Lim, Michael K. Brown Jianying Hu, "Writer independent on-line handwriting recognition using an HMM approach," *Pattern Recognition* , pp. 133-147, 2000.
- [18] Ahmed M. Zeki, Akram M. Zeki, Mustafa Ali Abuzaraida, "Segmentation Techniques for Online Arabic Handwriting Recognition: A Survey," in *3rd International Conference on Information and Communication Technology for the Muslim World (ICT4M)*, pp. 37-40, 2010.
- [19] Chaabouni Aymen , Alimi Adel, El Abed Haikal Boubaker Houcine, "Fuzzy Segmentation and Graphemes Modeling for Online Arabic

- Handwriting Recognition," in *12th International Conference on Frontiers in Handwriting Recognition*, pp. 695-700, 2010,
- [20] Hozeifa Adam Abd, Alshafy Mohamed, Elhafiz Mustafa, "Characters' Boundaries based Segmentation for Online Arabic Handwriting," in *International Conference On Computing, Electrical And Electronic Engineering*, pp.306-310, 2013.
- [21] Salem Meftah Jebriel, Mustafa Ali Abuzaraida, "The Detection of the Suitable Reduction Value of Douglas-Peucker Algorithm in Online Handwritten Recognition Systems," in *IEEE International Conference on Service Operations And Logistics, And Informatics*, pp. 82-87, 2015.
- [22] Paul L. Rosin, "Assessing the behaviour of polygonal approximation algorithms," *Pattern Recognition*, pp. 505 – 518, 2003.
- [23] Maylor K.H. Leung, Chai Quek , Siu-Yeung Cho, Dilip K. Prasad, "A Novel Framework For Making Dominant Point Detection Methods Non-Parametric," *Image and Vision Computing*, pp. 843–859, 2012.
- [24] Sherif Abdel Azeem, Hesham M. Eraqi, "An on-line arabic handwriting recognition system: Based on a new on-line graphemes segmentation technique," in *International Conference on Document Analysis and Recognition*, pp. 409-413, 2011.
- [25] Fatos T. Yarman-Vural, A. Alper Atici, "A Heuristic Algorithm for Optical Character Recognition of Arabic Script," *Signal Processing*, pp. 87-99, 1997.
- [26] Sherif Abdel Azeem, Hesham M. Eraqi, "A New Efficient Graphemes Segmentation Technique for Offline Arabic Handwriting," in *International Conference on Frontiers in Handwriting Recognition*, pp. 95-100, 2012.

- [27] Umapada Pal, Nilanjana Bhattacharya, "Stroke Segmentation and Recognition from Bangla Online Handwritten Text," in *International Conference on Frontiers in Handwriting Recognition*, pp. 740-745, 2012.
- [28] Anoop M Namboodiri, Prabhu Teja S, "A Ballistic Stroke Representation of Online Handwriting for Recognition," in *12th International Conference on Document Analysis and Recognition*, pp. 857-861. 2013.
- [29] Anuj Sharma, Indu Chhabra, Sukhdeep Singh, "A dominant points-based feature extraction approach to recognize online handwritten strokes," *IJDAR*, pp. 37-58, 2017.
- [30] K S Fu, P H Swain, "On Syntactic Pattern Recognition," in *Proceedings of the Third Symposium on Computer*, Florida, pp. 164, 1969.
- [31] V V Rozensveig, Shelya A Guberman, "Algorithm for the recognition of Handwritten Text," *Automatica i Telemekhanika*, pp. 122-129, 1976
- [32] J.R. Bellegarda, D. Nahamoo K.S, E.J. Bellegarda, "A probabilistic framework for on-line handwriting recognition," in *IWFHR III*, 1993, pp. 225-234.
- [33] I. Lossev, A.V. Pashintsev, S.A. Guberman, "Method and apparatus for recognizing cursive writing from sequential input information," US Patent WO 94/07214, 1994.
- [34] Koichi Higuchi, Youichi Yamada, Yunosuke Haga, Yoshiyuki Yamashita, "Classification of hand printed Kanji characters by the structural segment matching method," *Pattern Recognition Letters*, vol. 1, pp. 475-479, 1983.

- [35] Lei Guo, Tianyun Zhao, Xiaoliang Qian Bo YU, "A Curve Matching Algorithm Based on Freeman Chain Code," in *IEEE*, pp. 669-672, 2010
- [36] Vikas Kumar, "Online Handwriting Recognition Issues and Techniques," *MIT IJCSIT*, vol. 5, no. 1, pp. 16-24, 2014.
- [37] Mollah Masum Billah Azadi, Md. Abdur Rahman.M, M. A. Hashem Mohammad Badiul Islam, "Bengali Handwritten Character Recognition using Modified Syntactic Method," in *Procs. of the 2nd National Conference on Computer Processing of Bangla (NCCPB-2005)*, pp. 264-275, 2005
- [38] Dit-Yan Yeung, Xiaolin Li, "Online Handwritten Alpha-Numeric Recognition Using Dominant Points in Strokes," pp.31-44, 1996.
- [39] Anil K jain, Scott D Connel, "Template-Based online Character Recognition," *Pattern Recognition*, vol. 1, no. 34, pp. 1-14, 2001.
- [40] Satya Lahari Putrevu, C V Jawahar, Karteek Alahari, "Discriminant Substrokes for Online Handwriting Recognition," in *Eighth International Conference on Document Analysis and Recognition*, pp. 499-503, 2005.
- [41] Ahmad T Al-Taani, "An Efficient Feature Extraction Algorithm for the Recognition of Handwritten Arabic Digits," *Engineering and Technology, International Journal of Computer, Electrical, Automation, Control and Information Engineering, World Academy of Science*, pp.2221-2225, 2008.

- [42] Jihad El-sana, Nizar Habash, Fadi Biadisy, "Online Arabic Handwriting Recognition using Hidden Markov Models," in *The 10th International Workshop on Frontiers of Hand-writing Recognition*, pp. 108-116, 2006.
- [43] M Borahan Tumer, Tunga Gungor, Aleksei Ustimov, "A Low-Complexity Constructive Learning Automaton Approach to Handwritten Character Recognition," *Research in Computing Science*, pp. 311-322, 2010
- [44] Najiba Tagougui, Haikal El Abed, Monji Kherallah, Houcine Boubaker, "Graphemes Segmentation for Arabic Online Handwriting Modeling," *J Inf Process Systems*, vol. 10, no. 4, pp. 503-522, 2014
- [45] Santanu Chaudhury, Abhijith Dutta, "Bengali alpha-numeric character recognition using Curvature features," *Pattern Recognition*, vol. 26, no. 12, pp. 1757-1770, 1993.
- [46] Tanmoy Dasgupta, Samar Bhattacharya, Priyanka Das, "A Handwritten Bengali Consonants Recognition Scheme based on the detection of Pattern Primitives," in *Second International Conference on Research in Computational Intelligence and Communication Networks.*, pp.72-77, 2016.
- [47] Bikash K Guptha, Swapan K Parui, Ujjwal Bhattacharya, "Direction Code Based Features for Recognition of Online Handwritten Characters of Bangla," , *International Conference on Document Analysis and Recognition*. pp. 58-62, 2007.

- [48] Sunil Kumar Kopparappu, Lajish VL, "Fuzzy Directional Features for Unconstrained On-line Devanagari Handwriting Recognition," in *National Conference On Communications (NCC)*, pp. 1-5, 2010.
- [49] U. Bhattacharya, S. K. Parui, S. Dutta Chowdhury, "Handwriting Recognition Using Levenshtein Distance Metric," in *12th International Conference on Document Analysis and Recognition*, pp. 79-83, 2013.
- [50] S Al-Emami, M Usher, "Online Recognition of Handwritten Arabic Characters," *Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 7, p. 704, 1990.
- [51] S.G.El-Taweel T.S.El-Sheikh, "Real-time arabic handwritten character recognition," *Pattern Recognition*, pp. 1323-1332, 1990.
- [52] Beigi Homayoon, Subrahmonia Jayashree, Clary G.J., Maruyama Hiroshi, Nathan K.S, ".Real-Time On-Line Unconstrained Handwriting Recognition Using Statistical Methods.," in *ICASSP*, pp. 2619-2622, 1995.
- [53] Wojciech Bieniecki, Szymon Grabowski, Bartosz Paszkowski, "Preprocessing for real-time handwritten character recognition," in *Computer Recognition Systems: Proc. Int. Conf. mboxCORES*, pp. 470-476, 2005.
- [54] Alberto Beltrán, Sonia Mendoza, "Efficient algorithm for real-time handwritten character recognition in mobile devices," in *2011 8th International Conference on Electrical Engineering, Computing Science and Automatic Control*, pp. 1-6, 2011

- [55] Cheng-Lin Liu, Xiang-Dong Zhou, Da-Han Wang, "An approach for real-time recognition of online Chinese Handwritten Sentences," *Pattern Recognition*, pp. 3661–3675, 2012.
- [56] Zehua Gao, Qi Song, "Real Time Handwritten Digit Recognition on Mobile Devices," in *Fourth International Conference on Intelligent Control and Information Processing (ICICIP)*, pp. 487-490, 2013.
- [57] Vinh Dinh Nguyen, Jae Wook Jeon, Hyung-Min Jeon, "Real-Time Multi-Digit Recognition System Using Deep Learning on an Embedded System," in *IMCOM '18: The 12th International Conference on Ubiquitous Information*, pp. 1-6, 2018.
- [58] Chengfeng Han, E. M. Riseman, W. B. Croft R. Manmatha, "Indexing Handwriting Using Word Matching," in *First ACM Intl. Conf. on Digital Libraries*, pp. 1-9, 1996.
- [59] Chengfeng Han, E. M. Riseman R. Manmatha, "Word Spotting: A New Approach to Indexing," in *CVPR IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pp. 1-29, 1996.
- [60] Giorgos Sfikas, Basilis Gatos, Christophoros Nikou, Angelos P. Giotis, "A survey of document image word spotting techniques," *Pattern Recognition*, vol. 68, pp. 310-332, 2017.
- [61] Wasfi G. Al-Khatib, Sabri Mahmoud, Rashad Ahmed, "A Survey on handwritten documents word spotting," *International Journal of Multimedia Information Retrieval, Springer*, vol. 6, no. 1, pp. 31-47, 2017.

- [62] Srinivasan H, Babu P, Bhole C, Srihari S, "Handwritten arabic word spotting using the cedarabic document analysis system," in *Symposium on document image understanding technology*, pp. 123-132, 2005.
- [63] Li L, Tan C, Bai S, "Keyword spotting in document images through word shape coding," in *10th international conference on document analysis and recognition*, pp. 331-335, 2009
- [64] Nobile N, He CL, Suen C, Sagheer M, "A novel handwritten urdu word spotting based on connected components analysis," in *20th international conference on pattern recognition (ICPR)*, pp. 2013-2016, 2010.
- [65] Brook S, Aghbari ZA, "HAH manuscripts: a holistic paradigm for classifying and retrieving historical Arabic handwritten documents," *Expert System Applications*, pp. 10942–10951, 2009.
- [66] Duygulu P Can EF, "A line-based representation for matching words in historical manuscripts," *Pattern Recognition Letters*, pp. 1126–1138, 2011.
- [67] Andreas Keller, Volkmar Frinken, Horst Bunke, Andreas Fischer, "Lexicon-Free Handwritten Word Spotting Using Character HMMs," *Pattern Recognition Letters*, pp. 934-942, 2012.
- [68] Fernández David, "Handwritten Word Spotting in Old Manuscript Images using Shape Descriptors," *Pattern Recognition and Image Analysis*, pp. 628-635, 2010.
- [69] Lynn Wilcox Shingo Uchihashi, "Automatic index creation for handwritten notes," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, pp. 3453-3456. 1999.

- [70] Sushama S.N, Sharada.B, "Keyword Spotting in Historical Handwritten Devanagari Documents: A Graph Based Approach," *American International Journal of Research in Science, Technology, Engineering and Mathematics*, pp. 1-8, 2019.
- [71] Praveen Krishnan, Minesh Mathew. C.V.Jawahar, Kartik Dutta, "Towards Spotting and Recognition of Handwritten Words in Indic Scripts," in *16th ICFHR*, pp. 32-37, 2018.
- [72] Sushma S.N., Bharathlal, Sharada B., "Keyword Spotting in Historical Devanagari Manuscripts by Word Matching," *Data Analytics and Learning. Lecture Notes in Networks and Systems*, pp. 65-76, 2019.
- [73] Palanivel S, Ramalingam V, Sigappi A, "Handwritten document retrieval system for tamil language," *International Journal of Computer Applications*, vol. 31, pp. 42-47, 2011.
- [74] Da-Han Wang, Cheng-Lin, Liu Heng Zhang, "Character confidence based on N-best list for keyword spotting in online Chinese handwritten documents," *Pattern Recognition*, pp. 1880-1890, 2014.
- [75] Da-Han Wang, Cheng-Lin Liu, Heng Zhang, "Keyword Spotting from Online Chinese Handwritten Documents Using One-vs-All Trained Character Classifier," in *12th International Conference on Frontiers in Handwriting Recognition*, Kolkatta, pp. 271-276, 2010
- [76] H. Zhang, C. Liu, "A Lattice-Based Method for Keyword Spotting in Online Chinese Handwriting," in *International Conference on Document Analysis and Recognition*, Beijing, pp. 1064-1068, 2011.
- [77] Mohamed Cheriet Sebastian Pena Saldarriaga, "Indexing On-Line Handwritten Texts Using Word Confusion Networks," in *International Conference on Document Analysis and Recognition*, pp. 197-201, 2011.

- [78] Volkmar Frinkeney, Horst Bunke, Emanuel Indermuhl, "Mode Detection in Online Handwritten Documents Using BLSTM Neural Networks," in *International Conference on Frontiers in Handwriting Recognition*, pp. 302-307, 2012.
- [79] A. Balasubramanian, Million Meshesha, Anoop M Namboodiri, C. V. Jawahar, "Retrieval of online handwriting by synthesis and matching," *Pattern Recognition*, pp. 1445-1457, 2009.
- [80] Anoop M. Namboodiri Anil K. Jain, "Indexing and Retrieval of On-line Handwritten Documents," in *Seventh International Conference on Document Analysis and Recognition (ICDAR '03)*, pp. 655-659, 2003.
- [81] Kenneth Katzner, *The Languages of the world.:* Reutledge and Kegan Paul Ltd, 2002.
- [82] Soman.K. P, Antony. P. J, "Computational Morphology and Natural Language Parsing for Indian Languages : A Literature Survey," *IJCSET*, vol. 3, pp. 136–146, 2012.
- [83] Vivek P., Lajish V.L.. "Hidden markov model based keyword spotting for malayalam speech analytics," <http://hdl.handle.net/10603/213852>, Calicut, 2017.
- [84] Merriam Webster. (1828), Merriam Webster dictionary.
- [85] Coulmas. F, *The Blackwell's Encyclopedia of Writing Systems.:* Oxford: Blackwells, 1996.
- [86] WikipediA. (2019, October) [Online]. <https://en.wikipedia.org/wiki/Phonology>
- [87] Hi-Tech.(2019, November),<http://www.hitech-in.com>. [Online]. <http://www.hitech-in.com>

- [88] Y. B. Zhang, M. T. Kechadi, Bing Quan Huang, "Preprocessing Techniques for Online Handwriting Recognition," in *Studies in Computational Intelligence.: Springer*, pp. 25-45, 2009.
- [89] Rejean Plamondon, Wacef Guerfali, "Normalizing and restoring on-line handwriting," *Pattern Recognition*, vol. 26, no. 3, pp. 419-431, 1993.
- [90] Steven W. Smith, *The Scientist and Engineer's Guide to Digital Signal Processing.:* California Technical Publishing , 1999.
- [91] Bong-Kee Sin, Jin Hyung Kim, "Online Handwriting Recognition," in *Handbook of Document Image Processing and Recognition.:* Springer, pp. 887-915, 2019.
- [92] R. Haralick, S. Aksoy, "Feature normalization and likelihood-based similarity measures for image retrieval," *Pattern Recognition Letters-Special Issue on Image and Video Retrieval*, pp. 563-582, 2000.
- [93] Anitha Jayaraman, V. Srinivasa Chakravarthy, C. Chandra Shekhar, H. Swethalakshmi, "Online Handwritten Character Recognition of Devanagari and Telugu Characters using Support Vector Machines," *HAL*, pp. 100-107, 2006.
- [94] M. Sukumar, A.G. Ramakrishnan, Mahadeva Prasad M, "Divide and Conquer Technique in Online Handwritten Kannada Character Recognition," in *International Workshop on Multilingual OCR*, pp. 1-7, 2009.
- [95] T. S. S. V. L. L. K. B. Baiju, "Segmentation of Malayalam Handwritten Characters into Pattern Primitives and Recognition using SVM," *International Journal of Engineering and Advanced Technology* , vol. Vol. 9, no. 3, p. 1817–1822, February 2020.

- [96] Anil K Jain, Anoop M. Namboodiri, "Online handwritten script recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, pp. 124-130, 2004.
- [97] Toru Ayabe, Takahiro Nishizaki, Ningping Sun, "Efficient Spline Interpolation Curve Modeling," in *Third International Conference on Intelligent Information Hiding and Multimedia Signal Processing (IIH-MSP 2007)*, pp. 59-62, 2007.
- [98] Sriganesh Madhvanath, Bharath A, "Online Handwriting Recognition for Indic Scripts," in *OCR for Indic Scripts: Document Recognition and Retrieval*, Springer, pp. 209-234, 2008.
- [99] Jun S Huang, Chia-Wei Liao, "Stroke segmentation by bernstein-bezier curve fitting," *Pattern Recognition*, vol. 23, no. 5, pp. 475-484, 1990.
- [100] M.K. Brown, W. Turin Jianying Hu, "HMM based online handwriting recognition," *IEEE Transactions on Pattern Analysis and Machine Intelligence* , vol. 18, no. 10, pp. 1039 – 1045, 1996.
- [101] Keiji Taniguchi, Hideo Ogawa, "Thinning and stroke segmentation for handwritten Chinese character recognition," *Pattern Recognition*, vol. 15, no. 4, pp. 299-308, 1982.
- [102] D. G. Thakore, N. J. Randive, "Static Hand Gesture Recognition using Freeman Chain Code and Neural Network," *Int. J. Adv. Eng. Res. Dev. Sci. J.* , pp. 113–134, 2015.
- [103] Jayadevan.R, Sharma.N, Pal U, "Handwriting Recognition in Indian Regional Scripts: A Survey of Offline Techniques," *ACM Transactions on Asian Language Information Processing*, pp. 1-35, 2012.

- [104] Sabeerath K, Lajish V.L, Baiju.K.B, "A Quick Review on Features and Classification Techniques in Online recognition of handwritten Dravidian Scripts," *International Journal of Research in Advent Technology*, vol. 6, no. 11, pp. 3245- 3251, November 2018.
- [105] Swethalakshmi H, "Online handwritten character recognition of Devanagari and Telugu Characters using support vector machines," in *Tenth International Workshop on Frontiers in Handwriting Recognition*, pp. 1-6. 2006.
- [106] AM Al-Salman, H Alyahya, "Arabic online handwriting recognition: A survey," in *ACM International Conference Proceeding Series, Association for Computing Machinery*, 2017.
- [107] Marc Parizeau, Rejean Plamondon, Xiaolin Li, "Segmentation And Reconstruction Of On-Line Handwritten Scripts," *Pattern Recognition*, vol. 31, no. 6, pp. 675-684, 1998.
- [108] Nicholson J, Clapham C, *Oxford Concise Dictionary of Mathematics*, Gradient.: Addison-Wesley, 2009.
- [109] Han Shu, "On-line handwriting recognition using HMM," ME thesis, MIT, 1996.
- [110] Thomas Deselaers, Henry A. Rowley, Li-Lun Wang, Victor Carbune, Daniel Keysers, "Multi-Language Online Handwriting Recognition," *IEEE Transactions on Pattern Analysis And Machine Intelligence*, pp. 1180-1194, 2016.
- [111] K.H. Aparna, M. Kasirajan, G.V. Prakash, V.S. Chakravarthy, S. Madhvanath, Aparna, "Online Handwriting Recognition for Tamil," in *Ninth International Workshop on Frontiers in Handwriting Recognition*, Tokyo, 2004.

- [112] V Anoop, VS Chakravarthy, G Shankar, "LEKHAK [MAL]: A System for Online Recognition of Handwritten Malayalam Characters," in NCC IIT Madras, 2003.
- [113] M. Nakagawa, C. T. Nguyen, "Finite State Machine Based Decoding of Handwritten Text Using Recurrent Neural Networks," in *15th International Conference on Frontiers in Handwriting Recognition (ICFHR)*, pp. 246-251, 2016.
- [114] R. Fernandez, L. Sanchez, D. Alvarez, "Stroke-based intelligent character recognition using a deterministic finite automaton," in *Logic Journal of the IGPL*, vol. 23, no. 3, pp. 463-471, 2015.
- [115] Wikipedia((2019),Wikipedia[Online], https://en.wikipedia.org/wiki/Deterministic_finite_automaton#RS59
- [116] Pitts. W, McCulloch W. S, "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics*, pp. 115–133, 1943.
- [117] Scott D, Rabin M. O, "Finite automata and their decision problems," *IBM Journal of Research and Development*, pp. 114-125, 1959.
- [118] Wang D-H, Liu C-L, Zhang H, "Keyword Spotting From Online Chinese Handwritten Documents Using One-Vs-All Trained Character Classifier," in *International Conference on Frontiers in Handwriting Recognition*, pp. 271-276, 2010.
- [119] Nabil Aouadi AK, "Word spotting for Arabic handwritten historical document retrieval using generalized hough transform," in *Third International Conference on Pervasive Patterns Applications*, pp. 67-71, 2011.

- [120] Gil J, Pinto J, Sousa J, "Word indexing of ancient documents using fuzzy classification," *Fuzzy Systems IEEE Transactions*, pp. 852-862, 2007.
- [121] Andrew Tomkins, Daniel Lopresti, "On the Searchability of Electronic Ink," in *Fourth International Workshop on Frontiers in Handwriting Recognition*, pp. 156—165, 1997.
- [122] Yoshua Bengio, Yann Le Cun, "Word Normalization for On-Line Handwritten Word Recognition," in *Proceedings of the 12th IAPR International Conference on Pattern Recognition*, vol. 2, pp. 409-413, 1994.
- [123] Michael Unser, Akram Aldroubi, Murray Eden, "B-Spline Signal-Processing.2. Efficient Design and Applications," in *IEEE Transactions on Signal Processing*, vol. 41, pp. 834-848, 1993.
- [124] J. D'Errico, "MATLAB Central File Exchange.," 2016. [Online]. <https://www.mathworks.com/matlabcentral/fileexchange/34874-interparc>), [Accessed September 7, 2016 September 7, 2016. September 7, 2016].
- [125] Baiju.K.B, Lajish V.L, Sabna.T.S, "Online handwritten Malayalam word recognition from Ayurveda Prescriptions using Support Vector Machines," *Compliance Engineering Journal*, vol. 10, no. 11, pp. 243-249, 2019.
- [126] Rajib Ghosh, Gouranga Mandal, "A novel Approach of Skew Correction for Online Handwritten Words," *International Journal of Computer Applications*, vol. 48, no.9, pp. 45-48, 2012.
- [127] R. Manmatha, T. M. Rath, "Word image matching using dynamic time warping," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2003.

- [128] L. I. Jiaping Zhao, "shapeDTW: Shape Dynamic Time Warping," *Pattern Recognition*, vol. 74, pp. 171-184, 2018.
- [129] K. B. Urala, A. G. Ramakrishnan, S. Mohamed., "Recognition of open vocabulary, online handwritten pages in Tamil script," in *Proceedings of the International Conference on Signal Processing and Communications*, pp. 1-6, 2014.
- [130] N.V. Neeba, Anoop Namboodiri, C.V. Jawahar, P.J. Narayanan, "Recognition of Malayalam Documents," in *Guide to OCR for Indic Scripts.: Springer*, pp. 125–146, 2009.
- [131] Herve Abdi, "Centroid," Wiley, 2009. [Online]. Available: <http://www.wiley.com/wires/compstats>.
- [132] G Sita, A G Ramakrishnan, S Madhvanath, N Joshi, "Comparison of Elastic Matching Algorithms for Online Tamil Handwritten Character Recognition," in *IWFHR* , pp. 444-449, 2004.
- [133] Baiju. KB, "Effect of Directional Features in the Recognition of Online Handwritten Characters in Malayalam," in *Infinitude:Frontiers of Research in Mathematical and Computing Sciences*. UGC-HRDC, University of Calicut, 2018.

List of Publications by the Author

- [1] **K. B. Baiju**, T. S. Sabna, and V. L. Lajish, “Segmentation of Malayalam Handwritten Characters into Pattern Primitives and Recognition using SVM”, *International Journal of Engineering and Advent Technology*, vol. 9, no. 3, pp. 1817–1822, 2020.
- [2] **K. B. Baiju**, V. L. Lajish, “Primitive Segmentation of Online Malayalam Handwritten Strokes using Ramer-Douglas-Peucker Algorithm and Eight Direction Freeman Code”, *International. Journal of Research in Electronics, Computing and Engineering*, vol. 7, no. 2, pp. 87–90, 2019.
- [3] T. S. Sabna, **K. B. Baiju**, and V. L. Lajish, “Online Handwritten Malayalam Word Recognition from Ayurveda Prescriptions using Support Vector Machine”, *Compliance Engineering. Journal*, vol. 10, no. 11, pp. 242–250, 2019.
- [4] **K. B. Baiju**, “Effect of Directional Features in the Recognition of Online Handwritten Characters in Malayalam”, *Infinitude, UGC-HRDC, University of Calicut*, pp. 30–40, 2018
- [5] **K. B. Baiju**, K. Sabeerath, and V. L. Lajish, “A Quick Review on Features and Classification Techniques in Online Recognition of Handwritten Dravidian Scripts”, *International Journal of. Research in Advent Technology.*, vol. 6, no. 11, pp. 3245–3251, 2018.
- [6] K. Sabeerath, **K. B. Baiju**, , “Review of Feature Extraction and Classification Techniques for OHCR in Indian Scripts”, *International*

Advanced Research Journal of Science, Engineering and. Technology,
vol. 3, no. 10, pp. 131–135, 2016.

- [7] **K. B. Baiju**, K. Sabeerath, “Online Recognition of Malayalam Handwritten Scripts - A comparison using KNN, MLP and SVM”, *2016 IEEE International Conference on Advanced. Computing, Communication and Informatics (ICACCI)*, Sept. 21-24, 2016, Jaipur, India, pp. 2078–2083, 2016.
- [8] **K. B. Baiju**, K. Sabeerath, and V. L. Lajish, “Online Recognition of Malayalam Scripts with Minimized Feature Dimensions”, *International Journal of Advanced Research in Computing and. Communication Engineering.*, vol. 5, no. 9, pp. 472–476, 2016.