

An Investigation into Telugu Font and Character Recognition

*Thesis submitted
by*

Ghadiyaram Anuradha
(2kMCPC 04)

Under the Supervision of
Dr. Arun Agarwal & Dr. C. Raghavendra Rao

*in partial fulfillment of the requirements
for the award of the degree of*
Doctor of Philosophy
in
Computer Science



Department of Computer and Information Sciences,
School of MCIS
University of Hyderabad
Hyderabad-500046

April 2009



**UNIVERSITY OF HYDERABAD
HYDERABAD - 500 046, A.P. (INDIA)**

**DEPARTMENT OF COMPUTER AND INFORMATION SCIENCES,
SCHOOL OF MCIS**

CERTIFICATE

This is to certify that the thesis entitled “**An Investigation into Telugu Font and Character Recognition** ” being submitted to the University of Hyderabad, by **Mrs. G.Anuradha (Reg. No. 2KMCPC04)** in partial fulfillment for the award of **Doctor of Philosophy in Computer Science** is a bonafide work carried out by her under our supervision. The matter embodied in this thesis has not been submitted to any other University for the award of any degree or diploma.

Prof. Arun Agarwal,

(Supervisor)

Department of Computer and
Information Sciences,
University of Hyderabad,
Hyderabad.

Prof. C. Raghavendra Rao,

(Supervisor)

Department of Computer and
Information Sciences,
University of Hyderabad,
Hyderabad.

Prof. Arun Agarwal,

Head,
Department of Computer and
Information Sciences,
University of Hyderabad,
Hyderabad.

Prof. T. Amaranath,

Dean,
School of Mathematics and
Computer/Information Sciences,
University of Hyderabad,
Hyderabad.

DECLARATION

I hereby declare that the work embodied in this thesis has been carried out by me under the supervision of **Prof. Arun Agarwal**, and **Prof. C. Raghavendra Rao** in the department of Computer and Information Sciences, University of Hyderabad, Hyderabad and has not been submitted to any other University. Information derived from the published and unpublished work of others has been acknowledged and a list of references is given.

Place: Hyderabad

Date:

Name: G. Anuradha

Regn No.: 2kMCPC04

Abstract

Optical Character Recognition (OCR) for Indian scripts and for handwriting is an active area of research. OCR systems for Indian and many other oriental languages are not yet able to successfully recognize printed documents of varying scripts, quality, font styles and sizes. To address some of these issues, we have designed and developed a framework to simultaneously recognize fonts and characters from documents printed in Telugu. Telugu script is structurally complicated due to the large number of character shapes that can be formed. Specifically, we developed a system based on Rough Sets theory for recognizing Telugu fonts and characters. Using the theory of rough sets, we determine the most important features of characters from the point of view of classification.

The present work achieved 94.37%, and 90.17% accuracies for character and font recognitions respectively. These are the raw recognition accuracies without doing any post processing. The present model emphasizes the role of theory of rough sets in feature selection and dimension reduction in Telugu font and character recognition.

A comparative study with related earlier works and our contributions to the field of Telugu character recognition are presented.

The developed approach falls into the category of *a priori* classification, where domain knowledge of the content is not used. No explicit local feature analysis is used in this methodology. Applicability of the proposed approach is tested on documents printed in English, which is an entirely different script from Telugu. Thus, a general framework for script-independent character and font recognitions is suggestive from the present methodology.

Encouraging recognition accuracies in both the scripts (Telugu & English) are achieved with a scope for further improvement.

Acknowledgements

This document is the final result of the research that I began quite some time ago. While the journey had been a lengthy one with several twists and turns, it was not a solitary one. I would like to thank my entire committee—for their time and patience in helping me complete this thesis. I particularly thank my supervisors: Prof. Arun Agarwal, and Prof. C. Raghavendra Rao for their expert guidance, and constant encouragement that helped me focus on the light at the end of the tunnel at every stage in this journey.

I am grateful for the critical analysis from the DRC committee: Dr. Chakravarthy Bhagavathy, and Dr. Atul Negi. I am immensely benefited by their domain knowledge in the area of character recognition.

I sincerely thank Dr. T. Amarnath, Dean, School of MCIS, for his sympathetic approach and valuable advices in completing this thesis.

I express my gratitude to Prof. B.K. Rao, Chairman of Marconi Institute of Technology, for his encouragement and for providing necessary facilities whenever required in completing this work.

I thank my fellow- researchers and also each and every member of the Department for their cooperation.

On a more personal level, there are several people that deserve acknowledgement. My husband, Mr. B. Srinivas, with his invisible support made me carry out this tremendous work. I am grateful for the support of my children who sacrificed many pleasures that they duly deserve, for my research sake. I would like to thank my mother, all my family members, and friends whose good wishes enabled me to pursue and achieve my goal.

Contents

Abstract	iv
Acknowledgements	v
List of Figures	xiii
List of Tables	xvi
1 Introduction	[1-10]
1.1 Human and Machine Perception	1
1.2 Approaches to Pattern Recognition	2
1.3 Building blocks of Pattern Recognition System	4
1.3.1 Sensor/ Transducer	4
1.3.2 Preprocessing	5
1.3.3 Feature Extraction	5
1.3.4 Classification	6
1.3.5 Description	6
1.4 Applications of Pattern Recognition System	6
1.5 Motivation for the Present Work	7
1.6 Major Contributions	9
2 OCR System: A Literature Survey	[11-47]
2.1 History of machine recognition of scripts	11
2.1.1 First generation OCR systems	12
2.1.2 Second generation OCR systems	12

2.1.3 Third generation OCR systems	13
2.1.4 OCRs Today (Fourth generation OCR systems)	13
2.2 Components of an OCR System	14
2.2.1 Preprocessing	15
2.2.1.1 Binarization	16
2.2.1.2 Noise Removal	17
2.2.1.3 Skew detection and correction.....	17
2.2.1.4 Line, word, and character segmentation	19
2.2.1.5 Thinning	21
2.2.2 Recognition	21
2.2.2.1 Feature extraction and selection	22
i. Structural approach	23
ii. Statistical approach	23
2.2.2.2 Classification	24
2.3 Font Consideration	26
2.3.1 OCR Classification based on Fonts	28
2.3.1.1 Fixed font OCRs	28
2.3.1.2 Multi-font OCRs	28
2.3.1.3 Omni font OCRs	28
2.3.2 Font Recognition approaches	29
2.4 Indian language OCRs	30
2.4.1 Techniques used in different Indian script OCRs	31

2.4.1.1	Devnagari OCR	31
2.4.1.2	Bangla OCR	32
2.4.1.3	Gurmukhi (Punjabi) OCR	34
2.4.1.4	Telugu OCR	35
2.4.1.5	Gujarati OCR	38
2.4.1.6	Oriya OCR	38
2.4.1.7	Assamese OCR	40
2.4.1.8	Kannada OCR	41
2.4.1.9	Tamil OCR	41
2.4.1.10	Malayalam OCR	43
2.4.2	Performance Testing	43
2.5	Justification for the present work	46
2.5.1	Identification of fonts	46
2.5.2	Dimensionality reduction	47
2.5.3	Applicability to other languages	47
3	Feature Selection: A Mathematical Foundation [48-79]	
3.1	Feature Selection	48
3.2	Data dimensionality	50
3.3	Rough sets based Feature selection	51
3.3.1	Theory of rough sets	51
3.3.1.1	Information Systems	52
3.3.1.2	Indiscernibility	54

3.3.1.3 Set Approximations	55
3.3.1.4 Reduction of Attributes	57
3.3.2 Computing Reducts	58
3.3.2.1 Dependency Function-based Approaches	58
3.3.2.2 Discernibility Matrix-based Approaches	61
3.4 Discussion on Dimensionality Reduction Techniques	64
3.4.1 Principal Components Analysis (PCA)	64
3.4.2 Rough Set-based Attribute Reduction (RSAR)	65
3.5 Feature Dimensionality Reduction in Telugu OCR	66
3.5.1 Telugu Script Origin & Development	66
3.5.2 Complexity of recognition in Telugu	68
3.5.3 Feature Selection in Telugu OCR	70
3.5.4 Role of Rough Sets in Feature-Set Reduction	71
3.5.5 Proposed Feature-Set Reduction Algorithm	72
3.5.5.1 Algorithm PDA ()	72
3.6 Summary	79
4 Design of the Classifier	[80-111]
4.1 Data Sets Used	81
4.1.1 Dataset for Training (Dataset # 1)	81
4.1.2 Dataset for Testing (Dataset # 2)	83
4.1.3 Dataset for Designing (Dataset # 3)	85
4.2 Proposed Classification versus Traditional classification	85

4.3	Base Feature Extraction and Selection	86
4.3.1	Feature Dimensionality Reduction	89
4.4	Structure of the Classifier	91
4.4.1	Experimentation for Tuning the Parameters	91
4.4.1.1	Effect of Neighboring Features on Recognition	92
4.4.1.2	Effect of input character size on recognition	95
4.4.1.3	Effect of Nearest-Neighbor Classification	96
4.4.2	Design of a Multi-stage Classifier	97
4.4.2.1	Effect of Error Thresholds on Font and Character Recognition	98
4.4.2.2	Reducing the Confusion Set	101
4.4.2.2.1	Hu's invariant moments	101
4.4.2.2.2	EM parameters	104
4.4.2.3	Feature Selection for the Second Stage	105
4.4.2.4	Selection of Classifier for Second Stage	107
4.4.2.5	Threshold Selection Revisited	109
4.4.3	Proposed Classifier	110
4.5	Summary	110
5	Proposed OCR System & Performance Evaluation	[112-134]
5.1	Preprocessing	113
5.1.1	Binarization	113
5.1.2	Connected Component Extraction	114
5.1.3	Size Normalization	116

5.1.3.1 Linear Normalization	117
5.1.3.2 Nearest Neighbor Interpolation	118
5.1.3.3 Bi-linear Interpolation	119
5.1.3.4 Bi-cubic Interpolation	119
5.1.3.5 Proposed Normalization Method	121
5.2 Feature Extraction	122
5.3 Feature Reduction	122
5.4 Feature Selection	123
5.5 Training Database	123
5.6 Classification	124
5.7 Performance Evaluation	125
5.7.1 Results	126
5.7.2 Comparative Study with Earlier Works	129
5.8 Observations & Findings	130
6 Testing the Generality of the Approach: Case Study on English	[135-147]
6.1 Preparation Of Data And Preprocessing	135
6.1.1 Dataset for Training (Dataset # 1)	135
6.1.2 Dataset for Testing (Dataset # 2)	136
6.1.3 Dataset for Designing (Dataset # 3)	137
6.2 Feature Extraction	137
6.3 Feature Reduction	139
6.4 Feature Selection	140

6.5 Training Database Creation	140
6.6 Test Database Creation	140
6.7 Designing the Classifier	141
6.8 Testing and Results	143
6.9 Observations	145
6.10 Summary	147
7 Conclusions And Future Scope	[148-151]
7.1 Conclusions	148
7.2 Future Scope	149
APPENDIX-A	
Prototype Character Images	[151-162]
DRISHTI Templates	[163-170]
APPENDIX-B	
Distinct Prototype Characters	[171- 180]
APPENDIX-C	
Font Names and Font Codes	181
References	[182-192]

List of Figures

Figure No.	Figure Caption	Page No.
1.1	Structure of a typical pattern recognition system	4
2.1	OCR-A font	13
2.2	OCR-B font	13
2.3	OCR process	14
2.4	Steps in an OCR	16
2.5	Document image binarization	17
2.6	An image with skew	18
2.7	An image and its horizontal projection profile	21
2.8	An image before and after thinning	22
2.9	Recognition process	22
2.10	Arial font families in italic, bold and plain styles	27
2.11	Arial font face	28
2.12	Chain code, and graphical representations	34
3.1	Feature Selection Process	50
3.3	Quick Reduct algorithm	61
3.4	Johnson Reducer algorithm	64
3.5	Telugu Vowels	67
3.6	Telugu Consonants	68

3.7	Maatras / Vowel modifiers	68
3.8	Sample Votthus / Consonant modifiers	68
3.9	Some Hraswaksharas	68
3.10	A character and its modifiers	68
3.11	Some compound characters	68
3.12	A Telugu akshara and its connected components	70
3.13	Horizontal crossing counts for the character "Ka"	71
4.1	Example test documents used in the present work	85
4.2	Calculation of neighboring features	94
4.3	Size Vs Character Recognition	97
4.4	Size Vs Font recognition	97
4.5	Role of Distance thresholds on (a) character and (b) font recognitions	100
4.6	Total Costs for (a) character, and (b) font recognitions	101
4.7	Overall Recognitions Vs Distance threshold	102
4.8	Hu moments for a sample character in different sizes	104
5.1	Block diagram of the TFCR system	113
5.2	A document image (a) before and (b) after binarization	115
5.3	(a) An input document and (b) its connected components highlighted with rectangular boundaries	116
5.4	A character image (a) before, and (b) after normalization	118
5.5	An image resized with a) linear normalization, b) after column-wise smoothing, and c) after row and column-wise smoothing	119

5.6	A character image of 34X36 size normalized to 50 X50	120
5.7	Crossing counts using (a) Nearest Neighbor, (b) Bi-linear, and (c) Bi-Cubic Interpolation methods	121
5.8	An image of size (a) 34X36 zoomed up to (b) 150 X 150, down to (c) 50 X50	122
5.9	100 crossing counts using our method	123
5.10	Block diagram of the 2-stage classifier	125
5.11	Character and Font Recognitions for different fonts	128
5.12	Character and Font Recognitions for different sizes	129
5.13	Some touching characters	131
5.14	Some similar-looking character pairs	132
5.15	Some broken characters	132
6.1	Effect of Threshold distance on (a) character and (b) font recognitions (English)	141
6.2	Costs for (a) character and (b) font recognitions (English)	142
6.3	Character and Font Recognitions for different fonts (English)	144
6.4	Character and Font Recognitions for different sizes (English)	145

List of Tables

Table No.	Table Heading	Page No.
2.1	Performance results of OCRs developed at the RCILTs	45
2.2	Feature sets and classifiers of different OCRs	46
3.1	An example information system	54
3.2	Discernibility matrix for Table 3.1	63
3.3	An example decision table with two decision- attributes	75
3.4	Decision table with respect to Charcode	76
3.5	Partial discernibility matrix after comparing C0	76
3.6	Partial discernibility matrix after comparing C2	76
3.7	Partial discernibility matrix after comparing C1	77
3.8	Final discernibility matrix	77
3.9	Discernibility matrix after removing rows with $F4=1$	78
3.10	Predominant and decision (Charcode) attributes	78
3.11	Decision table with Fontcode	79
3.12	Predominant and decision (Fontcode) attributes	79
4.1	Some prototype characters in 6 fonts	83
4.2	Font names and font codes	84
4.3	Number of Test characters: Size-wise, Font-wise	86
4.4	Feature vectors for 5 Sample characters	88
4.5	Sample database with 30-D feature vectors	91
4.6	Effect of neighboring features on recognition	96
4.7	Results of nearest neighbor classification	98
4.8	Hu moments for a sample character in different sizes	105
4.9	Evaluation of Features	107
4.10	Evaluation of classifiers	110

4.11	Effect of threshold distance on final recognition rates	110
5.1	Performance results	127
5.2	Comparison of Results	130
5.3	Performance results on clear data	133
5.4	Same character(s) in different fonts	134
6.1	Prototype characters from the five selected fonts	137
6.2	Number of Test characters(Dataset # 2): Size-wise, Font-wise	138
6.3	103-D feature vectors for 5 characters	139
6.4	Sample database with 21-D feature vectors	142
6.5	Performance results on English characters	145
6.6	Similar-looking characters	145
6.7	Some characters that appear similar in different fonts	146

Chapter 1

Introduction

1.1 Human and Machine Perception

Achieving human functions like reading, recognizing and thinking through machines is an ancient dream. Much of our interaction with environment requires recognition of 'things' such as sounds, smells, shapes in a scene (text characters, faces, flowers, plants), etc. It is natural that we should seek to design and build machines that can recognize such patterns. In several real world problems like fingerprint identification, automated speech recognition, optical character recognition, DNA sequence identification etc., reliable and accurate pattern recognition by machines would be immensely useful [Duda, Hart, & Stork, 2000]. If these tasks have to be automated, the machines should have some capability to recognize patterns in the environment.

It is generally easy for a person to differentiate the sound of a human voice from that of a musical instrument; a handwritten digit "5" from the alphabet "S"; and the aroma of a flower from that of an onion. The human brain can easily recognize the characters in hundreds of different sizes and fonts. Computers, however, are not as smart as people. These kinds of perceptual problems are difficult for the computer because of voluminous data with composite and hidden information each pattern usually contains.

These problems are best studied under the domain of *Pattern Recognition (PR)*, which is the study of how machines can observe the environment, learn to distinguish patterns of interest, and make reasonable decisions about the categories of these patterns [Jain, Duin, & Mao, 2000]. Pattern recognition is defined as the science that concerns the description or classification (recognition) of measurements [Schalkoff, 1992].

"Cognition" is the mental process of *knowing*: including aspects such as awareness, perception, reasoning, and judgment. Re-cognition means awareness that something perceived has been perceived before [Farlex, 2003]. "The word *pattern* is derived from the same root as the word *patron* and, in its original use, means something which is set up as a

perfect example to be imitated. Thus pattern recognition means the identification of the ideal which a given object was made after” [Pavlidis, 1977].

We are often influenced by the knowledge of how patterns are modeled and recognized in nature when we develop pattern recognition algorithms. The essential problem of pattern recognition is to identify an object (pattern) as belonging to a particular group. Assuming that the objects associated with a particular group share more common attributes than with objects in other groups, the problem of assigning an unlabeled object to a group can be accomplished by determining the attributes/features of the object, and identifying the group of which those attributes are most representatives. If information about the universe of all possible objects and the groups to which they can be assigned is known, then the identification problem is straightforward: the attributes that best discriminate among groups, and the mapping from attributes to groups can both be determined with certainty. If the information about the identification problem is imperfect or incomplete, then the attributes and mappings must be *inferred* from example objects whose group membership is known. The best pattern recognizers are humans, yet we do not understand how humans recognize patterns [Jain, Duin, & Mao, 2000].

1.2 Approaches to Pattern Recognition

Pattern recognition techniques can be broadly classified into the following 3 categories:

1. Statistical Approach
 2. Syntactic approach, and
 3. Structural Approach
-
1. In statistical approach, classes and objects within the classes are modeled statistically. In statistical pattern recognition we focus on the statistical properties of the patterns, generally expressed in probability densities [Duda, Hart, & Stork, 2000]. A pattern is represented as a vector of D-features and is viewed as a point in D- dimensional space. A *feature extraction* mechanism measures certain “features” or “properties” of the objects, and a *classification* scheme does the actual job of

classifying observations, depending on the extracted features, and based on some measures.

2. In syntactic PR (SyntPR), patterns are viewed as composed of simpler and simpler sub patterns leading eventually to atomic patterns which cannot anymore be decomposed into sub patterns. Pattern recognition is then the study of these atomic patterns (primitives) and the language between relations of these atomic patterns. The theory of formal languages forms the basis of syntactic pattern recognition [Tohka, 2006]. A pattern is represented by a sentence in a language which is specified by a grammar. The rules governing the composition of primitives into patterns are specified by the so called "pattern grammar." The practical utility of the syntactic approach depends upon our ability to recognize the simple pattern primitives and their relationships represented by the composition operations [Fu & Rosenfeld, 1976]. Pattern classes are thus represented by means of formal structures such as grammars, automata, strings, etc. Syntactic Pattern Recognition describes the input to the pattern recognition system as an aggregate of parts, but it places major emphasis on the rules of composition [Pavlidis, 2000].
3. In structural PR, domain knowledge (description) is used to discriminate among objects belonging to different groups based on the arrangement of their shape-based (structural) features, and extraction of these features. Structural pattern recognition emphasizes on the description of the structure, namely how some simple sub-patterns compose one pattern. Structural pattern recognition systems require domain knowledge. Knowledge acquisition techniques necessary to obtain this domain knowledge from experts are tedious and often fail to produce a complete and accurate knowledge base. Hence, structural pattern recognition systems are difficult to apply to new domains.

A central aspect in virtually every pattern recognition problem is that of achieving such a "good" representation, one in which the structural relationships among the components is simply and naturally revealed, and one in which the true (unknown) model of the patterns can be expressed [Duda, Hart, & Stork, 2000].

1.3 Building blocks of Pattern Recognition System

General composition of a PR system is given below in Figure 1.1, and each of the sub-systems is explained next.

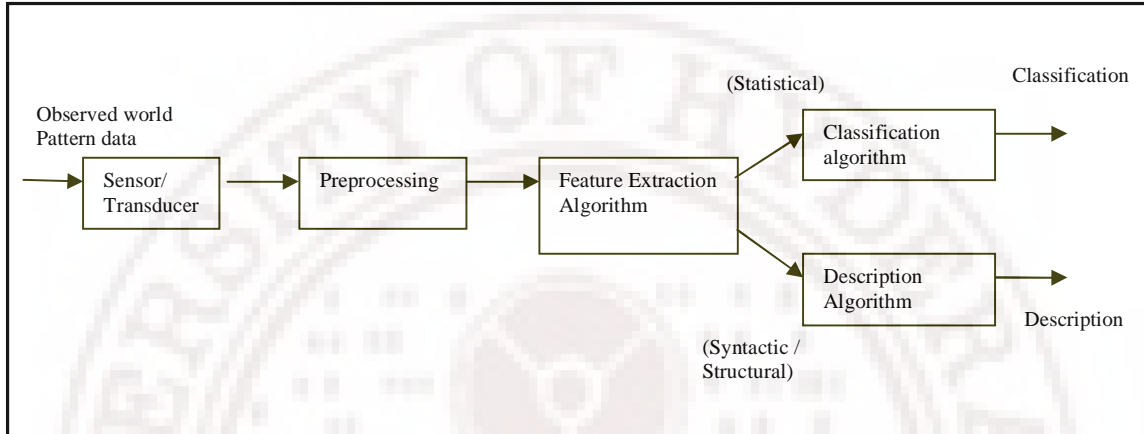


Figure 1.1 Structure of a typical pattern recognition system

1.3.1 Sensor/ Transducer

The purpose of a pattern recognition program is thus to analyze a scene in the real world with the aid of an input device, which is usually in the form of a transducer such as digital camera or scanner, and to arrive at a description of the scene. A *transducer* converts an energy pattern from the “real” physical world, such as a light intensity pattern into a digital pattern in the computer. Normally, sensors are considered as outside the pattern recognition system. Thus, a typical pattern recognition system consists of:

1. A preprocessing stage in which the quality of the digital image is enhanced to make it suitable for pattern recognition.
2. A feature extraction mechanism that computes numeric or symbolic information from the observations; and
3. A classification *or* description scheme that does the actual job of classifying or describing observations, depending on the extracted features, and based on some measures.

1.3.2 Preprocessing

Preprocessing is the name given to a set of procedures for smoothing, enhancing, filtering a digital image so that subsequent algorithms can be made simple and more accurate.

1.3.3 Feature Extraction

Feature extraction is again a set of procedures for measuring the relevant shape information contained in a pattern. It is necessary to choose the features carefully, and hence achieve a representation that enables reasonably successful pattern classification. One criterion for designing a feature extractor is that of finding as few features as possible that adequately differentiate the patterns into their corresponding pattern classes. There is no theory either in computer science or in psychology to solve this problem optimally [Toussaint, 1997]. Feature extraction is very problem-dependent. Usually one feature is not enough to differentiate between objects from different categories. Multiple features representing the same object are organized into *feature vectors*. The set of all possible feature vectors is called the *feature space*.

Characteristics of a good feature set

- Feature set should have a good discriminating power in order to enable the correct identification even among very similar symbols. Thus, features which are good for classification ensure that objects from the same class have similar feature values; objects from different classes have different values.
- A good feature set should be efficient with respect to computation-time.
- As far as possible the feature set should be invariant to rotation, scaling, and translation. This is because we often seek recognition of objects even when they are in any arbitrary position, with angular variations, and scaled to different sizes.
- Feature set should accord some immunity to noise by being robust with respect to occlusion, distortion, deformation, and variations in environment.

In order to solve a pattern recognition problem, we need features that are information preserving (or the loss of information is controlled), and the features whose

resulting feature vectors cluster tightly within class and are far apart for different classes [Pavlidis, 2000].

1.3.4 Classification

Statistical PR systems employ classifiers. The task of a classifier is to use the feature vector provided by the feature extractor, and to assign the object to a category. Essentially the classifier divides the feature space into regions corresponding to different categories. The degree of difficulty of the classification problem depends on the variability in the feature values for objects *in the same category* relative to the feature value variation *between* the categories. Variability is natural or it is due to noise. If the features have the good qualities mentioned above, and have a sufficient number of representative samples, then a good classifier will complete the solution [Pavlidis, 2000].

1.3.5 Description

In syntactic/structural PR, structure of an entity is important, and it may be used for the description [Shalkoff, 1992]. Description is an alternative to classification, where a structural description of the input pattern is given. It is common to use linguistic or structural models in description. The interrelationships or interconnections of features yield important structural information, which facilitates structural description.

1.4 Applications of Pattern Recognition System

Pattern Recognition finds applications that range from the classical ones such as automatic character recognition and medical diagnosis to the more recent ones in data mining such as consumer sales analysis and credit card transaction analysis. Some of the applications of Pattern Recognition are listed in Table 1.1(from Jain, Duin, & Mao, 2000).

Table 1.1 Applications of Pattern recognition

S. No	Problem Domain	Application	Input Pattern	Pattern Classes
1	Document image analysis	Optical character recognition	Document image	Characters, words
2	Document classification	Junk mail filtering	Email	Junk/non-junk
3	Document classification	Internet search:	Text document	Semantic categories
4	Speech recognition	Telephone directory assistance	Speech waveform	Spoken words
5	Data mining	Searching for meaningful patterns	Points in multidimensional Space	Compact and well-separated cluster
6	Biometric recognition	Personal identification	Face, iris, fingerprint	Authorized users for access control
7	Health	Medical Diagnosis	Microscopic image	Cancerous/healthy cell
8	Military	Automatic target recognition	Optical or infrared image	Target type
9	Industrial automation	Printed circuit board inspection	Intensity or range image	Defective/non-defective product
10	Industrial automation	Fruit sorting	Images taken on a conveyor belt	Grade of quality
11	Remote sensing	Forecasting crop yield	Multi-spectral image	Land use categories, growth pattern of crops
12	Bioinformatics	Sequence analysis	DNA sequence	Known types of genes

1.5 Motivation for the Present Work

Over the last few decades, machine reading is changing from a dream to reality. Optical Character Recognition systems (OCR) deal with the recognition of printed or handwritten characters (Serial number 1 in Table 1.1). Methodically, character recognition is a subset of the pattern recognition area. However, it was character recognition that gave

the impetus to the Pattern Recognition and Image Analysis to become matured fields of science.

At present, reasonably efficient and inexpensive OCR packages are commercially available to recognize printed texts in languages such as English, Chinese, and Japanese. On the contrary, there is only limited research effort made in the recognition of Indian and other oriental languages. While research in OCR for printed Roman script has reached a point of diminishing returns, OCR for handwriting and for printed non-Roman scripts continues to be a very active field [Kasturi et al, 2002].

Apart from character recognition, recognizing the font[†] of a printed document is not even attempted on Indian language documents; while some successful studies are made in English. Font recognition approaches for documents printed in English are described in section 2.3.2. Typographically, a font is a particular instantiation of a typeface design, often in a particular size, weight and style [Rubinstein, 1988]. A number of OCR systems have been developed for different languages across the globe, with reasonable accuracy, but the performance of these recognizers is fair as long as the same font is maintained. Since this requirement is not practical, often we get poor results. In many occasions, printed documents may contain words in various font faces and sizes. For Indian and many other oriental languages, OCR systems are not yet able to successfully recognize printed document images of varying scripts, quality, size, style and font [Rawat, et al, 2006]. Reasons for all this slow development in Indian script OCRs could be attributed to the peculiarities of the shape of the character sets. Some of these peculiar characteristics are as follows:

Most of the Indian scripts are two-dimensional compositions of symbols: core characters in the middle strip, optional modifiers above and/or below core characters. In Indian language scripts, the concept of upper case and lower-case characters is absent; however, the alphabet itself contains more number of symbols (vowel and consonant characters) than that of English. Apart from vowel and consonant characters called basic characters, in most of the Indian languages, there is a practice of having more than twelve forms each for all the consonants, giving rise to modified characters, depending on whether

[†] A description of font types and styles is given in section 2.3.

the vowel is placed to the left, right, top or bottom of the consonant. These compound characters exist in most Indian script alphabet systems except Tamil and Gurumukhi scripts [Pal & Chaudhuri, 2004]. The shape of a compound character is usually more complex than the constituent basic characters. The net result is that there are several thousand different shapes or patterns, which may, in addition, be connected with each other without any visible separation. Compared to European languages, Indian languages have many additional challenges like lack of standard test databases, lack of support from browsers, operating system, and keyboard, etc. This makes the Indian script OCRs difficult to develop.

In the present work, recognition of fonts and characters from documents printed in Telugu, which is one of the major scheduled languages of India, is chosen. Telugu is the official language of Andhra Pradesh and the second widely spoken language in Tamilnadu, and Karnataka. To our knowledge, no studies were reported earlier to recognize Telugu fonts. Font recognition is a fundamental issue in document analysis and recognition, and is often a difficult and time-consuming task [Zhu, Tan & Wang, 2001]. When no font information is available, the recognition process becomes more time consuming and the accuracy decreases as more fonts are added [Shi and Pavlidis, 1997]. Font information has been used in many multi-font text recognition systems. Thus, OCR for Telugu is an emerging area of research, and recognizing Telugu fonts and characters simultaneously from printed documents is a novel attempt, which is the main focus of the present work. During our investigation, it is found that the theory of rough sets is useful in feature selection and can be used efficiently in pattern recognition applications [Swinarski, 2001]. So, usefulness of rough sets in reducing the feature dimensionality of the Telugu OCR problem is also explored in the present work.

1.6 Major Contributions

We believe that our work[‡] has contributed to the field of Telugu character recognition in the following areas:

[‡] Parts of the present work had appeared in four international conference proceedings, and three international journals.

1. A new approach for simultaneously recognizing Telugu characters and font typefaces is developed with better accuracies.
2. A novel method of *semantics-preserving* dimensionality reduction is introduced for feature set reduction in Telugu OCRs using Rough sets theory.
3. Developing a framework for script-independent character and font recognition is also investigated.
4. Several parameters that influence the performance of the classifier are analyzed and fine-tuned by extensive experimentation.

The above contributions have been segregated and organized into the present thesis as follows:

Chapter 1 gives introduction and motivation for the present work. Chapter 2 presents literature survey on font and character recognition approaches, along with an overview of the research in Indian OCRs. Chapter 3 discusses feature selection, and complexity of feature selection in Telugu script. In this chapter, we introduce a new algorithm for reducing feature dimensionality. Our algorithm is rooted in the strong mathematical principles, called Rough Set Theory (RST). Detailed design procedure of the classifier is presented in Chapter 4, where several parameters that influence the recognition efficiency are given careful consideration. Implementation issues of the proposed Telugu Font and Character Recognition System (TFCRS), along with the performance results are presented in Chapter 5. Applicability of the proposed method for script-independent recognition is also investigated, considering English printed characters, which is presented in detail, in Chapter 6. Chapter 7 concludes the thesis with suggestions for the future scope of our work.

Chapter 2

OCR System: A Literature Survey

2.1 History of machine recognition of scripts

The overwhelming volume of paper-based data in corporations and offices challenges their ability to manage documents and records. Computers, working faster and more efficiently than human operators, can be used to perform many of the tasks required for efficient document and content management. Computers understand alphanumeric characters as ASCII code typed on a keyboard where each character or letter represents a recognizable code. However, computers cannot distinguish characters and words from scanned images of paper documents. Therefore, where alphanumeric information must be retrieved from scanned images such as commercial or government documents, tax returns, passport applications and credit card applications, characters must first be converted to their ASCII equivalents before they can be recognized as readable text. Optical character recognition system (OCR) allows us to convert a document into electronic text, which we can edit and search etc. It is performed off-line after the writing or printing has been completed, as opposed to on-line recognition where the computer recognizes the characters as they are written. For these systems to effectively recognize hand-printed or machine-printed forms, individual characters must be well separated. This is the reason why most typical administrative forms require people to enter data into neatly spaced boxes and force spaces between letters entered on a form. Without the use of these boxes, conventional technologies reject fields if people do not follow the structure when filling out forms, resulting in a significant overhead in the administration cost.

Optical character recognition for English has become one of the most successful applications of technology in pattern recognition and artificial intelligence. OCR is the machine replication of human reading and has been the subject of intensive research for

more than five decades. To understand the evolution of OCR systems from their challenges, and to appreciate the present state of the OCRs, a brief historical survey of OCRs is in order now.

Depending on the versatility, robustness and efficiency, commercial OCR systems may be divided into the following four generations [Line, 1993; Pal & Chaudhuri, 2004]. It is to be noted that this categorization refers specifically to OCRs of English language.

2.1.1 First generation OCR systems

Character recognition originated as early as 1870 when Carey invented the retina scanner, which is an image transmission system using photocells. It is used as an aid to the visually handicapped by the Russian scientist Tyurin in 1900. However, the first generation machines appeared in the beginning of the 1960s with the development of the digital computers. It is the first time OCR was realized as a data processing application to the business world [Mantas, 1986]. The first generation machines are characterized by the “constrained” letter shapes which the OCRs can read. These symbols were specially designed for machine reading, and they did not even look natural. The first commercialized OCR of this generation was IBM 1418, which was designed to read a special IBM font, 407. The recognition method was template matching, which compares the character image with a library of prototype images for each character of each font.

2.1.2 Second generation OCR systems

Next generation machines were able to recognize regular machine-printed and hand-printed characters. The character set was limited to numerals and a few letters and symbols. Such machines appeared in the middle of 1960s to early 1970s. The first automatic letter-sorting machine for postal code numbers from Toshiba was developed during this period. The methods were based on the structural analysis approach. Significant efforts for standardization were also made in this period. An American standard OCR character set: *OCR-A* font (Figure 2.1) was defined, which was designed to facilitate optical recognition, although still readable to humans. A European font *OCR-B* (Figure 2.2) was also designed.

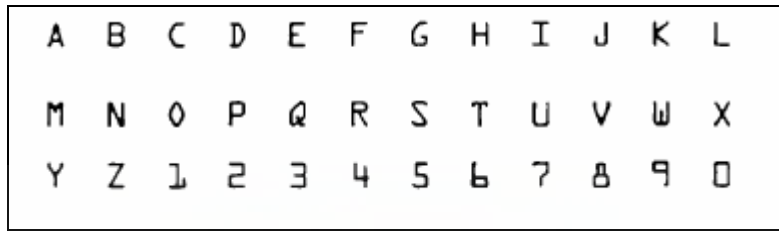


Figure 2.1 OCR-A font

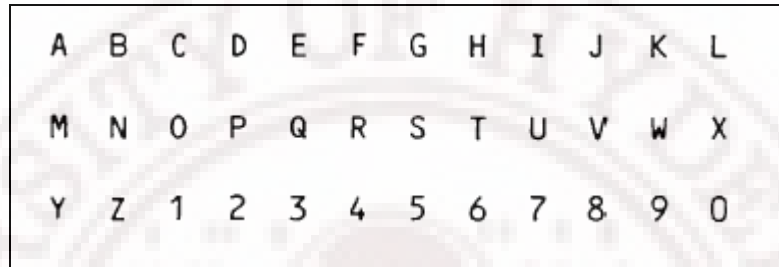


Figure 2.2 OCR-B font

2.1.3 Third generation OCR systems

For the third generation of OCR systems, the challenges were documents of poor quality and large printed and hand-written character sets. Low cost and high performance were also important objectives. Commercial OCR systems with such capabilities appeared during the decade 1975 to 1985.

2.1.4 OCRs Today (Fourth generation OCR systems)

The fourth generation can be characterized by the OCR of complex documents intermixing with text, graphics, tables and mathematical symbols, unconstrained hand-written characters, color documents, low-quality noisy documents, etc. Among the commercial products, postal address readers, and reading aids for the blind are available in the market.

Nowadays, there is much motivation to provide computerized document analysis systems. OCR contributes to this progress by providing techniques to convert large volumes of data automatically. A large number of papers and patents advertise recognition rates as high as 99.99%; this gives the impression that automation problems seem to have been solved. Although OCR is widely used presently, its accuracy today is still far from that of a

seven-year old child, let alone a moderately skilled typist [Nagy, Nartker & Rice, 2000]. Failure of some real applications show that performance problems still exist on composite and degraded documents (i.e., noisy characters, tilt, mixing of fonts, etc.) and that there is still room for progress.

Various methods have been proposed to increase the accuracy of optical character recognizers. In fact, at various research laboratories, the challenge is to develop robust methods that remove as much as possible the typographical and noise restrictions while maintaining rates similar to those provided by limited-font commercial machines [Belaid,1997].

Thus, current active research areas in OCR include handwriting recognition, and also the printed typewritten version of non-Roman scripts (especially those with a very large number of characters).

2.2 Components of an OCR System

Before we present a survey on various approaches used in the literature for recognizing fonts and characters, a brief introduction to the general OCR techniques is given now.

The objective of OCR software is to recognize the text and then convert it to editable form. Thus, developing computer algorithms to identify the characters in the text is the principal task of OCR. A document is first scanned by an optical scanner, which produces an image form of it that is not editable. Optical character recognition involves translation of this text image into editable character codes such as ASCII. Any OCR implementation consists of a number of preprocessing steps followed by the actual recognition, as shown in Figure 2.3.

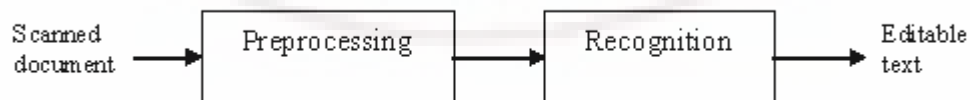


Figure 2.3 OCR process

The number and types of preprocessing algorithms employed on the scanned image depend on many factors such as age of the document, paper quality, resolution of the scanned image, amount of skew in the image, format and layout of the images and text, kind of the script used and also on the type of characters: printed or handwritten [Anbumani & Subramanian, 2000]. After preprocessing, the recognition stage identifies individual characters, and converts them into editable text. Figure 2.4 depicts these steps and they are described in the following section.

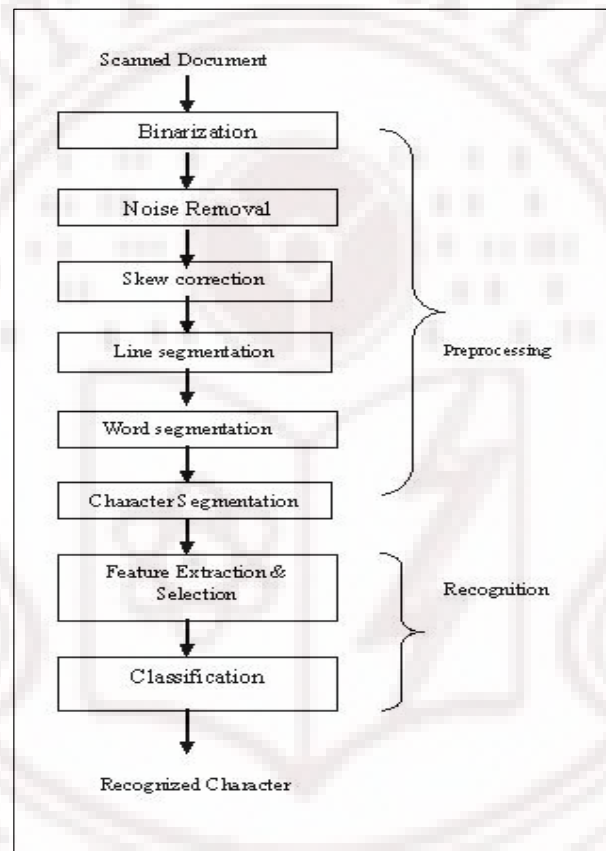


Figure 2.4 Steps in an OCR

2.2.1 Preprocessing

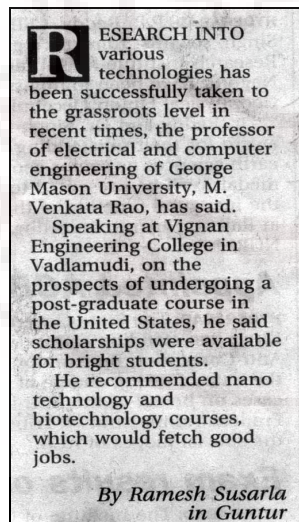
Typical preprocessing includes the following stages:

- Binarization,
- Noise removing,

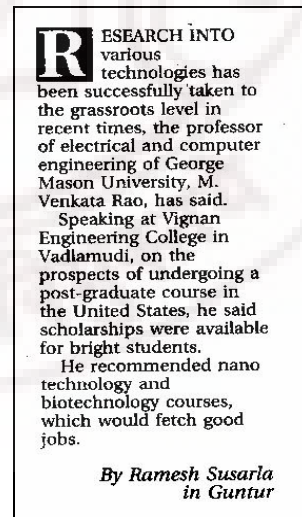
- Skew detection and correction,
- Line segmentation,
- Word segmentation,
- Character segmentation, and
- Thinning

2.2.1.1 Binarization

A printed document is first scanned and is converted into a gray scale image. Binarization is a technique by which the gray scale images are converted to binary images. In any image analysis or enhancement problem, it is very essential to identify the objects of interest from the rest. In a document image this usually involves separating the pixels forming the printed text or diagrams (foreground) from the pixels representing the blank paper (background). The goal is to remove only the background, by setting it to white, and leave the foreground image unchanged. Thus, binarization separates the foreground and background information. This separation of text from background is a prerequisite to subsequent operations such as segmentation and labeling [Bunke & Wang, 1997]. Figure 2.5 shows a gray image (a), and binary image (b) of a newspaper document.



(a) Gray image



(b) Binary image

Figure 2.5

Document image binarization

The most common method for binarization is to select a proper intensity threshold for the image and then convert all the intensity values above the threshold to one intensity value, and to convert all intensity values below the threshold to the other chosen intensity. Thresholding (Binarization) methods can be classified into two categories: global and adaptive thresholding. Global methods apply one threshold value to the entire image. Local or adaptive thresholding methods apply different threshold values to different regions of the image [Wu & Amin, 2003]. The threshold value is determined by the neighborhood of the pixel to which the thresholding is being applied. Among those global techniques, Otsu's thresholding technique [Otsu, 1979] has been cited as an efficient and frequently used technique [Leedham et al., 2002]. Niblack's method [Niblack, 1986] and Sauvola's method [Sauvola & PietikaKinen, 2000] are among the most well known approaches for adaptive thresholding.

2.2.1.2 Noise Removal

Scanned documents often contain noise that arises due to printer, scanner, print quality, age of the document, etc. Therefore, it is necessary to filter this noise before we process the image. Commonly used approach is to process the image through a low-pass filter and use it for later processing. The objective in the design of a noise- filter is that it should remove as much of the noise as possible while retaining the entire signal [Kasturi et. al, 2002].

2.2.1.3 Skew detection and correction

When a document is fed to the scanner either mechanically or by a human operator, a few degrees of tilt (skew) is unavoidable. Skew angle is the angle that the lines of text in the digital image make with the horizontal direction. Figure 2.6 shows an image with skew.



Figure 2.6 An image with skew

A number of methods have previously been proposed in the literature for identifying document image skew angles. Mainly, they can be categorized into the following groups [Lu & Tan, 2003]: (i) methods based on projection profile analysis, (ii) methods based on nearest- neighbor clustering, (iii) methods based on Hough transform, (iv) methods based on cross-correlation, and (v) methods based on morphological transforms. Survey on different skew correction techniques can be found in [Hull, 1998; Chaudhuri & Pal, 1997].

Most existing approaches use the Hough transform or enhanced versions [Srihari & Govindaraj, 1989; Pal & Chaudhuri, 1996; Amin & Fischer, 2000]. Hough transform detects straight lines in an image. The algorithm transforms each of the edge pixels in an image space into a curve in a parametric space. The peak in the Hough space represents the dominant line and its skew. The major draw back of this method is that it is computationally expensive and is difficult to choose a peak in the Hough space when text becomes sparse [Shivakumara et al., 2003].

Approaches based on the correlation [Yan, 1993] use cross correlation between the text lines at a fixed distance. It is based on the observation that the correlation between vertical lines in an image is maximized for a skewed document; in general if one line is shifted relatively to the other lines such that the character base line levels for two lines are coincident. The algorithm can be seen as a special case of the Hough transform. It is accurate for skew angles less than ± 10 degrees.

Postl [Postl, 1986] proposed a method in which the horizontal projection profile is calculated at a number of different angles. A projection profile is a one-dimensional array with a number of locations equal to the number of rows in an image. Each location in the projection profile stores a count of the number of black pixels in the corresponding row of the image. This histogram has the maximum amplitude and frequency when the text in the image is skewed at zero degrees since the number of co-linear black pixels is maximized in this condition [Hull, 1998]. To determine the skew angle of a document, the projection profile is computed at a number of angles, and for each angle, the difference between peak and trough heights is measured. The maximum difference corresponds to the best alignment

with the text line direction. This, in turn, determines the skew angle. The image is then rotated according to the detected skew angle.

Nearest neighbor methods described by Hoashizume et al. [Hoashizume et al., 1986] finds nearest neighbors of all connected components, the direction vector for all nearest neighbor pairs are accumulated in a histogram and the histogram peak is found to obtain a skew angle. Since only one nearest neighbor connectivity is made for each component, connection with noisy sub parts of characters reduce the accuracy of the method.

Approaches using morphological operations [Chen & Haralick, 1994] remove ascenders and descenders and merge adjacent characters to produce one connected component for each line of text. A line is then fit to the pixels in each component using a least-squares technique. Histogram of the angles of the lines detected is constructed in a document and a search procedure is applied to the histogram to determine the skew of the document.

In the methods based on Fourier Transform [Postl, 1986], the direction for which the density of the Fourier space is the largest, gives the skew angle. Fourier method is computationally expensive for large images.

2.2.1.4 Line, word, and character segmentation

Once the document image is binarized and skew corrected, actual text content must be extracted. Commonly used segmentation algorithms in document image analysis are: Connected component labeling, X-Y tree decomposition, Run-length smearing, and Hough transform [Bunke & Wang, 1997].

In connected component labeling, each connected component in the binary document image is assigned a distinct label. A connected component algorithm scans an image and groups its pixels into components based on pixel connectivity, *i.e.* all pixels in connected component share similar pixel intensity values and are in some way connected to each other. Once all groups have been determined, each pixel is labeled according to the component (group) it was assigned to. This technique assigns to each connected component of a binary image a distinct label. The labels are natural numbers starting from 1 to the total number of connected components in the image [Rosenfeld & Kak, 1976].

The X-Y tree decomposition [Nagy & Seth, 1984] uses Horizontal projection profile of the document image to extract the lines from the document. If the lines are well separated, the horizontal projection will have separated peaks and valleys, which serve as the separators of the text lines. These valleys are easily detected and used to determine the location of boundaries between lines (Figure 2.7). Similarly, gaps in the vertical projection of a *line image* are used in extracting words in a line, as well as extracting individual characters from the word. Overlapping, adjacent characters in a word (called *kerned* characters) cannot be segmented using zero-valued valleys of the vertical projection profile. Special techniques/heuristics have to be employed to solve this problem.

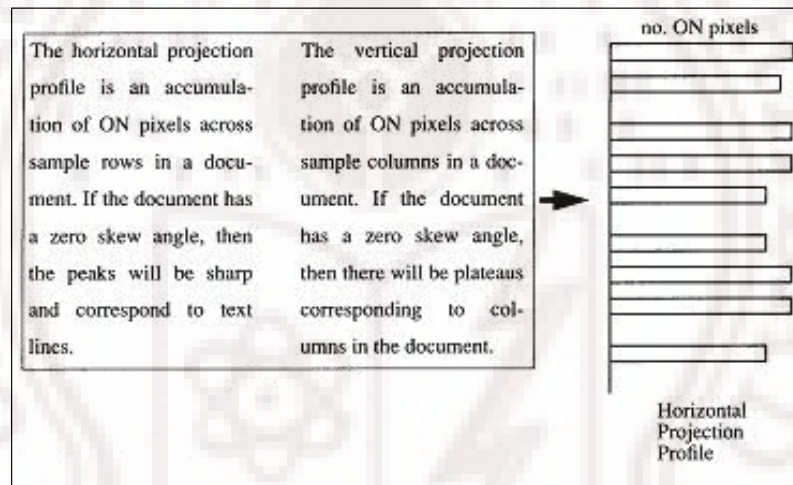


Figure 2.7 An image and its horizontal projection profile

Run-length smearing algorithm (RLSA) [Wong, Casey & Wahl, 1982] first detects all white runs (sequence of 1's) of the line. It then converts those runs whose length is shorter than a predefined threshold T , to black runs. To obtain segmentation, RLSA is applied line-by-line first; and then column-by-column, yielding two distinct bitmaps; these are then combined by a logical AND operation.

Hough-transform based methods, described in earlier section, doesn't require connected or even nearby edge points. They work successfully even when different objects are connected to each other.

2.2.1.5 Thinning

Thinning, or, skeletonization is a process by which a one-pixel-width representation (or the skeleton) of an object is obtained, by preserving the connectedness of the object and its end points [Gonzalez & Woods, 2002]. The purpose of thinning is to reduce the image components to their essential information so that further analysis and recognition are facilitated. For instance, an alphabet can be handwritten with different pens giving different stroke thicknesses, but the information presented is the same. This enables easier subsequent detection of pertinent features. Letter 'e' is shown in Figure 2.8 before and after thinning. A number of thinning algorithms have been proposed and are being used. The most common algorithm used is the classical Hilditch algorithm [Hilditch, C.J., 1969] and its variants. For recognizing large graphical objects with filled regions which are often found in logos, region boundary detection is useful, but for small regions such as those which correspond to individual characters, neither thinning nor boundary detection is performed, and the entire pixel array representing the region is forwarded to the subsequent stage of analysis [Kasturi et al, 2002].

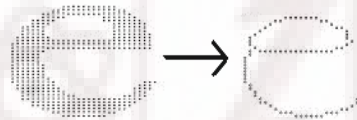


Figure 2.8 An image before and after thinning

2.2.2 Recognition

By character recognition, the character symbols of a language are transformed into symbolic representations such as ASCII, or Unicode. The basic problem is to assign the digitized character into its symbolic class. This is done in two steps: (i) Feature extraction, and selection, and (ii) classification; and is shown in Figure 2.9.

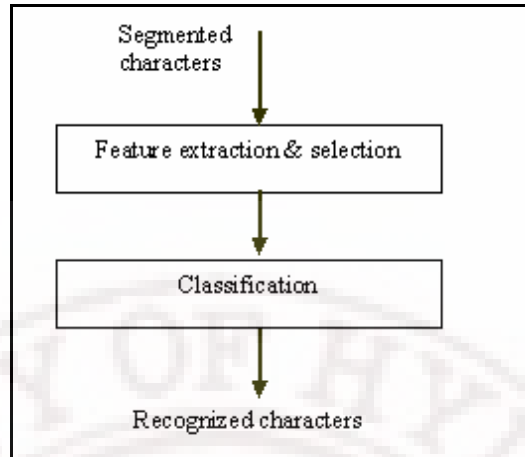


Figure 2.9 Recognition process

2.2.2.1 Feature extraction and selection

The heart of any optical character recognition system is the formation of feature vector to be used in the recognition stage. Feature extraction can be considered as finding a set of parameters (features) that define the shape of the underlying character as precisely and uniquely as possible. The term *feature selection* refers to algorithms that select the best subset of the input feature set. Methods that create new features based on transformations, or combination of original features are called *feature extraction* algorithms. However, the terms *feature selection* and *feature extractions* are used interchangeably in literature [Jain, Duin & Mao, 2000]. The features are to be selected in such a way that they help in discriminating between characters. Selection of feature extraction methods is probably the single most important factor in achieving high performance in recognition [Trier, Jain & Taxtt, 1996]. A large number of feature extraction methods are reported in literature; but the methods selected depend on the given application. There is no universally accepted set of feature vectors in document image understanding. Features that capture topological and geometrical shape information are the most desired ones. Features that capture the spatial distribution of the black (text) pixels are also very important [Bunke & Wang, 1997]. Hence, two types of approaches are defined as follows:

i. Structural approach

In structural approaches features that describe the geometric and topological structures of a symbol are extracted. Structural features may be defined in terms of character strokes, character holes, end points, intersections between lines, loops and other character attributes such as concavities. Compared to other techniques, structural analysis gives features with high tolerance to noise and style variations. However, these features are only moderately tolerant to rotation and translation [Line, 1993]. Structural approaches utilize structural features and decision rules to classify characters. The classifier is expected to recognize the natural variants of a character but discriminate between similar looking characters such as 'O' and 'Q', 'c' and 'e', etc.

ii. Statistical approach

In statistical approach, a pattern is represented as a vector: an ordered, fixed length list of numeric features [Jain, Duin, & Mao, 2000]. Many samples of a pattern are used for collecting statistics. This phase is known as the training phase. The objective is to expose the system to natural variants of a character. Recognition process uses this statistics for identifying an unknown character. Features derived from the statistical distribution of points include number of holes, geometrical moments, black-to-white crossing counts, width, height, and aspect ratio. Representation of a character image by statistical distribution of points takes care of style variations to a large extent.

Template matching is also one of the most common and oldest classification methods. In template matching, individual image pixels are used as features. Classification is performed by comparing an input character image with a set of templates (or prototypes) from each character class. The template, which matches most closely with the unknown, provides recognition. The technique is simple and easy to implement and has been used in many commercial OCR machines. However, it is sensitive to noise and style variations and has no way of handling rotated characters.

Statistical approach and structural approach both have their advantages and disadvantages. Statistical features are more tolerant to noise than structural descriptions provided the sample space over which training has been performed is representative and

realistic. The variation due to font or writing style can be more easily abstracted in structural descriptions. In hybrid approach, these two approaches are combined at appropriate stages for representation of characters and utilizing them for classification of unknown characters. Segmented character images are first analyzed to detect structural features such as straight lines, curves, and significant points along the curves. For regions corresponding to individual characters or graphical symbols, local features such as aspect ratio, compactness (ratio of area to square of perimeter), asymmetry, black pixel density, contour smoothness, number of loops, number of line crossings and line end points etc. are used.

Feature extraction techniques are evaluated based on: (a) robustness against noise, distortion, size, and style/font variation, (b) Speed of recognition, c) complexity of implementation, and d) how independent the feature set is without requiring any supplementary techniques [Line, 1993].

2.2.2.2. Classification

Classification stage in an OCR process assigns labels to character images based on the features extracted and the relationships among the features [Kasturi et al, 2002]. In simple terms, it is this part of the OCR which finally recognizes individual characters and outputs them in machine editable form. A number of approaches are possible for the design of classifier; and the choice often depends on which classifier is available, or best known to the designer. Decision-theoretic methods are used when the description of the character can be numerically represented in a feature vector. The principal approaches to decision-theoretic recognition are: minimum-distance classifiers, statistical classifiers and neural networks. Jain, Duin & Mao [Jain, Duin & Mao, 2000] identified the following two main categories.

1. The simplest approach is based on matching / identification of a similar neighbor pixel with the nearest distance. ‘Matching’ covers the groups of techniques based on similarity measures where the distance between the feature vector describing the extracted character and the description of each class is calculated. Different measures may

be used, but the common is the Euclidean distance. This minimum distance classifier works well when the classes are well separated, that is when the distance between the *mean* values is sufficiently large compared to the spread of each class.

When the entire character is used as input to the classification, and no features are extracted (template-matching), a correlation approach is used. Here the distance between the character image and prototype images representing each character class is computed.

A special type of classifier is Decision tree [Brieman et al., 1984] which is trained by an iterative selection of individual features that are most salient at each individual node of the tree.

Another category is to construct decision boundaries directly by optimizing certain error criterion. Examples are: Fisher's linear discriminant [Fisher, 1936; Chernoff & Moses, 1959; Fukunaga, 1990] that minimizes the Mean Squared Error (MSE) between classifier output and the desired labels.

Neural networks are another category. Considering a back-propagation neural network, the network is composed of several layers of interconnected elements. A feature vector enters the network at the input layer. Each element of the layer computes a weighted sum of its input and transforms it into an output by a nonlinear function. During training, the weights at each connection are adjusted until a desired output is obtained. A problem of neural networks in OCR may be their limited predictability and generality, while an advantage is their adaptive nature [Line, 1993]. Other approaches include Single layer and Multilayer Perceptrons [Raudys, 1998], and Support vectors [Vapnik, 1995].

2. The second main concept used in the classifier design is based on the probabilistic approaches such as Bayes decision rule and maximum likelihood rule [Bayes, 1763; Chow, 1957; Fukunaga, 1990]. The principle is to use a classification scheme that is optimal in the sense that, on average, it gives the lowest probability of making classification errors. A classifier that minimizes the total average loss is called the Bayes' classifier. Given an unknown symbol described by its feature vector, the probability that the symbol belongs to class c is computed for all classes $c = 1 \dots N$. The symbol is then assigned

the class which gives the maximum probability. For this scheme to be optimal, the probability density functions of the symbols of each class must be known, along with the probability of occurrence of each class.

Having dealt with the general techniques of character recognition, we now consider the importance of font recognition.

2.3 Font Consideration

Optical character recognition systems deal with the recognition of printed or handwritten characters. Printed characters may have various fonts and sizes. Typographically, a font is a particular instantiation of a typeface design, often in a particular size, weight and style [Rubinstein, 1988]. A font *family* (for example, Arial) includes several styles (plain, italic, bold); the angular slope given to vertical strokes would result into *italic* variation of the same font. Arial font family in italic, bold and plain styles is shown in Figure 2.10.

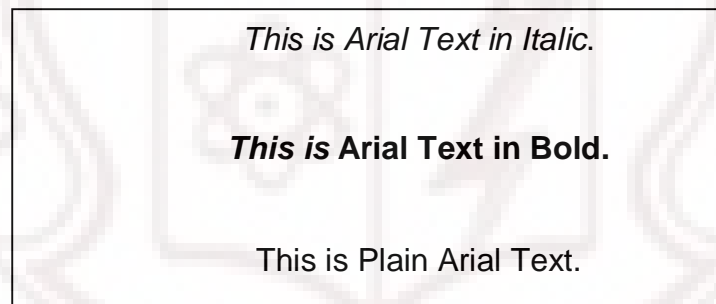


Figure 2.10 Arial font families in italic, bold and plain styles

The *type font family* can be identified through its stylistic renderings of elements, weights, transformation variations and sizes. For example: Times Roman have thin smooth serifs at the end of vertical strokes, so this stylistic feature will be observed in all its family members. Different styles of two font families are shown below for the character A.

A A A ---> Plain, Italic, and Bold styles in Times New Roman font family.

A A A ---> Plain, Italic, and Bold styles in Arial font family.

A font *face* (for example, Arial, Regular) is a complete set of a single *style*, in all sizes, as shown in Figure 2.11.



Figure 2.11 Arial font *face*

Font is an important factor for both character recognition and script identification. At present, many attempts were made to construct OCRs with reasonable accuracy, but only for limited fonts. The performance of these OCRs is expected to be well as long as the same font size and type are maintained. Since this requirement is not practical, often we get poor results. The recognition accuracy often drops significantly when a document printed in a different font is encountered.

A document reader must cope with many sources of variations notably that of font and size of the text. In commercial devices, the multi-font aspect was for a long time neglected for the benefit of speed and accuracy, and substitution solutions were proposed. At first, the solution was to work on customized fonts such as OCR-A and OCR-B. The accuracy was quite good even on degraded images on the condition that the font is carefully selected. However, recognition scores drop rapidly when fonts or sizes are changed. An optical character recognition system that works efficiently on documents that contain any font is highly desirable.

2.3.1 OCR Classification based on Fonts

Based on the OCR systems' capability to recognize different character sets, a classification [Line, 1993], by the order of difficulty is as follows.

2.3.1.1 Fixed font OCRs

OCR machines of this category deal with the recognition of characters in only one specific typewritten font. Examples of such fonts are OCR-A, OCR-B, Pica, Elite, etc. These fonts are characterized by fixed spacing between each character. The OCR-A and OCR-B are the American and European standard fonts specially designed for optical character recognition, where each character has a unique shape to avoid ambiguity with other characters similar in shape. Using these character sets, it is quite common for commercial OCR machines to achieve a recognition rate as high as 99.99% with a high reading speed. The first generation OCRs were fixed font machines, and the methods applied were usually based on template matching and correlation.

2.3.1.2 Multi-font OCRs

Multi-font OCR machines recognize characters from more than one font, as opposed to a fixed font system, which could only recognize symbols of one specific font. For the earlier generation OCRs, the limit in the number of recognized fonts was due to the pattern recognition algorithm used: template matching, which required that a library of bit map images of each character from each font was stored (requirement of a huge database). The accuracy is quite good, even on degraded images, as long as the fonts in the library are selected with care.

2.3.1.3 Omni font OCRs

An omni font OCR machine can recognize symbols of most non-stylized fonts without having to maintain huge databases of specific font information. Usually omni font-technology is characterized by the use of feature extraction. The database of an omni font system will contain a description of each symbol class instead of the symbols themselves. This gives flexibility in automatic recognition of characters from a variety of fonts. A

number of current OCR-systems for English claim to be omni font. Although omni font is the common term used for these OCRs, this does not mean that they recognize characters from all existing fonts.

Font classification can reduce the number of alternative shapes for each class, leading essentially to single-font character recognition [Zhu, Tan & Wang, 2001]. Following is the overview of approaches used in the literature for recognizing fonts in English.

2.3.2 Font Recognition approaches

There are basically two approaches used for font identification in English [Zramdini & Ingold, 1998]:

1. *A priori* font classification that identifies the font without any knowledge of the content of characters, and,
2. *A posteriori* font classification approach that classifies the font using the knowledge of the characters.

In the first one, global features are extracted from word/ line/ paragraph. These are the features that are generally detected by non-experts in typography (text density, size, orientation and spacing of the letters, serifs, etc.).

In the second approach, local features are extracted from individual characters. These features are based on letter peculiarities like the shapes of serifs and the representation of particular letters like ‘g’ and ‘g’, ‘a’ and ‘a’. This kind of approach may derive substantial benefit from the knowledge of the letter classes.

Zramdini and Ingold [Zramdini & Ingold, 1993] aim at the identification of the global typographical features such as font weight, typeface, slope and size of the text from an image block from a given set of already learned fonts. Jung, Shin and Srihari [Jung, Shin & Srihari 1999] proposed a font classification method based on the definition of typographical attributes such as ascenders, descenders and serifs, and the use of a neural network classifier.

Bazzi *et al.* [Bazzi *et al.*, 1997] focused on the problem of language independent recognition i.e., script-independent features are used. For example, a line is divided into a

sequence of cells, and features such as intensity, vertical derivative of intensity, horizontal derivative of intensity are used, but script-dependent features are not used.

In the method used by Khoubyari and Hull, [Khoubyari & Hull, 1996], clusters of word images are generated from an input document and matched to a database of function words derived from fonts and document images. The font or document that matches best provides the identification of the predominant font and function words.

Khorsheed and Clocksin [Khorsheed & Clocksin, 2000] transformed each word into a normalized polar image, and a two-dimensional Fourier transform is applied to the polar image. The resultant spectrum tolerates variations in size, rotation or displacement.

Allier and Emptoz [Allier & Emptoz, 2003] treated the printed document as texture at character level. Multi-channel Gabon filtering approach is used. Each filter is applied to the original textured image, and a simple feature vector is created using statistical calculations. Classification is done using Bayesian theory.

Shi and Pavlidis [Shi & Pavlidis, 1997] use a hybrid font recognition approach that combines an *a priori* approach and an *a posteriori* approach. Page properties such as histogram of word length and stroke slopes are used for font feature extraction. This font information is extracted from either the entire page, or, some selected short words like a, an, am, as. This system utilizes contextual information at word-level.

Following a similar method, Zhu, Tan and Wang [Zhu, Tan & Wang, 2001] considered document as an image containing some specific textures and regarded font recognition as texture identification. The original image is preprocessed to form a uniform block of text. A block of text printed in each font can be seen as having a specific texture. The spatial frequency and orientation contents represent the features of each texture. These texture features are used to identify different fonts. Multi-channel Gabor filtering technique is used to extract features from this uniform text block. A weighted Euclidean distance classifier is used to identify the fonts.

2.4 Indian language OCRs

At present, reasonably efficient and inexpensive OCR packages are commercially available to recognize printed texts in widely used languages such as English. These

systems can process documents that are typewritten, or printed. While a large amount of literature is available for the recognition of Roman, Chinese and Japanese language characters, relatively less work is reported for the recognition of Indian language scripts. Nevertheless, under the aegis of TDIL Programme, thirteen Resource Centers for Indian Language Technology Solutions (RCILTS) have been established at various educational institutes and R&D organizations covering all Indian Languages [JLT, July 2003]. Under this program, OCRs, human-machine interface systems and other tools are being developed in different Indian languages. Thus, OCR systems for Indian scripts have just started appearing. A brief summary of the techniques used in these OCRs as given in the news letter [JLT, October 2003] along with other reported works are described in this section. A detailed performance reports as given in Language Technology Products Testing Reports from July 2004 news letter of TDIL [JLT, July 2004] are also presented in this section.

2.4.1 Techniques used in different Indian script OCRs

We now present the studies in OCRs of different Indian scripts, along with a detailed description of the methods used in them.

2.4.1.1 Devnagari OCR

OCR work on printed Devnagari script started in 1970s. Sinha and Mahabala [Sinha & Mahabala, 1979] presented a syntactic pattern analysis system with an embedded picture language for the recognition of handwritten and machine printed Devnagari characters. For each symbol of the Devnagari script, the system stores structural description in terms of primitives and their relationships.

Pal and Chaudhuri [Pal & Chaudhuri, 1997] reported a complete OCR system for printed Devnagari. In this, headline deletion is used to segment the characters from the word. Text lines are divided into three horizontal zones for easier recognition procedure. After preprocessing, and segmentation using zonal information and shape characteristics; the basic, modified and compound characters are separated. A structural feature-based tree classifier recognizes modified and basic characters, while compound characters are

recognized by a tree classifier followed by template matching approach. The method reports about 96% accuracy.

Bansal [Bansal, 1999] described Devnagari OCR in her doctoral thesis. Here, segmentation is done using a two-stage, hybrid approach. The initial segmentation extracts the header line, and delineates the upper strip from the rest. This yields vertically separated character boxes that could be conjuncts, touching characters, shadow characters, lower modifiers or a combination of these. Segmentation is done based on structural information obtained from boundary traversal in the second stage. Vertical bar features, horizontal zero crossings; number and position of vertex points, and moments, etc are used in the classification. For a feature, the distances from the reference prototypes for the candidate characters are computed; a classifier based on the distance matching is employed for recognition. An error detection and correction phase is also included as post processing. Performance of 93% accuracy at character level is reported.

Problems that arise in developing OCR systems for noisy images are addressed in the work by Iyer et al., [Iyer et al, 2005]. Lines are segmented into word-like units, based on the dips in the vertical projection profile of the line. Some statistical data such as minimum and average widths, height, etc, are computed. Basic geometrical shapes such as full vertical bar, a horizontal line, diagonal lines in both the orientations, circles and semicircles of varying radii, and orientations are used to form the feature vector. Characters are classified using a rule-based approach. The rule base consists of more than one rule for a given character to account for different font-specific representations of the same character. Hamming distance metric is employed for classification. Character recognition rate of only 55% is reported. The authors also trained a feed-forward back propagation neural network, with a single hidden layer. Character recognition rate of 76% is reported with this neural network approach.

2.4.1.2 Bangla OCR

Ray and Chatterjee presented a recognition system based on a nearest neighbor classifier employing features extracted by using a string connectivity criterion [Ray & Chatterjee, 1984].

Chaudhuri and Pal reported a complete OCR for printed Bangla [Chaudhuri & Pal, 1998], in which a combination of template and feature-matching approach is used. A histogram-based thresholding approach is used to convert the image into binary images. Skew angle is determined from the skew of the headline. Text lines are partitioned into three zones and the horizontal and vertical projection profiles are used to segment the text into lines, words, and characters. Primary grouping of characters into the basic, modified and compound characters is made before the actual classification. A few stroke features are used for this purpose along with a tree classifier where the decision at each node of the tree is taken on the basis of presence/absence of a particular feature. Recognition of compound characters is done in two stages: In the first stage, characters are grouped into small sub-sets by the above tree classifier. At the second stage, characters in each group are recognized by a run-based template matching approach. Some character level statistics like individual character occurrence frequency, bi-gram and tri-gram statistics etc. are utilized to aid the recognition process. For single font, clear documents 99.1% character level recognition accuracy is reported.

Recognition of isolated and continuous printed multi-font Bengali characters is reported by Mahmud et al., [Mahmud et al., 2003]. This is based on Freeman-chain code features, which are explained as follows. When objects are described by their skeletons or contours, they can be represented by chain coding, where the ON pixels are represented as sequences of connected neighbors along lines and curves. Instead of storing the absolute location of each ON pixel, the direction from its previously coded neighbor is stored. The chain codes from center pixel are 0 for east, 1 for North- East, and so on. This is represented pictorially in Figure 2.12 (a) and (b). Chain code gives the boundary of the character image; slope distribution of chain code implies the curvature properties of the character. In this work, connected components from each character are divided into four regions with the center of mass as the origin. Slope distribution of chain code, in these four regions is used as local feature. Using chain code representation, classification is done by a feed forward neural network. Testing on three types of fonts with accuracy of approximately 98% for isolated characters and 96% for continuous characters is reported.

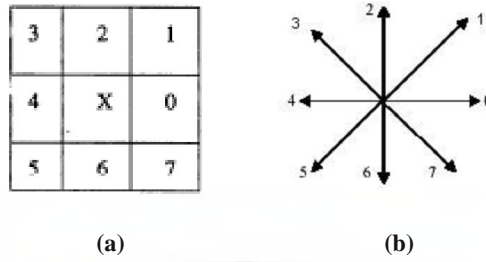


Figure 2.12 Chain code, and graphical representations

2.4.1.3. Gurmukhi (Punjabi) OCR

Lehal and Singh presented an OCR system for printed Gurumukhi script [Lehal & Singh, 2000]. The skew angle is determined by calculating horizontal and vertical projections at different angles at fixed interval in the range 0° to 90° . The angle, at which the difference of the sum of heights of peaks and valleys is maximum is identified as the skew angle. For line and word segmentation horizontal and vertical projection profiles are respectively used. Each word is segmented into connected components or sub-symbols, where each sub-symbol corresponds to the connected portion of the character lying in one of the three zones. Connected components are formed by grouping together black pixels having 8-connectivity. Primary feature set is made up of features which are expected to be font and size invariant such as number of junctions with the headline equals 1, presence of sidebar, presence of a loop, and loop along the headline. The secondary feature set is a combination of local and global features: number of endpoints and their location, number of junctions and their location, horizontal projection count, right profile depth, left profile depth, right and left profile directions, and aspect ratio. Binary tree classifier is used for primary features, and the nearest neighbor classifier with a variant sized vector was used for the secondary features. This multi-stage classifier is used to classify the sub -symbols and they are then combined using heuristics and finally converted to characters. A recognition rate of 96.6% was reported.

Lehal and Singh also developed a post processor for Gurmukhi [Lehal & Singh, 2002]. In this, statistical information of Punjabi language such as word length, shape of the words, frequency of occurrence of different characters at specific positions in a word,

information about visually similar-looking words, grammar rules of Punjabi language, and heuristics are utilized.

2.4.1.4. Telugu OCR

The first reported work on OCR of Telugu Characters is by Rajasekaran and Deekshatulu [Rajasekaran & Deekshatulu, 1977]. It identifies 50 primitive features and proposes a two-stage syntax-aided character recognition system. A knowledge-based search is used in the first stage to recognize and remove the primitive shapes. In the second stage, the pattern obtained after the removal of primitives is coded by tracing along points on it. Classification is done by a decision tree. Primitives are joined and superimposed appropriately to define individual characters.

Rao and Ajitha utilized the characteristic feature of Telugu characters as composing of circular segments of different radii [Rao & Ajitha, 1995]. Recognition consists of segmenting the characters into the constituent components and identifying them. Feature set is chosen as the circular segments, which preserve the canonical shapes of Telugu characters. The recognition scores are reported as ranging from 78 to 90% across different subjects, and from 91 to 95% when the reference and test sets were from the same subject.

Sukhaswami, Seetharamulu and Pujari proposed a neural network based system [Sukhaswami, Seetharamulu & Pujari, 1995]. Hopfield model of neural network working as an associative memory is chosen for recognition purposes initially. Due to the limitation in the storage capacity of the Hopfield neural network, they later proposed a Multiple Neural Network Associative Memory (MNNAM). These networks work on mutually disjoint sets of training patterns. They demonstrated that storage shortage could be overcome by this scheme.

Negi, Bhagvati and Krishna reported an OCR for Telugu [Negi, Bhagvati & Krishna 2001]. Instead of segmenting the words into characters as usually done, words are split into connected components (glyphs). Run Length Smearing Algorithm and Recursive XY Cuts [Nagy, Seth & Vishwanathan, 1992] methods are used to segment the input document image into words. About 370 connected components (depending on the font) are identified

as sufficient to compose all the characters including punctuation marks and numerals. Template matching based on the fringe distance [Brown, 1994] is used to measure the similarity or distance between the input and each template. The template with the minimum fringe distance is marked as the recognized character. The template code of the recognized character is converted into ISCII, the Indian Standard Code for Information Interchange. Raw OCR accuracy with no post processing is reported as 92%.

Pujari, Naidu and Jinaga proposed a recognizer that relies on wavelet multi-resolution analysis for capturing the distinctive characteristics of Telugu script [Pujari, Naidu & Jinaga, 2002]. Gray level input text images are line-segmented using horizontal projections; and vertical projections are used for the word segmentation. Images are uniformly scaled to 32x32 sizes using zero-padding technique. Wavelet representation with three levels of down-sampling reduces a 32x32 image into a set of four 8x8 images, of which only an average image is considered for further processing. Character images of size 8x8 are converted to binary images using the mean value of the grey level as the threshold. The resulting bit string of 64 bits is used as the signature of the input symbol. A Hopfield-based Dynamic Neural Network is designed for the recognition purpose. For 444 number of testing patterns, the system correctly recognized 415, giving a recognition rate of 93.46%. The authors reported that the same system, when applied to recognize English characters, resulted in very low recognition rate since the directional features that are prevalent in Latin scripts are not preserved during signature computation with wavelet transformation.

Lakshmi and Patvardhan presented recognition of basic Telugu symbols [Lakshmi & Patvardhan, 2003]. Each character (basic symbol) is resized to 36 columns, while maintaining the original aspect ratio. A preliminary classification is done by grouping all the symbols with approximately same height (rows). Feature vector is computed out of a set of seven invariant moments from the second and third order moments. Recognition is done using k-nearest neighbor algorithm on these feature vectors. A single font type is used for both training and test data. Testing is done on noisy character images with Gaussian noise, salt and pepper noise and speckle noise added. Preprocessing such as line, word, and character segmentation is not addressed in this work. The authors extended the work to

multi font OCR [Lakshmi & Patvardhan, 2002]. Preprocessing stages such as binarization, noise removal, skew correction using Hough transform method, Lines and words segmentation using horizontal and vertical projections are included in this work. Basic symbols from each word are obtained using connected components approach. After preliminary classification as in the previous work, pixel gradient directions are chosen as the features. Recognition is done again using the k-nearest neighbor algorithm on these feature vectors. The training vectors are created with three different fonts and three sizes: 25, 30 and 35. Testing is done on characters with different sizes, and also with some different fonts. Recognition accuracy of more than 92% for most of the images is claimed.

DRISHTI is the OCR for Telugu language developed by the Resource Center for Indian Language Technology Solutions (RCILTS), at the University of Hyderabad [JLT, July 2003]. The techniques used in Drishti are as follows: For binarization three options are provided: global (the default), percentile based and iterative method. Skew Detection and Correction are done by maximizing the variance in horizontal projection profile. Text and Graphics Separation is done by horizontal projection profile. Multi-column Text Detection is done using Recursive X-Y Cuts technique. It is based on recursively splitting a document into rectangular regions using vertical and horizontal projection profiles alternately. Word segmentation is done using a combination of Run-Length Smearing Algorithm (RLSA) and connected-component labeling. Words are decomposed into glyphs by running the connected component labeling algorithm again. Recognition is based on template matching using fringe distance maps. The template with the best matching score is output as the recognized glyph.

A semi-automatic, adaptive OCR is developed by Rawat, et al., for recognition of Indian Languages [Rawat, et al., 2006]. Features in the Eigenspace are used for classification using a Support Vector Machine (SVM). Principal Component Analysis (PCA) is used for feature dimensionality reduction. To resolve confusing characters during the recognition process, a resolver module that uses language-specific information is designed. Postprocessor is based on contextual information and language-based reverse dictionary approach to correct any wrongly recognized words. Performance of the prototype system is tested on datasets taken from four Indian languages: Hindi, Telugu, Tamil and

Malayalam. Accuracies of 97.15% and 93.14% are reported on good-quality data, and degraded data respectively for Telugu.

2.4.1.5. Gujarati OCR

Antani and Agnihotri described recognition of Gujarati characters [Antani & Agnihotri, 1999]. Subsets of similar-looking Gujarati characters were classified by different classifiers: Euclidean Minimum Distance classifier, Nearest Neighbor classifier were used with regular and invariant moments, and the Hamming Distance classifier was also used in the binary feature space. However, a low recognition rate of 67% is reported.

A working prototype of Gujarati OCR is developed by Maharaja Sayajirao University, Baroda [JLT, July 2003]. It employs the template matching technique for recognition and nearest neighbor technique for classification. The input image is assumed to be skew-corrected, with one column of text only. Output is in Unicode format as plain text file. Recognition accuracy of initial results is reported to be good, but not specified.

2.4.1.6. Oriya OCR

The features of Oriya OCR developed at the Indian Statistical Institute, Kolkata [Chaudhuri, Pal & Mitra, 2002] are similar to the Bangla OCR developed by the same team [JLT, July2003], and are as follows:

Scanning resolution is at 200 to 300 dots per inch (dpi). Histogram-based thresholding approach is used to convert the images into two-tone images. The threshold value is chosen as the midpoint between the two peaks of the histogram. Hough transform based technique is used for estimating the skew angle using only the uppermost and lowermost pixels of each component. The lines of a text block are segmented by finding the valleys of the horizontal projection profile. Oriya text lines are partitioned into three zones: lower zone contains only modifiers and the *halant* marker, while the upper zone contains modifiers and portions of some basic characters. After line segmentation, the zones in each line are detected. Vertical projection profile is used for word segmentation. To segment each word into individual characters, the image is scanned in the vertical direction starting

from the mean line of the word. If during a scan, the base line without encountering any black pixel is reached, and then this scan marks the boundary between two characters. To segment the touching characters, principle of water overflow from a reservoir [Garain & Chaudhuri, 2002] is used. After preprocessing, individual characters are recognized using a combination of stroke and run-number based features, along with features obtained from the concept of water overflow from a reservoir. Topological features, stroke-based features as well as features obtained from the concept of water overflow are considered for character recognition. Stroke-based features like the number and position of vertical lines are used for the initial classification of characters in a tree classifier. The topological features used include existence of holes and their number, position of holes with respect to the character bounding box, and ratio of hole- height to character height, etc. Water reservoir features include position of the reservoirs with respect to the character bounding box, the height of each reservoir, the direction of water overflow, etc. In the first stage, the characters are grouped into small subsets by a feature based tree classifier. In the second stage, characters in each group are recognized using a matching approach based on number of black runs (a black run is a set of black pixels with white pixel at either end). On an average, the system reports an accuracy of about 96.3%.

Mohanty and Behera described a complete OCR development system for Oriya script [Mohanty & Behera, 2004]. For skew detection and correction, angular projection profiles are prepared for -8^0 to $+8^0$ with an interval of 0.05^0 and a strip-wise histogram is constructed using these projection profiles. A global maximum at a particular angular direction gives the global skew angle; the whole document is then rotated to remove the skew. Structural features such as upper-part circular, a vertical line on the right most part, holes, horizontal run code, vertical run code, number and position of holes, are extracted from the 16x16 pixel matrix. A tree-based classifier is used in which each node denotes a particular feature. All leaf nodes contain individual characters, modifiers (*matras*), digits and composite characters. The recognition phase has two parts: In the first phase individual characters, and left and right modifiers are recognized based on structural features; whereas in the second stage upper and lower modifiers are recognized based on run length code. Recognition accuracy is not mentioned.

Resource Center for Indian Language Technology Solutions (RCILTS) for Oriya is established at Utkal University, Bhubaneswar [JLT, October 2003]. The features of the OCR, DIVYADRUSTI developed at Utkal University are as given below.

Binarization is done using dynamic thresholding technique. Repeated angular projection profiles are used for skew detection. Lines are extracted through strip-wise vertical histogram analysis. Character segmentation is achieved using region growing and labeling, and *matra* extraction is done using region analysis. Connected components are handled by forward and backward chaining of appropriate mask. A tree-based classification method is used in which each node denotes a particular feature and all leaf nodes contain individual characters, modifiers, digits and composite characters.

2.4.1.7. Assamese OCR

The features of Assamese OCR are based on the Bangla OCR developed at Indian Statistical Institute, Kolkata [JLT, October 2003], and are as given below.

Histogram based global threshold approach is used for binarization. Skew detection and correction is done using the method proposed by Chaudhary and Pal, by finding the headline of the Assamese script in the document image [Chaudhary & Pal, 1998]. Words are segmented using the connected component analysis. Segmenting individual characters from a word is by deleting the headline from the word. Simple stroke features like vertical and horizontal lines, horizontal and vertical black runs are used. Two types of classifiers are used: Classifier-1 detects simple stroke features like vertical and horizontal lines. This is designed to separate the basic characters, modifiers and compound characters. Characters like punctuation marks, special marks like quotes are also recognized by this classifier. Classifier-2 extracts features from individual characters by counting the horizontal and vertical black runs. Recognition is done by calculating distance (dissimilarity) measure between character and stored prototypes. In the post processing stage dictionary match and morphological analyzer are used to select the correct word from the set of alternatives.

2.4.1.8. Kannada OCR

Ashwin and Sastry developed a font and size-independent OCR for Kannada [Ashwin & Sastry, 2002]. Text page is binarized using a global threshold computed automatically. Skew correction is done by a windowed Hough transform technique. Line and word segmentation are done by projection profile based methods. For segmentation, the words are first split into three vertical zones based on the horizontal projection for the word. The three zones are then horizontally segmented using their vertical projections. A character is segmented into its constituents, i.e. the base consonant, the vowel modifier and the consonant conjunct.

Features of Kannada OCR developed at RCILTS, Indian Institute of Science, Bangalore are as follows [JLT, July 2003]. Input image scanning is done at 300 dpi. Binarization is done using global threshold. Hough transform technique is used in skew detection and correction. Line and Word Segmentation are based on projection profiles. Words are segmented into sub-character level so that each akshara may be composed of many segments. Distribution of the ON pixels in the radial and the angular directions are extracted to capture the rounded shape of the Kannada characters. Classification is based on the Support Vector Machines.

2.4.1.9. Tamil OCR

Siromony, Chandrasekaran and Chandrasekaran described a method for recognition of machine printed letters of the Tamil alphabet using an encoded character string dictionary [Siromony, Chandrasekaran & Chandrasekaran, 1978]. The scheme employs string features extracted by row and column-wise scanning of the character matrix. The features in each row /column are encoded suitably depending upon the complexity of the script to be recognized.

Chinnuswamy and Krishnamoorthy proposed an approach for hand-printed Tamil character recognition [Chinnuswamy & Krishnamoorthy, 1980]. Here, the characters are assumed to be composed of line-like elements called primitives, satisfying certain relational constraints. Labeled graphs are used to describe the structural composition of characters in

terms of the primitives and the relational constraints satisfied by them. The recognition procedure consists of converting the input image into a labeled graph representing the input character and computing correlation coefficients with the labeled graphs stored for a set of basic symbols. The algorithm uses topological matching procedure to compute the correlation coefficients and then maximizes the correlation coefficient.

Aparna and Chakravarthy detailed a complete OCR for Tamil magazine documents [Aparna & Chakravarthy, 2002]. Radial basis function neural network is used for separating text and graphics. For skew correction, cumulative scalar products (CSP) of windows of the text boxes at different orientations with the Gabor filters are computed. Orientation with the maximum CSP gives the skew. Ostu's method is used for binarization. Line segmentation using horizontal projection is employed. Inclined projections are used for segmenting lines into words and characters. A Radial basis function neural network is trained for character recognition. Response of 40 Gabor filters with 10 filters in each of the 4 directions is computed. The recognition accuracy is reported to be varying from 90-97 %.

Tamil OCR developed at Indian Institute of Science, Bangalore [Aparna & Ramakrishnan, 2001; JLT, July 2003] has the following features: Input image scanning is done at 300 dpi into a binary image. Skew detection and correction are achieved through Hough transform and Principal Component Analysis. Horizontal and vertical projection profiles are employed for line and word detection, respectively. Connected component analysis is performed to extract the individual characters. The segmented characters are normalized to predefined size and thinned before recognition phase. Depending on the spatial spread of the characters in the vertical direction, they are grouped into 4 classes. These classes are further divided into groups based on the type of ascenders and descenders in the characters. Second order moments are employed as features to perform this grouping. Truncated Discrete Cosine transform (DCT) based features are used for the final classification with a nearest neighbor classifier. Recognition accuracy of 98% on a sample size of 100 is reported.

2.4.1.10. Malayalam OCR

NAYANA is the Malayalam OCR developed at C-DAC, Thiruvananthapuram [JLT, July 2003]. Binarization is done using histogram based thresholding approach (Otsu's algorithm). Skew detection is done using the projection profile based technique. Linguistic rules are applied to the recognized text in the post processing module to correct classification errors. Recognition accuracy of 97% for good quality printed documents is reported.

2.4.2 Performance Testing

In order to address the issues of quality in the language technology products developed at the RCILTs (described in Section 2.4), Department of Information Technology (DIT) initiated a project entitled- *Software Quality Engineering* in Indian Language Products [JLT, July 2004]. The OCRs developed at the RCILTs are tested for functionality, usability and portability. In all cases of testing, text from newspapers, books and laser printed output were used as test inputs. Standardization Testing and Quality Certification (STQC) division of DIT was designated as the third party for evaluation of the language technology tools and products developed at TDIL programme. Performance results of the OCRs developed at the RCILTs and tested by STQC are given in the news letter of TDIL [JLT, July 2004], and are tabulated below in Table 2.1. We made a comparison on the feature sets and classifiers used in the OCRs developed at the RCILTs. This is summarized in Table 2.2.

Table 2.1 Performance results of OCRs developed at the RCILTs

OCR Name	Developed by	Platform	Font size	Accuracy [§]	Speed (CPS)	Skew angle correction	Input format	Output file format	Supports	Resolution
Bangla:	ISI,Kolkata	DOS	14-36 pts.	30-96%	13 to 47.	+/- 5 °.	TIFF	PC	Single column, and also bold characters, but not symbols.	300 dpi
Gurmukhi	TIET, Patiala,	WIN 9X.	12-20 pts.	93-99%	12 to 59	+/- 5 °	BMP	Punjabi font in Word.	Single column, and bold characters, but not symbols.	300 dpi
Tamil	IISC, Bangalore	WIN 95/98.	12-18 pts.	62-98%	1 to 19	+/- 10 °.	BMP,PGM	TAB, RTF.	Single column, also bold characters, & special symbols.	300 dpi
Devnagari	ISI Kolkota,	WIN95(Min) / Linux.	10-24 pts.	48-97%	20-44	+/- 5 °.	TIFF	PC	Single column, and bold characters also, but not special symbols.	>=300 dpi
Hindi/ Marathi:	CDAC, Pune	WIN 95X/NT/2000.	10-36pts	85-98%	93-368.	+/- 5 °.	BMP, JPG, TIFF	RTF UNICODE/ ISCII	Multi column. And bold characters, but not special symbols	270-320
Telugu	DCIS, University of Hyderabad	Linux.	12-20 pts.	84-87%	23-29	NM*	PGM.	ACI	Single column.	300 dpi.
Hindi	CDAC, Noida,	Win 98/2000.	12-36 pts.	5-96%	9-188	+/-5 °.	TIFF,BMP	RTF	Single column, and bold characters, but not special symbols.	280-320 dpi.
Oriya	Utkal University, Bhubaneswar	Windows	NM*	74-86%	NM*	NM*	BMP	TXT	single column text	300dpi.
Malayalam	CDAC, Thiruvanthapuram	windows	NM*	93-97%	NM*	+/-5 °.	BMP, TIFF, JPEG and GIF	RTF,HTML, ACI , TXT	multiple font sizes, formats, multicolumn layouts containing both text & images	300 dpi.

[§] Depending on input

*NM=Not mentioned

Table 2.2 Feature sets and classifiers of different OCRs

OCR	Developed by	Feature Set	Classification
Assamese	ISI, Kolkata	Simple stroke features like vertical and horizontal lines, horizontal and vertical black runs	2-stage classifiers- Similarity matching using distance measure
Bangla:	ISI, Kolkata	Structural features	Basic and modified characters by tree classifier; compound characters by a tree classifier followed by template matching
Gurmukhi	TIET, Patiala,	Structural features	Hybrid classification scheme consisting of binary decision trees and nearest neighbors
Kannada	IISC, Bangalore	Distribution of black pixels in radial and the angular directions	Support Vector Machines
Tamil	IISC, Bangalore	Number of dots, second order geometric moments, Truncated Discrete Cosine Transform (DCT) coefficients	Three level, tree structured classifier & nearest neighbor classifier.
Devnagari	ISI Kolkota,	Structural features	Tree classifier followed by template matching
Telugu	DCIS, University of Hyderabad	Template matching technique	Similarity matching using fringe distance measure
Gujarathi	Maharaja Sayajirao University of Baroda, Vadodara	Template matching technique	Nearest neighbor
Oriya	Utkal University, Bhubaneswar	Structural features	Tree-based
	ISI Kolkota,	Topological features, stroke and black runs, water reservoir features	Feature-based tree classifier in first stage, run-number based matching approach in second stage

From this literature survey, it can be noted that OCR research for Indian scripts is a very active field currently. Also, several aspects like feature dimensionality reduction, and font recognition still have scope. In the next section, we present a detailed justification for selecting the present research topic.

2.5 Justification for the present work

For the Indian languages, OCR systems are not yet able to successfully recognize printed document images with varying style and font; and also, there were no known studies for font recognition. Thus, OCR for Telugu is an emerging area of research, and recognizing Telugu fonts and characters simultaneously from printed documents is a novel attempt, which is the main focus of the present work.

The following salient features explain the novelty of the present work and justifies its importance in the field of character recognition.

2.5.1 Identification of fonts

A scanned document image may contain words in different font typefaces and styles. In many occasions, different font styles are used to emphasize certain words scattered in the normal text. Font recognition is a fundamental issue in document analysis and recognition, and is often a difficult and time-consuming task. Often, the font style is not the same for a whole document; it is a word feature, rather than a document feature, and its detection can be used to discriminate between different regions of the document, such as title, Figure caption, or normal text [Abuhaiba, 2005].

Also, the orthographic complexity of Telugu script makes Telugu characters not ready for directly using with OCR. Since mono-font OCR systems achieve better results than multi-font ones, it is a good idea to use an Optical Font Recognition (OFR) system first to determine the font types in it, and then, use the appropriate mono-font OCR for text recognition. So, our font recognition system can also be used as a preprocessing step in a fixed-font OCR. To our knowledge, no attempts were made earlier to recognize Telugu fonts. Developing a methodology for simultaneously recognizing font types and characters from Telugu printed documents is a novel attempt in our work.

2.5.2 Dimensionality reduction

Feature selection plays the most important factor in achieving high performance in any OCR. From the above survey on Indian OCRs, it can be understood that feature set reduction was attempted by a very few researchers only. Reducing the feature dimensionality is important because it helps in reducing the time and space complexities of the system. With a small feature set, it becomes possible to apply very efficient methodologies in the design of the classifier. The orthographic complexity of Telugu characters makes the feature selection process a complicated one. We explored and demonstrated the usefulness of the theory of rough sets in feature dimensionality reduction in Telugu OCRs. The present model emphasizes the role of rough sets in feature selection and data reduction in Telugu font and character recognition. Thus, our feature reduction techniques are based on the strong mathematical principles, which is quite different from the earlier works.

2.5.3 Applicability to other languages

Features used in the present work are not dependent on the underlying script of any one particular language. It does not use any domain knowledge. The methodology adapted here is equally applicable to OCR of any language script. This makes the proposed approach language-independent. Thus, the present methodology paves way for designing a framework of a general OCR for recognizing characters as well as fonts used in a document, irrespective of the script in which it is written. The applicability of the approach is also successfully demonstrated on the font and character recognitions for English.

Chapter 3

Feature Selection: A Mathematical Foundation

Feature selection is the process of finding a subset of features, from the original set of pattern features, optimally according to some pre-defined criterion. An optimal feature selection thus guarantees the accomplishment of a processing goal while minimizing feature set. Typically, the main aim of feature selection is to determine a minimal feature subset from a problem domain while retaining a suitably high accuracy in representing the original features [Jensen & Shen, 2007]. In an OCR application, it is appropriate to select features which are invariant and which capture the essential information of the character by filtering out all the other features that cause changes in the appearance of the character.

In this chapter, a new approach for feature selection in Telugu OCR, using the theory of rough sets, is presented.

3.1 Feature Selection

Features may be irrelevant (having no effect on the processing performance), or relevant (having an impact on the processing performance). Features may have different discriminatory powers. By removing most irrelevant and redundant features from the data, feature selection helps improving the performance of learning models.

Typical Feature Selection architecture works as a *generate-and-test* iterative algorithm: Generated feature (sub) sets are iteratively evaluated for a stopping criterion and are generated again, until the stopping criterion is met, at which point the subset is validated. This is illustrated in Figure 3.1 (adapted from Dash & Liu, 1997).

1. The *Subset Generation* procedure implements a search method that generates subset of features. It may start with a) no features, b) all features, c) a selected feature set or d) some random feature subset. Those methods that start with an initial subset usually

select these features heuristically beforehand. Features are added (*forward selection*) or removed (*backward elimination*) iteratively in the first three cases. In the last case, features are either added or removed or produced randomly thereafter.

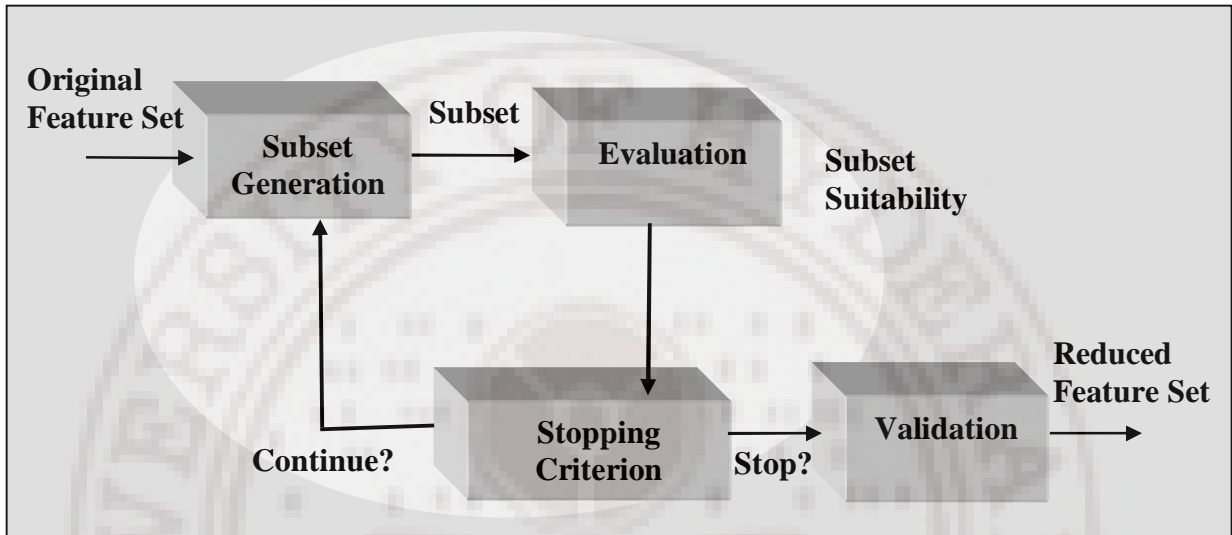


Figure 3.1 Feature Selection Process

2. The *Evaluation* function calculates the suitability of a feature subset produced by the generation procedure and compares this with the previous best candidate, replacing it if found to be better.
3. A *Stopping criterion* is tested at each iteration step to determine whether the feature selection process should continue or not. Once the stopping criterion has been satisfied, the loop terminates. One example criterion to halt the selection process could be when a pre-decided number of features have been selected.
4. The resulting subset of features may then be *validated* for optimality. Determining subset optimality is a challenging problem. There is always a trade-off between subset *minimality* and subset *suitability*. For some domains, particularly where the classification rate using the selected features must be extremely high, even a non-minimal set of features would be acceptable. In other areas where it is expensive or

impractical to maintain many features, it is much more desirable to have a smaller, less accurate feature subset.

3.2 Data dimensionality

In general, if the performance obtained with a given set of features is inadequate, new features that will help separate the confused pairs may be added. In pattern recognition applications, adding new features is a common practice to improve the performance. Although increasing the number of features increases the cost and complexity of both the feature extractor and the classifier, it is often reasonable to believe that the performance will improve [Duda, Hart & Stork, 2000]. The number of attributes/features in an application domain is termed as *dimensionality*. The amount of data required for training the system, and the amount of computation required for recognition grows exponentially with the increase of dimensionality of the feature vector. Thus, the time and space complexities of most learning algorithms greatly increase with the dimension of the input, which is called the “Curse of dimensionality” [Bellman, 1957]. According to this, the dataset size needed to approximate a multivariate function grows exponentially with the number of variables of the function. As an example, assume that ten samples are required for each attribute to train a machine-learning system. A similar, two-dimensional domain will require 10^2 samples to sample the plane. An n -dimensional domain is likely to require 10^n samples, rapidly leading to insurmountable storage and training problems.

Data dimensionality is an obstacle for both the training and runtime phases of a learning system. Many systems exhibit non-polynomial complexity with respect to dimensionality. The curse of dimensionality effectively limits the applicability of learning systems to only small and well-analyzed domains. Very elegant methodologies also become incapable of performing satisfactorily on arbitrary domains due to the curse of dimensionality [Shen & Chouchoulas, 2001]. Hence, it is necessary to reduce the dimension of the original data prior to any modeling.

The aim of feature dimension reduction techniques is to capture the essential information (for discrimination) of the high dimensional feature vector into a set of smaller number of ‘useful’ features. The usefulness of a feature or feature subset is determined by

both its *relevancy* and *redundancy*. A feature is said to be *relevant* if it is predictive of the decision feature(s), otherwise it is irrelevant. A feature is considered to be *redundant* if it does not contribute any additional knowledge than is possible with other features. Hence, the search for a good feature subset involves finding those features that are *not* redundant with respect to the decision feature(s), as well as other selected features.

A straightforward technique of best feature selection could be choosing a minimal feature subset that fully describes all the concepts (for example, classes in classification) in a given data set. The idea of feature selection, with the minimum concept criterion, can be extended using the concept of *reducts* defined in the theory of rough sets introduced by Zdzislaw Pawlak [Pawlak, 1982].

Basic concepts of rough sets, and feature selection using rough sets are described in the following section.

3.3 Rough sets based Feature selection

Rough set theory (RST) was proposed by Professor Pawlak for knowledge discovery in databases and experimental data sets. RST is a formal methodology that can be employed to reduce the dimensionality of datasets as a preprocessing step in training a learning system on the data. The approach is highly efficient, relying on simple set operations, which makes it suitable as a preprocessor for techniques that are more complex.

3.3.1 Theory of rough sets

Philosophers, logicians and mathematicians have tackled the problem of imperfect knowledge for a long time. Recently it became also a crucial issue for computer scientists, particularly in the area of artificial intelligence. There are many approaches to the problem of how to understand and manipulate imperfect knowledge. Fuzzy set theory proposed by Zadeh is one successful approach [Zadeh, 1965]. Rough set theory presents yet another attempt to this problem. Rough set theory can be viewed as a specific implementation of idea of vagueness, i.e., imprecision in this approach is expressed by a boundary region of a set, and not by a partial membership, like in fuzzy set theory.

Basically, rough set theory deals with the approximation of sets that are difficult to describe with the available information. Rough set theory found many interesting applications, and is of fundamental importance to AI and cognitive sciences; especially in the areas of machine learning, knowledge representation, knowledge acquisition, pattern recognition, decision analysis, knowledge discovery from databases, inductive reasoning and expert systems. Rough sets have been found to be particularly useful for rule induction and feature selection. Rough sets have been used for missing-data estimation, as well as for understanding HIV. Rough sets based tools find applications in the following fields [Busse, 1995].

- Global warming,
- Assessing pre-term labor risk for pregnant women.
- Diagnosis of melanoma,
- Prediction of behavior under mental retardation,
- Analysis of animal models for prediction of self-injurious behavior,
- Nursing,
- Natural language, and
- Data transmission.

Basic concepts in Rough sets Theory are now explained in the following section.

3.3.1.1 Information Systems

Data are often presented as a table in which, each row represents an event or an object; each column represents an attribute, or an observation that can be measured for each object. Entries of the table represent attribute values. Such tables are known as “information systems”, or “information tables”. Table 3.1 shows an example information system in which information about 8 sample characters are represented by their features. “Charcode” is the character code; attributes are represented in the columns labeled F1, F2, F3, and F4, which are the features of the character image such as width, height, and density of black pixels etc.,. Thus each row of the table can be seen as information about a specific sample. For example, for sample 1, the features (F1=1), (F2=0), (F3=2), (F4=2), and its code is C0.

An information system can be represented as

$$S = \langle U, Q, V, f \rangle \dots\dots\dots(3.1)$$

where U is the universe, which is a finite set of N objects (x_1, x_2, \dots, x_N), Q is a finite set of attributes. With every attribute $q \in Q$, we associate a set V_q of its *values*, called the *domain* of q .

Table 3.1 An example information system

<i>Samples</i>	<i>F1</i>	<i>F2</i>	<i>F3</i>	<i>F4</i>	<i>Charcode</i>
1	1	0	2	2	C0
2	0	1	1	1	C2
3	2	0	0	1	C1
4	1	1	0	2	C2
5	1	0	2	0	C1
6	2	2	0	1	C1
7	2	1	1	1	C2
8	0	1	1	0	C1

$$V = \bigcup_{q \in Q} V_q \dots\dots\dots(3.2)$$

The mapping $f : (U \times Q) \rightarrow V$ is the *information function*, or the *total decision function* such that $f(x, q) \in V_q$ for every $q \in Q$, and $x \in U$. For example, in the above information system given in Table 3.1, $f(2, F1) = 0$, and $f(5, F3) = 2$.

In many applications there is an outcome of classification that is known. This *a posteriori* knowledge is expressed by some distinguished attributes called “*decision attributes*”. The other attributes are referred to as *condition attributes*. Thus, if an information table contains two classes of attributes called *decision attributes* (D) and *condition attributes* (C), such a table is called *decision table* [Ohrn & Rowland, 2000]. A decision table is an information system where $Q = (C \cup D)$.

In Table 3.1, the outcome of the attribute *Charcode* depends on the features of the sample character. For any sample, if ($F1=1$), and ($F2=0$), and ($F3=2$), and ($F4=2$), then, its *Charcode*=C0. From rows 2, 4, and 7,

If $\left([(F1=0), \text{ and } (F2=1), \text{ and } (F3=1), \text{ and } (F4=1)], \text{ OR } [(F1=1), \text{ and } (F2=1), \text{ and } (F3=0), \text{ and } (F4=2)], \text{ OR } [(F1=2), \text{ and } (F2=1), \text{ and } (F3=1), \text{ and } (F4=1)] \right)$ then its $charcode = C2$.

Similar decisions can be inferred from the other rows. Each row of a decision table determines a *decision rule*, which specifies decisions (*actions*) that should be taken when conditions pointed out by condition attributes are satisfied. Thus, in a decision table, each row is a rule, and each column in that row is either a condition or action for that rule. Thus, this table has four condition attributes (F1, F2, F3, and F4), and one decision attribute *Charcode*.

3.3.1.2 Indiscernibility

A decision table expresses all the knowledge about the model. This table may be unnecessarily large in part because it is redundant in at least two ways: The same or indiscernible objects may be represented several times (redundant rows), or some of the attributes may be superfluous (redundant columns). We often face a question whether we can remove some data from a data table, while preserving its basic properties. The notion of *indiscernibility* is fundamental to rough set theory. Informally, two objects in a decision table are indiscernible if one cannot distinguish between them on the basis of a given set of attributes. Elements that exhibit the same information are indiscernible (similar) and form blocks. An indiscernibility relation partitions the set of cases or objects into a number of *equivalence classes*. In Table 3.1, for example, samples 2, 7, 8 are indiscernible with respect to the attributes *F2*, and *F3*; and also samples 1, and 5. Indiscernibility, and our ability to form set approximations, depends on which columns or attributes in the data table that we consider.

The indiscernibility relation is defined as follows:

A subset of attributes $A \subseteq Q$ defines an *equivalence relation* (called an *indiscernibility relation, IND*) on U :

$$IND(A) = \{(x, y) \in U : \text{for all } a \in A, f(x, a) = f(y, a)\} \quad \dots\dots\dots(3.3)$$

Thus, $(x, y) \in IND(A)$ if and only if $f(x, a) = f(y, a)$ for all $a \in A$.

Obviously $IND(A)$ is an equivalence relation. The family of all equivalence classes of $IND(A)$ i.e., partition determined by A , will be denoted by $U/IND(A)$, or simply U/A .

Thus, in Table 3.1, if $A = \{F2, F3\}$, the partitions created by A on U are: $U/IND(A) = \{\{1,5\}, \{2,7,8\}, \{3\}, \{4\}, \{6\}\}$.

The equivalence classes discussed above form the basic building blocks to define *set approximations*, the fundamental concepts of rough set theory.

3.3.1.3 Set Approximations

Rough sets approximate traditional sets by using a pair of sets, namely the lower and upper approximations of the set in question. With every rough set we associate two crisp sets, called its lower and upper approximation. Let $X \subseteq U$ be a subset of the universe.

Intuitively, the lower approximation of a set consists of all elements that surely belong to the set, whereas the upper approximation of the set constitutes of all elements that *possibly* belong to the set. The difference of the upper and the lower approximation is a *boundary region*. The *boundary region* of a set X with respect to U is the set of all objects, which can be classified neither as X nor as *not-X* with respect to U . It consists of all elements that cannot be classified uniquely to the set or its complement, by employing available knowledge. Thus any rough set, in contrast to a crisp set, has a non-empty boundary region.

The *lower approximation* of a set of objects (with respect to a given set of attributes) is the collection of objects whose equivalence classes are *fully* contained in the set of objects we want to approximate.

$$\underline{P}X = \bigcup \{Y \in U/IND(P) : Y \subseteq X\} \dots\dots\dots (3.4)$$

For example, in Table 3.1, $U/IND(P) = \{\{1, 5\}, \{3\}, \{2, 7, 8\}, \{4\}, \{6\}\}$
 $= \{\{1, 5\}, \{2, 7, 8\}, \{3\}, \{4\}, \{6\}\}$

The *lower approximation* of samples $X = \{2, 4, 7\}$ where $charcode = "C2"$, with respect to $P = \{F2, F3\}$ is $\underline{P}X = \{4\}$.

The *upper approximation* of a set of objects (with respect to a given set of attributes) is defined as the collection of objects whose equivalence classes are at least partially contained in (i.e., overlap with) the set of objects we want to approximate.

$$\overline{P} X = \bigcup \{Y \in U / IND(P) : Y \cap X \neq \Phi\} \quad \dots\dots\dots (3.5)$$

In Table 3.1, the *upper approximation* $\overline{P} X = \{ \{2,7,8\}, \{4\} \} = \{2,4,7,8\}$. If the lower and upper approximations are equal, then the set of interest can be defined crisply (Exact). A *rough set* is any set defined through its lower and upper approximations. Now we give the definition of rough sets.

Let U be any finite universe of discourse. Let R be any equivalence relation defined on U. Clearly, the equivalence relation R partitions U. (U, R) is the collection of all equivalence classes, which is called the approximation space. Let W_1, W_2, \dots, W_n be the elements of the approximation space (U,R). This collection is called *knowledge base*. For any subset A of U, the lower and upper approximations respectively are defined as follows:

$$\left. \begin{aligned} \underline{R}A &= \bigcup \{W_i / W_i \subseteq A\} \\ \overline{R}A &= \bigcup \{W_i / W_i \cap A \neq \Phi\} \end{aligned} \right\} \dots\dots\dots (3.6)$$

In general, $\underline{R}A \subseteq A \subseteq \overline{R}A$. If $\underline{R}A = \overline{R}A$ then A is called *exact*. The lower approximation of A is called the positive region of A and is denoted by $POS(A)$ and the complement of upper approximation of A is called the negative region of A and is denoted by $NEG(A)$. Its boundary is defined as $BND(A) = \overline{R}A - \underline{R}A$. Hence, it is trivial that if $BND(A) = \Phi$, then A is exact. The ordered pair $(\underline{R}A, \overline{R}A)$ is called a rough set [Ganesan & Rao, 2006].

- Set A is *crisp* (exact with respect to R), if the boundary region of A is empty.
- Set A is *rough* (inexact with respect to R), if the boundary region of A is nonempty. Thus a set is *rough* (imprecise) if it has nonempty boundary region; otherwise the set is *crisp* (precise). In the above considered example of Table 3.1,

$$POS(X) = \{4\},$$

$$NEG(X) = \text{Complement of } (\{2, 4, 7, 8\}) = \{1, 3, 5, 6\}, \text{ and}$$

$$\text{BND}(X) = \{2, 4, 7, 8\} - \{4\} = \{2, 7, 8\}.$$

Since the boundary region for X is non-empty, the set X is Rough with respect to $P \{F2, F3\}$, and its representation is $(\{4\}, \{2, 4, 7, 8\})$.

3.3.1.4 Reduction of Attributes

In many cases, not all the measured data are important for understanding the underlying phenomena of interest. One natural dimension of reducing data is to identify equivalence classes. i.e., objects that are indiscernible using the available attributes. Since only one element of the equivalence class is needed to represent the entire class, we obtain the reduction [Komorowski, Polkowski & Skowron, 1999].

It is often the case that some of the attributes are superfluous. Certain attributes in a decision table may be redundant and can be eliminated without losing essential classificatory information. The other dimension in reduction is to keep only those attributes that preserve the indiscernibility relation and, consequently, set approximation. The rejected attributes are redundant since their removal cannot worsen the classification. The main challenge in inducing rules from decision tables lies in determining which attributes should be included in the conditional part of the rule. This is where the notion of “*reducts*” comes to our rescue. Given a dataset with discretized attribute values, it is possible to find a subset (called *reduct*) of the original attributes that possesses most of the information; all the other attributes are redundant, and hence can be removed from the dataset without any loss of information. In other words, a reduct is the minimal subset of attributes, and remaining attributes that do not belong to a reduct are superfluous with regard to classification of elements of the universe [Pawlak, 1982].

A subset B of the attribute set A is a reduct if and only if $IND(B) = IND(A)$ and B is minimal with this property, i.e.,

$$IND(B - \{a\}) \neq IND(A) \text{ for all } a \in B. \dots\dots\dots(3.7)$$

Reducts enable us to discard redundant information. Thus, Rough sets provide a method to determine the most important attributes of a database from the classificatory power point of view.

Let B be a subset of A and let ‘ a ’ belongs to B .

- We say that a is *dispensable* in B if $IND(B) = IND(B - \{a\})$; otherwise a is *indispensable* in B .
- Set B is *independent* if all its attributes are indispensable.
- Subset B_I of B is a *reduct* of B if B_I is independent and $IND(B_I) = IND(B)$.

The set of attributes, which is common to all reducts, is called the **core**. The core is the set of attributes that *cannot be removed* from the information system without causing collapse of the equivalence class structure. The rejected attributes are redundant since their removal cannot worsen the classification. In practical terms, reducts help us construct smaller and simpler models, and provide a focus on the decision-making process. Typically, a decision table may have many reducts.

3.3.2 Computing Reducts

We are typically interested in reducing the information down to minimal sets of attributes, called *reducts*. In terms of computational complexity and memory requirements, calculating all possible subsets of a given set is an NP-hard task. There are two main approaches in finding rough set reducts:

- i) Those that consider the degree of dependency, and
- ii) Those that consider the discernibility matrix.

First one incorporates the degree of dependency measure (or extensions), and the second approach applies heuristic methods to generated discernibility matrices.

3.3.2.1 Dependency Function-based Approaches

A set of attributes D depends totally on a set of attributes C , denoted $C \Rightarrow D$, if all values of attributes from D are uniquely determined by values of attributes from C . In other words, D depends totally on C , if a functional dependency exists between values of D and C . If only some values of D are determined by values of C , it is called partial dependency.

RST introduces a measure of dependency of two subsets of attributes. Let P and Q be equivalence relations over U , then the positive region can be defined as:

$$POS_P(Q) = \bigcup_{X \in U/Q} PX \dots\dots\dots(3.8)$$

The positive region contains all objects of U that can be classified to classes of U/Q using the information in attributes P . For example in Table 3.1, let $P = \{F2, F3\}$ and $Q = \{Charcode\}$, then

$$POS(\{1\}) = \Phi$$

$$POS(\{2, 4, 7\}) = \{4\},$$

$$POS(\{3, 5, 6, 8\}) = \{3, 6\}$$

$$POS_P(Q) = \bigcup \{\Phi, \{4\}, \{3, 6\}\} = \{3, 4, 6\}.$$

Using this definition of the positive region, the rough set *degree of dependency* of a set of attributes Q on a set of attributes P , denoted by $\gamma_P(Q)$ is defined in the following way:

$$\begin{aligned} \gamma_P(Q) &= \text{cardinality}^1(POS_P(Q)) / \text{cardinality}(U) \dots\dots\dots(3.9) \\ &= 3/8 = 0.375. \end{aligned}$$

The degree of dependency $\gamma_P(Q)$ provides a measure of how important P is in mapping the dataset examples into Q . If $\gamma_P(Q) = 0$, then classification Q is independent of the attributes in P , hence the attributes P are of no use to this classification. If $\gamma_P(Q) = 1$, then Q is completely dependent on P , hence the attributes are indispensable. Values $0 < \gamma_P(Q) < 1$ denote partial dependency, which shows that only some of the attributes in P may be useful.

In the above example, a value of 0.375 shows that out of the eight samples, only three (sample 3, 4, and 6) can be mapped unambiguously into *Charcode*, given conditional attributes *F2* and *F3*. The other five samples represent ambiguous information.

Reduction of attributes is achieved by comparing equivalence relations generated by sets of attributes. Attributes are removed so that the reduced set provides the same predictive capability of the decision feature as the original. A reduct R is defined as a subset of minimal cardinality of the conditional attribute set C such that $\gamma_R(D) = \gamma_C(D)$.

¹ **Cardinality** of a set is a measure of the number of elements of the set.

Generating and processing power sets is an NP-hard task, experiencing combinatorial explosion. This makes an exhaustive search for reducts infeasible. In order to reduce computational complexity and memory requirements, the reduct search may be implemented as a Hill-climbing task by the QUICKREDUCT algorithm [Jensen & Shen, 2007] given in Figure 3.3. It attempts to calculate a reduct from the decision table which is available globally, without exhaustively generating all possible subsets. The reduct subset search space is treated as a tree traversal. Each node of the tree represents the addition of one input attribute to an initially empty reduct set R . Instead of generating the whole tree and picking the best path on it, the path is chosen progressively. Starting with the empty set, attributes are chosen and progressively added to the candidate reduct using the following heuristic: the next attribute chosen is the attribute that adds the most to the reducts dependency.

```

QUICKREDUCT( $C, D$ )
   $C$ , the set of all conditional features;
   $D$ , the set of decision features.

(1)  $R \leftarrow \{\}$ 
(2) do
(3)    $T \leftarrow R$ 
(4)    $\forall x \in (C - R)$ 
(5)     if  $\gamma_{R \cup \{x\}}(D) > \gamma_T(D)$ 
(6)        $T \leftarrow R \cup \{x\}$ 
(7)    $R \leftarrow T$ 
(8) until  $\gamma_R(D) = \gamma_C(D)$ 
(9) return  $R$ 

```

Figure 3.3 Quick Reduct algorithm

QuickReduct is a forward searching hill climbing algorithm. It starts off with an empty set R and adds in turn, one at a time, those attributes that result in the greatest increase in the rough set dependency metric, until this produces its maximum possible value for the dataset. The hill climb ends when the dependency reaches one, or when no more variables are left. In doing so, QuickReduct converts an otherwise exhaustive search of the

entire variable power set into a best-first search. According to this algorithm, the dependency of each attribute is calculated, and the best candidate is chosen. Referring to Table 3.1, attribute $F4$ generates the highest dependency degree, therefore that attribute is chosen and the sets $\{F1, F4\}$, $\{F2, F4\}$ and $\{F3, F4\}$ are evaluated. This process continues until the dependency of the reduct equals the consistency of the dataset (if the dataset is consistent). In the example, the algorithm terminates after evaluating the subset $\{F2, F4\}$.

QuickReduct, however, is not guaranteed to find a *minimal* subset. But, it does result in a close-to-minimal subset, though, which is still useful in greatly reducing dataset dimensionality [Jensen & Shen, 2007].

3.3.2.2 Discernibility Matrix-based Approaches

Many applications make use of *Decision-Relative Discernibility* matrices for finding reducts. If $D = (U, C \cup d)$ is a decision table where C = set of conditional attributes, and d is the decision attribute, the Decision-Relative discernibility matrix of D is a symmetric $|U| \times |U|$ matrix defined as:

$$C_{ij} = \{a \in C \mid a(x_i) \neq a(x_j) \quad \text{when} \quad d(x_i) \neq d(x_j)\} \quad i, j = 1, 2, \dots, |U|$$

.....(3.10)

The entry C_{ij} is the set of all attributes, which discern objects x_i and x_j when the corresponding decision attributes differ. For example, it can be seen from Table 3.1 that samples 1 and 2 differ in *Charcode*, and also in all the four attributes $F1, F2, F3, F4$. Hence, $C_{21} = \{F1, F2, F3, F4\}$.

Although some attributes in samples 2 and 4 differ, their corresponding decision attributes are the same; so no entry appears in the discernibility matrix. $C_{42} = \Phi$. Pair-wise object discernibilities are calculated in this manner, and are entered in Table 3.2.

Grouping all entries containing single attributes forms the core of the dataset (those attributes appearing in *every* reduct). In this example, the core of the dataset is $\{F4\}$. From this, the discernibility function can be defined.

Table 3.2 Discernibility matrix for Table 3.1

Samples	1	2	3	4	5	6	7	8
1								
2	F1,F2,F3,F4							
3	F1,F3,F4	F1,F2,F3						
4	F2,F3		F1,F2,F4					
5	F4	F1,F2,F3,F4		F2,F3,F4				
6	F1,F2,F3,F4	F1,F2,F3		F1,F2,F4				
7	F1,F2,F3,F4	F2,F3		F1,F2,F3,F4	F2,F3			
8	F1,F2,F3,F4	F4	F1,F3,F4				F1,F4	

The *discernibility function* f_D is a concise notation of how each object within the dataset may be distinguished from the others. A discernibility function f_D is a boolean function of m boolean variables a_1, \dots, a_m defined as below:

$$f_D(a_1, \dots, a_m) = \bigwedge_{\substack{1 \leq j \leq i \leq |U| \\ C_{ij} \neq \emptyset}} C_{ij}^* \quad \dots \dots \dots (3.11)$$

where $C_{ij}^* = \vee \{a \mid a \in C_{ij}\} \quad \dots \dots \dots (3.12)$

All the minimal reducts of a system may be determined by finding the set of all *prime implicants* of the discernibility function.

For the above example, $C_{21}^* = (F1 \vee F2 \vee F3 \vee F4)$, and $C_{42}^* = \Phi$, and so on. From Table 3.2, we can write the discernibility function (after removing duplicates) as:

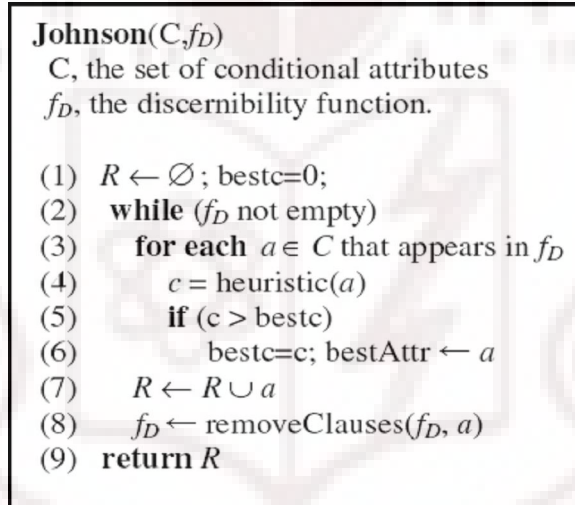
$$f_D(F1, F2, F3, F4) = \{ F1 \vee F2 \vee F3 \vee F4 \} \wedge \{ F1 \vee F3 \vee F4 \} \wedge \{ F2 \vee F3 \} \wedge \{ F4 \} \wedge \\ \{ F1 \vee F2 \vee F3 \} \wedge \{ F1 \vee F2 \vee F4 \} \wedge \{ F2 \vee F3 \vee F4 \} \wedge \{ F1 \vee F4 \}$$

Further simplification yields

$$f_D(F1, F2, F3, F4) = \{ F2 \vee F3 \} \wedge \{ F4 \} \\ = \{ F2 \wedge F4 \} \vee \{ F3 \wedge F4 \}.$$

Hence, the reducts are $\{F2, F4\}$ and $\{F3, F4\}$. Although this procedure guarantees discovering all minimal subsets, it is a costly operation, rendering the method impractical even for medium-sized datasets.

A simple greedy heuristic algorithm that is often applied to discernibility functions to find a single reduct is *Johnson Reducer* [Øhrn, 1999], given in Figure 3.4. Reducts found by this process have no guarantee of minimality, but are generally of a size close to the minimal.



```

Johnson( $C, f_D$ )
   $C$ , the set of conditional attributes
   $f_D$ , the discernibility function.

  (1)  $R \leftarrow \emptyset$ ; bestc=0;
  (2) while ( $f_D$  not empty)
  (3)   for each  $a \in C$  that appears in  $f_D$ 
  (4)      $c = \text{heuristic}(a)$ 
  (5)     if ( $c > \text{bestc}$ )
  (6)       bestc=c; bestAttr  $\leftarrow a$ 
  (7)    $R \leftarrow R \cup a$ 
  (8)    $f_D \leftarrow \text{removeClauses}(f_D, a)$ 
  (9) return  $R$ 

```

Figure 3.4 Johnson Reducer algorithm

The algorithm begins by setting the current set of reduct candidates R to the empty set. Then, each conditional attribute appearing in the discernibility function is evaluated according to the heuristic measure. For the standard Johnson Reducer algorithm, the typical heuristic measure is the count of the number of appearances an attribute makes within clauses; attributes that appear more frequently are considered to be more significant. The attribute with the highest heuristic value is added to the reduct candidate; and all clauses in the discernibility function containing this attribute are removed. As soon as all clauses have

been removed, the algorithm terminates and returns the reduct R . R is assured to be a reduct as all clauses contained within the discernibility function have been addressed. However, no perfect heuristic exists, and hence there is still no guarantee of subset optimality [Jensen, & Shen, 2007].

3.4 Discussion on Dimensionality Reduction Techniques

There are numerous methodologies to perform this dimension reduction task. Principal component analysis (PCA), Projection Pursuits and Factor Analysis are some of the widely used dimension reduction methods. A survey on the dimension reduction methods published in the statistics, signal processing and machine learning literature can be found in [Fodor, 2002]. A brief comparison of Rough Set-based Attribute Reduction (RSAR) with PCA is outlined here.

3.4.1 Principal Components Analysis (PCA)

Principal component analysis is a well-known tool for data analysis and transformation. Informally, PCA works by maximizing the variance of data sample vectors along their axes by locating a new co-ordinate system and transforming suitably the samples. The new axes are constructed in order of decreasing variance, so that the first variable in the new, transformed data has the most variance. Correlations between variables in the former sample space are reduced or altogether removed in the new one, hence redundancies are also reduced. In essence, PCA seeks to reduce the dimension of the data by finding a few orthogonal linear combinations (the PCs) of the original variables with the largest variance. The first PC is the linear combination with the largest variance. The second PC is the linear combination with the second largest variance and orthogonal to the first PC, and so on. There are as many PCs as the number of the original variables. For many datasets, the first several PCs explain most of the variance, so that the rest can be disregarded with minimal loss of information. It is thus possible to reduce the dimensionality of a dataset by transforming it with PCA and then selecting an appropriate number of principal components (variables), starting with the first one. All remaining transformed variables are discarded.

PCA has certain intrinsic shortcomings: it irreversibly destroys the underlying semantics of the dataset, prohibiting the use of PCA as a pre-processor for symbolic or descriptive fuzzy modeling approaches. By implication, only purely numerical (non-symbolic) datasets may be processed by PCA. It is only able to deal with linear projections, ignoring any nonlinear structure in the data. Finally, PCA does not guarantee that the selected first principal components will be the most adequate ones for classification. Human intervention on an application-specific basis is required to decide how many of the principal components are to be retained in the sample space. Thus, the operator's task is to balance information loss against dimensionality reduction to suit the task at hand.

3.4.2 Rough Set-based Attribute Reduction (RSAR)

RSAR works on discretized data by selecting the most information-rich features in a dataset. It attempts this task by neither transforming the data, nor removing information needed for the classification task. RSAR offers a number of advantages:

1. It preprocesses datasets without altering the attribute values themselves, thus maintaining the semantics.
2. It is not a *lossy* algorithm; it removes an input attribute from the dataset only if that action removes absolutely no information (with respect to the classification).
3. RSAR reduces the number of measurements required to train a classifier.
4. The dimensionality reduction does not require a human input, or the setting of variance thresholds.

However, RSAR also suffers from two problems. First one is the scalability problem when dealing with very large dimensionality datasets (of more than approximately 80,000 variables). Secondly, as with any non-adaptive learning system, RSAR assumes that the training data is representative of all future unseen information —this is rarely the case in real world applications where datasets comprise uncertain data due to noise or other defects.

PCA requires that the input features to be continuous in nature, whereas RSAR works on discrete data. Since the main features used in the present work are discrete ones, RSAR is our natural choice for feature dimensionality reduction.

3.5 Feature Dimensionality Reduction in Telugu OCR

Telugu is one of the major scheduled languages of India. It is the official language of Andhra Pradesh and second widely spoken language in Tamilnadu, and Karnataka. The total size of Telugu character set is very large due to compound characters that result from the combination of basic characters and modifiers. This orthographic complexity of Telugu script affects especially the feature selection process, and makes it a complicated one. In this chapter, we present the orthography of Telugu script, complexity of feature selection process in Telugu OCR, and the role of rough sets in reducing the feature dimensionality that aids an efficient design of a character recognition system.

3.5.1 Telugu Script Origin & Development

Telugu is written in Telugu script which is derived from Ashokan Brahmi used in the South India circa 2nd A.D. The Southern Brahmi also known as *Draavidi Brahmi* of 2nd c. A.D. gave rise to *Veengi-Chaalukyan* script, also known as Telugu-Kannada script. By the end of 13th Century A.D., the Telugu and Kannada scripts got separated. Unlike in the Roman alphabet used for English, in the Telugu alphabet the correspondence between the symbols (graphemes) and sounds (phonemes) is more or less exact [JLT, April 2000].

Telugu script consists of 51 letters of which, 16 are vowels (*Achchu*), and 35 are consonants (*hallus*) (Figures 3.5 and 3.6). In addition to the vowels and consonants, several semi-vowel symbols called *maatras* or *Vowel modifiers* (Figure 3.7) are used in conjunction with *hallus*. Half-consonants called *voththus*, (or) *Consonant modifiers* (Figure 3.8) are used in consonant clusters. A *hraswakshara* is a Consonant without any Maatra. Figure 3.9 shows some of the *Hraswaksharas* in Telugu.

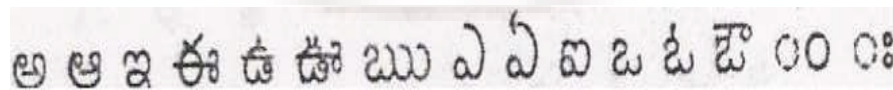


Figure 3.5 Telugu Vowels

Figure 3.6 Telugu Consonants

Figure 3.7 *Maatras / Vowel modifiers*

Figure 3.8 **Sample *Votthus* / Consonant modifiers**

Figure 3.9 Some *Hraswaksharas*

67

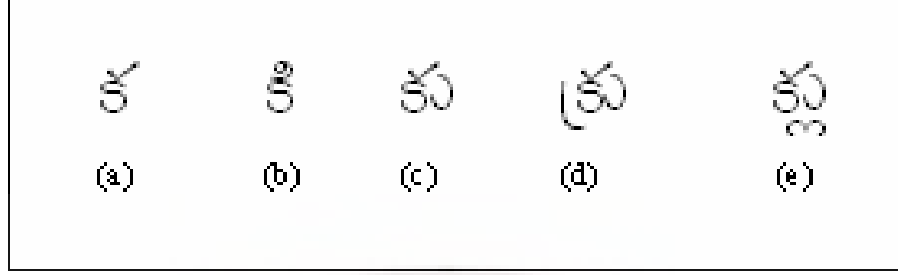


Figure 3.10 A character and its modifiers
 (a) A character (b), (c), (d), and (e) modifiers on top; right; left and right; and right and bottom

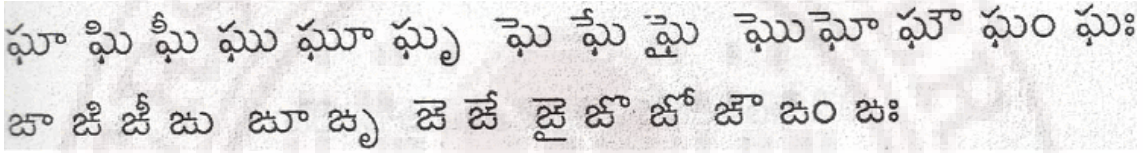


Figure 3.11 Some compound characters

3.5.2 Complexity of recognition in Telugu

Achchus, *hallus*, *maatras* and *voththus* together provide roughly 100 basic orthographic units that are combined together in different ways to represent all the frequently used syllables (estimated between 5000 and 10000) in the language [Negi, Bhagvati & Krishna, 2001]. Approximate calculations given below show how large the character set in Telugu script.

Number of possible consonant(C)–vowel (V) combinations (CV form) are $36 \times 16 = 576$.

Number of possible CCV combinations are $36 \times 36 \times 16 = 20736$.

Number of combinations with a basic character and a single Voththu = 18375

Number of combinations with a basic characters and two Voththus = 5, 83, 125

Number of combinations with a basic characters and three Voththus = 2, 24, 09, 375

These calculations are by taking all the 3 combinations into account. In actual script, however, very rarely we use three levels of Voththus, but two levels of Voththus are very common. Also, there are some combinations that never appear in the language. Even if we

eliminate all these cases, the total number of different character shapes that exist commonly is more than 5 lakhs [Venkat Rao, et al, 2006]. Thus, Telugu script is structurally complicated due to the large set of character shapes that can be formed. This is in direct contrast with the simple alphabetic structure of English, which has no modifiers.

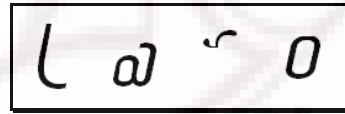
One of the major problems in building a Telugu OCR is thus the large number of recognition classes. Because of the many variations in the position and form of modifiers, the feature selection stage in the character recognition poses several complications. With about 5, 00,000 character shapes in general use, it is a complex task to build a system that can efficiently and reliably recognize far more classes than are required for Latin text.

So, in order to reduce the orthographic complexity, it is reasonable to treat a single connected entity as a basic unit of segmentation and recognition in Telugu OCR. A single connected component may be any individual or multiple glyphs, which appear connected in the script.

In English, each character/symbol in the alphabet is a separate entity. That is, each character is a connected component in English, whereas a complete Telugu character (*Akshara*) is usually constituted by one or more connected components as explained above. For example, the single *Akshara* in Figure 3.12 (a) is formed by the 4 connected components shown in Figure 3.12 (b):





(a) A Telugu *akshara*



(b) connected components

Figure 3.12 A Telugu *akshara* and its connected components

This connected component approach had been used by earlier researchers also [Negi, Bhagvati & Krishna 2001; Lakshmi & Patvardhan, 2003; Jawahar, et al., 2003] for reducing the complexity in segmentation and recognition. About 370 connected components are identified by Negi, Bhagvati and Krishna as sufficient to compose all the normally used characters in Telugu script [Negi, Bhagvati & Krishna 2001]. Lakshmi and Patvardhan identified 386 connected components for Telugu [Lakshmi & Patvardhan,

2003]. To some extent, this number depends on the font used in the document as explained now. Some characters constitute more connected components in one font than in other fonts. As an example, consider a character  in Srilipi font, which occurs as a single character, where as in Hemalatha font, it is written as  which is a combination of two connected components.

In the present work also, connected components are considered as the basic recognizable units. Henceforth, the term “character” is used to mean a connected component in the present thesis.

3.5.3 Feature Selection in Telugu OCR

Feature selection is probably the most important factor in achieving high performance in any OCR. A large number of feature selection methods are reported in literature; but the methods selected depend on the given application. Except in the work by Jawahar et al. [Jawahar, et al., 2003], no earlier attempts were found to reduce feature dimensionality in Telugu OCR problem. In their work, Principal Component Analysis (PCA) is used for dimensionality reduction.

It is found that zero-crossing features are often used in commercial OCR systems because they can be computed at high speed and require low complexity [Line, 1993]. They are frequently used in Chinese, Japanese, and other Asian character recognition systems because they demonstrated high recognition abilities. Zero-crossing-counts are also called *black-to-white transition* counts. They are defined as the number of times a character shape is crossed by vectors along certain directions. Figure 3.13 shows horizontal crossing counts at some selected rows for the character “Ka”. In the first row, there is only one black-to-white crossing; and in next row, there are 2 crossings, and so on.

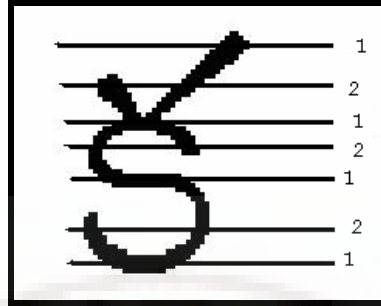


Figure 3.13 Horizontal crossing counts for the character "Ka"

Merits of zero- crossing features are:

- * They capture the shape information,
- * Preserve the shape information irrespective of the changes in character size,
- * Takes care of variations in font styles to a large extent,
- * They can be computed at high speed,
- * Doesn't require complex data structures for storing.

In the present work, black-to-white crossing counts in both horizontal and vertical directions are chosen as the main features.

3.5.4 Role of Rough Sets in Feature-Set Reduction

Consider a character (connected component) image fixed in its Minimum Bounding Rectangle (MBR) of $m \times n$ size. An MBR is a rectangle that a character will just sufficiently fit into it in both width and height directions. We then have m horizontal crossing features, and n vertical crossing features, totaling $m+n$ features. More (finer) information can be obtained if m and/or n are increased further. Telugu characters require high resolution of the normalized raster image to enable discrimination of complex shaped compound characters. But, this leads to a feature space dimensionality of prohibitive size. Large feature-dimensions also increase the computational effort required for classification. In order to acquire reasonably good resolution for recognizing complex shaped characters while still maintaining low dimensions, it is necessary to use feature dimensionality reduction methods.

In the context of OCR, one can consider feature set reduction as the process of finding a smaller (than the original one) set of features with the same or close classificatory

power as the original set [Swinarski , 2001]. To obtain decision rules that are minimal and yet describe the data accurately, one can use RSAR technique by computing the reducts of section 3.3. Since reducts determine the most important attributes of a database from the point of view of classificatory power, redundant information can be eliminated, and thus reduce the feature dimensions. We only have to compute the reducts per case relative to the outcome attribute, and read off the attribute values for that case [Ohrn & Rowland, 2000]. Using RSAR for feature reduction in OCR is a novel attempt in the present work. To the best of our knowledge, this approach was not attempted in Telugu OCR problem so far.

3.5.5 Proposed Feature – Set Reduction Algorithm

Our proposed feature reduction algorithm using RSAR is presented below. Aim of the algorithm is to find the set of predominant attributes among a set of N features for classifying characters and fonts.

3.5.5.1 Algorithm PDA ()

// Computes predominant attributes for the two decision attributes: Charcode, & Fontcode

Input: ‘N’ crossing-count features for each sample character in the Training set.

Output: Set of Predominant attributes for classifying characters, and fonts both.

Begin

1. Prepare a decision table D_T with $N+2$ columns (‘N’ crossing count features + Charcode + Font code).
2. First compute Reducts from D_T with only Charcode as the decision attribute. These are the predominant attributes for Charcode. Let P_C = Predominant attributes for Charcode.
3. Compute Reducts from D_T with Fontcode as the decision attribute. These are the predominant attributes for Fontcode. Let P_F = Predominant attributes for Fontcode.
4. Total set of Predominant Attributes $P_{CF} = P_C \cup P_F$

End Algorithm

In order to recognize the font and character codes simultaneously, we need to construct a decision table with two decision attributes: “Fontcode” and “Charcode” with ‘N’ rows, where N is the length of the feature vector for each sample in the Training set. It is likely that some of these N features may not be required for classification, and are superfluous. Since our aim is to recognize font and characters both, it is necessary to compute the reducts for both the classes, namely, Charcode-class, and Fontcode-class to reduce the feature set dimensions. After obtaining the predominant attributes for each of the two classes, the algorithm combines them by taking their union. The result is the total set of predominant attributes required for correctly classifying the font and character codes. Thus, we reduce the feature set dimensionality by computing the predominant attributes that are just required to classify the font and character codes.

Algorithm for computing reducts for a single decision attribute [Ramadevi, Rao & Reddy, 2007] is as follows:

Algorithm Reduct()

// computes the predominant attributes for a single decision attribute from the given decision table.

Input: Decision table.

Output: Set of Predominant attributes for classifying the decision attribute.

Begin

1. Read the decision table.
2. Initialize the Set of Predominant Attributes (PDA) to null.
3. For each pair of rows that differ in the values of the decision attribute, construct a Binary Matrix (BM) as follows:
 Assign a ‘1’ to an element if the corresponding independent attribute values are different, otherwise assign ‘0’.
4. Repeat steps 5 and 6 until the rows in the BM are zeros or null matrix.
5. Pick up the attribute ‘a’, which has the maximum sum & append it to PDA.
6. Remove all the rows from BM for which the elements are ‘1’ for ‘a’.
7. If BM is null, then output the Set of Predominant Attributes (PDA).

8. If BM is not-null, then print “The PDA roughly explains the decision attribute”, and output PDA.

End algorithm.

The algorithm works on the discernibility-function based approach, explained in section 3.3.2.2. Attributes in the decision table are the crossing count features for each character image. Each pair of distinct characters/fonts is compared for their individual features. A crossing count feature that has the same value in both cases is not an important one from classification point of view. The other features are the likely candidates for the set of Predominant Attributes. For these features, we count the number of appearances they make within all character/ font classes. The features with the highest frequency count are added to the set of Predominant Attributes; and all clauses in the discernibility function containing this attribute are removed. As soon as all clauses have been removed, the algorithm terminates and returns the set of Predominant Attributes, PDA.

The predominant attributes computed using the above algorithm result in the reduced set of features required for identifying the character/font codes.

An illustrative example on the 8 sample characters of Table 3.1 is as follows. For the sake of convenience, Table 3.1 is reproduced here with the addition of Fontcode field.

Table 3.3 An example decision table with two decision- attributes.

<i>Samples</i>	<i>F1</i>	<i>F2</i>	<i>F3</i>	<i>F4</i>	<i>Charcode</i>	<i>Fontcode</i>
1	1	0	2	2	C0	F1
2	0	1	1	1	C2	F1
3	2	0	0	1	C1	F1
4	1	1	0	2	C2	F1
5	1	0	2	0	C1	F2
6	2	2	0	1	C1	F2
7	2	1	1	1	C2	F2
8	0	1	1	0	C1	F2

First let us compute the predominant attributes for Charcode. Fontcode field is eliminated, and the remaining table with respect to Charcode attribute is shown in Table

3.4. Decision attribute is Charcode, and there are 3 distinct character codes in this table: (C0, C1, and C2).

Table 3.4 Decision table with respect to Charcode

<i>Samples</i>	<i>F1</i>	<i>F2</i>	<i>F3</i>	<i>F4</i>	<i>Charcode</i>
1	1	0	2	2	C0
2	0	1	1	1	C2
3	2	0	0	1	C1
4	1	1	0	2	C2
5	1	0	2	0	C1
6	2	2	0	1	C1
7	2	1	1	1	C2
8	0	1	1	0	C1

Sample 1 (Charcode C0) is compared with other samples. Charcodes of all remaining samples are different from C0, so it is compared with all the samples 2, 3, 4, 5, 6, 7 and 8. For each pair, if the corresponding features are different, Binary matrix entry will be “1”, otherwise, “0”. For example, samples 1, and 2 differ in all F1, F2, F3, and F4. Therefore, entries are 1 1 1 1, as shown under the column “2” in the Table 3.5 below. For pair (1, 3), only feature F2 value is same (=0). Other feature values are different. So, entries are 1 0 1 1. Similarly, the entries for pairs (1,4), (1,5), (1,6), (1,7), and (1,8) are made, as shown below.

Table 3.5 Partial discernibility matrix after comparing C0

Samples	1	2	3	4	5	6	7	8
1	---	1 1 1 1	10 11	0 1 1 0	0 0 0 1	1 1 1 1	1 1 1 1	1 1 1 1

Now, sample 2 (Charcode C2) is compared with other samples. Samples 4, and 7 are C2 itself, and so are not used; comparison with sample 1 is already done in the previous step; so sample 2 is now compared only with samples 3, 5, 6, and 8, and is shown in Table 3.6.

Table 3.6 Partial discernibility matrix after comparing C2

Samples	1	2	3	4	5	6	7	8
2	-	-	1 1 10	--	1 1 1 1	1 1 1 0	--	0 0 0 1

Now, sample 3 (C1) need not be compared with 5, 6, and 8 (all are C1 only), it is already compared with 1, and 2, so it is compared with samples 4 and 7 as shown in Table 3.7.

Table 3.7 Partial discernibility matrix after comparing C1

Samples	1	2	3	4	5	6	7	8
3	--	---	---	1 1 0 1	---	---	0 1 1 0	---

Similarly, remaining pairs of samples are compared and the final entries in the Boolean matrix are as shown in Table 3.8.

Table 3.8 Final discernibility matrix

S.No	Pairs	Features			
		F1	F2	F3	F4
1	(1,2)	1	1	1	1
2	(1,3)	1	0	1	1
3	(1,4)	0	1	1	0
4	(1,5)	0	0	0	1
5	(1,6)	1	1	1	1
6	(1,7)	1	1	1	1
7	(1,8)	1	1	1	1
8	(2,3)	1	1	1	0
9	(2,5)	1	1	1	1
10	(2,6)	1	1	1	0
11	(2,8)	0	0	0	1
12	(3,4)	1	1	0	1
13	(3,7)	0	1	1	0
14	(4,5)	0	1	1	1
15	(4,6)	1	1	0	1
16	(4,8)	1	0	1	1
17	(5,7)	1	1	1	1
18	(6,7)	0	1	1	0
19	(7,8)	1	0	0	1
	SUM=	13	14	14	14

Maximum number of appearances occurs for features F2, F3, and F4 as shown in the last row of the above table. First F4 is added to the set of Predominant Attributes (PDA), all rows in which F4=1 are removed from BM. After removing, BM becomes Table 3.9 shown below.

Table 3.9 Discernibility matrix after removing rows with F4=1

S.No	Features Pairs	F1	F2	F3	F4
3	(1,4)	0	1	1	0
8	(2,3)	1	1	1	0
10	(2,6)	1	1	1	0
13	(3,7)	0	1	1	0
18	(6,7)	0	1	1	0
	SUM=	2	5	5	0

From this, after removing rows in which F3=1, BM becomes a null matrix. Hence, the set of predominant attributes $P_C = \{F4, F3\}$. New table with only these predominant and decision attributes after removing duplicate rows is Table 3.10.

Table 3.10 Predominant and decision (Charcode) attributes

F3	F4	CharCode
2	2	C0
0	1	C1
2	0	C1
1	0	C1
1	1	C2
0	2	C2

We can construct decision rules from these reducts as follows:

If (F3=2) and (F4=2) then Charcode= C0.

If (F3=0) and (F4=1) OR if (F3=2) and (F4=0) OR if (F3=1) and (F4=0) then Charcode= C1.

If $(F3=1)$ and $(F4=1)$ **OR** if $(F3=0)$ and $(F4=2)$ then Charcode= C2.

Thus, out of 4 features F1, F2, F3, and F4 with which we started, two are eliminated, and finally F3 and F4 are remained as the predominant features. This is clearly a 50% reduction in the feature set dimension.

Similarly, a decision table with only Font code as decision attribute is shown in Table 3.11.

Table 3.11 Decision table with Fontcode

<i>Samples</i>	<i>F1</i>	<i>F2</i>	<i>F3</i>	<i>F4</i>	<i>Fontcode</i>
1	1	0	2	2	F1
2	0	1	1	1	F1
3	2	0	0	1	F1
4	1	1	0	2	F1
5	1	0	2	0	F2
6	2	2	0	1	F2
7	2	1	1	1	F2
8	0	1	1	0	F2

Similar calculations for the Fontcode attribute result in the set of predominant features {F4, F2}. $P_F = \{F4, F2\}$. Predominant and decision attributes after removing duplicate rows is given in Table 3.12.

Table 3.12 Predominant and decision (Fontcode) attributes

<i>F2</i>	<i>F4</i>	<i>Fontcode</i>
0	2	F1
1	1	F1
0	1	F1
1	2	F1
0	0	F2
2	1	F2
1	1	F2
1	0	F2

From these reducts, we can write the decision rules as follows:

If $(F2=0)$ and $(F4=2)$ **OR** if $(F2=1)$ and $(F4=1)$ **OR** if $(F2=0)$ and $(F4=1)$ **OR**
if $(F2=1)$ and $(F4=2)$ then Fontcode= F1.

*If (F2=0) and (F4=0) **OR** if (F2=2) and (F4=1) **OR** if (F2=1) and (F4=1) **OR** if (F2=1) and (F4=0) then Fontcode= F2.*

Total set of predominant features required for correctly classifying the font and character codes is the union of P_C and P_F .

$$P_{CF} = P_C \cup P_F = \{F4, F3\} \cup \{F4, F2\} = \{F2, F3, F4\}.$$

3.6 Summary

In this chapter, we discussed several aspects of feature selection process and noted that feature selection is crucial in all applications of pattern recognition. We also observed the orthographic complexity of Telugu script that makes the feature selection process a difficult one in Telugu OCRs. We proposed a novel solution for this problem using the theory of rough sets. Our algorithm for reducing feature set dimensionality in Telugu OCRs is introduced in this chapter.

Chapter 4

Design of the Classifier

This chapter introduces the basic methodology used in designing the classifier for the present work. The task of a classifier is to use the feature vector provided by the feature extractor, and to assign the object to a category. The space of all possible feature vectors is called the *feature space*. Essentially, a classifier divides the feature space into regions corresponding to different categories.

There are two types of machine learning: *supervised* learning and *unsupervised* learning. We also use terms *supervised classification* and *unsupervised classification*. In a supervised classification, we present examples of the correct classification (a feature vector along with its correct class) in training the classifier. Based on these examples (also termed as *prototypes*, or *training samples*), the classifier then learns how to assign a given feature vector to a correct class. The generation of the prototypes (i.e., the classification of feature vectors/objects they represent) has to be done manually in most cases [Tohka, 2006]. Supervised learning is of particular use when systems under training are intended to perform tasks that have previously been performed by humans with a certain degree of success. In such cases a relation between data is known to exist, but the rules governing it are not known, or are difficult to obtain. The system to be trained effectively learns by example, generalizing the knowledge to apply it to the entire domain.

Unsupervised classification or *clustering* uses sample feature vectors without class labels. The classification of the feature vectors must be based on *similarity* between them based on which they are divided into natural groupings. Whether any two feature vectors are similar depends on the application. Obviously, unsupervised classification is more difficult than supervised classification and supervised classification is the preferable option when possible.

The design of a pattern recognition system is usually an iterative process [Tohka, 2006]. Once the system is designed, it is tested for its efficiency. If it is not good enough, design process is inspected again to improve some of the stages of the system. After that, the system is tested again and is improved if it still does not meet its requirements. This process is repeated until the system meets the requirements. The design process is based on the knowledge about the particular pattern recognition application, and is usually trial and error procedure. In what follows next, we give a detailed presentation of our methodology in designing the classifier for the proposed font and character recognition system for Telugu.

4.1 Data Sets Used

The following datasets are used for training, designing, and testing purposes in the present work.

4.1.1 Dataset for Training (Dataset # 1)































In the present work, training data is prepared from the following Telugu font typefaces:

1. *Amma*,
2. *Gowthami*,
3. *Hemalatha*,
4. *Srilipi* (*ShreeG* series),
5. *Vaaritha*, and
6. *Vennela*.

Prototype characters (connected components) are generated synthetically (created in the pre-selected size, form all these selected fonts, using computer). Since they are computer-generated images, and are not scanned, prototype images are assumed to be noise-free. About 370 connected components of 72 point size are generated in each of the 6 fonts (approximately 2200 in total). It is to be recalled from section 3.5.2, that the number of connected components varies from font to font. These computer-generated prototypes are

used for training. Table 4.1 shows five of such prototype characters along with their codes (*Charcodes* and *Font codes*) in all the six chosen fonts.

Table 4.1 Some prototype characters in six fonts









S.No.	CharCode	Amma Font (Font code=1)	Gowthami Font (Font code=2)	Hemalatha Font (Font code=3)	Srilipi (G- series) Font (Font code=4)	Vartha Font (Font code=5)	Vennela Font (Font code=6)
1	1						
2	2						
3	3						
4	4						
5	5						

In addition to these prototypes, we also included template images provided by the developers of DRISHTI in our training database. DRISHTI is the Telugu OCR developed by the Resource Center for Indian Language Technology Solutions at the University of Hyderabad. In order to use the data corpus available with the DRISHTI package, and also due to the possibility of embedding the proposed system with DRISHTI in future, these templates are included. DRISHTI templates are from the following two fonts: Hemalatha, and Srilipi (*Shree9* series). These templates are of 32 X 32 sizes [Negi, Bhagvati & Krishna 2001]. There are about 730 templates from these two fonts which are included in our training data. Appendix – A shows our prototype images, and also the templates from DRISHTI.

Total number of prototype images used in our training database is then 2930 (2200 + 730) from eight (6+2) sources. All these 2930 prototype images are given their respective font codes, and character codes. Table 4.2 shows font codes along with one sample character, for all these fonts. These 2930 prototype images constitute Dataset #1. Among these, there are about 481 distinct prototypes in total. Each of these 481 prototype

characters represents one distinct class in the character classification task. These 481 prototypes are shown in Appendix – B.

Table 4.2 Font names and font codes

S.No	Font name	Font code	Example Character
1	Amma	1	
2	Gowthami	2	
3	Hemalatha	3	
4	Srilipi-G series	4	
5	Vartha	5	
6	Vennela	6	
7	Srilipi-9 series (DRISHTI)	7	
8	Hemalatha (DRISHTI)	8	

4.1.2 Dataset for Testing (Dataset # 2)

Our test data is created from real-life documents such as news-papers, story books, and laser-printed pages. All these documents are scanned and segmented to obtain the test data. Example test documents used in the present work are shown in Figure 4.1. Since newspaper print size is assumed to be 9 points [Negi , Bhagvati & Krishna, 2001], our test characters' sizes start from as small as 9 points. In order to obtain test characters in various sizes in the above mentioned fonts, characters are typed in electronic documents in the required sizes, hard copy prints of which are then scanned. Figure 4.1 (a) shows an example of such a laser-printout in which characters of Amma font are typed in size 60. Other sizes include 16, 20, 24, 36, 48, 56, 65, and 72 points. Thus, sizes of our test characters vary from

9 to 72 points. Totally there are 10,166 test characters collected in this manner, and all these are assigned their respective font codes, and character codes. This constitutes Dataset #2. Table 4.3 presents the total number of test characters in each size, and in each font.

యూ యు రరారిరీరురూ
రెరే రో రోరావవావివీవు
పూవెవేవోవోశశాశిశీశు
శూశాశే శోశోశాశిషాషిషీ
షుషూషేషే షో షో షా ష
సాసిసిపెసెసాసోసాహహా
హి హి హుహుహాహా హా హా

(a) Laser-printed page in Amma font

గుంటూరు, ఆగస్టు 30 (ప్రభాతవార్త): వ్యర్థ పదార్థాలతో
వస్తువులు తయారు చేసే పరిశ్రమలు నెలకొల్పాలని ఎ.పి.
పొల్యూషన్ కంట్రోల్ బోర్డు సీనియర్ ఇంజనీర్ బి.మధుసూ
ధనరావు అన్నారు. సోమవారం విజ్ఞాన్ ఇంజనీరింగ్ కళాశా
లలో ఆసోసియేషన్ ఆఫ్ కెమికల్ ఇంజనీర్స్, కళాశాల ఇం
జనీరింగ్ విభాగం ఆధ్వర్యంలో 'పర్యావరణ పరిరక్షణ చ
ట్టం- అనుసరణీయ విధానాలు' అనే అంశంపై సదస్సు జ
రిగింది. ఈ సందర్భంగా జరిగిన సభలో ఆయన మాట్లాడు
తూ, కాలుష్య నియంత్రణ మండలికి ఉన్న ప్రత్యేక అ

(b) Newspaper document in Vartha font

శ్రీశైలం క్షేత్రాన్ని దర్శించే భక్తులను సాక్షిగణపతి తన చిట్టాలో వ్రాసుకుని
తైలానంలో శివునికి వివేదించి ఆ భక్తులు శ్రీశైలం క్షేత్రాన్ని దర్శించినట్లు సాక్షం
చెబుతాడని ప్రతీతి. అందువల్లనే ఈ గణపతికి సాక్షిగణపతి అనే పేరు వచ్చిందని
చెబుతారు. అపంకపురం జిల్లా రాయదుర్గంలో మూడు గణపతి ఆలయాలున్నాయి.
కోటలో కొండపైకి వెళ్ళే మార్గంలో వున్న దశభుజ గణపతి ఆలయం అత్యంత
ప్రసిద్ధమైంది.
పది అడుగుల ఎత్తున వున్న ఈ విగ్రహం భక్తులను విశేషంగా ఆకర్షిస్తుంది.
ద్రాక్షారామంలోని భీమేశ్వర ఆలయంలో తూర్పుదిశగా వున్న గోపురానికి సమీపంలో
స్వత్య గణపతి ఆలయం వున్నది.
శ్రీకాకుళం జిల్లా శ్రీముఖలింగ క్షేత్రంలో చింతామణి గణపతి సాక్షిగణపతి,
దుర్తి విఘ్నేశ్వర, స్వత్య గణపతి మూర్తులు ఉన్నాయి. చిత్తూరు జిల్లా గుడిమర్లం
పరశురామేశ్వర ఆలయంలో స్వత్య గణపతి శిల్పం వుంది. చిత్తూరు జిల్లా శ్రీకాళహస్తి
ఆలయంలో రెండో ప్రాకారం, మూడవ ప్రాకారం మధ్యలో సువిశాలమైన అవరణలో
పాతాళవిఘ్నేశ్వర ఆలయం వుంది. ఇటువంటి ఆలయం ఇది ఒక్కటే ఉండడం
విశేషం.
చిత్తూరు జిల్లా పుంగమూరు సమీపంలోని లద్దిగంలో ఇరుంగళేశ్వర దేవాలయ
అవరణలో కొండమనూరులోని ఆదిత్యేశ్వర ఆలయంలోనూ స్వత్య భంగిమలో గణపతి
విగ్రహాలు గల ఆలయాలున్నాయి. నెల్లూరు పట్టణంలో మూలాపేటలోని
మూలస్థానేశ్వర ఆలయంలో దక్షిణ దిశగా అతి పెద్ద గణపతి విగ్రహం గల ఆలయం
వుంది. ఆలయం మొత్తం ఒకే పెద్ద గణపతి విగ్రహంతో నిండిపోయి ఉండడం ఇక్కడ
విశేషం.
ఖమ్మం జిల్లా భద్రాచలం కొండరామస్వామి ఆలయ పరిసరాల్లో గణపతి
ఆలయం వుంది. జహీరాబాద్ సమీపంలోని కొండల మత్స్య గణపతి ఆలయం వుంది.
ఈ గణపతి స్వయంభువని ప్రతీతి. శ్రీకాకుళం జిల్లాలోని అరసవెల్లి సూర్య
నారాయణస్వామి దేవాలయంలో గణపతి విగ్రహం ఉంది. భీమేశ్వర క్షేత్రంలో స్వత్య
గణపతి ఆలయం వుంది. కాశీలోపల కుడిచేతిమీద కొండం కలిగిన గణపతి విగ్రహం
ఇక్కడ ఉండడం విశేషం.
కరీంనగర్ జిల్లా వేములవాడలోని శ్రీరాజరాజేశ్వర ఆలయ ప్రాంగణంలో
లక్ష్మీగణపతి ఆలయం ఉంది. భక్తులు తూర్పుదిశలో గల ద్వారం నుండి ఆలయ

(c) Page from a book in Hemalatha font

జాతీయ గీతం
గురుదేవ్ శ్రీ రవీంద్రనాథ్ ఠాకూర్
జన గణ మన అధినాయక జయహే
భారత భాగ్యవిధాతా
పంజాబ సింధు గుజరాత్ మరాఠా
ద్రావిడ ఉత్తర వంగ
వింధ్య హిమాలయమునా గంగా
ఉచ్చల జలధి తరంగా
తనశుభ నామే జాగే
తన శుభ ఆశిష మాగే
గాహితవ జయ గాధా
జన గణ మంగళ దాయక జయహే
భారత భాగ్యవిధాతా
జయహే జయహే జయహే
జయ జయ జయ జయహే

(b) Page in Srilipi font

Figure 4.1 Example test documents used in the present work

Table 4.3 **Number of Test characters: Size-wise, Font-wise**

Font code	Size																	
	9	16*	18	20	24	30	32*	36	40	48	50	56	58	60	65	68	72	Total
1				19	22	21		21	22	21	417		21	360	22	22	310	1278
2					55	56		49	50	53		53		375	54		370	1115
3					43	24		399	23	43	20	21	20	20	20	20	385	1038
4					50	50		376	53	53	68	69	54	54	55	54	362	1298
5	1010				35	35		35	35	35	35	35	35	35	37	33	395	1790
6	---				281	34		24	34	365	26	15	16	377	16	16	379	1583
7			173	296			409											878
8		134					1052											1186
Total	1010	134	173	315	486	220	1052	1313	217	570	566	193	146	1221	204	145	2201	10,166

*These test characters are collected from books whose character sizes are not known, sizes are assumed to be 16, and 32, based on the visual observation.

4.1.3 Dataset for Designing (Dataset # 3)

During the design phase, we conducted several experiments to determine the influence of various parameters on the recognition accuracies, and there by improving the performance. About 600 characters (100 from each of the 6 fonts) are created as test characters for this purpose. These character images are computer-generated in the following sizes: 18, 24, 30, 36, 40, 48, 56, 60, 65 and 72. This forms Dataset #3, which is used only during the design phase to fine-tune some of the design parameters.

4.2 Proposed Classification Versus Traditional Classification

There are two distinct differences of the proposed classification method compared to the traditional classification approaches.

i) Our prototypes are clean images, and test characters are from scanned data, which are assumed to have been subjected to noise. Hence, significant deviation is expected in the test data from the prototypes. In traditional classification, both the training and test samples are usually collected from the same environment.

ii) In traditional classification methods, usually a large number of samples of same class are available. Hence clustering is used on these samples. In the present work, since all the characters in a font are distinct, *within-class* variation is more than the *between-class* variation, for classifying fonts. For classifying characters, there are only 8 samples at the most (from the 8 fonts) for each character. Hence in the present work, *clustering* is not employed as in traditional classification methods.

4.3 Base Feature Extraction and Selection

All the prototype characters are normalized to 50 x 50 size using our size-normalization algorithm given in section 5.2. In the feature extraction stage, crossing count features in both horizontal and vertical directions from each character image are extracted. A feature vector constituting 50 horizontal crossing-counts from the 50 rows and 50 vertical crossing-counts from the 50 columns is formed. Thus, initially, dimension of the feature vector is 100 for each prototype image. Horizontal crossing-counts are stored as HCV vector; and vertical crossing-counts as VCV vector. HCV_i denotes the i^{th} horizontal crossing value, and VCV_i denotes the i^{th} vertical crossing value.

Table 4.4 shows the 100-D crossing count feature vectors (HCV 1 to 50, VCV 1 to 50) for a sample set of 5 characters from *Vaaritha* font.

Table 4.4 Feature vectors for 5 Sample characters

S. No	Features	అ	ఆ	ఇ	ఈ	ఉ
1	HCV1	2	2	1	1	1
2	HCV2	2	2	1	1	1
3	HCV3	2	2	2	1	1
4	HCV4	2	2	2	1	1
5	HCV5	2	2	2	1	1
6	HCV6	3	2	2	1	1
7	HCV7	3	2	2	1	1
8	HCV8	4	4	2	1	1
9	HCV9	4	4	3	2	1
10	HCV10	4	4	3	2	1
11	HCV11	4	4	3	2	1
12	HCV12	4	4	3	2	1
13	HCV13	4	4	3	2	1
14	HCV14	4	4	3	2	1
15	HCV15	4	4	3	2	1
16	HCV16	4	4	3	2	1
17	HCV17	4	4	2	2	2
18	HCV18	4	4	2	2	2
19	HCV19	4	4	2	2	2
20	HCV20	4	4	1	2	2
21	HCV21	4	3	1	4	2
22	HCV22	4	3	1	4	1
23	HCV23	3	3	2	4	1
24	HCV24	3	2	2	4	1
25	HCV25	3	2	2	4	1
26	HCV26	3	2	2	3	1
27	HCV27	3	3	3	3	1
28	HCV28	2	3	3	3	3
29	HCV29	2	2	3	3	3
30	HCV30	2	2	3	5	2
31	HCV31	2	2	3	1	2
32	HCV32	2	2	2	1	2
33	HCV33	2	2	2	1	2
34	HCV34	2	2	2	1	3
35	HCV35	2	2	2	4	4
36	HCV36	2	2	2	4	4
37	HCV37	2	2	2	4	3
38	HCV38	2	2	2	4	3
39	HCV39	2	2	2	4	3
40	HCV40	2	2	2	4	3
41	HCV41	2	2	2	4	3
42	HCV42	2	2	2	4	3
43	HCV43	1	1	2	4	3
44	HCV44	1	1	2	3	3
45	HCV45	1	1	2	3	4
46	HCV46	1	1	2	2	2
47	HCV47	1	1	2	2	2
48	HCV48	1	1	1	1	2
49	HCV49	1	1	1	1	2

50	HCV50	1	1	1	1	2
51	VCV1	1	1	2	1	1
52	VCV2	1	1	2	1	1
53	VCV3	1	1	2	1	1
54	VCV4	1	1	2	1	1
55	VCV5	1	1	2	1	1
56	VCV6	2	2	3	1	1
57	VCV7	2	2	3	1	1
58	VCV8	3	2	3	1	1
59	VCV9	3	2	3	2	1
60	VCV10	3	3	3	3	2
61	VCV11	3	3	3	4	2
62	VCV12	3	3	3	4	2
63	VCV13	3	3	3	4	3
64	VCV14	3	3	3	4	3
65	VCV15	3	3	3	3	3
66	VCV16	3	3	3	3	3
67	VCV17	3	3	3	3	3
68	VCV18	2	3	3	3	3
69	VCV19	2	2	3	3	3
70	VCV20	2	2	3	3	3
71	VCV21	3	3	3	3	3
72	VCV22	2	3	3	3	4
73	VCV23	2	3	3	4	4
74	VCV24	2	2	3	4	3
75	VCV25	2	2	3	4	3
76	VCV26	2	2	3	4	3
77	VCV27	2	2	3	4	3
78	VCV28	2	2	3	4	4
79	VCV29	2	2	3	4	4
80	VCV30	2	2	3	2	4
81	VCV31	2	3	3	2	3
82	VCV32	3	3	3	2	3
83	VCV33	3	3	3	2	3
84	VCV34	3	3	3	2	3
85	VCV35	2	3	3	2	4
86	VCV36	3	3	2	2	4
87	VCV37	3	3	2	2	4
88	VCV38	3	3	2	3	4
89	VCV39	3	3	2	2	4
90	VCV40	3	3	2	3	4
91	VCV41	3	3	2	3	4
92	VCV42	3	3	2	3	4
93	VCV43	2	3	1	3	3
94	VCV44	2	3	1	3	3
95	VCV45	2	3	2	3	3
96	VCV46	2	2	2	3	3
97	VCV47	1	1	2	3	2
98	VCV48	1	1	2	3	2
99	VCV49	1	2	2	2	2
100	VCV50	1	2	1	1	2

4.3.1 Feature Dimensionality Reduction

One of the fundamental steps in the design of a classifier is reducing the pattern dimensionality through Feature Selection (FS).

The actual training database contains 100 crossing count features for the 2930 prototypes of Dataset #1. Hence the training database dimensions are 2930 X 100. We then applied our dimensionality reduction technique (Algorithm *PDA*) by computing reducts as explained in chapter in 3.5.5. Here, 'N' is the number of features for each prototype, which is 100 in this case.

Applying the algorithm on *Fontcode* produced 23 features as predominant attributes for classifying all the fonts in the training set.

$\therefore P_F = \{ \text{'HCV9','HCV12','HCV16','HCV24','HCV34','HCV38','HCV42','HCV44','HCV45','HCV46','VCV3','VCV6','VCV10','VCV16','VCV22','VCV29','VCV31','VCV33','VCV41','VCV43','VCV45','VCV48','VCV50'} \}$ (Total 23).

Similarly, applying the algorithm on *Charcode* produced 21 features as predominant attributes. These 21 features that are adequate for classifying all the characters in the training set are:

$P_C = \{ \text{'HCV11','HCV16','HCV24','HCV31','HCV38','HCV45','HCV46','VCV6','VCV10','VCV12','VCV15','VCV19','VCV22','VCV27','VCV29','VCV33','VCV41','VCV43','VCV47','VCV48','VCV50'} \}$ (Total 21).

Of these 44 features, 14 are common for both. Hence, the combined predominant feature set, after eliminating repeated features, consists of only **30** features given below.

$$\begin{aligned} P_{CF} &= P_C \cup P_F \\ &= \{ \text{HCV9, HCV11, HCV12, HCV16, HCV24, HCV31, HCV34, HCV38, HCV44, HCV45, HCV46, VCV3, VCV6, VCV10, VCV12, VCV15, VCV16, VCV18, VCV19, VCV22, VCV27, VCV29, VCV31, VCV33, VCV41, VCV43, VCV45, VCV47, VCV48, VCV50} \} \end{aligned}$$

Out of these 30 features, seven are *charcode*-specific, nine are *fontcode*-specific, and common predominant attributes are fourteen (Table 4.5).

Table 4.5 Sample database with 30-D feature vectors

S. No	Property of the Feature	Features	అ	ఆ	ఇ	ఈ	ఉ
1	Common	HCV16	4	4	3	2	1
2	Common	HCV24	3	2	2	4	1
3	Common	HCV38	2	2	2	4	3
4	Common	HCV45	1	1	2	3	4
5	Common	HCV46	1	1	2	2	2
6	Common	VCV6	2	2	3	1	1
7	Common	VCV10	3	3	3	3	2
8	Common	VCV22	2	3	3	3	4
9	Common	VCV29	2	2	3	4	4
10	Common	VCV33	3	3	3	2	3
11	Common	VCV41	3	3	2	3	4
12	Common	VCV43	2	3	1	3	3
13	Common	VCV48	1	1	2	3	2
14	Common	VCV50	1	2	1	1	2
15	Character Specific	HCV11	4	4	3	2	1
16	Character Specific	HCV31	2	2	3	1	2
17	Character Specific	VCV12	3	3	3	4	2
18	Character Specific	VCV15	3	3	3	3	3
19	Character Specific	VCV19	2	2	3	3	3
20	Character Specific	VCV27	2	2	3	4	3
21	Character Specific	VCV47	1	1	2	3	2
22	Font Specific	HCV9	4	4	3	2	1
23	Font Specific	HCV12	4	4	3	2	1
24	Font Specific	HCV34	2	2	2	1	3
25	Font Specific	HCV42	2	2	2	4	3
26	Font Specific	HCV44	1	1	2	3	3
27	Font Specific	VCV3	1	1	2	1	1
28	Font Specific	VCV16	3	3	3	3	3
29	Font Specific	VCV31	2	3	3	2	3
30	Font Specific	VCV45	2	3	2	3	3

The seven Char code-specific features are: HCV 11, HCV 31, VCV12, VCV 15, VCV 19, VCV 27, and VCV47.

The nine font code specific features are: HCV 9, HCV 12, HCV 34, HCV 42, HCV 44, VCV 3, and VCV16, VCV31, and VCV45.

Fourteen features that are common for both font code and char code are: HCV 16, HCV 24, HCV 38, HCV 45, HCV 46, VCV 6, VCV 10, VCV 22, VCV 29, VCV 33, VCV 41, VCV 43, VCV 48, and VCV 50.

Rough sets method (reducts) thus showed an ability to reduce the feature dimensionality significantly from 100 to 30.

For the 2930 prototype characters, a new database is now created with these 30 predominant features. This new database with 30-dimensional (30-D) feature vector is shown in Table 4.5 for the same five characters.

4.4 Structure of the Classifier

The classifier assigns labels to character images by using a classification algorithm based on the features extracted and the relationships among the features. Before arriving at the final design of the classifier, we conducted several experiments on deciding the influence of other factors on the recognition accuracies. Dataset #3, which contains 600 characters of different sizes, is used for this purpose.

4.4.1 Experimentation for Tuning the Parameters

We first tested with the 30 reduct-based features obtained in Section 4.3.1. A minimum-distance classifier is applied on the feature vectors of 600 test characters in Dataset #3. Minimum distance classifier uses a distance metric to compute distance between samples. An unknown sample is classified by computing its distance to the training samples (prototypes), and producing the one with the least distance. One frequently used distance metric is the Euclidean distance. In this, root mean square error of the features is taken as the measure of resemblance between known and unknown characters.

Out of these 600 test characters, 442 and 421 correct classifications are obtained resulting 73.6% and 70.16% accuracies for character and font recognitions respectively. To explore the possibility of improving the classification results further, we then conducted the following experiments.

It is possible that imperfections in the scanning process would result in some error. Also, some digitization errors are unavoidable in scaling the images to the required size of 50x50. To compensate for these errors, and to gain good recognition accuracy, we tested the effect of the following factors on the recognition accuracy before proposing the final structure of the classifier. These experiments are aimed at fine-tuning various parameters that would influence the recognition accuracies. The details of these experiments are given now.

4.4.1.1 Effect of Neighboring Features on Recognition

When normalizing the sizes of the characters to a uniform size (50 X 50), some digitization errors are bound to occur in the preprocessing phase. This will have a great influence in the recognition accuracy. Any minor digitization error in fixing a character into its MBR will capture incorrect feature; hence a robust method of capturing the right features with good accuracy need to be developed. Following is our feature capturing method developed to compensate for the digitization errors. We tested the role of the features that are adjacent to the 30 predominant features obtained in 4.2.1. If HCV_i is a reduct-based feature, the crossing counts in $(i-1)^{th}$ row, and $(i+1)^{th}$ row may also influence the value of HCV_i . Similarly, if VCV_j is the reduct-based feature, neighboring crossing counts in $(j-1)^{th}$, and $(j+1)^{th}$ columns also need to be tested for their effect on recognition accuracy. Figure 4.2 illustrates this concept clearly. If HCV_5 is a predominant feature obtained through our algorithm $PDA()$, then, neighboring features are HCV_4 , and HCV_6 . In a similar manner, for a predominant feature VCV_3 , VCV_2 and VCV_4 are tested for their influence on recognition accuracy.

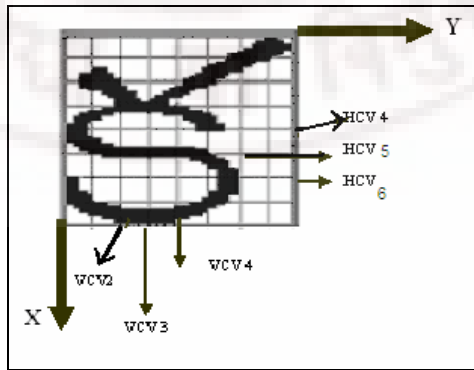


Figure 4.2 Calculation of neighboring features

The following experiments (Feature capturing methods) on the neighboring features are carried out on Dataset #3 consisting of 600 test characters of different sizes from all the selected fonts. In the following experiments, a reduct-based feature is denoted as CC.

- i. FLOOR5: In the first experiment, two neighboring features on either side of CC are considered. i.e., for each of the reduct-based features (HCV_i) obtained, average of the five values: HCV_{i-2} , HCV_{i-1} , HCV_i , HCV_{i+1} , and HCV_{i+2} , are calculated, and is rounded to the nearest lower integer value. This value is used in place of HCV_i .

$$HCV_i = \text{floor}(\text{average}(HCV_{i-2}, HCV_{i-1}, HCV_i, HCV_{i+1}, HCV_{i+2})) \text{ ----- (4.1)}$$

Similarly, for each VCV_i , two of its adjacent values above and below: VCV_{i-2} , VCV_{i-1} , VCV_{i+1} , and VCV_{i+2} are added, and their average is rounded to the nearest lower integer value, which is then used in place of VCV_i .

$$VCV_i = \text{floor}(\text{average}(VCV_{i-2}, VCV_{i-1}, VCV_i, VCV_{i+1}, VCV_{i+2})) \text{ ----- (4.2)}$$

- ii. CEIL5: In the second experiment, instead of rounding to the nearest lower integer, the mean of the 5 adjacent features is rounded to the next higher integer value.

$$HCV_i = \text{ceil}(\text{average}(HCV_{i-2}, HCV_{i-1}, HCV_i, HCV_{i+1}, HCV_{i+2})) \text{ ----- (4.3)}$$

$$VCV_i = \text{ceil}(\text{average}(VCV_{i-2}, VCV_{i-1}, VCV_i, VCV_{i+1}, VCV_{i+2})) \text{ ----- (4.4)}$$

- iii. FLOOR3: In the third experiment, one neighboring feature on either side of CC is considered. i.e., for each of the reduct-based features obtained, crossing counts in the immediately adjacent row/column are only considered. For each HCV_i , the adjacent values on either side: HCV_{i-1} , and HCV_{i+1} , are added, and the average of these three values is rounded to the nearest lower integer value.

$$HCV_i = \text{floor}(\text{average}(HCV_{i-1}, HCV_i, HCV_{i+1})) \text{ ----- (4.5)}$$

$$VCV_i = \text{floor}(\text{average}(VCV_{i-1}, VCV_i, VCV_{i+1})) \text{ ----- (4.6)}$$

- iv. CEIL3: In this experiment, instead of rounding to the nearest lower integer, the mean of the 3 adjacent features is rounded to the nearest higher integer value.

$$HCV_i = \text{ceil} (\text{average} (HCV_{i-1}, HCV_i, HCV_{i+1})) \quad \text{-----}(4.7)$$

$$VCV_i = \text{ceil} (\text{average} (VCV_{i-1}, VCV_i, VCV_{i+1})) \quad \text{-----} (4.8)$$

In the fifth and sixth experiments, most frequently occurred feature (*mode*) is used instead of the average value of neighboring features.

- v. MODE 5: In experiment 5, two adjacent values each on both sides of CC are considered.

$$HCV_i = \text{mode} (HCV_{i-1}, HCV_{i-2}, HCV_i, HCV_{i+1}, HCV_{i+2}) \quad \text{-----} (4.9)$$

$$VCV_i = \text{mode} (VCV_{i-1}, VCV_{i-2}, VCV_i, VCV_{i+1}, VCV_{i+2}) \quad \text{-----} (4.10)$$

- vi. MODE 3: In experiment 6, single adjacent value each on both sides of CC is considered.

$$HCV_i = \text{mode} (HCV_{i-1}, HCV_i, HCV_{i+1}) \quad \text{-----} (4.11)$$

$$VCV_i = \text{mode} (VCV_{i-1}, VCV_i, VCV_{i+1}) \quad \text{-----} (4.12)$$

- vii. ORIGINAL: Actual values of the original features are considered in this experiment. i.e., without using the adjacent features, only the predominant features (CC) are used as mentioned in section 4.4.1.

Testing is carried out on Dataset #3 consisting of 600 test characters of different sizes from all the selected fonts. Results are given in Table 4.6.

Table 4.6 Effect of neighboring features on recognition

Feature capturing method	Experiment	Correctly recognized (characters)	Character Recognition Accuracy	Correctly recognized (fonts)	Font Recognition Accuracy
i	FLOOR5	547	91.16%	493	82.16 %
ii.	CEIL5	541	90.16%	488	81.33%
iii.	FLOOR3	507	84.52%	459	76.5 %
iv.	CEIL3	499	83.16%	452	75.3%
v.	MODE5	522	87%	464	77.3 %
vi.	MODE3	464	77.3%	457	76.16 %
vii.	ORIGINAL	442	73.6%	421	70.16 %

Considering a band of 2 features on both sides of predominant features, taking the average of them, and rounding to the nearest lower integer (floor) as in experiment1 (FLOOR5), gave the best results for both font and character recognitions. By comparing this with that of original predominant features (last row in Table 4.6), it can be observed that the recognition accuracies are improved by about 18%, and 12% for character and font respectively.

So, for all the 30 predominant features, HCV and VCV values are set as per FLOOR5 (Equations 4.1, and 4.2).

4.4.1.2 Effect of input character size on recognition

In this experiment, recognition accuracies for different font sizes of the input character images are tested. Dataset #3 containing computer-generated characters in sizes 18, 24, 30, 36, 40, 48, 56, 60, 65 and 72 are used in this experiment. We then plotted the recognition accuracies (Y-axis) with respect to size (X-axis). Results are shown in Figure 4.3 for character recognition, and in Figure 4.4 for font recognition. In both cases, maximum recognition (100%) is obtained at size 72, which is the size of the prototype

characters. This experiment gives the conclusion that better results are possible if the training data is selected near the font size of the expected test input characters.

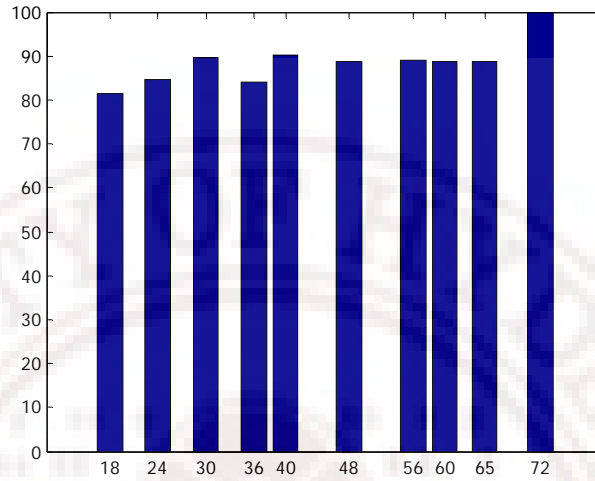


Figure 4.3 Size Vs Character Recognition

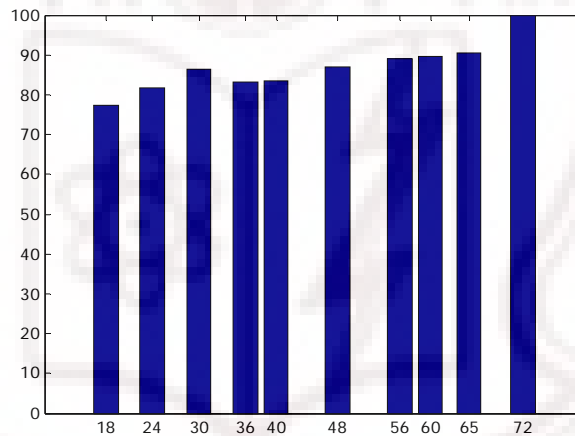


Figure 4.4 Size Vs Font recognition

4.4.1.3 Effect of Nearest-Neighbor Classification

Nearest neighbor classifiers are used quite frequently and effectively in statistical pattern recognition. In these, feature vectors of several samples of each class are stored as exemplars of the class. When an unknown pattern is to be classified, the Euclidean distance between the feature vectors of the unknown pattern from the feature vectors of each of the exemplars is computed. The k smallest of these distances is used to classify the given

pattern according to a majority-voting rule [Bunke & Wang, 1997]. If $k = 1$, this is just the Euclidean minimum distance classifier. The advantage of considering k nearest neighbors is that a particular symbol on the boundary may be nearer to a noisy symbol of a different class but majority of k symbols would bring out the correct identification with a higher probability.

Dataset # 3 containing 600 characters in different sizes is used in this experiment. For any $k=1, 2$, or 3 , let the k^{th} minimum distance be KNN. All the prototype characters/fonts which are at a distance of KNN or less from the test character are then found. Then, a majority voting is applied which produces the prototype which occur maximum number of times (maximum frequency) as the output. We tested with $k=1, 2$, and 3 nearest neighbor classifiers to arrive at a suitable k value for better recognition. The classifier computes the Euclidean distance between the feature vectors of the test characters from the feature vectors of each of the prototype characters. For each test character, the distances to each of the prototypes are then sorted. For the least distance, $k=1$. The results are shown in Table 4.7. From these results, it can be observed that 1NN gives better performance than the other two.

Table 4.7 Results of nearest neighbor classification

K-Nearest neighbor	Character Recognition	Font Recognition
1NN (K=1)	91%	82%
2NN (K=2)	84%	77%
3NN (K=3)	75 %	64 %

4.4.2 Design of a Multi-stage Classifier

In order to improve the results obtained with the nearest neighbor classifier, an alternative multi-stage classifier is then explored. The details of the multi-stage classifier are as follows: Preliminary classification is to be done in the first stage. First stage operates at some *threshold distance* instead of *minimum distance*. This threshold value (Th) will be

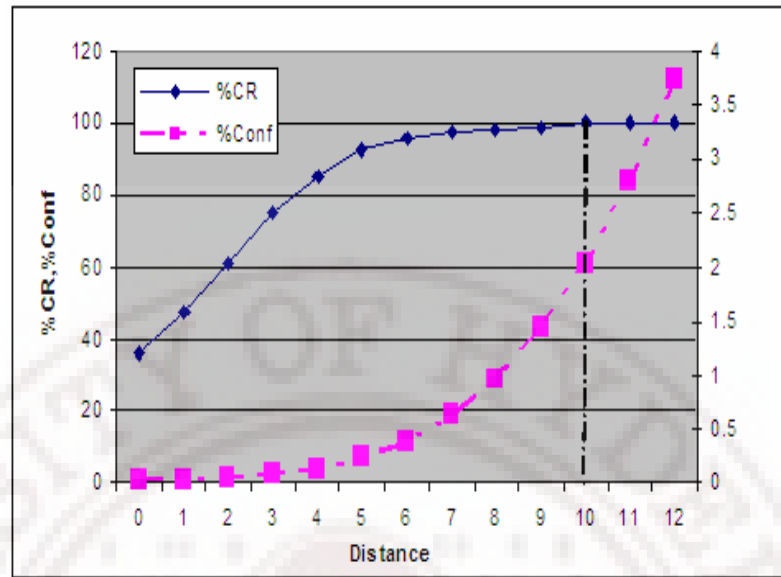
determined from the following experiment. First stage outputs a group of prototypes which are within the threshold distance Th from the test characters. Since this output usually includes some wrong (confused) characters/fonts also, a second stage is to be designed with an aim to reduce this confusion set. Threshold distance value Th for the first stage is determined based on the results of the following experiment on Dataset # 3.

4.4.2.1 Effect of Error Thresholds on Font and Character Recognition

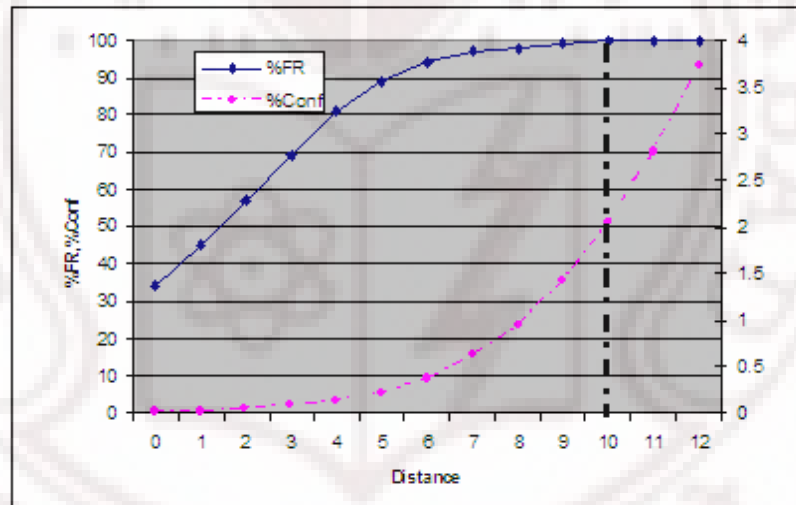
Feature vectors of test characters are compared with the feature vectors of each of the prototype characters. This experiment is conducted by varying the threshold distance from 0 to 12. All prototype characters in the training database whose distance falls below some threshold value are produced as the outputs. Figure 4.5 shows the effect of varying threshold distance from 0 to 12 on (a) character and (b) font recognitions. PCR is the Percentage Character Recognition, and PFR is the Percentage Font Recognition.

Upper curves in Figure 4.5 show the relation between threshold distance and the percentage of possibility of the test character/font “include” in the output (P_I), i.e., Confusion set. Lower curves show the effect of varying the error threshold on the confusion set. This is the percentage of proportion of confusion (P_C). With increasing threshold, as the probability of inclusion is increased, so also the confusion set. So, as the upper curve is increasing, lower curve also increases.

From these curves, it can be observed that distance values ≥ 10.0 produces peak results (100%) for both font and character recognitions, though the confusion set is not negligible in these cases.



(a)



(b)

Figure 4.5 Role of Distance thresholds on (a) character and (b) font recognitions

However, the idea is to remove this confusion set at the next stage by including additional features if necessary. The threshold distance can be determined by one of the following methods.

i. Cost based Method:

Cost- curves are plotted for both font and character recognitions as follows. Let P_I = Percentage of inclusion and, let P_C = Percentage of confusion as mentioned above. If C_m is the cost of missing, i.e., cost associated with missing a character because of choice of threshold, and C_c is the cost of confusion, i.e., cost associated to resolve confusion ($C_c \ll C_m$), then the total cost involved (C) for a specified distance threshold is:

$$C = C_m(100 - P_I) + C_c \cdot P_C$$

Assuming $C_m = 100$, and $C_c = 5$, graphs are plotted between Cost(C) and Distance Threshold, on semi-log graph sheets, as shown in Figure 4.6 (a) for PCR, and in Figure 4.6 (b) for PFR. Minimum cost is obtained at Threshold = 10 and, the cost is increasing afterwards in both the curves. So, Distance Threshold value of 10 is found to be the ideal choice for the threshold.

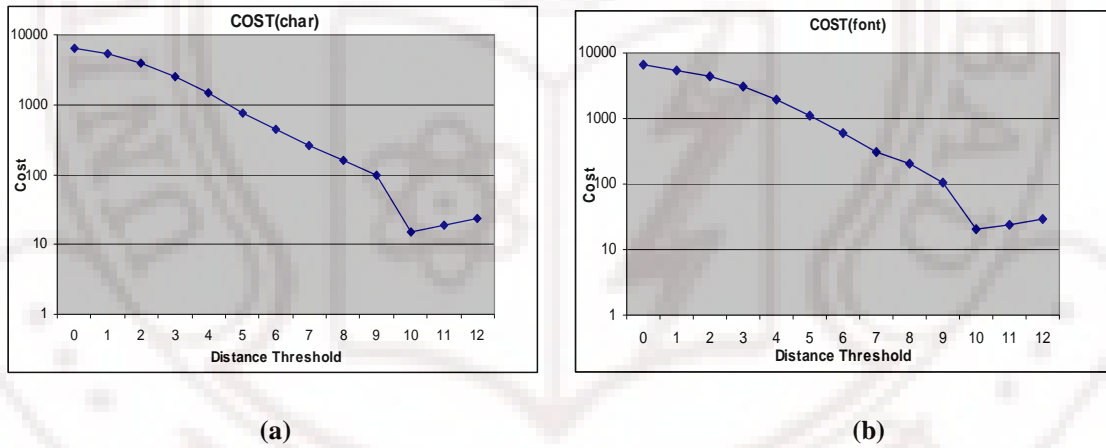


Figure 4.6 Total Costs for (a) character and (b) font recognitions

ii. Empirical Method:

Experimentation for the threshold selection is continued in the second stage also. That is, after designing our two-stage classifier, threshold distance values are varied from 6 to 11, and, overall recognition values (after second stage) are computed, and the results are presented in section 4.4.2.5. Figure 4.7 shows the graph of these results. Peak recognition

values can be noticed at threshold 10. From this figure also, it is observed that threshold value 10 would be the ideal choice.

Therefore, our first stage of the classifier is set at a threshold distance of 10.0 to maintain reasonably good values for the final recognition rates.

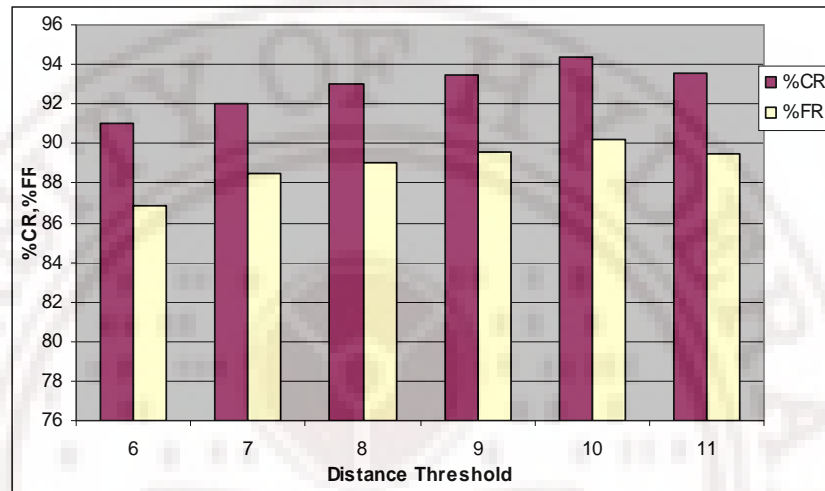


Figure 4.7 Overall Recognitions Vs. Distance threshold

4.4.2.2 Reducing the Confusion Set

Now, the second stage of the classifier is to be designed to reduce the confusion set produced in the first stage. This may require considering additional features. For this, we experimented with two popular types of features along with the crossing-count features: Invariant moment features, and Expectation Maximization (EM) parameters. These features are used frequently in the field of character recognition, and are explained below.

4.4.2.2.1 Hu's invariant moments

Classical moment invariants were introduced by Hu [Hu, 1962] and they were successfully used in numerous pattern recognition tasks in image processing, computer vision and related fields [Flusser & Suk, 1994]. Moment invariants are the features based on statistical moments of characters. Certain weighted averages (*moments*) of the image pixels' intensities are used to describe certain properties of the image. When applied to images, they describe the image content (or distribution) with respect to its axes.

They are designed to capture both global and detailed geometric information about the image. Simple properties of the image which are found *via* image moments are: area (or total intensity), its centroid, and information about its orientation.

Hu described a set of six moments that are rotation, scaling, and translation invariant. In addition, he also described a seventh moment that is skew invariant to help distinguish mirror images.

For a 2-D continuous function $f(x,y)$ the moment of order $(p + q)$ is defined as

$$M_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} x^p y^q f(x, y) dx dy \quad \text{for } p, q = 0, 1, 2, \dots$$

For a scalar (greytone) image with pixel intensities $I(x, y)$, raw image moments M_{ij} are calculated by

$$M_{ij} = \sum_x \sum_y x^i y^j I(x, y)$$

Central moments (moments about the mean), which are translational invariant are defined as:

$$\mu_{pq} = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} (x - \bar{x})^p (y - \bar{y})^q f(x, y) dx dy, \text{ where}$$

$$\bar{x} = \frac{M_{10}}{M_{00}}, \quad \bar{y} = \frac{M_{01}}{M_{00}}. \quad \bar{x} \text{ and } \bar{y} \text{ are the components of the centroid.}$$

If $f(x, y)$ is a digital image, then the previous equation becomes

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

Invariance to both translation and changes in scale can be achieved by dividing the corresponding central moment by the properly scaled μ_{00} , using the following formula.

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{(1+\frac{i+j}{2})}} \quad \text{where } (i+j) \geq 2.$$

The seven Hu's invariant moments are as follows:

$$\begin{aligned} I_1 &= \eta_{20} + \eta_{02} \\ I_2 &= (\eta_{20} - \eta_{02})^2 + (2\eta_{11})^2 \\ I_3 &= (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \\ I_4 &= (\eta_{30} + \eta_{12})^2 + (\eta_{21} + \eta_{03})^2 \\ I_5 &= (\eta_{30} - 3\eta_{12})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] + \\ &\quad (3\eta_{21} - \eta_{03})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] \\ I_6 &= (\eta_{20} - \eta_{02})[(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2] + 4\eta_{11}(\eta_{30} + \eta_{12})(\eta_{21} + \eta_{03}) \\ I_7 &= (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2] - \\ &\quad (\eta_{30} - 3\eta_{12})(\eta_{21} + \eta_{03})[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2]. \end{aligned}$$

Hu's moments are invariant under translation, rotation and scaling. We, therefore, selected them to use in the second stage of the classifier, for reducing confusion set. The seven moment invariants (I_1 , I_2 , I_3 , I_4 , I_5 , I_6 , and I_7) are calculated for the character images as per the above equations. Table 4.8 below lists them for a sample character in different sizes, and is plotted in Figure 4.8 below.

Table 4.8 Hu moments for a sample character in different sizes

Char Size	I1	I2	I3	I4	I5	I6	I7
20	6.600376	17.12091	23.13567	30.91836	58.38332	40.91917	42.94311
24	6.249609	15.59953	21.74801	23.6345	48.59275	32.54232	35.57808
30	6.200777	15.38347	21.95384	26.05034	50.1522	33.90114	40.19422
36	6.290162	15.46284	22.28159	26.97041	51.91079	35.02548	39.86631
40	6.155983	15.21114	21.76129	25.77891	49.75067	34.68091	37.83597
48	6.209461	15.29114	22.06333	26.8543	51.54903	34.61525	39.94634
50	6.182331	15.28769	21.82815	24.94221	49.87181	34.25902	37.04239
56	6.247388	15.51529	22.09425	26.14619	51.33234	34.57172	38.65241
58	6.209516	15.41152	21.89692	25.71418	50.17711	33.90017	38.41007
60	6.17783	15.29558	21.76888	25.46339	50.99933	33.9983	37.82138
65	6.234601	15.48571	21.99321	26.56434	51.06423	34.47571	40.43885
68	6.19956	15.38197	21.89534	26.2298	51.94944	35.58678	38.27365
72	6.413166	15.99024	22.51419	26.34163	51.53861	34.77612	40.47154

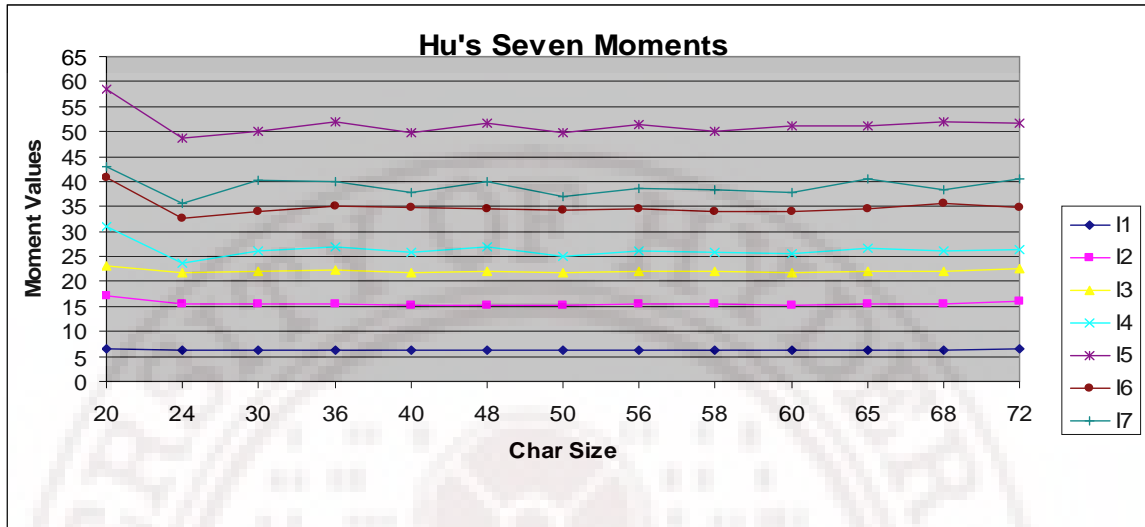


Figure 4.8 Hu moments for a sample character in different sizes

It can be seen from these that only the first 3 moments (I_1 , I_2 , I_3) are stable over the entire range of character sizes. From I_4 onwards, moment values are varying largely with size.

4.4.2.2.2 EM parameters:

Another set of features that are considered is Expectation-Maximization (EM) parameters. EM application domains include Pattern Recognition, Machine Learning, Natural Language Processing, Statistics, Data-Mining, Genetic Algorithms, Medical Imagery, and Software Engineering. A brief introduction to EM parameters is given now.

Objects in a pattern recognition application may have *missing* features due to imperfections in data collection and feature generation. Instead of discarding the cases with missing data, EM algorithm estimates the parameters that maximize the likelihood of the observed data (non-missing features). The EM algorithm is used to approximate a Probability Density Function (*PDF*). The basic idea in the EM algorithm is to iteratively estimate the likelihood given the data that is present [Duda & Hart, 1982]. It is used to classify a test sample even when it has some missing features. The approach is to first

assume that the quantity is represented as a value in some parameterized probability distribution (the popular application is a Gaussian mixture). The EM procedure, then, is:

1. Initialize the distribution parameters
2. Repeat until *convergence*:
 - * E-Step: estimate the [E]xpected value of the unknown variables, given the current parameter estimate
 - * M-Step: re-estimate the distribution parameters to [M]aximize the likelihood of the data, given the expected estimates of the unknown variables.

The choice of initial parameters technically does not matter, but in practice a poor choice can lead to a bad estimate. Convergence, though guaranteed, may take a long time to get to. In practice, if the values of the missing variables and parameters do not change significantly between two iterations, then the algorithm terminates.

Using the EM algorithm [Wajahat, 2008], we computed the parameters: Mixing proportion, Mean, and Standard deviation associated with the foreground (black) pixels of the character images. These values are stored in the original training and test databases along with the 100 crossing count features.

4.4.2.3 Feature Selection for the Second Stage

To select the suitable features for the second stage of the classifier, we experimented with different combinations of the following features: i) Crossing count (both Predominant and all the hundred) features, ii) 3 Hu's moments, and iii) EM parameters. Nearest neighbor classifier (1NN) with 10 folds-cross validation scheme is used. A problem which often occurs in training/testing the classifiers is the lack of readily available training/test data. These instances must be pre-classified which is typically time-consuming. Cross-validation is a method used in classifiers to circumvent the requirement of pre-classification. Cross-validation works as follows:

1. Separate data into fixed number of partitions (or folds)
2. Select the first fold for testing, whilst the remaining folds are used for training.
3. Perform classification and obtain performance metrics.
4. Select the next partition as testing and use the rest as training data.

5. Repeat classification until each partition has been used as the test set.
6. Calculate an average performance from the individual experiments.

Experience of many machine learning experiments suggest that using 10 partitions (tenfold cross-validation) often yields the same error rate as if the entire data set had been used for training [Roberts, 2005].

For the cross-validation tests, free software called WEKA is used in the present work. WEKA stands for the “Waikato Environment for Knowledge Analysis”. WEKA is a popular suite of machine learning algorithms written in Java, developed at the University of Waikato. It is available under the GNU General Public License. Before applying any learning algorithm, WEKA expects the data to be in ARFF format (Attribute Relation File Format). An ARFF file consists of a list of all the instances, with the attribute values for each instance being separated by commas [Witten & Frank, 2000]. So, our data is first converted to ARFF. We then used Weka's main user interface: *Explorer* [Kirkby & Frank, 2006], which is used to preprocess a dataset, feed it into a learning scheme, and analyze the resulting classifier and its performance. Using the 1NN classifier, 10-folds cross-validation tests are then conducted on the above mentioned combinations of features. Table 4.9 gives the summary of the results.

Table 4.9 Evaluation of Features

S. No.	Features	% Font Recognition (Approx)	% Character Recognition (Approx)
1	Only EM	33	16
2	Only Hu moments	59	47
3	Only PDA	78	76
4	Hu + EM	61	47
5	PDA + EM	80	77
6	PDA + 3Hu	82	78
7	PDA + EM +3Hu	82	79
8	100 CC	88	81
9	100CC+ 3Hu	89	82

Looking at the first 3 rows, one can observe that the predominant (PDA) crossing count features gave superior performance compared to *only EM values*, or *only Hu moments*.

For the combination of features, Hu moments + EM values (Row 4) are not giving any better performance than the PDA features alone. Also, addition of Hu moments to the PDA features in Row 6 (PDA +3Hu) improved the results than adding EM values (Row 5); but inclusion of EM values in Row7 (PDA + EM +3Hu) improved the results only marginally. Hence, we can summarize as Hu moments are better choice than EM values to use in combination with PDAs.

Taking all the 100 crossing count (CC) features (Row 8) clearly improved the performance. This observation leads to explore the combination of 100 CC + 3 Hu moments. Since we are now considering the second stage whose input is from the output of the first stage, and hence will consist of only a small number of characters, applying all the 100 +3 features on this small set may not be an overhead.

All the 100 crossing count features along with the 3 Hu moments are tried on 1 NN, 10- Folds (Row 9). Recognition rates of 89 % and 82% are obtained for font and character recognitions respectively. Hence, considering the 100 crossing count features resulted in a gain of 4 to 7% compared to that of PDA + 3 Hu moments. Increasing the number of features in the second stage is justified not only due to the performance gain, but also due to the fact that these features will now be applied on a small set of data. Hence, applying the 100 features on this small set in the second stage will not be an overhead considering the performance gain it offers. So, after carefully considering the merits and demerits of the performance gain versus memory requirement and computational complexity, 100 crossing count features along with the first 3 Hu moments (100CC + 3 Hu) are recommended as the features for the second stage.

4.4.2.4 Selection of Classifier for Second Stage

To select the classification method, we tried 3 types of classifiers namely: One Nearest Neighbor (1NN), Two- Nearest Neighbor (2NN), and Support Vector Machines

(SVM) classifiers, all using WEKA tool, again. *Sequential Minimal Optimization* (SMO) class implements the SMO algorithm of Platt [Platt, 1998], for training a support vector machines type of classifier.

Support Vector Machines [Vapnik, 1995; Gunn, 1998] are pair-wise discriminating classifiers with the ability to identify the decision boundary with maximal margin. Viewing input data as two sets of vectors in an n -dimensional space, an SVM will construct a separating hyper plane in that space, one which maximizes the *margin* between the two data sets. To calculate the margin, two parallel hyper planes are constructed, one on each side of the separating hyper plane, which are "pushed up against" the two data sets. Intuitively, a good separation is achieved by the hyper plane that has the largest distance to the neighboring data points of both classes, since in general the larger the margin the better the generalization error of the classifier [Wikipedia, 2009].

Sequential Minimal Optimization, or SMO is an algorithm for training support vector machines. Training a support vector machine requires the solution of a very large quadratic programming (QP) optimization problem. SMO breaks this large QP problem into a series of smallest possible QP problems. These small QP problems are solved analytically, which avoids using a time-consuming numerical QP optimization as an inner loop. SMO chooses to solve the smallest possible optimization problem at every step. For the standard SVM QP problem, the smallest possible optimization problem involves two Lagrange multipliers, because the Lagrange multipliers must obey a linear equality constraint. At every step, SMO chooses two Lagrange multipliers to jointly optimize, finds the optimal values for these multipliers, and updates the SVM to reflect the new optimal values [Platt, 1998].

Data consisting of the 100 crossing count features + the first 3 Hu moments is applied to these 3 classifiers using WEKA Explorer. Results are summarized in Table 4.10. For the character recognition, SVM classifier could not be tested on the actual data as the algorithm quickly ran out of memory even after increasing the heap size to 1 GB. This could be due the large number of classes, and/or discrete nature of the crossing count features used in our data. However, observing the performance on font recognition, it may be noted that 1 NN gives the better results over 2 NN or SMO types.

Table 4.10 Evaluation of classifiers

Features	Classifier	% Font Recognition	% Char Recognition
100CC + 3 Hu	1 NN , 10 Folds	89	82
100CC + 3 Hu	Two NN,10 folds	83	77
100CC + 3 Hu	SMO	54	---

Hence, a minimum distance (1 NN) classifier with a 103-dimension feature vector consisting of 100 crossing count features + the first 3 Hu moments is recommended for the second stage of the classifier.

4.4.2.5 Threshold Selection Revisited

After fixing the feature sets for first and second stages, the threshold distance at first stage which was set as 10.0 is reviewed. In section 4.4.2.1, effect of threshold distance is analyzed, and arrived at an optimum value of 10.0 for the first stage. Now, to verify our choice further, a similar experiment is conducted by varying the threshold value from 6 to 11 and computed the final recognition results (after second stage). Results of this experiment are tabulated below in Table 4.11. Peak recognition values are obtained at threshold= 10.0. From these results also, it can be observed that the chosen value of 10.0 is justified as the threshold distance.

Table 4.11 Effect of threshold distance on final recognition rates

S.No	Threshold Distance	% Character Recognition	% Font Recognition
1	6.0	90.87	86.88
2	7.0	92.37	88.53
3	8.0	93.09	89.04
4	9.0	93.56	89.60
5	10.0	94.37	90.17
6	11.0	93.52	89.51

4.4.3 Proposed Classifier

Based on the results of the above experiments, a two-stage classifier is proposed as follows:

Algorithm Two_Stage_Classifier ()

Begin

// **I Stage: Uses 30-D feature vector.**

1. Read the test character.
2. Compute the 100 crossing count features for the test character.
3. Select the 30 predominant features from the above.
4. Compute the Euclidian distance of the 30-D feature vector of test character to that of each of the prototype characters in the training database.
5. Obtain a set of characters whose Euclidian distance is less than, or equal to 10.0 from the test character.

// **II Stage: uses 103-D feature vector.**

6. For the set of characters obtained at the output of the first stage, get all the 100 crossing count features + first 3 Hu moments from the original training database.
7. Compute Euclidian distance for the 103 - D feature vectors between the test characters to each of the characters in the set.
8. Output the character/font with the minimum distance as the recognized character/font.

End Algorithm

4.5 Summary

In this chapter, various intrinsic issues in designing a suitable classifier for Telugu Font and Character Recognition System (TFCRS) are studied. A systematic procedure for designing the classifier is followed where several parameters that influence the recognition

efficiency are given careful consideration. Various algorithms are developed and implemented to support this work. In addition, effectiveness of rough sets in feature dimensionality reduction in Telugu OCR is also demonstrated.



Chapter 5

Proposed OCR System & Performance Evaluation

Development of a Telugu Font and Character Recognition System (TFCRS) is presented in this chapter. Our proposed system recognizes font and characters simultaneously from Telugu printed documents. Performance of the proposed system is evaluated and the results are presented. Block diagram of the proposed TFCRS is depicted in Figure 5.1.

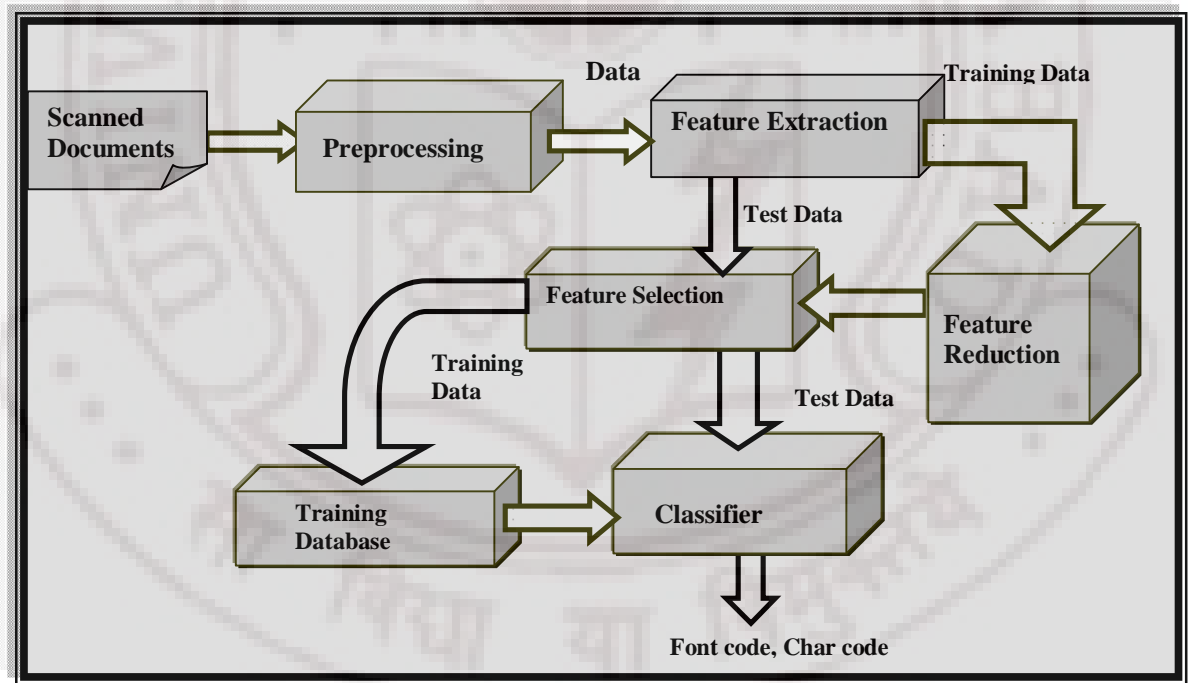


Figure 5.1 Block diagram of the TFCRS system

Proposed system integrates the following modules, details of which are given next.

1. **Preprocessing:** Prepares data from the scanned documents.
2. **Feature Extraction:** Computes the base features from the data.

3. **Feature Reduction:** Implements the *PDA* algorithm described in Chapter 3, and finds the predominant features.
4. **Feature Selection:** Given a set of attributes (by the feature reduction module) and a test dataset, extracts and outputs only the specified features from the dataset.
5. **Classification:** Designed as described in chapter 4, and outputs the recognized font and character codes of the test data.

5.1 Preprocessing

In the present work, training data is prepared from the prototype characters (Dataset #1) of the following Telugu font typefaces-Vaatha, Vennela Hemalatha, Gowthami, Amma, and Srilipi (Shree9 series, and ShreeG series). These font families are selected to represent typical variations that are expected in a printed document. For example, “Hemalatha” and Srilipi are C-DAC approved fonts; “Vartha” is a popular Telugu newspaper font; “Amma” is a font with variable stroke-widths; and “Vennela” is a font which consists of strokes rather than curved structure of normal Telugu script.

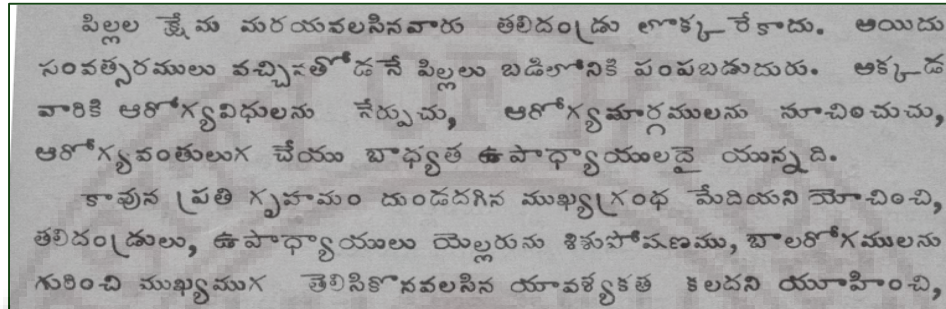
Telugu printed documents are first scanned at 300 dpi, and are then preprocessed. In the preprocessing stage, first the scanned image is binarized, connected components are extracted from it, and size-normalization is done to each connected component before extracting features. The following sections explain the details of the preprocessing steps.

5.1.1 Binarization

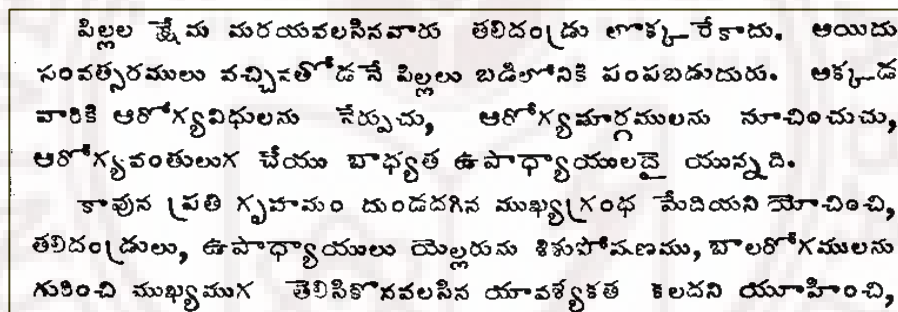
In any image analysis or enhancement problem, it is essential to identify the objects of interest from the rest. Binarization is required to separate the foreground text portion from background in the document images.

In the present work, Otsu's method [Otsu, 1979] is used to perform histogram shape-based image thresholding. The algorithm assumes that the image to be binarized contains two classes of pixels (e.g. foreground and background), it then calculates the optimum threshold separating those two classes so that their combined spread (within-class variance) is minimal. In Otsu's method we exhaustively search for the threshold that minimizes the

within-class variance, defined as a weighted sum of variances of the two classes. Figure 5.3(a) shows the scanned image of a paper document printed in Telugu. Figure 5.2 (b) is the same image after binarization in which the text (foreground) pixels are separated from the background.



(a) Original gray image



(b) Binarized image

Figure 5.2 A document image (a) before and (b) after binarization

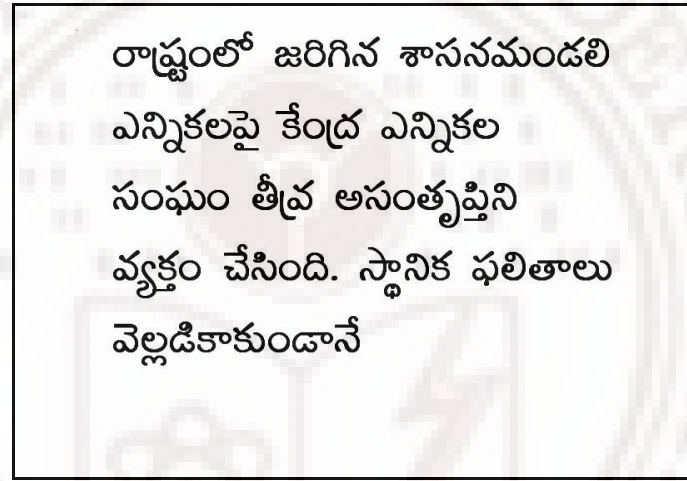
5.1.2 Connected Component Extraction

After binarization, the document image is then segmented into connected components. If the document contains some skew, a skew-correction algorithm (as explained in chapter 2) could be easily integrated into the system, if necessary. Connected components are extracted as explained below:

The algorithm scans the image from left-to-right, and top-to bottom. On the first line containing black pixels, a unique label is assigned to each contiguous run of black pixels. For each black pixel of the next and succeeding lines, neighboring pixels on the previous line, and the pixel to the left are examined. If any of these neighboring pixels has

been labeled, the same label is assigned to the current black pixel; otherwise, the next unused label is assigned. This procedure continues till the bottom line of the image. After the scanning process, the labeling is completed by unifying conflicting labels, and reassigning unused labels.

The above algorithm is applied using 8-connectivity on the document image, and all the connected components are extracted. Figure 5.3 shows an input document (a), and the connected components highlighted with rectangular boundaries of different colors/gray shades in (b).



(a)



(b)

Figure 5.3 (a)An input document, and (b) its connected components highlighted with rectangular boundaries

5.1.3 Size Normalization

Our recognizable units are *connected components* as mentioned in 3.5.2. Since test characters are expected to occur in different sizes, it is required to apply size-normalization. Size normalization in character recognition is a transformation of an input character image of arbitrary size into an output character image of a pre-specified size. Size-normalization is essential because each input feature vector to the classifier must be of fixed length. Size normalization involves image dimension changes, which implies re-sampling of the image pixels by different factors in the orthogonal dimensions. So, each connected component is required to be brought to a uniform size in this step. A value of 50X50 is selected for the uniform size. This value is arrived at by using the heuristics that a lower value may not capture all the essential information; and a higher value may actually reduce the accuracy due to artifacts. With the orthogonal complexity, Telugu characters pose special problems like shape variations with scaling. Hence, for the purpose of good classification, an extremely well representation of the discriminating features is required. A value of 50 x 50 is suitable for the crossing count features used in this work.

Let us illustrate the transformation of a character image of size $W_1 \times H_1$ into a normalized image of size $W_2 \times H_2$. W_1 , and H_1 are width and height of original image; W_2 , and H_2 are width and height of image after normalization. The transformation can be accomplished by forward mapping or backward mapping. Let us denote the original image as $f(x, y)$ and the normalized image as $g(x', y')$. The normalized image is generated by the following coordinate mapping:

Let α and β denote the ratios of transformation, given by

$$\alpha = W_2 / W_1,$$

$$\beta = H_2 / H_1.$$

$$\left. \begin{array}{l} x' = \alpha x \\ y' = \beta y \end{array} \right\} \text{ for forward mapping}$$

$$\left. \begin{array}{l} x = x' / \alpha \\ y = y' / \beta \end{array} \right\} \text{ for backward mapping}$$

In forward mapping, x and y are discrete but x' and y' are not necessarily discrete; while in backward mapping, x' and y' are discrete but x and y are not necessarily discrete. Further, in forward mapping, the mapped coordinates (x', y') do not necessarily fill all pixels in the normalized plane. Therefore, coordinate discretization or pixel interpolation is needed in the implementation of normalization [Liu et al, 2004]. By discretization, the mapped coordinates approximate with the closest integer numbers. Let (x_0, y_0) and (x_1, y_1) be the coordinates of upper-left, and lower-right corners of a character in the original plane. After normalization, they become $(1, 1)$ and (x', y') respectively. This is illustrated in Figure 5.4.

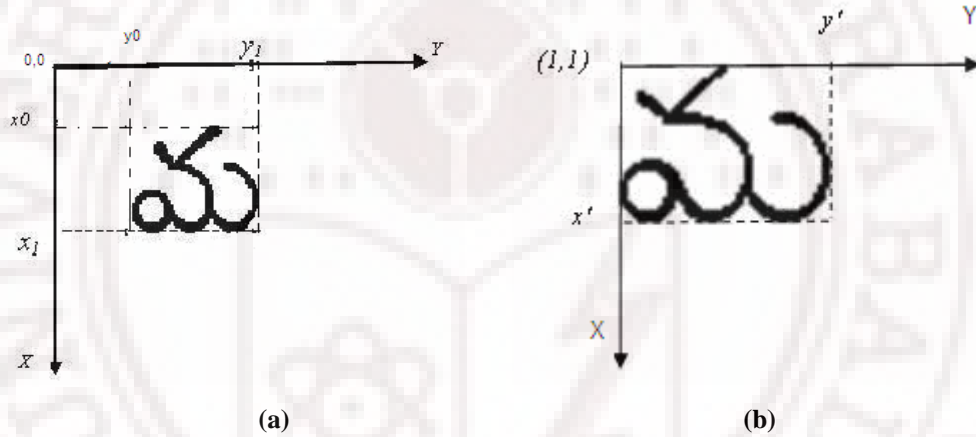


Figure 5.4 A character image (a) before, and (b) after normalization

5.1.3.1 Linear Normalization

For an image of size $W \times H$ to scale it to a size of $NewW \times NewH$, the usual linear normalization method computes the new coordinates $Newx$, and $Newy$ as the following:

Algorithm Linear_Norm ()

// W, H width and height of original image ; $NewW, NewH$ width and height of image after normalization. x, y and $Newx$, and $Newy$ are the coordinates before and after normalization.

Begin

For $x=1$ to $NewH$
{


```

For y=1 to NewW
{
Newx = Round ((x/ NewH)*H);
Newy = Round ((y/ NewW)* W);
Intensity [x, y] = Intensity [Newx, Newy];
}
}
End

```

The above linear normalization computation introduces some holes (white pixels) in the image due to rounding operations. Some form of smoothing operations is required to fill the gaps. Figure 5.5 illustrates this.

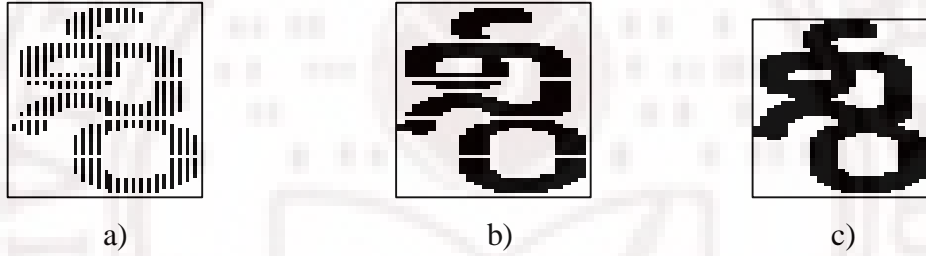


Figure 5.5 An image resized with a) linear normalization, b) after column-wise smoothing, c) after row and column-wise smoothing

Our target image size is square (50X50), but the input images are of any size. Usually the input image aspect ratio does not match with that of target image in these methods. The resulting images contain discretization errors (due to rounding operation), and are also distorted. This causes large drifts in the zero crossing count features. The normalization algorithms available in literature are found unsuitable for the feature set used in this work. We used Dataset # 3, and tested the crossing counts from the images scaled to 50X50 using the following 3 methods: Nearest-neighbor interpolation, Bi-linear interpolation, and Bi-cubic interpolation.

5.1.3.2 Nearest Neighbor Interpolation

Nearest Neighbor Interpolation determines the grey level value from the closest pixel to the specified input coordinates, and assigns that value to the output coordinates. This

method does not really interpolate values; it just copies existing values. Since it does not alter values, it is preferred if subtle variations in the gray level values need to be retained.

5.1.3.3 Bi-linear Interpolation

Bi-linear Interpolation determines the value of a new pixel based on a weighted average of the pixels in the nearest 2×2 neighborhood of the reference pixel in the original image.

5.1.3.4 Bi-cubic Interpolation

In the Bi-cubic Interpolation, new pixel value is a bi-cubic function using the weighted average of the 16 pixels in the nearest 4×4 neighborhood of the reference pixel in the original image. Here, two cubic interpolation polynomials, one for each plane direction, are used.

For testing the suitability of normalization methods, the images were taken from 24, 36, and 48 sizes from Dataset #3. These are scaled independently to 50×50 sizes with each of the above methods. From these normalized images, 100 zero-crossing counts in vertical and horizontal directions ($50 + 50$) were computed. The original 72- sized training character images are also scaled to 50×50 using the same methods. The 100 crossing counts of both are then compared. Results for normalizing the size of a single character (Figure 5.6) are plotted in Figure s 5.7 (a,b,c).

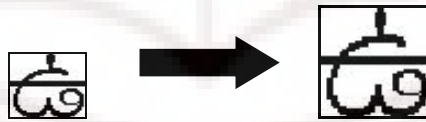
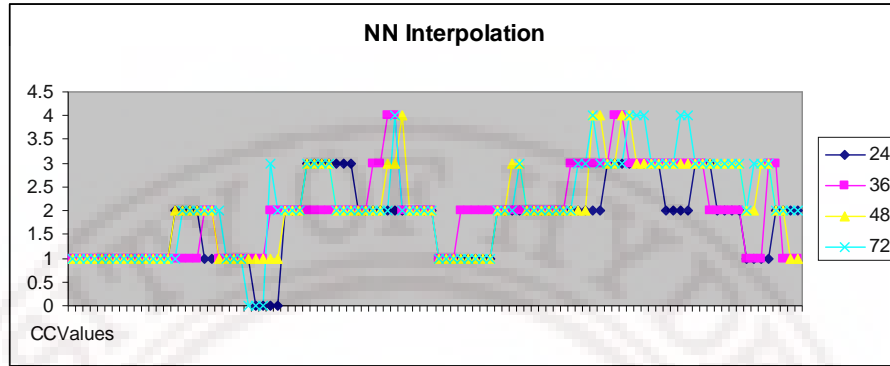


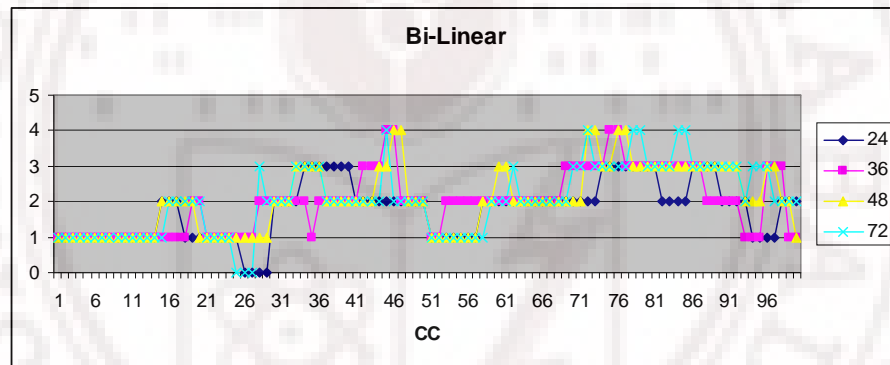
Figure 5.6 A character image of 34×36 size normalized to 50×50

Figure 5.7 (a) represent the 100 crossing counts using Nearest Neighbor Interpolation (NN) method for all the 24, 36, 48, and 72 sizes. Figure 5.7 (b) is by using Bi-linear Interpolation method, and Figure 5.7 (c) is by using Bi-cubic Interpolation method. The 100 Crossing Count (CC) features vary on X- axis from 1 to 100, while the values of these 100 CC features vary on Y axis from 1 to 4. All the above interpolation methods produced large variations in the zero-crossing features (CC values on Y axis) when

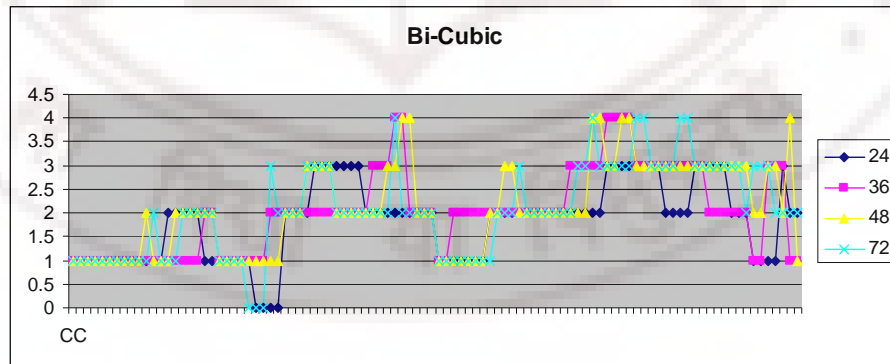
compared with the CC values of 72 sized images scaled down to 50X50. For example, in Figure 5.7 (a, b, c), all the four curves are drifting from one another at several places.



(a)



(b)



(c)

Figure 5.7 Crossing counts using (a) Nearest Neighbor, (b) Bi-linear, and (c) Bi-Cubic Interpolation methods

5.1.3.5 Proposed Normalization Method

From the Figures 5.7 (a, b, c), large variations in the crossing count values (Y-axis) can be seen among the 4 selected sizes in all the 3 methods. Hence, we developed a novel normalization method in which the input image is scaled twice using linear normalization technique. Our algorithm zooms up the input image to a large size (150X150), and again zooms down to the required size of 50X50. The idea is to smooth out any irregularities that may arise due to rounding operations. This double- scaling is shown in Figure 5.8 for a sample character.

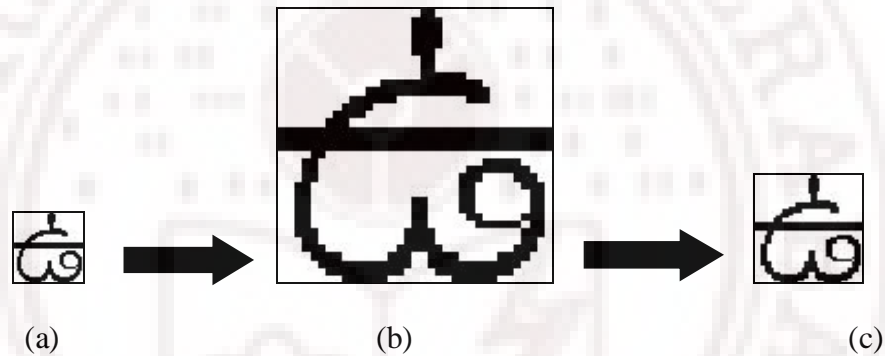


Figure 5.8 An image of size (a) 34X36 zoomed up to (b) 150 X 150, down to(c) 50 X50

The 100 crossing count features from these normalized images are found to be uniform mostly. This is shown in Figure 5.9 for the same sample character. In contrast with Figures 5.7 (a,b,c) , variations in 100 crossing counts are more or less consistent in all the four sizes (24, 36, 48 and 72). For example, there is no difference in the values of crossing counts from 1 to 16 (on X- axis) and all have the same value ($y = 1$) in all the four cases. Between 16 and 21, deviation is observed only in two cases. Again, till 28, all have the same crossing count values. Whenever there is any change, all the four curves are changing almost in unison. This is in contrast with the curves in Figures 5.7 (a, b, c) where crossing count values are changing randomly for the four cases. Hence, this double scaling technique is chosen as the normalization method.

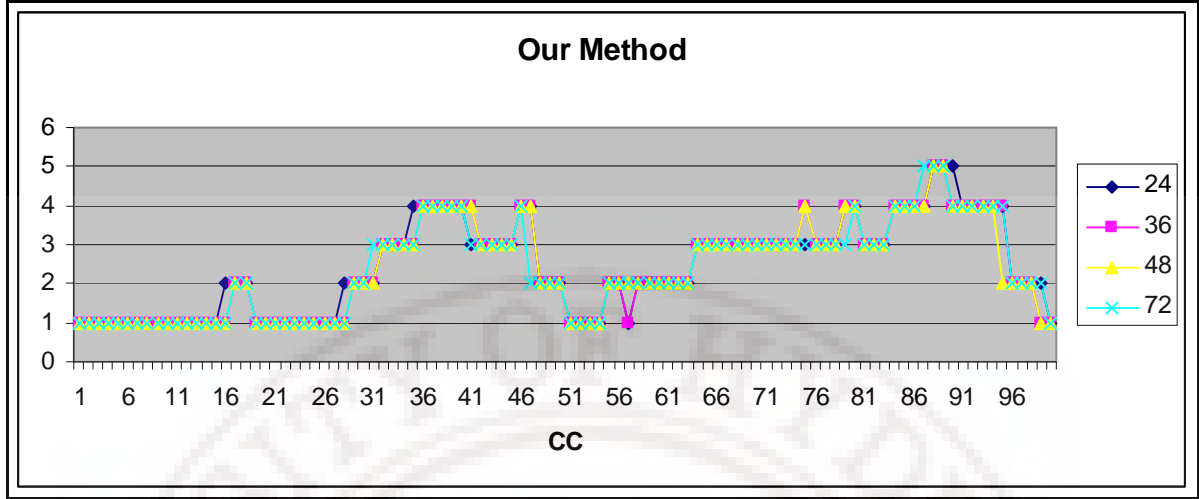


Figure 5.9 100 crossing counts using our method

Thus, in our proposed system, preprocessing stage contains Binarization, Connected component extraction, and Size normalization operations that are just required to prepare the data for the subsequent stages. Thinning and other elaborate pre-processing steps are not used, making the system fast and efficient.

5.2 Feature Extraction

In the present work, black-to-white crossing counts in both horizontal and vertical directions are chosen as the main features. Since all the characters are normalized to 50 x 50 size, this algorithm computes the feature vectors constituting the 50 horizontal crossing-counts from 50 rows and another 50 from the 50 columns. Horizontal crossing-counts are stored as HCV vector; and vertical crossing-counts as VCV vector. A total of 2930 prototypes are used for training from all the fonts.

In addition to crossing count features, this algorithm also computes the first 3 Hu moments, which are required in the second stage of the classifier.

5.3 Feature Reduction

The complete training database dimensions are thus 2930 X 103. Excluding the 3 moment features, database contains 100 crossing count features on which dimensionality reduction is to be applied. The PDA algorithm explained in Section 3.5.5 is applied on this

data for each of the decision attributes *Fontcode* and *Charcode* for computing reducts, as mentioned in Section 4.3.1. The reduced feature sets are reproduced here.

For *font code*, the algorithm produced 23 features as predominant attributes for classifying all the fonts in the training set. They are : 'HCV9' , 'HCV12', 'HCV16' , 'HCV24', 'HCV34' , 'HCV38', 'HCV42', 'HCV44', 'HCV45', 'HCV46' , 'VCV3' , 'VCV6', 'VCV10', 'VCV16', 'VCV22', 'VCV29', 'VCV31', 'VCV33', 'VCV41' , 'VCV43', 'VCV45', 'VCV48', and 'VCV50' (Total 23).

Similarly, the algorithm produced 21 features as predominant attributes for *charcode*. These 21 features that are adequate for classifying all the characters in the training set are : 'HCV11' , 'HCV16', 'HCV24', 'HCV31', 'HCV38', 'HCV45', 'HCV46', 'VCV6', 'VCV10', 'VCV12', 'VCV15', 'VCV19', 'VCV22', 'VCV27', 'VCV29', 'VCV33', 'VCV41', 'VCV43', 'VCV47', 'VCV48' , and 'VCV50' (Total 21).

Of these 44 features, 14 are common for both. Hence, the combined predominant feature set consists of only **30** features.

5.4 Feature Selection

Given a set of attributes (by the feature reduction module) and dataset, this algorithm extracts and outputs the required features from the dataset. As discussed in 4.4.1, two neighboring features on either side of each of the 30 predominant features are extracted; their average is computed, and rounded to the nearest lower integer (floor) as per Equations 4.1, and 4.2. For the first stage of the classifier, it sends the 30 crossing counts only. For the second stage, the algorithm sends all the 100 crossing counts + the first 3 Hu moments.

5.5 Training Database

Original training database dimensions are 2930 X 103. With the 30 predominant features, a new database is now created. Table 4.5 in Section 4.3.1 shows a sample database for the reduced feature set.

5.6 Classification

Classification is done by the two- stage classifier as designed in section 4.4.3. Our first stage of the classifier is set with a threshold distance of 10, with 30 predominant features. At the second stage, we added 100 crossing count features + 3 Hu moments. Figure 5.10 shows the block diagram of the two-stage classifier designed as mentioned in section 4.4.3.

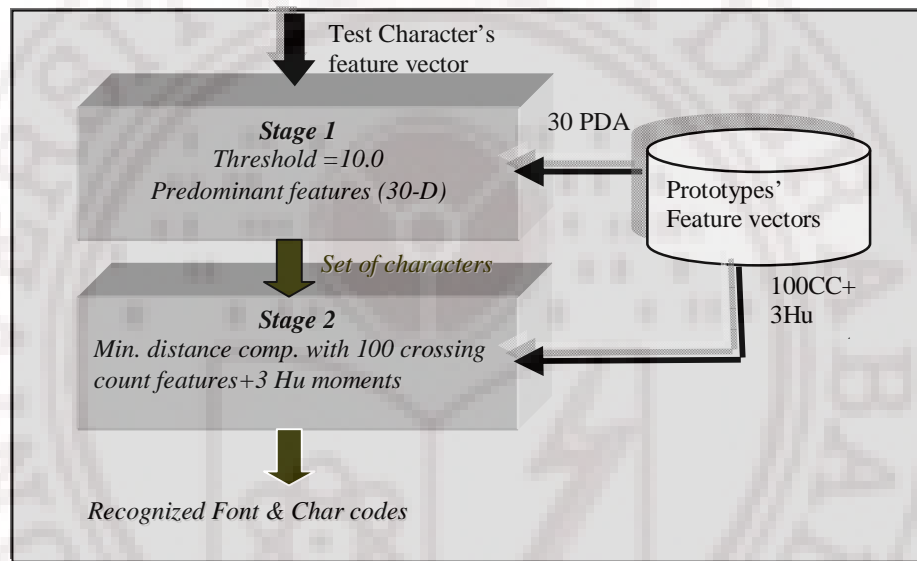


Figure 5.10 Block diagram of the 2-stage classifier

Feature vector with 30 predominant features for each test component is applied to the first stage. An Euclidian-distance classifier compares the distance between the 30-D feature vectors of test character and prototype characters. It outputs the characters/ fonts from the prototypes, whose distance is less than or equal to the threshold value 10.0. For this output set of characters, feature vectors with the 100 crossing counts + 3 Hu moments is formed. In the second stage, a minimum distance classifier compares the distance between the 103 features of the test character with that of prototype characters. Characters/fonts, with minimum distance are produced as the recognized ones.

5.7 Performance Evaluation

The proposed TFCR system designed as explained above is then tested for its performance evaluation. Results of these tests are presented in this section.

Performance of an OCR system is highly dependent on the quality of the input, but no standardized test sets exist for Telugu character recognition. This makes it difficult to evaluate and compare different systems. Still, recognition rates are often given, and usually presented as the percentage of characters correctly classified. However, this does not say anything about the errors committed. Hence, three different performance criteria are used in rating any OCR systems [Line, 1993]:

- i. **Correct Recognition Rate:** The proportion of correctly classified characters from the total number of test characters.
- ii. **Rejection Rate:** The proportion of characters which the system was unable to recognize. Rejected characters can be flagged by the OCR system, and therefore are easily retractable for manual corrections.
- iii. **Error Rate:** The proportion of characters that are erroneously recognized. Misrecognized characters require manual detection and correction.

There is usually a tradeoff between different recognition rates. A low error rate may lead to a higher rejection rate and a lower recognition rate. Because of the time required to detect and correct OCR errors, error rate is the most important factor in evaluating an OCR system for its cost-effectiveness. The rejection rate is less critical. An example from barcode reading may illustrate this. Here a rejection while reading a bar-coded price tag will only lead to rescanning of the code or manual entry, while a wrongly coded price tag might result in the customer being charged for the wrong amount. In the barcode industry, error rates as low as one in a million labels and rejection rates of one in a hundred are acceptable. Recognition rate is highly affected by many factors, such as the font type, size, quality of printing, thickness of the paper, background of text, resolution of scanner and even the slight skewing of the text image to be scanned. In view of this, it is clear that measuring the performance of OCR systems only by the recognition rate is not appropriate.

5.7.1 Results

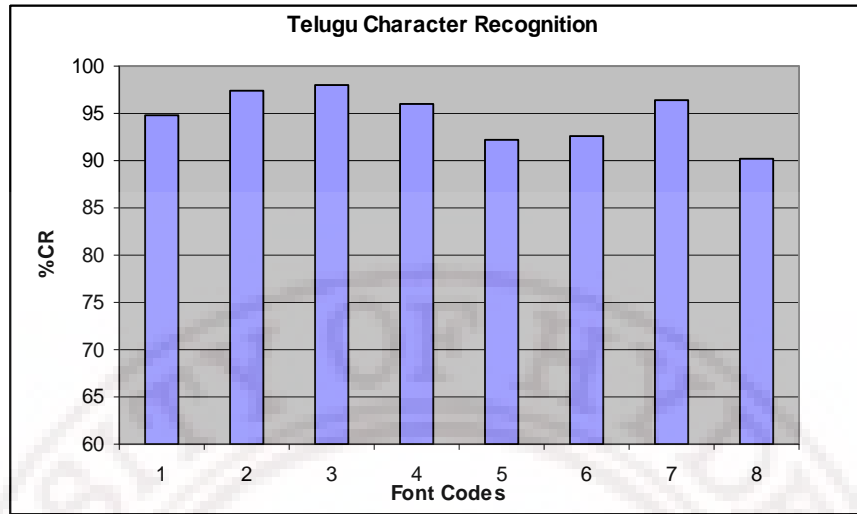
The developed system is tested with Dataset # 2 of section 4.1.2. Our test data set contains scanned and segmented characters from news papers, books and also laser-printed characters with different sizes. Total test characters are in sizes that range from as low as 9 points to 72 points, as given in Table 4.3. Table 5.1 summarizes the results obtained with the two-stage classifier designed as above.

Table 5.1 Performance results

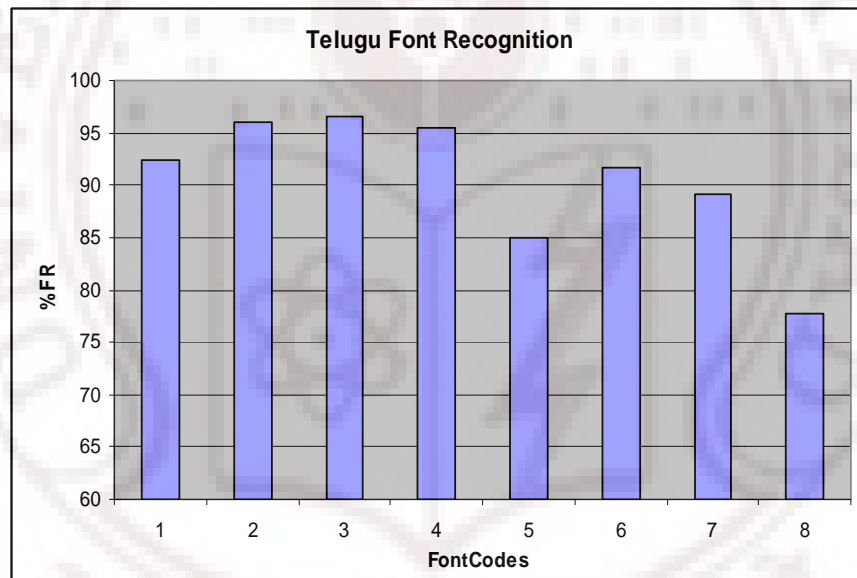
Test	Total	Correctly recognized	Rejected	Misrecognized	Recognition	Rejection	Substitution Error
Character Recognition	10,166	9,594	09	563	94.37%	0.09%	5.54 %
Font Recognition	10,166	9,167	09	990	90.17%	0.09%	9.74 %

The present work achieved 94.37%, and 90.17% accuracies for character and font recognitions respectively. These results are obtained without applying any post processing operations. Still, they show good recognition values with low rejection rates. A rather high substitution error values, appeared to be due to the poor quality of the data, are analyzed in Section 5.8, in more detail.

Character and font recognition values are plotted in Figures 5.11(a) and (b) for different fonts in the test data. Variation in character recognition (%CR) among different fonts is from 90 to 98, whereas variation in font recognition (%FR) among different fonts is from 77 to 96. Relatively low recognitions are observed for Font code=8, where most of the test data is from old books.



(a)

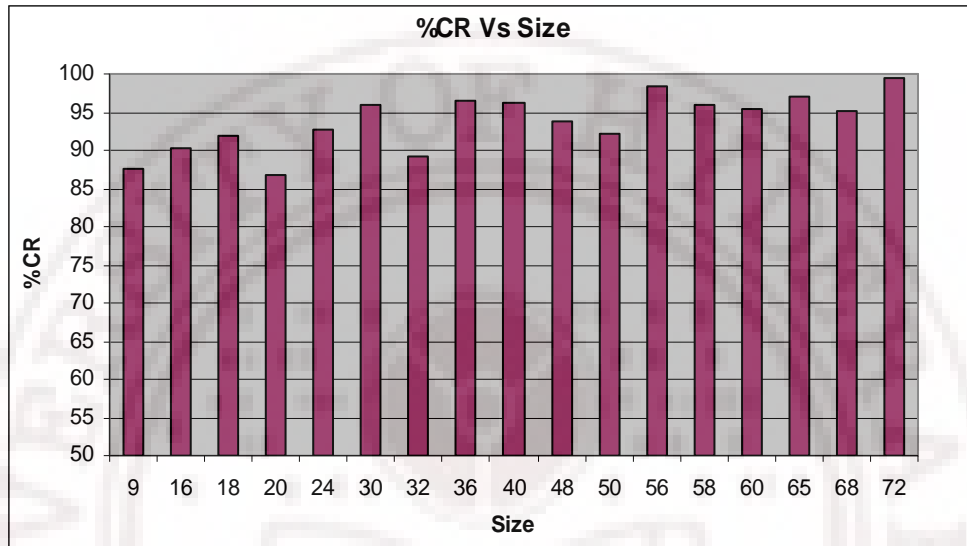


(b)

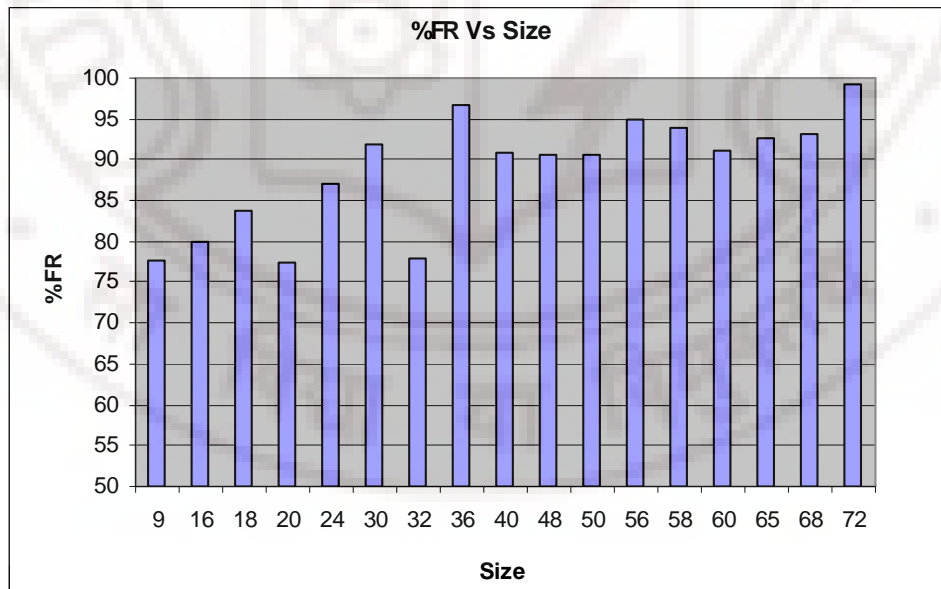
Figure 5.11 Character and Font Recognitions for different fonts

Recognition values for test characters of different sizes are plotted in Figures 5.12(a), and (b). Character recognition (%CR) varies from 88 to 99.6, and font recognition (%FR) varies from 76 to 99.3 among different sizes. Font recognition values are homogeneous in the sizes between 36 and 72. At sizes smaller than 36, font recognition is non-homogeneous and is also less than those at larger sizes. Features that discriminate one

font among other fonts might not have got captured at the smaller sizes. So, small size of the test characters appears to be a detrimental factor for good font-recognition.



(a)



(b)

Figure 5.12 Character and Font Recognitions for different sizes

In all the cases, character recognition is found to be better than font recognition, which indicates the complexity of recognizing fonts in the printed documents. Section 5.8 presents a critical analysis on the substitution errors along with other observations, in detail.

5.7.2 Comparative Study with Earlier Works

Performance of the proposed method is now compared with relevant earlier works in Telugu. This comparison is based on the type of features used, recognition rates, test data size, test character sizes and font types.

- * Pujari, Naidu and Jinaga proposed a recognizer that relies on wavelet multi-resolution analysis [Pujari, Naidu & Jinaga, 2002]. The system was tested on 444 characters from different fonts. Signatures of each of the images are computed with different filter banks. Characters recognition accuracy is reported to be 93.46% using Battle-Lemarie filter. According to the authors, the same system produced low recognition rate when applied to recognize English characters since the directional features which are prevalent in Latin scripts are not preserved during signature computation. Font recognition is not addressed in this work.
- * Negi , Bhagvati and Krishna proposed a Telugu OCR in which, template matching is used [Negi , Bhagvati & Krishna,2001]. Test data consists of 2,524 components with sizes varying from 9pts to 22 pts. Training characters are taken from *TL-TT Hemalatha* font. Testing is done for character recognition in *TL -TT Harshapriya* font, a newspaper font called *eenadu* and some unknown fonts from books and novels. Raw OCR accuracy of 92% is reported. Font recognition is not addressed.
- * Lakshmi and Patvardhan reported a multi-font OCR [Lakshmi & Patvardhan, 2002]. Pixel gradient directions are chosen as the features. K-nearest neighbor classifier is used. Testing is done on characters with different sizes, and also with some different fonts. More than 92% accuracy for most of the images is reported. Font recognition is not addressed.
- * The present work achieved 94.37% and 90.17% accuracies for character and font recognitions respectively. Our test data contains 10,166 characters in several font types (Dataset #2). Test data sizes are varying over a wide range from 9 pts to 72

pts. Applicability of the proposed method is also tested on English character recognition with very promising results. Raw accuracies of 96% and 92% are obtained for English character and font recognitions respectively, without using any post processing like spell-checking. Details of this case study are presented in Chapter 6. These observations are summarized in the following table, Table 5.2.

Table 5.2 Comparison of Results

Authors	Features	Test data Type	Test data size	Test character sizes	Char Rec	Font Rec	Other Remarks
Pujari et al (2002).	Wavelet representation	Pages from different books	444	Not available	93.46 %	-----	Very low recognition rate for English
Negi, et al (2001)	Template matching	Newspapers, books and laser printed output.	2,524	9 pts to 22 pts.	92%.	-----	-----
Lakshmi & Patvardhan (2002)	Real-valued feature vector containing the seven moment values	Images added with synthetic noise	1544	25, 30 and 35pts	92%	-----	-----
Our approach	Crossing counts, Hu moments	Newspapers, books, and laser prints.	10,166	9 pts to 72 pts.	94.37 %	90.17%	Script-independent recognition (Good recognition rates for English)

5.8 Observations & Findings

The results of our approach shown in Tables 5.1 are without any post processing. Still, they show good recognition values with low rejection rates. After analyzing the wrongly recognized data (Substitution errors), the following observations are made.

- i. Most of the substitution errors are found to be due to the touching characters. Two or more connected components touch each other due to the poor quality of the documents (especially newspapers and old books). Then, the segmentation (connected component extraction) algorithm can not separate them correctly into individual components. Some of these are shown below in Figure 5.13. These poorly segmented characters are one main reason for recognition errors.



Figure 5.13 Some touching characters

- ii. Some pairs of characters appear very similar in shape. Unless the print quality is good and preprocessing stage is efficient, they are many times recognized wrongly as each other. This degrades the performance in both character and font recognitions. Some of these characters are shown in Figure 5.14.

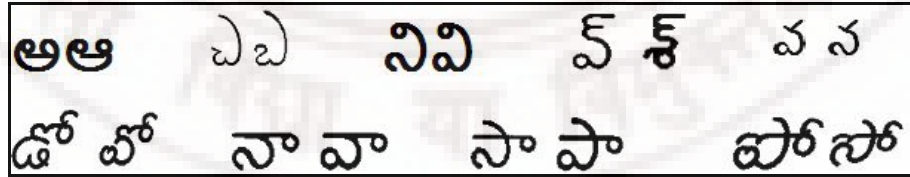


Figure 5.14 Some similar-looking character pairs

- iii. Sometimes, imperfections in scanning and binarization stages result in broken characters as shown in Figure 5.15. Since part of these characters is missing, recognition becomes difficult. Depending on which part of the character is missing,

and where the gaps occur, these are either recognized wrongly or not recognized at all. In general, a smoothing algorithm that follows the binarization stage will help bridging the gaps and improve the image quality.

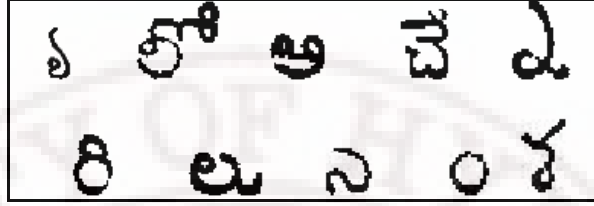


Figure 5.15 Some broken characters

Performance is also measured on clear documents of good quality, and the results are presented in Table 5.3. In this, we excluded newspapers, but retained pages from good quality books. As can be seen in Table 5.3, substitution errors are greatly reduced, and recognition accuracies are increased for clear data. This emphasizes the effect of touching and broken characters, and hence the need for good quality input for better performance.

Table 5.3 Performance results on clear data

Test	Total	Correctly recognized	Rejected	Misrecognized	Recognition	Rejection	Substitution Error
Character Recognition	6494	6289	04	201	96.84%	0.06%	3.1%
Font Recognition	6,494	6,129	04	361	94.38%	0.06%	5.56%

These observations suggest that a good preprocessing stage that will enhance the quality of the document image solves most of the recognition errors that are encountered here. Hence, further improvement in the performance of the developed method can be achieved by adopting more robust preprocessing techniques, and some additional post-processing operations.

- iv. Several characters appear very similar in two or more fonts. Table 5.4 illustrates this clearly. Same character (that also appear the same) in different fonts is shown in each row. These characters appear more or less the same, except in thickness in some cases, in all the fonts shown. It is difficult to identify the font codes, even visually, from these cases. Hence, what makes objects (characters) differ in their fonts is not very clear, and the concept of a *font* need to be established more clearly. This explains the higher substitution error in font recognition compared to that of character recognition. Incorporating font-variant features may enhance the font recognition.

Table 5.4 Same character(s) in different fonts *

Hemalatha (FC=3)	Gowthami (FC=2)	Srilipi (FC=7)	Vartha (FC=5)
	ಲ	ಲ	ಲ
ಲ	ಲ	ಲ	
	ಇ	ಇ	
ಈ	ಈ	ಈ	ಈ
ಲ	ಲ	ಲ	ಲ
ಜ	ಜ	ಜ	ಜ
	ಁ	ಁ	
ಐ		ಐ	
	ಏ	ಏ	
ಎ	ಎ	ಎ	ಎ
ಠ			ಠ
ಬ	ಬ	ಬ	ಬ
	ಱ	ಱ	

* FC=Fontcode

An interesting observation about the proposed system is that the entire method is content-independent. That is, no explicit local feature analysis is used in the entire work. Specifically, zero crossing features which are used in this work are not only size invariant, they are also independent of the underlying script. Other features used in the present work are Hu moments; describing a character with moments implies that global properties of the character are used rather than local properties. Thus, our proposed approach doesn't use any local / structural features that depend on the language used in the printed document. Examples of such structural features are: Shirorekha/Headline for Hindi, stroke features for English and Chinese characters, and curved features of some Indian languages like Kannada, Telugu etc. Not using such features makes the proposed approach a global and content-independent one.

This content-independent property of the developed methodology is particularly suitable for proposing a general framework for character and font recognitions. Applicability of the proposed approach in the script-independent recognition is demonstrated on English, which is the subject matter of the next chapter.

Chapter 6

Testing the Generality of the Approach: Case Study on English

Since the proposed approach does not use the knowledge about the script used in the printed documents, we now attempt to suggest a new technique for script-independent recognition. The content-independent property of our approach observed in the previous chapter (Section 5.8) is tested for its generality on a popular language, English. We applied the same methodology to simultaneously recognize fonts and characters printed in English. Optical character recognition for English is one of the most successful applications of automatic pattern recognition. To the best of our knowledge, simultaneously recognizing English fonts and characters is not attempted by earlier researchers. The details and results of this case study are described now.

Proposed system integrates the same modules shown in Figure 5.1 of Chapter 5.

6.1 Preparation of Data and Preprocessing

The following datasets are used for training, designing, and testing purposes in the present work.

6.1.1 Dataset for Training (Dataset # 1)

For training set prototypes, binary images are created from the following five fonts:

1. Arial,
2. Comic sans MS,
3. Monotype Corsiva,
4. Times new Roman, and
5. Verdana.

Training set is prepared from 26 lowercase and 26 upper case letters from these five fonts. Training characters size is fixed at 36 points. Total number of prototype characters is then $52 \times 5 = 260$. Table 6.1 shows the characters in these fonts.

Table 6.1 **Prototype characters from the 5 selected fonts**

FONT NAME	UPPER-CASE	LOWER CASE
Arial	ABCDEFGHIJKLMNOPQRSTUVWXYZ	abcdefghijklmnopqrstuvwxyz
Comic Sans	ABCDEFGHIJKLMNOPQRSTUVWXYZ	abcdefghijklmnopqrstuvwxyz
Monotype corsiva	ABCDEFGHIJKLMNOPQRSTUVWXYZ	abcdefghijklmnopqrstuvwxyz
Times New Roman	ABCDEFGHIJKLMNOPQRSTUVWXYZ	abcdefghijklmnopqrstuvwxyz
Verdana	ABCDEFGHIJKLMNOPQRSTUVWXYZ	abcdefghijklmnopqrstuvwxyz

6.1.2 Dataset for Testing (Dataset # 2)

For testing, character images are extracted from scanned documents in different sizes in all the five selected fonts. Sizes from 24 to 46 are used with a total of 1560 characters. All these are assigned their respective font codes, and character codes. This constitutes Dataset #2, which is used for testing. Table 6.2 presents the number of test characters in each size, in each of the five fonts.

Table 6.2 **Number of Test characters (Dataset # 2): Size-wise, Font-wise**

Font Name	Font code	Size						
		24	26	30	36	40	46	Total
Arial	100	52	52	52	52	52	52	312
Comic Sans MS	200	52	52	52	52	52	52	312
Monotype Corsiva	300	52	52	52	52	52	52	312
Times new Roman	400	52	52	52	52	52	52	312
Verdana	500	52	52	52	52	52	52	312
Total		260	260	260	260	260	260	1560

6.1.3 Dataset for Designing (Dataset # 3)

For fine-tuning certain parameters as in TFCRS, a small set of characters are used during the design phase. About 100 characters are created as test characters from all the five chosen fonts for this purpose. These character images are computer-generated in different sizes. These 100 characters constitute Dataset # 3, which is used only during the design phase to fine-tune some of the design parameters.

All these character images are then normalized to 50x50 sizes using the algorithm explained in section 4.2.

6.2 Feature extraction

After each character is normalized to 50X50 sizes, black-to-white transitions in both vertical and horizontal directions are computed for each row and column. Since all the characters are of 50 x 50 sizes, this gives rise to a feature vector of 100 dimensions for each character. The first 3 Hu moments, which are required in second stage of the classifier, are also computed along with zero-crossing features for these normalized images.

Table 6.3 shows the feature vectors of 103-Dimension for five characters from Arial font. Each feature vector consists of 100 crossing counts (HCV 1 to 50, VCV 1 to 50) + 3 Hu moment values (H1, H2, H3). In order to find the *reducts* and obtain predominant attributes from the 100 crossing count features, a decision table is now created with 260 rows and 102 columns (100 crossing count feature values + 2 decision attributes: “*Charcode*” and “*Font code*”).

Table 6.3 103-D feature vectors for five characters

S. No	Features	A	B	C	D	E
1	HCV1	1	1	1	1	1
2	HCV2	1	1	1	1	1
3	HCV3	1	1	1	1	1
4	HCV4	1	1	1	1	1
5	HCV5	1	1	1	1	1
6	HCV6	1	1	1	1	1
7	HCV7	1	1	1	1	1
8	HCV8	2	2	2	2	1

9	HCV9	2	2	2	2	1
10	HCV10	2	2	2	2	1
11	HCV11	2	2	2	2	1
12	HCV12	2	2	2	2	1
13	HCV13	2	2	2	2	1
14	HCV14	2	2	2	2	1
15	HCV15	2	2	2	2	1
16	HCV16	2	2	2	2	1
17	HCV17	2	2	2	2	1
18	HCV18	2	2	1	1	1
19	HCV19	2	2	1	1	1
20	HCV20	2	2	1	1	1
21	HCV21	2	1	1	1	1
22	HCV22	2	1	1	1	1
23	HCV23	2	1	1	1	1
24	HCV24	2	1	1	1	1
25	HCV25	2	1	1	1	1
26	HCV26	2	1	1	1	1
27	HCV27	2	1	1	1	1
28	HCV28	2	2	1	1	1
29	HCV29	2	2	1	1	1
30	HCV30	1	2	1	1	1
31	HCV31	1	1	1	1	1
32	HCV32	1	1	1	1	1
33	HCV33	1	1	2	1	1
34	HCV34	1	1	2	1	1
35	HCV35	1	1	1	2	1
36	HCV36	2	1	1	2	1
37	HCV37	2	1	1	2	1
38	HCV38	2	1	2	2	1
39	HCV39	2	1	2	2	1
40	HCV40	2	1	2	2	1
41	HCV41	2	2	2	2	1
42	HCV42	2	2	2	2	1
43	HCV43	2	2	2	2	1
44	HCV44	2	2	2	2	1
45	HCV45	2	1	1	1	0
46	HCV46	2	1	1	1	0
47	HCV47	2	1	1	1	0
48	HCV48	1	1	1	1	0
49	HCV49	1	1	1	1	0
50	HCV50	1	1	1	1	0
51	VCV1	0	0	1	0	0
52	VCV2	0	0	1	0	0
53	VCV3	0	0	1	0	0
54	VCV4	0	0	1	0	0
55	VCV5	0	0	1	0	0
56	VCV6	0	0	1	0	0
57	VCV7	0	0	1	0	0
58	VCV8	0	0	1	0	0
59	VCV9	1	0	2	1	0
60	VCV10	1	0	2	1	0
61	VCV11	1	2	2	1	2
62	VCV12	1	2	2	1	2
63	VCV13	1	2	2	1	2

64	VCV14	1	2	2	1	2
65	VCV15	1	2	2	1	2
66	VCV16	1	2	2	1	2
67	VCV17	1	2	2	1	2
68	VCV18	2	2	2	1	2
69	VCV19	2	2	1	1	2
70	VCV20	2	2	1	1	2
71	VCV21	2	2	1	1	2
72	VCV22	2	2	1	1	2
73	VCV23	2	2	1	1	2
74	VCV24	2	2	1	1	2
75	VCV25	2	2	1	1	2
76	VCV26	2	2	1	1	2
77	VCV27	2	2	1	1	2
78	VCV28	2	2	1	1	2
79	VCV29	2	2	1	1	2
80	VCV30	2	2	1	1	2
81	VCV31	2	2	1	1	2
82	VCV32	2	2	1	1	2
83	VCV33	2	2	1	2	2
84	VCV34	2	2	1	2	2
85	VCV35	1	2	1	2	2
86	VCV36	1	2	2	2	2
87	VCV37	1	2	2	2	2
88	VCV38	1	3	2	2	2
89	VCV39	1	3	2	2	2
90	VCV40	1	2	2	2	2
91	VCV41	1	2	2	2	2
92	VCV42	1	2	2	2	2
93	VCV43	1	2	2	1	2
94	VCV44	0	2	2	1	2
95	VCV45	0	2	2	1	2
96	VCV46	0	2	2	1	2
97	VCV47	0	2	2	1	1
98	VCV48	0	2	2	1	1
99	VCV49	0	1	2	1	0
100	VCV50	0	1	1	1	0
101	H1	0.397006	0.358516	0.585406	0.492398	0.442421
102	H2	0.004462	0.010849	0.023392	0.006873	0.034557
103	H3	0.041203	0.000404	0.007775	0.003725	0.002078

6.3 Feature Reduction

The PDA algorithm explained in Section 3.5.5 is applied on this decision table with the two decision attributes *Fontcode* and *Charcode* for computing reducts.

For *Font code*, the algorithm produced 15 features as predominant attributes for classifying all the fonts in the training set. They are: HCV3, HCV7, HCV18, HCV27,

HCV40, HCV45, HCV49, VCV11, VCV12, VCV17, VCV28, VCV33, VCV37, VCV43, and VCV50.

Similarly, the algorithm produced 14 features as predominant attributes for *Charcode*. These 14 features that are adequate for classifying all the characters in the training set are: HCV3, HCV7, HCV19, HCV22, HCV35, HCV40, HCV44, HCV49, VCV12, VCV23, VCV28, VCV29, VCV43, and VCV50.

Of these, 8 features are common for both font and character classifications. So, eliminating these repeated features, 21 combined features are resulted totally. This implies a total reduction of 79 out of 100 features.

6.4 Feature Selection

Given a set of attributes (by the *Feature Reduction* module) and a dataset, this algorithm outputs the 21 predominant features from the dataset. For the first stage of the classifier, it selects and sends the 21 crossing count features only. For the second stage, it sends all the 100 crossing counts + the first 3 Hu moments.

6.5 Training Database Creation

As discussed in Section 4.4.1.1, two features on each side of 21 predominant features are considered, their average is computed, and rounded to the nearest lower integer as per Equations 4.1, and 4.2 (FLOOR5). For the 260 training samples, a new database is now created. This new database with 21-dimensional (21-D) feature vector is shown in Table 6.4 for the same five sample characters from Arial font.

6.6 Test Database Creation

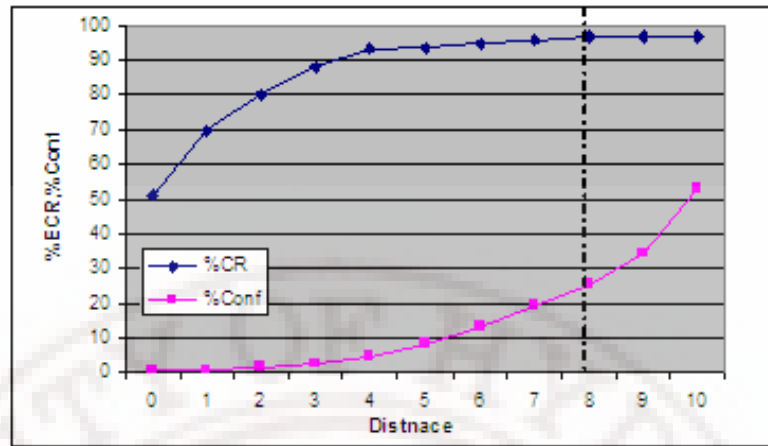
All the 100 Crossing-count features, and the 3 Hu moments are computed for each of the test characters in Dataset #2. A test database with 1560 X 103 dimensions is then created.

Table 6.4 Sample database with 21-D feature vectors

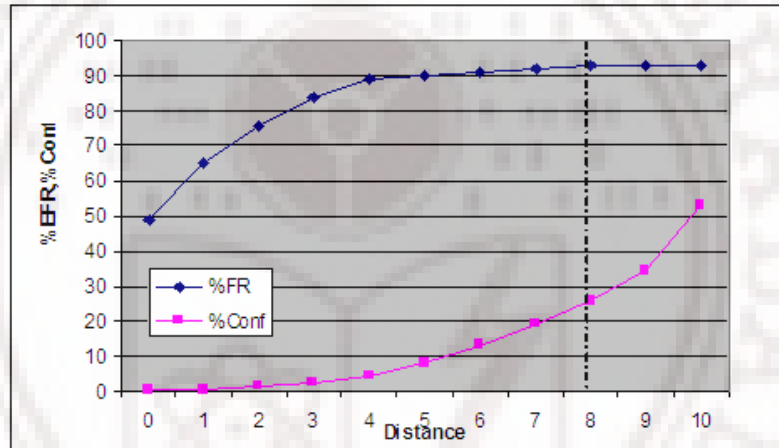
S.No.	Features	A	B	C	D	E
1	HCV3	1	1	1	1	1
2	HCV7	1	1	1	1	1
3	HCV18	2	2	1	1	1
4	HCV19	2	1	1	1	1
5	HCV22	2	1	1	1	1
6	HCV27	2	1	1	1	1
7	HCV35	1	1	1	1	1
8	HCV40	2	1	2	2	1
9	HCV44	2	1	1	1	0
10	HCV45	2	1	1	1	0
11	HCV49	1	0	1	0	0
12	VCV11	1	1	2	1	1
13	VCV12	1	1	2	1	1
14	VCV17	1	2	1	1	2
15	VCV23	2	2	1	1	2
16	VCV28	2	2	1	1	2
17	VCV29	2	2	1	1	2
18	VCV33	1	2	1	1	2
19	VCV37	1	2	1	2	2
20	VCV43	0	2	2	1	2
21	VCV50	0	1	1	1	0

6.7 Designing the Classifier

To design the first stage of the classifier, a suitable threshold distance similar to Telugu, has to be found. Using the Dataset #3, effect of the threshold distance on the classification is measured by varying the distance from 0 to 10. Increasing the threshold distance also increases the confusion set. Results are plotted in Figures 6.1 (a) and (b). Figure 6.1 (a) is for character recognition, and Figure 6.1 (b) is for font recognition.



(a)



(b)

Figure 6.1 Effect of Threshold distance on (a) character and (b) font recognitions (English)

Upper curves in Figure 6.1 show the relation between threshold distance and the percentage of possibility of the test font/character “include” in the output (P_I). Lower curves show the effect of varying the error threshold on the confusion set. It appears that a threshold distance near 8.0 is a compromise value for obtaining high accuracy, and at the same time not having more number of confused characters in the output. Our aim is to remove this confusion set at the next stage by including additional features. Any threshold value below 8.0 at the first stage seems to reduce recognition rates still further after the second stage.

However, to fix the optimum threshold value, Cost- curves are plotted for both font and character recognitions, as in section 4.4.2.1. Assuming $C_m=100$, and $C_c=5$, graphs are plotted between Cost(C) associated with missing/ resolving a confused character and the Distance Threshold. These are shown in Figure 6.2 (a) for character recognition, and Figure 6.2 (b) for font recognition. Minimum cost is obtained at Distance=8 and afterwards, the cost is increasing in both the curves. So, distance threshold value of 8 turns out to be the ideal choice for the threshold.

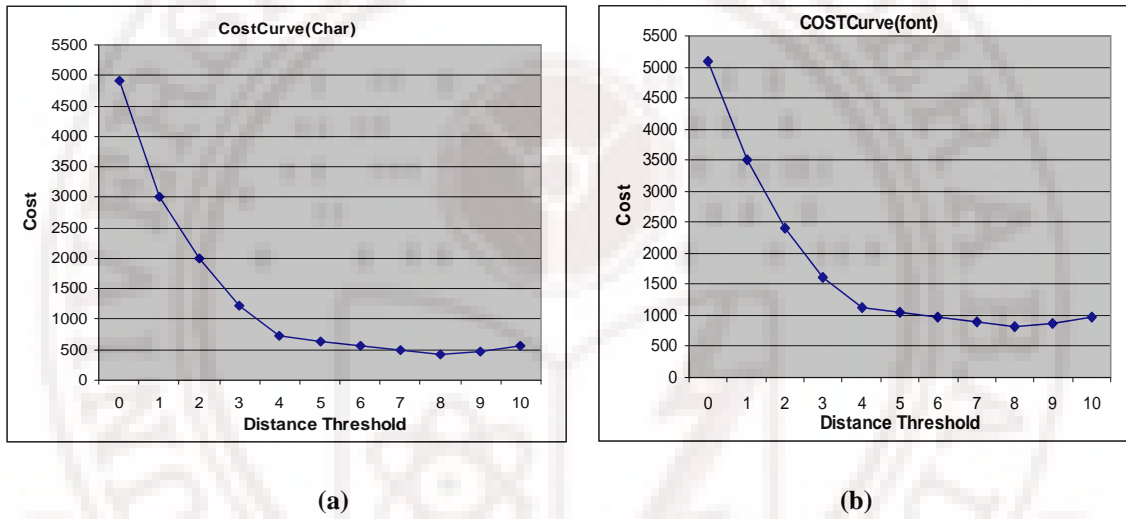


Figure 6.2 Costs for (a) character and (b) font recognitions (English)

Accordingly, our first stage of the classifier is set at a threshold distance of 8 to maintain reasonably good values for the final recognition rates; and a two- stage classifier similar to that of Telugu is constructed. Second stage is implemented as a minimum distance classifier with a 103-dimension feature vector consisting of 100 crossing counts and 3 Hu moments.

6.8 Testing and Results

Proposed methodology is then tested for its performance evaluation. Results of these tests are presented in this section. Test data contains 1,560 characters. With the 2-stage

classifier designed as above, 96.54% character recognition and 92.24 % font recognition are achieved. These results are presented in Table 6.5 below.

Table 6.5 Performance results on English characters

Test	Total	Correctly recognized	Rejected	Mis-recognized	Recognition	Rejection	Substitution Error
Character Recognition	1,560	1506	00	54	96.54%	0%	3.46%
Font Recognition	1,560	1439	00	121	92.24 %	0%	7.76%

Character and font recognition values are plotted in Figures 6.3 for different fonts in the test data. Character recognition (%CR) among different fonts is varying from 94 to 99, whereas font recognition (%FR) among different fonts is varying from 85 to 99. In all the cases, character recognition is found to be better than font recognition. Low font recognition values are obtained for Arial and Verdana fonts (Font codes 100, and 500 respectively), mainly due to the similarity of characters in these two fonts. Section 6.9 analyzes these errors in detail.

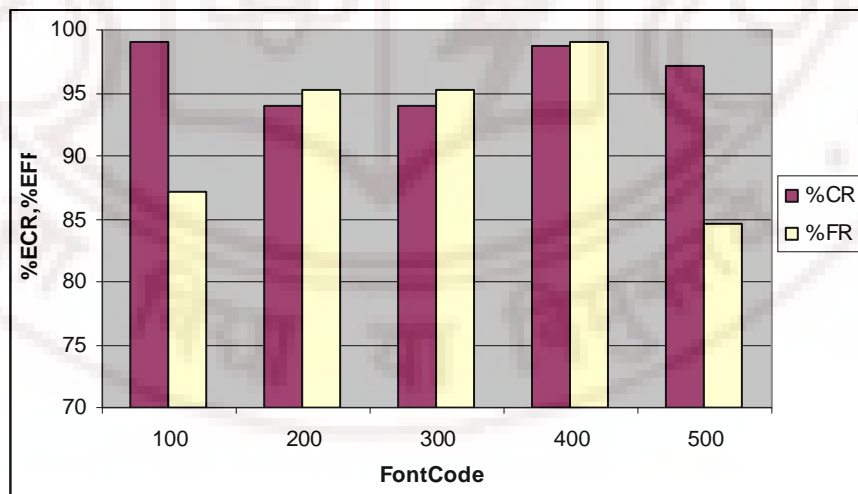


Figure 6.3 Character and Font Recognitions for different fonts (English)

Recognition values for test characters of different sizes are plotted in Figure 6.4. Character recognition (%CR) varies from 95 to 99, and font recognition (%FR) varies from 88 to 99 among different sizes. Peak recognitions are obtained at size 36, which is the size of the training set characters.

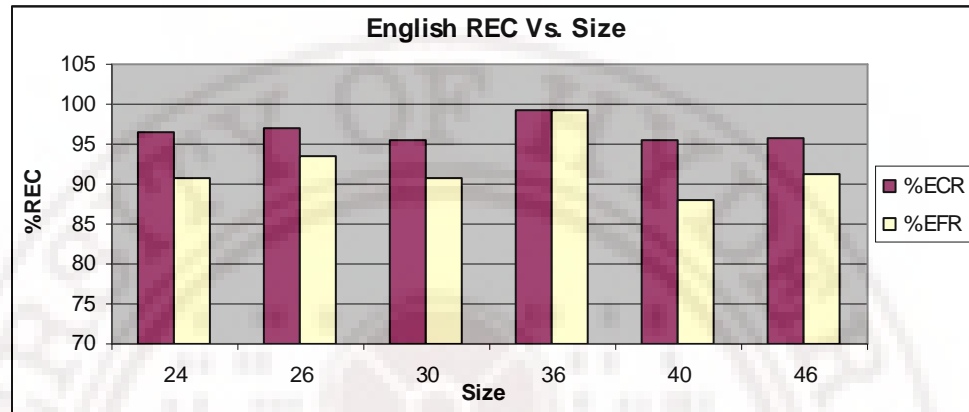


Figure 6.4 Character and Font Recognitions for different sizes (English)

6.9 Observations

Some interesting observations are made in English character and font recognitions.

- i. Cases of misrecognition are mainly because of very similar shapes. Sometimes, characters of one font are exactly looking similar to some other characters of another font. Ex: Uppercase letter I in Arial with lowercase L in Arial and ComicSans. This affects both character and font recognitions. Examples of some confused characters/fonts are shown in Table 6.6.

Table 6.6 Similar-looking characters

Test character	Recognized character
I (Uppercase letter I in Arial)	I (lowercase L in Arial, Verdana, ComicSans)
o (Letter O)	0 Digit 0

- ii. Several characters appear very similar in two or more fonts. Table 6.7 shows some of these characters. Each row shows the characters that appear the same in different fonts. These characters appear more or less the same in the fonts shown. From these cases, it is difficult to identify the font codes, even visually. This explains the higher substitution error in font recognition compared to that of character recognition. Font-variant features may enhance the results.

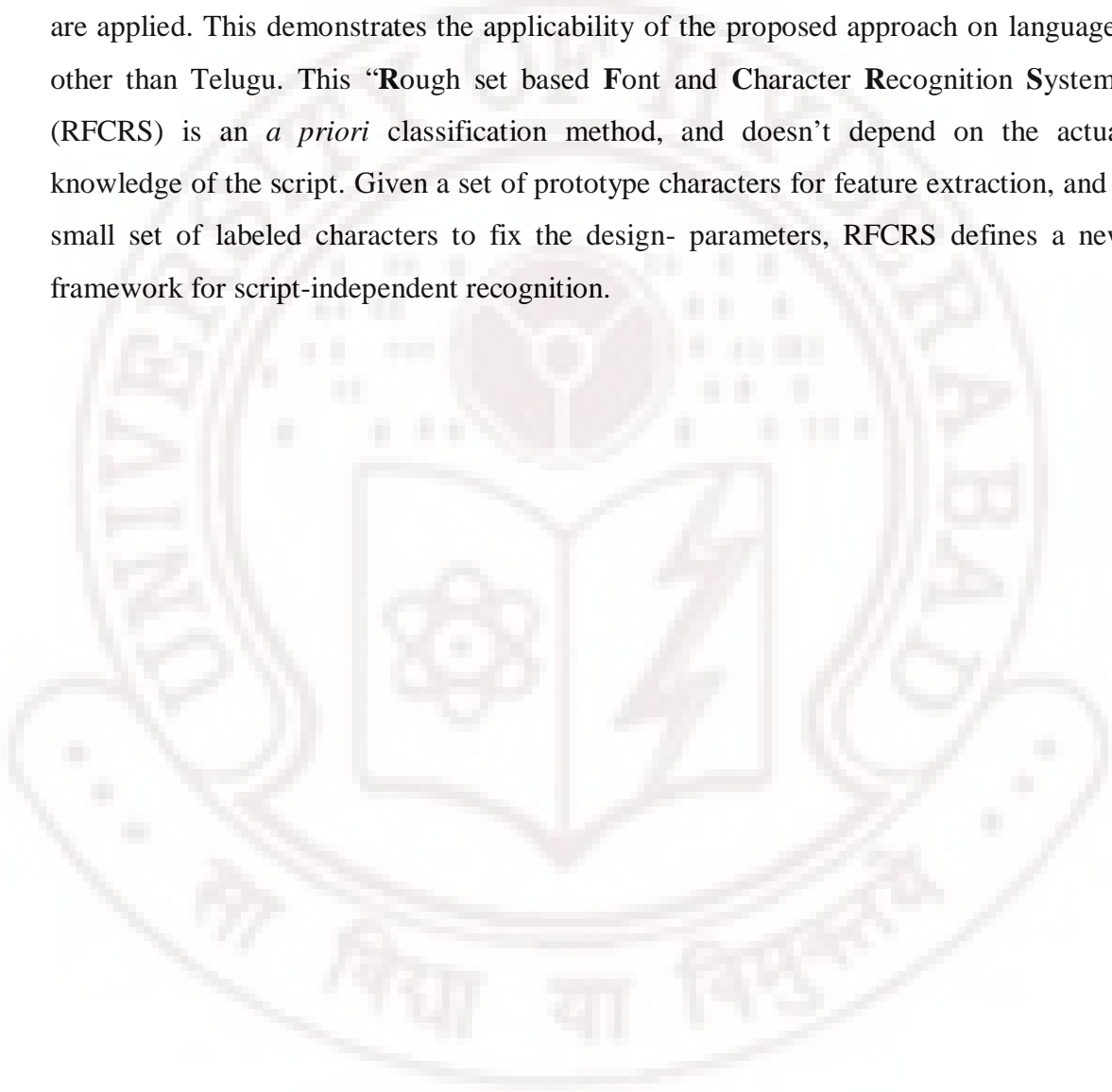
Table 6.7 Some characters that appear similar in different fonts

Arial	Verdana	ComicSans
	b	b
e	e	
i, o	i, o	i,o
v, x	v, x	v, x
p,q,s	p,q,s	
A,B,F,L,Z	A,B,F,L,Z	

- The moderate recognition accuracies reported in our work in Table 6.5 are only raw accuracies, i.e., without using any post processing like dictionary spell-checking. Many commercial OCRs employ morphological (n-gram) and lexical techniques to correct recognition errors. Further improvement in the recognition accuracy may be possible by adopting more robust preprocessing techniques, and some additional post-processing operations. The above results prove the applicability of the proposed approach to the OCR for English, or any other language with a scope for further improvement in accuracies.

6.10 Summary

The content-independent property of the developed methodology is tested in this chapter, for its generality on a popular language like English. Results of font and character recognitions are quite encouraging, even though no post processing techniques are applied. This demonstrates the applicability of the proposed approach on languages other than Telugu. This “**Rough set based Font and Character Recognition System**” (RFCRS) is an *a priori* classification method, and doesn’t depend on the actual knowledge of the script. Given a set of prototype characters for feature extraction, and a small set of labeled characters to fix the design- parameters, RFCRS defines a new framework for script-independent recognition.



Chapter 7

Conclusions and Future Scope

7.1 Conclusions

Development of a system for recognizing Telugu fonts and characters based on Rough sets theory is presented in this thesis. A survey of general OCR methodology in recent research was also summarized at the beginning of this report.

We conclude that this thesis has contributed to the field of Telugu character recognition in the following major areas:

1. A new approach for simultaneously recognizing Telugu characters and font faces is developed. Identification of fonts is generally a neglected task in the existing OCRs. Recognition of fonts and characters simultaneously is, hence, a major contribution of this thesis.
2. Explored the usefulness of rough sets in feature dimensionality reduction in the context of Telugu OCR. A great reduction that is also *semantics-preserving* is obtained in the feature set dimensions using RSAR.
3. Simple crossing count features are used in this work, which are size invariant, and also independent of the underlying script. Since no domain knowledge is used in the present work, we suggest a framework for script-independent character and font recognition. As a case study, the proposed methodology is applied on English, which is an entirely different script from Telugu. Encouraging recognition accuracies are achieved with a scope for further improvement.
4. Classifier design is based on extensive experimentation for fine-tuning several parameters that influence the performance.
5. A novel algorithm for size normalization of the images is developed. Though several normalization techniques were available in literature, they were not found suitable for

the selected feature set. Hence, a normalization algorithm that preserves the selected feature properties is developed.

6. Parts of the present work has evolved into the following international conference papers, and international journal papers:

International conference papers:

1. **“An efficient Binarization technique for old documents”**, proceedings of *International conference on Systemics, Cybernetics, and Informatics (ICSCI2006)*, Hyderabad, Jan 2006, pp.771-775.
2. **“Telugu character recognition”**, proceedings of *Int. conference on Systemics, Cybernetics, and Informatics*, Hyderabad, India, Jan2007, pp. 654-659.
3. **“A model for a multi font character recognition system”**, presented at *Fourteenth international conference of the Forum for interdisciplinary Mathematics. (CMASM2007- FIM XIV)* at Chennai, India. January 2007.
4. **“Telugu Font and Character Recognition using Rough sets”** proceedings of *“International conference on Cognition and Recognition”* at Mysore, Karnataka. April 2008, pp. 36-45.

International Journal papers:

1. **“Rough Sets based Feature Selection in Character Recognition”** *International Journal of Computational Intelligence Research & Applications*, Vol 2(1), Jan-June 2008, pp 73-81.
2. **“An Overview of OCR Research in Indian Scripts”** *International Journal of Computer science and Engineering systems* , Vol.2, No.2, April 2008, pp 141-153.
3. **“Telugu Font Recognition System”**, *IETECH Journal of Advanced Computations*, Vol: 2, No.4, 2008, pp.220-224.

7.2 Future Scope

In this research, it is proved that Rough sets theory can be effectively used in reducing feature dimensionality in Telugu OCR. The scope, however, is not only limited to

Telugu. The developed methodology can be made as a unified approach for OCR of any language, provided a training set is available. Thus, it can also be used in other language scripts, as well as in handwritten character recognition.

Developing commercial OCR systems which can maintain high recognition rates regardless of the irregularities such as the quality of the input documents, and varying font styles is a challenging task for Telugu and other Indian scripts. Compared to European languages, Indian languages have many additional challenges like larger character set due to modifiers, lack of standard test databases, and lack of support from browsers, operating system, and keyboard, etc. Telugu character segmentation poses additional problems due to touching, and overlapping characters. Factors listed by Bhagvati et al. [Bhagvati et al., 2003] such as identification of glyph position information, recognizing punctuation marks from the width and height information, handling of confusion pairs of glyphs and touching characters would help improving the performance of Telugu OCRs further. Also, hybridizing the proposed methodology with some of the popular methods such as N-grams may be thought over to enhance the performance to suit commercial OCRs.

~~~~~

# APPENDIX – A

## Prototype Character\* Images AMMA FONT

|             |             |             |             |              |              |             |              |              |              |             |
|-------------|-------------|-------------|-------------|--------------|--------------|-------------|--------------|--------------|--------------|-------------|
| 0           | 1           | 2           | 3           | 4            | 5            | 8           | 9            | క            | కా           | కి          |
| a172L14.jpg | a172L15.jpg | a172L16.jpg | a172L17.jpg | a172L18.jpg  | a172L19.jpg  | a172L31.jpg | a172L32.jpg  | a172L33.jpg  | a172L34.jpg  | a172L35.jpg |
| క్రీ        | ల           | లా          | కె          | కే           | కొ           | కో          | కౌ           | ఖ            | ఖా           | ఖి          |
| a172L36.jpg | a172L38.jpg | a172L42.jpg | a172L43.jpg | a172L44.jpg  | a172L45.jpg  | a172L46.jpg | a172L47.jpg  | a172L51.jpg  | a172L52.jpg  | a172L53.jpg |
| వీ          | ము          | మూ          | వౌ          | వో           | వొ           | వో          | వౌ           | గ            | .            | గా          |
| a172L54.jpg | a172L55.jpg | a172L56.jpg | a172L71.jpg | a172L72.jpg  | a172L73.jpg  | a172L74.jpg | a172L75.jpg  | a172L76.jpg  | a172L84.jpg  | a172L91.jpg |
| గి          | గీ          | గె          | గే          | 6            | 7            | ల           | గొ           | గో           | గా           | చ           |
| a172L92.jpg | a172L93.jpg | a172L98.jpg | a172L99.jpg | a172L110.jpg | a172L111.jpg | a238.jpg    | a272L11.jpg  | a272L12.jpg  | a272L13.jpg  | a272L14.jpg |
| చా          | చి          | చీ          | చె          | చే           | చొ           | చో          | చౌ           | చు           | జ            | జా          |
| a272L15.jpg | a272L16.jpg | a272L17.jpg | a272L21.jpg | a272L22.jpg  | a272L23.jpg  | a272L24.jpg | a272L25.jpg  | a272L31.jpg  | a272L34.jpg  | a272L35.jpg |
| జి          | జీ          | జె          | జొ          | జో           | లా           | యి          | యా           | యె           | యా           | ర           |
| a272L36.jpg | a272L45.jpg | a272L46.jpg | a272L47.jpg | a272L51.jpg  | a272L55.jpg  | a272L56.jpg | a272L62.jpg  | a272L63.jpg  | a272L111.jpg | a272L16.jpg |
| రా          | రి          | రీ          | రె          | రో           | రో           | రో          | రో           | రా           | వ            | వా          |
| a272L17.jpg | a272L21.jpg | a272L22.jpg | a272L27.jpg | a272L28.jpg  | a272L29.jpg  | a272L31.jpg | a272L32.jpg  | a272L33.jpg  | a272L34.jpg  | a272L35.jpg |
| వీ          | వె          | వో          | వొ          | వో           | వౌ           | శ           | శా           | శి           | శీ           | శె          |
| a272L36.jpg | a272L54.jpg | a272L55.jpg | a272L56.jpg | a272L57.jpg  | a272L58.jpg  | a272L71.jpg | a272L72.jpg  | a272L73.jpg  | a272L74.jpg  | a272L79.jpg |
| శొ          | శో          | శౌ          | ష           | ఁ            | షొ           | ష           | ఁ            | శే           | ఁ            | షు          |
| a272L81.jpg | a272L82.jpg | a272L83.jpg | a272L84.jpg | a272L85.jpg  | a272L86.jpg  | a272L87.jpg | a272L88.jpg  | a272L710.jpg | a272L810.jpg | a472L11.jpg |
| షూ          | =           | =           | షొ          | షొ           | స            | సా          | సా           | సో           | సొ           | వా          |
| a472L13.jpg | a472L15.jpg | a472L17.jpg | a472L19.jpg | a472L21.jpg  | a472L22.jpg  | a472L24.jpg | a472L31.jpg  | a472L32.jpg  | a472L33.jpg  | a472L34.jpg |
| ఁ           | హో          | హల          | హలా         | హో           | హో           | హౌ          | హో           | ళు           | ళూ           | తె          |
| a472L35.jpg | a472L36.jpg | a472L43.jpg | a472L45.jpg | a472L51.jpg  | a472L53.jpg  | a472L55.jpg | a472L110.jpg | a572L11.jpg  | a572L12.jpg  | a572L13.jpg |
| తే          | తొ          | తో          | తౌ          | ణ            | ణా           | ణో          | ణో           | ణె           | ణే           | ణొ          |
| a572L14.jpg | a572L15.jpg | a572L16.jpg | a572L21.jpg | a572L22.jpg  | a572L23.jpg  | a572L24.jpg | a572L25.jpg  | a572L28.jpg  | a572L31.jpg  | a572L32.jpg |
| ణో          | ణొ          | ట           | టా          | టీ           | టీ           | టు          | టూ           | టె           | టే           | టొ          |
| a572L33.jpg | a572L34.jpg | a572L37.jpg | a572L41.jpg | a572L42.jpg  | a572L43.jpg  | a572L44.jpg | a572L45.jpg  | a572L46.jpg  | a572L51.jpg  | a572L52.jpg |
| టో          | టొ          | ఎ           | పా          | ఎ            | ఁ            | ఁ           | గా           | ఁ            | ఎ            | సొ          |
| a572L53.jpg | a572L54.jpg | a572L55.jpg | a572L57.jpg | a572L58.jpg  | a572L59.jpg  | a572L64.jpg | a572L65.jpg  | a572L66.jpg  | a572L67.jpg  | a572L68.jpg |

\* Connected components

## Prototype Character Images AMMA FONT (Contd.)

|             |             |             |              |             |             |             |              |              |              |              |
|-------------|-------------|-------------|--------------|-------------|-------------|-------------|--------------|--------------|--------------|--------------|
| పో          | పౌ          | బ           | బా           | బి          | బీ          | .           | బు           | బూ           | బె           | బే           |
| a672L11.jpg | a672L12.jpg | a672L13.jpg | a672L15.jpg  | a672L17.jpg | a672L19.jpg | a672L21.jpg | a672L31.jpg  | a672L33.jpg  | a672L35.jpg  | a672L37.jpg  |
| ను          | బ           | బా          | బు           | బూ          | బె          | బే          | బొ           | బో           | బౌ           | లి           |
| a672L39.jpg | a672L51.jpg | a672L52.jpg | a672L55.jpg  | a672L56.jpg | a672L61.jpg | a672L62.jpg | a672L63.jpg  | a672L64.jpg  | a672L65.jpg  | a672L73.jpg  |
| లీ          | లు          | లో          | లో           | లౌ          | త           | తా          | నూ           | తి           | తీ           | తో           |
| a672L74.jpg | a672L75.jpg | a672L83.jpg | a672L84.jpg  | a672L85.jpg | a672L86.jpg | a672L88.jpg | a672L311.jpg | a772L11.jpg  | a772L13.jpg  | a772L21.jpg  |
| తో          | తౌ          | ద           | దా           | ది          | దు          | రా          | దె           | దే           | దొ           | దో           |
| a772L23.jpg | a772L25.jpg | a772L27.jpg | a772L28.jpg  | a772L29.jpg | a772L31.jpg | a772L33.jpg | a772L34.jpg  | a772L35.jpg  | a772L36.jpg  | a772L37.jpg  |
| దౌ          | న           | నా          | ని           | నీ          | ను          | నూ          | నె           | నే           | నొ           | నో           |
| a772L41.jpg | a772L42.jpg | a772L43.jpg | a772L44.jpg  | a772L45.jpg | a772L46.jpg | a772L51.jpg | a772L52.jpg  | a772L53.jpg  | a772L54.jpg  | a772L55.jpg  |
| నౌ          | ం           | ఁ           | ం            | ఁ           | ి           | ి           | తె           | తే           | దీ           | వీ           |
| a772L56.jpg | a772L61.jpg | a772L62.jpg | a772L63.jpg  | a772L67.jpg | a772L68.jpg | a772L69.jpg | a772L111.jpg | a772L113.jpg | a772L210.jpg | a772L510.jpg |
| ై           | ఎ           | ఎ           | ఎ            | ని          | వీ          | ి           | క            | గ            | క            | క            |
| a872L11.jpg | a872L12.jpg | a872L13.jpg | a872L14.jpg  | a872L15.jpg | a872L16.jpg | a872L17.jpg | a872L21.jpg  | a872L22.jpg  | a872L23.jpg  | a872L24.jpg  |
| ద్          | డ్          | చ్          | వ్           | న్          | ఖ్          | య్          | ర్           | మై           | బ్           | ణ్           |
| a872L25.jpg | a872L26.jpg | a872L27.jpg | a872L28.jpg  | a872L31.jpg | a872L32.jpg | a872L34.jpg | a872L35.jpg  | a872L36.jpg  | a872L41.jpg  | a872L42.jpg  |
| బ్          | బ్          | త్          | ల్           | జ్          | వ్          | వ           | రా           | వి           | మె           | మే           |
| a872L43.jpg | a872L44.jpg | a872L45.jpg | a872L46.jpg  | a872L51.jpg | a872L52.jpg | a872L54.jpg | a872L57.jpg  | a872L58.jpg  | a872L61.jpg  | a872L62.jpg  |
| మొ          | మొ          | మా          | వీ           | ా           | ా           | ి           | రా           | ఱ            | ఎ            | ె            |
| a872L63.jpg | a872L64.jpg | a872L65.jpg | a872L510.jpg | a872L43.jpg | a872L47.jpg | a872L56.jpg | a872L57.jpg  | a872L58.jpg  | a872L59.jpg  | a872L64.jpg  |
| డ           | దా          | డే          | డీ           | దు          | డూ          | డె          | డే           | దొ           | దో           | డా           |
| a972L71.jpg | a972L72.jpg | a972L73.jpg | a972L74.jpg  | a972L75.jpg | a972L76.jpg | a972L81.jpg | a972L82.jpg  | a972L83.jpg  | a972L84.jpg  | a972L85.jpg  |
| అ           | ఆ           | ఇ           | ఈ            | ఉ           | ఊ           | ఎ           | ఐ            | ఒ            | ఓ            | శి           |
| a872L.jpg   | a872L.jpg   | a872L.jpg   | a872L.jpg    | a872L.jpg   | a872L.jpg   | a872L.jpg   | a872L.jpg    | a872L.jpg    | a872L.jpg    | a872L.jpg    |



# Prototype Character Images

## GOWTHAMI FONT

|             |             |             |             |             |             |             |             |             |             |             |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| అ           | ఆ           | ఇ           | ఈ           | ఉ           | ఊ           | ఋ           | ఎ           | ఐ           | ఒ           | ఓ           |
| g001.png    | g002.png    | g003.png    | g004.png    | g005.png    | g006.png    | g007.png    | g008.png    | g010.png    | g011.png    | g012.png    |
| ఔ           | ం           | క           | ఖ           | గ           | ఙ్          | చ           | ఝ           | ట           | ఠ           | ణ           |
| g013.png    | g014.png    | g015.png    | g016.png    | g017.png    | g018.png    | g019.png    | g020.png    | g021.png    | g022.png    | g023.png    |
| త్          | ద్          | వ్          | బ్          | మ్          | య్          | ర్          | ల్          | ళ్          | ఫ్          | శ్          |
| g024.png    | g025.png    | g026.png    | g027.png    | g028.png    | g029.png    | g030.png    | g031.png    | g032.png    | g033.png    | g034.png    |
| ఖ           | జ           | ఙ           | ఞ           | ట           | ణ           | బ           | ల           | ఱ           | ృ           | ౄ           |
| g212.png    | g215.png    | g218.png    | g220.png    | g221.png    | g225.png    | g233.png    | g238.png    | g244.png    | g211.png    | g315.png    |
| ౌ           | ౠ           | ౡ           | ౢ           | ౣ           | ౤           | ౥           | ౦           | ౧           | ౨           | ౩           |
| g316.png    | g317.png    | g318.png    | g320.png    | g321.png    | g322.png    | g323.png    | g324.png    | g325.png    | g326.png    | g327.png    |
| ౪           | ౫           | ౬           | ౭           | ౮           | ౹           | ౺           | ౻           | ౼           | ౽           | ౾           |
| g329.png    | g330.png    | g331.png    | g332.png    | g333.png    | g334.png    | g335.png    | g336.png    | g337.png    | g338.png    | g340.png    |
| క           | కా          | కొ          | కో          | కె          | కే          | కి          | క్రి        | కా          | ఖా          | భా          |
| g21311.png  | g21312.png  | g21313.png  | g21314.png  | g21317.png  | g21318.png  | g21320.png  | g21321.png  | g21322.png  | g21322.png  | g21323.png  |
| భో          | ఖు          | ఖూ          | భె          | భే          | భి          | భీ          | భౌ          | గ           | గా          | గౌ          |
| g21324.png  | g21325.png  | g21326.png  | g21327.png  | g21328.png  | g21330.png  | g21331.png  | g21332.png  | g21331.png  | g21332.png  | g21333.png  |
| గో          | గు          | గూ          | గె          | గే          | గి          | గీ          | గౌ          | ఘ           | ఘా          | ఘౌ          |
| g21334.png  | g21335.png  | g21336.png  | g21337.png  | g21338.png  | g21339.png  | g21340.png  | g21341.png  | g21342.png  | g21343.png  | g21344.png  |
| ఘా          | ఘౌ          | జా          | జొ          | జో          | జు          | జూ          | జె          | జే          | జి          | జీ          |
| g214316.png | g214322.png | g215312.png | g215313.png | g215314.png | g215315.png | g215316.png | g215317.png | g215318.png | g215320.png | g215321.png |
| జౌ          | చ           | చా          | చొ          | చో          | చె          | చే          | చి          | చీ          | చౌ          | జా          |
| g215322.png | g216311.png | g216312.png | g216313.png | g216314.png | g216317.png | g216318.png | g216320.png | g216321.png | g216322.png | g216323.png |
| జొ          | జో          | జు          | జూ          | జె          | జే          | జి          | జీ          | జౌ          | రు          | రౌ          |
| g218313.png | g218314.png | g218315.png | g218316.png | g218317.png | g218318.png | g218320.png | g218321.png | g218322.png | g218323.png | g218324.png |
| రౌ          | రౌ          | రౌ          | రి          | రీ          | రౌ          | శా          | శొ          | శో          | శౌ          | శౌ          |
| g219314.png | g219317.png | g219318.png | g219320.png | g219321.png | g219322.png | g220312.png | g220313.png | g220314.png | g220317.png | g220318.png |
| శో          | శో          | శౌ          | టా          | టౌ          | టో          | టు          | టూ          | టె          | టే          | టి          |
| g220320.png | g220321.png | g220322.png | g221312.png | g221313.png | g221314.png | g221315.png | g221316.png | g221317.png | g221318.png | g221320.png |
| టీ          | టా          | డ           | డా          | డొ          | డో          | డె          | డే          | డి          | డీ          | డౌ          |
| g221321.png | g221322.png | g223311.png | g223312.png | g223313.png | g223314.png | g223317.png | g223318.png | g223320.png | g223321.png | g223322.png |
| ణా          | ణౌ          | ణో          | ణు          | ణూ          | ణె          | ణే          | ణి          | ణీ          | ణౌ          | త           |
| g225312.png | g225313.png | g225314.png | g225315.png | g225316.png | g225317.png | g225318.png | g225320.png | g225321.png | g225322.png | g226311.png |



## Prototype Character Images GOWTHAMI FONT (Contd.)

|    |    |    |    |    |    |    |    |    |    |    |
|----|----|----|----|----|----|----|----|----|----|----|
| తా | తో | తో | తె | తే | తి | తీ | తౌ | ద  | దా | దౌ |
| దో | దె | దే | ది | దీ | దౌ | న  | నా | నౌ | నో | నె |
| నే | ని | నీ | నౌ | పా | పొ | పో | పా | బా | బౌ | బో |
| బు | బూ | బె | బే | బి | బీ | బౌ | భ  | భా | భి | భీ |
| భౌ | మ  | మొ | మౌ | మె | మీ | మి | మీ | య  | యౌ | యె |
| యే | యి | యా | యౌ | ర  | రా | రౌ | రో | రె | రీ | రి |
| రీ | రా | లా | లౌ | లో | లు | లూ | లె | లే | లి | లీ |
| లౌ | వ  | వా | వౌ | వో | వె | వే | వి | వీ | వౌ | శ  |
| శౌ | శొ | శో | శె | శే | శి | శీ | శౌ | షా | షౌ | షో |
| షౌ | సా | సౌ | సో | సా | జా | జౌ | జో | జు | జూ | జె |
| జే | జి | జీ | జౌ | ళ  | ళా | ళౌ | ళో | ళె | ళే | ళి |
| ళి | ళౌ | కు | కూ | చు | చూ | రూ | రూ | రూ | డు | డూ |
| తు | తూ | దు | దూ | ను | నూ | భు | భూ | మా | ము | మూ |
| మౌ | యా | యు | యూ | రు | రూ | పు | పూ | శు | శూ | హౌ |
| ళు | ళూ | 0  | 1  | 2  | 3  | 4  | 5  | 6  | 7  | 8  |
| ఎ  | ఎ  | ని | ౧  | ౨  | ౩  | ౪  | ౫  | ౬  | ౭  | ౮  |

# Prototype Character Images

## HEMALATHA FONT

|             |             |             |             |             |             |             |             |             |             |             |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| అ           | ఆ           | ఇ           | ఈ           | ఉ           | ఊ           | ఋ           | ఎ           | ఐ           | ఒ           | ఓ           |
| h001.png    | h002.png    | h003.png    | h004.png    | h005.png    | h006.png    | h007.png    | h008.png    | h010.png    | h011.png    | h012.png    |
| ఔ           | ం           | క           | గ           | జ్          | చ్          | ట్          | డ           | ణ్          | త్          | ద్          |
| h013.png    | h014.png    | h015.png    | h017.png    | h018.png    | h019.png    | h021.png    | h022.png    | h023.png    | h024.png    | h025.png    |
| వ్          | బ్          | మ్          | య్          | ర్          | ల్          | ల్          | ళ్          | శ్          | జ           | జ           |
| h026.png    | h027.png    | h028.png    | h029.png    | h030.png    | h031.png    | h032.png    | h033.png    | h034.png    | h215.png    | h216.png    |
| ఇ           | ట           | ణ           | బ           | ల           | ఱ           | ✓           | ౞           | ౞           | ౞           | ౞           |
| h220.png    | h221.png    | h225.png    | h233.png    | h238.png    | h244.png    | h311.png    | h313.png    | h314.png    | h315.png    | h316.png    |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h317.png    | h318.png    | h320.png    | h321.png    | h322.png    | h323.png    | h324.png    | h325.png    | h326.png    | h327.png    | h328.png    |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h329.png    | h330.png    | h331.png    | h332.png    | h333.png    | h334.png    | h335.png    | h337.png    | h338.png    | h340.png    | h360.png    |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h363.png    | h398.png    | h408.png    | h211311.png | h211312.png | h211313.png | h211314.png | h211317.png | h211318.png | h211320.png | h211321.png |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h211322.png | h213311.png | h213312.png | h213313.png | h213314.png | h213315.png | h213316.png | h213317.png | h213318.png | h213320.png | h213321.png |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h213322.png | h215312.png | h215313.png | h215314.png | h215315.png | h215316.png | h215317.png | h215318.png | h215320.png | h215321.png | h215322.png |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h216311.png | h216312.png | h216313.png | h216314.png | h216317.png | h216318.png | h216320.png | h216321.png | h216322.png | h216323.png | h216324.png |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h218314.png | h218315.png | h218316.png | h218317.png | h218318.png | h218320.png | h218321.png | h218322.png | h220312.png | h220313.png | h220314.png |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h220317.png | h220318.png | h220320.png | h220321.png | h220322.png | h221312.png | h221313.png | h221314.png | h221315.png | h221316.png | h221317.png |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h221318.png | h221320.png | h221321.png | h221322.png | h223311.png | h223312.png | h223313.png | h223314.png | h223317.png | h223318.png | h223320.png |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h223321.png | h223322.png | h225313.png | h225314.png | h225315.png | h225316.png | h225317.png | h225318.png | h225320.png | h225321.png | h225322.png |
| ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           | ౞           |
| h225322.png | h226311.png | h226312.png | h226313.png | h226314.png | h226317.png | h226318.png | h226320.png | h226321.png | h226322.png | h226323.png |

## Prototype Character Images HEMALATHA FONT (Contd.)

|                |                |                |                |                |                |                |                |                |                |                |
|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|----------------|
| దా             | దొ             | దో             | దె             | దే             | ది             | దీ             | దౌ             | న              | నా             | నొ             |
| h228312.jpg    | h228313.jpg    | h228314.jpg    | h228317.jpg    | h228318.jpg    | h228320.jpg    | h228321.jpg    | h228322.jpg    | h230311.jpg    | h230312.jpg    | h230313.jpg    |
| నో             | నె             | నే             | ని             | నీ             | నౌ             | పా             | పొ             | పో             | పౌ             | బా             |
| h230314.jpg    | h230317.jpg    | h230318.jpg    | h230320.jpg    | h230321.jpg    | h230322.jpg    | h231312.jpg    | h231313.jpg    | h231314.jpg    | h231322.jpg    | h233312.jpg    |
| బొ             | బో             | బు             | బూ             | బె             | బే             | బి             | బీ             | బౌ             | భ              | భా             |
| h233313.jpg    | h233314.jpg    | h233315.jpg    | h233316.jpg    | h233317.jpg    | h233318.jpg    | h233320.jpg    | h233321.jpg    | h233322.jpg    | h234311.jpg    | h234312.jpg    |
| భి             | భీ             | భౌ             | మ              | మొ             | మో             | మె             | మే             | మి             | మీ             | య              |
| h234320.jpg    | h234321.jpg    | h234322.jpg    | h235311.jpg    | h235313.jpg    | h235314.jpg    | h235317.jpg    | h235318.jpg    | h235320.jpg    | h235321.jpg    | h236311.jpg    |
| యో             | యె             | యే             | యి             | యీ             | యౌ             | ర              | రా             | రొ             | రో             | రె             |
| h236314.jpg    | h236317.jpg    | h236318.jpg    | h236320.jpg    | h236321.jpg    | h236322.jpg    | h237311.jpg    | h237312.jpg    | h237313.jpg    | h237314.jpg    | h237317.jpg    |
| రే             | రి             | రీ             | రౌ             | లా             | లొ             | లో             | లు             | లూ             | లే             | లే             |
| h237318.jpg    | h237320.jpg    | h237321.jpg    | h237322.jpg    | h238312.jpg    | h238313.jpg    | h238314.jpg    | h238315.jpg    | h238316.jpg    | h238317.jpg    | h238318.jpg    |
| లి             | లీ             | లౌ             | వ              | వా             | వొ             | వో             | వె             | వే             | వి             | వీ             |
| h238320.jpg    | h238321.jpg    | h238322.jpg    | h239311.jpg    | h239312.jpg    | h239313.jpg    | h239314.jpg    | h239317.jpg    | h239318.jpg    | h239320.jpg    | h239321.jpg    |
| వౌ             | శ              | శా             | శొ             | శో             | శె             | శే             | ని             | నీ             | శౌ             | షా             |
| h239322.jpg    | h240311.jpg    | h240312.jpg    | h240313.jpg    | h240314.jpg    | h240317.jpg    | h240318.jpg    | h240320.jpg    | h240321.jpg    | h240322.jpg    | h241312.jpg    |
| షొ             | షో             | షౌ             | సా             | సొ             | సో             | సౌ             | టా             | టొ             | టో             | ఱు             |
| h241313.jpg    | h241314.jpg    | h241322.jpg    | h242312.jpg    | h242313.jpg    | h242314.jpg    | h242322.jpg    | h244312.jpg    | h244313.jpg    | h244314.jpg    | h244315.jpg    |
| ఱూ             | ఱె             | ఱే             | ఱి             | ఱీ             | ఱౌ             | ళ              | ళా             | ళొ             | ళో             | ళె             |
| h244316.jpg    | h244317.jpg    | h244318.jpg    | h244320.jpg    | h244321.jpg    | h244322.jpg    | h245311.jpg    | h245312.jpg    | h245313.jpg    | h245314.jpg    | h245317.jpg    |
| ళే             | ళి             | ళీ             | ళౌ             | కు             | కూ             | చు             | చూ             | డు             | డూ             | తు             |
| h245318.jpg    | h245320.jpg    | h245321.jpg    | h245322.jpg    | h211311315.jpg | h211311316.jpg | h216311315.jpg | h216311316.jpg | h223311315.jpg | h223311316.jpg | h226311315.jpg |
| తూ             | దు             | దూ             | ను             | నూ             | భు             | భూ             | మా             | ము             | మూ             | మౌ             |
| h226311316.jpg | h228311315.jpg | h228311316.jpg | h230311315.jpg | h230311316.jpg | h234311315.jpg | h234311316.jpg | h235311312.jpg | h235311315.jpg | h235311316.jpg | h235311322.jpg |
| యా             | యు             | యూ             | రు             | రూ             | వు             | వూ             | శు             | శూ             | ళు             | ళూ             |
| h236311312.jpg | h236311315.jpg | h236311316.jpg | h237311315.jpg | h237311316.jpg | h239311315.jpg | h239311316.jpg | h240311315.jpg | h240311316.jpg | h245311315.jpg | h245311316.jpg |
| 0              | 1              | 2              | 3              | 4              | 5              | 6              | 7              | 8              | ఎ              | 9              |
| HL11.jpg       | HL12.jpg       | HL13.jpg       | HL14.jpg       | HL15.jpg       | HL16.jpg       | HL17.jpg       | HL18.jpg       | HL19.jpg       | HL25.jpg       | HL110.jpg      |

# Prototype Character Images

## SRILIPi (G-Series) FONT

|              |              |              |              |              |              |              |              |              |              |              |
|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| 0            | 1            | 2            | 3            | 4            | 5            | 6            | 7            | 8            | 9            | అ            |
| s4860.jpg    | s4960.jpg    | s5060.jpg    | s5160.jpg    | s5260.jpg    | s5360.jpg    | s5460.jpg    | s5560.jpg    | s5660.jpg    | s5760.jpg    | an001.jpg    |
| ఆ            | ఇ            | ఈ            | ఉ            | ఊ            | ఋ            | ఎ            | ఐ            | ఒ            | ఓ            | ఔ            |
| an002.jpg    | an003.jpg    | an004.jpg    | an005.jpg    | an006.jpg    | an007.jpg    | an008.jpg    | an010.jpg    | an011.jpg    | an012.jpg    | an013.jpg    |
| ం            | క            | ఖ            | గ            | ఙ            | చ            | ఝ            | ట            | డ            | ణ            | త            |
| an014.jpg    | an015.jpg    | an016.jpg    | an017.jpg    | an018.jpg    | an019.jpg    | an020.jpg    | an021.jpg    | an022.jpg    | an023.jpg    | an024.jpg    |
| ద్           | వ్           | బ్           | మ్           | య్           | ర్           | ళ్           | ల్           | ళ్           | ళ్           | ఖ            |
| an025.jpg    | an026.jpg    | an027.jpg    | an028.jpg    | an029.jpg    | an030.jpg    | an031.jpg    | an032.jpg    | an033.jpg    | an034.jpg    | an012.jpg    |
| జ            | ఝ            | ఇ            | ట            | ణ            | బ            | ల            | ఱ            | ఱ            | ఱ            | ఱ            |
| an215.jpg    | an218.jpg    | an220.jpg    | an221.jpg    | an225.jpg    | an233.jpg    | an238.jpg    | an244.jpg    | an311.jpg    | an313.jpg    | an314.jpg    |
| ృ            | ౌ            | ృ            | ౄ            | ౅            | ె            | ే            | ై            | ౉            | ొ            | ో            |
| an315.jpg    | an316.jpg    | an317.jpg    | an318.jpg    | an320.jpg    | an321.jpg    | an322.jpg    | an323.jpg    | an325.jpg    | an326.jpg    | an327.jpg    |
| చ            | ఛ            | ఞ            | ణ            | ట            | ఠ            | ద            | ధ            | న            | న            | న            |
| an328.jpg    | an329.jpg    | an330.jpg    | an331.jpg    | an332.jpg    | an333.jpg    | an334.jpg    | an335.jpg    | an337.jpg    | an338.jpg    | an339.jpg    |
| ఝ            | క            | కా           | కొ           | కో           | కె           | కి           | క్రి         | క్రి         | క్రి         | ఖా           |
| an340.jpg    | an21311.jpg  | an21312.jpg  | an21313.jpg  | an21314.jpg  | an21317.jpg  | an21318.jpg  | an21320.jpg  | an21321.jpg  | an21322.jpg  | an21322.jpg  |
| ఖా           | ఖో           | ఖు           | ఖూ           | ఖౌ           | ఖో           | ఖి           | ఖీ           | ఖౌ           | గ            | గా           |
| an212313.jpg | an212314.jpg | an212315.jpg | an212316.jpg | an212317.jpg | an212318.jpg | an212320.jpg | an212321.jpg | an212322.jpg | an21311.jpg  | an21312.jpg  |
| గా           | గో           | గె           | గి           | గి           | గి           | గౌ           | జా           | జా           | జో           | జె           |
| an213313.jpg | an213314.jpg | an213317.jpg | an213318.jpg | an213320.jpg | an213321.jpg | an213322.jpg | an215312.jpg | an215313.jpg | an215314.jpg | an215317.jpg |
| జే           | జి           | జి           | జౌ           | చ            | చా           | చా           | చో           | చె           | చె           | చి           |
| an215318.jpg | an215320.jpg | an215321.jpg | an215322.jpg | an216311.jpg | an216312.jpg | an216313.jpg | an216314.jpg | an216317.jpg | an216318.jpg | an216320.jpg |
| చీ           | చా           | జా           | జా           | జో           | జా           | జా           | జె           | జే           | జి           | జి           |
| an216321.jpg | an216322.jpg | an218312.jpg | an218313.jpg | an218314.jpg | an218315.jpg | an218316.jpg | an218317.jpg | an218318.jpg | an218320.jpg | an218321.jpg |
| జౌ           | ఝ            | ఝ            | ఝౌ           | ఝ            | ఝ            | ఝ            | ఝ            | ఝ            | ఝ            | ఝ            |
| an218322.jpg | an219311.jpg | an219313.jpg | an219314.jpg | an219317.jpg | an219318.jpg | an219320.jpg | an219321.jpg | an219322.jpg | an220312.jpg | an220313.jpg |
| ఝో           | ఝ            | ఝ            | ఝ            | ఝ            | ఝ            | ఝ            | ఝ            | ఝ            | ఝ            | ఝ            |
| an220314.jpg | an220317.jpg | an220321.jpg | an220322.jpg | an220321.jpg | an220322.jpg | an221312.jpg | an221313.jpg | an221314.jpg | an221315.jpg | an221316.jpg |
| టౌ           | టౌ           | టౌ           | టౌ           | టౌ           | టౌ           | టౌ           | టౌ           | టౌ           | టౌ           | టౌ           |
| an221317.jpg | an221318.jpg | an221320.jpg | an221321.jpg | an221322.jpg | an223311.jpg | an223312.jpg | an223313.jpg | an223314.jpg | an223317.jpg | an223318.jpg |



## Prototype Character Images SRILIPi (G-Series) FONT (Contd.)

|             |             |             |             |             |             |             |             |             |             |             |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| డి          | డీ          | డౌ          | ణా          | ణో          | ణో          | ణు          | ణూ          | ణె          | ణే          | ణి          |
| sn23320.jpg | sn23321.jpg | sn23322.jpg | sn25312.jpg | sn25313.jpg | sn25314.jpg | sn25315.jpg | sn25316.jpg | sn25317.jpg | sn25318.jpg | sn25320.jpg |
| ణీ          | ణౌ          | ళ           | ళా          | ళో          | ళో          | ళె          | ళే          | ళి          | ళీ          | ళౌ          |
| sn25321.jpg | sn25322.jpg | sn26311.jpg | sn26312.jpg | sn26313.jpg | sn26314.jpg | sn26317.jpg | sn26318.jpg | sn26320.jpg | sn26321.jpg | sn26322.jpg |
| ద           | దా          | దో          | దో          | దె          | దే          | ది          | దీ          | దౌ          | న           | నా          |
| sn26312.jpg | sn26312.jpg | sn26313.jpg | sn26312.jpg | sn26317.jpg | sn26318.jpg | sn26320.jpg | sn26321.jpg | sn26322.jpg | sn26311.jpg | sn26312.jpg |
| నొ          | నో          | నె          | నే          | ని          | నీ          | నొ          | ప           | పా          | పా          | పా          |
| sn26313.jpg | sn26314.jpg | sn26317.jpg | sn26318.jpg | sn26320.jpg | sn26321.jpg | sn26322.jpg | sn26311.jpg | sn26312.jpg | sn26313.jpg | sn26314.jpg |
| పా          | బా          | బొ          | బో          | బు          | బూ          | బొ          | బో          | బి          | బీ          | బొ          |
| sn26322.jpg | sn26322.jpg | sn26323.jpg | sn26324.jpg | sn26325.jpg | sn26326.jpg | sn26327.jpg | sn26328.jpg | sn26329.jpg | sn26330.jpg | sn26332.jpg |
| భ           | భా          | భొ          | భో          | భి          | భీ          | భొ          | మ           | మొ          | మొ          | మొ          |
| sn26331.jpg | sn26332.jpg | sn26333.jpg | sn26334.jpg | sn26335.jpg | sn26336.jpg | sn26337.jpg | sn26338.jpg | sn26339.jpg | sn26340.jpg | sn26341.jpg |
| మే          | మి          | మీ          | య           | యే          | యె          | యే          | యి          | యా          | యా          | ర           |
| sn26342.jpg | sn26343.jpg | sn26344.jpg | sn26345.jpg | sn26346.jpg | sn26347.jpg | sn26348.jpg | sn26349.jpg | sn26350.jpg | sn26351.jpg | sn26352.jpg |
| రా          | రా          | రా          | రె          | రే          | రి          | రీ          | రొ          | లా          | లా          | లా          |
| sn26353.jpg | sn26354.jpg | sn26355.jpg | sn26356.jpg | sn26357.jpg | sn26358.jpg | sn26359.jpg | sn26360.jpg | sn26361.jpg | sn26362.jpg | sn26363.jpg |
| లు          | లూ          | లొ          | లో          | లి          | లీ          | లొ          | వ           | వా          | వొ          | వో          |
| sn26364.jpg | sn26365.jpg | sn26366.jpg | sn26367.jpg | sn26368.jpg | sn26369.jpg | sn26370.jpg | sn26371.jpg | sn26372.jpg | sn26373.jpg | sn26374.jpg |
| వె          | వే          | వి          | వీ          | వొ          | శ           | శా          | శొ          | శో          | శె          | శే          |
| sn26375.jpg | sn26376.jpg | sn26377.jpg | sn26378.jpg | sn26379.jpg | sn26380.jpg | sn26381.jpg | sn26382.jpg | sn26383.jpg | sn26384.jpg | sn26385.jpg |
| శి          | శీ          | శొ          | షా          | షొ          | షో          | షొ          | సా          | సొ          | సో          | సొ          |
| sn26386.jpg | sn26387.jpg | sn26388.jpg | sn26389.jpg | sn26390.jpg | sn26391.jpg | sn26392.jpg | sn26393.jpg | sn26394.jpg | sn26395.jpg | sn26396.jpg |
| ఱా          | ఱొ          | ఱో          | ఱె          | ఱే          | ఱి          | ఱీ          | ఱొ          | ళ           | ళా          | ళొ          |
| sn26397.jpg | sn26398.jpg | sn26399.jpg | sn26400.jpg | sn26401.jpg | sn26402.jpg | sn26403.jpg | sn26404.jpg | sn26405.jpg | sn26406.jpg | sn26407.jpg |
| ళో          | ళె          | ళే          | ళి          | ళీ          | ళొ          | కు          | కూ          | చు          | చూ          | ఝా          |
| sn26408.jpg | sn26409.jpg | sn26410.jpg | sn26411.jpg | sn26412.jpg | sn26413.jpg | sn26414.jpg | sn26415.jpg | sn26416.jpg | sn26417.jpg | sn26418.jpg |
| ఝు          | ఝూ          | ఝొ          | ఝా          | దు          | దూ          | ను          | నూ          | భు          | భూ          | మా          |
| sn26419.jpg | sn26420.jpg | sn26421.jpg | sn26422.jpg | sn26423.jpg | sn26424.jpg | sn26425.jpg | sn26426.jpg | sn26427.jpg | sn26428.jpg | sn26429.jpg |
| ము          | మూ          | మొ          | యా          | యొ          | యొ          | రు          | రూ          | వు          | వూ          | శ           |
| sn26430.jpg | sn26431.jpg | sn26432.jpg | sn26433.jpg | sn26434.jpg | sn26435.jpg | sn26436.jpg | sn26437.jpg | sn26438.jpg | sn26439.jpg | sn26440.jpg |
| శా          | ళ           | ళొ          |             |             |             |             |             |             |             |             |
| sn26441.jpg | sn26442.jpg | sn26443.jpg |             |             |             |             |             |             |             |             |

# Prototype Character Images VARTHA FONT

|              |              |              |              |              |              |              |              |              |              |              |
|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| అ            | ఆ            | ఇ            | ఈ            | ఉ            | ఊ            | ఋ            | ఎ            | ఐ            | ఒ            | ఓ            |
| va1.jpg      | va2.jpg      | va3.jpg      | va4.jpg      | va5.jpg      | va6.jpg      | va7.jpg      | va8.jpg      | va9.jpg      | va10.jpg     | va11.jpg     |
| ఔ            | ౦            | ౧            | ౨            | ౩            | ౪            | ౫            | ౬            | ౭            | ౮            | ౯            |
| va13.jpg     | va14.jpg     | va15.jpg     | va16.jpg     | va17.jpg     | va18.jpg     | va19.jpg     | va20.jpg     | va21.jpg     | va22.jpg     | va23.jpg     |
| త్           | ద్           | వ్           | బ్           | మ్           | య్           | ర్           | ళ్           | ల్           | ళ్           | ళ్           |
| va24.jpg     | va25.jpg     | va26.jpg     | va27.jpg     | va28.jpg     | va29.jpg     | va30.jpg     | va31.jpg     | va32.jpg     | va33.jpg     | va34.jpg     |
| ఖ            | జ            | ఙ            | ఞ            | ట            | ణ            | బ            | ల            | ళ            | ✓            | ౦            |
| va212.jpg    | va215.jpg    | va218.jpg    | va220.jpg    | va221.jpg    | va225.jpg    | va223.jpg    | va238.jpg    | va244.jpg    | va311.jpg    | va315.jpg    |
| ౧            | ౨            | ౩            | ౪            | ౫            | ౬            | ౭            | ౮            | ౯            | ౧            | ౨            |
| va316.jpg    | va317.jpg    | va318.jpg    | va320.jpg    | va321.jpg    | va323.jpg    | va324.jpg    | va325.jpg    | va326.jpg    | va327.jpg    | va328.jpg    |
| ౩            | ౪            | ౫            | ౬            | ౭            | ౮            | ౯            | ౧            | ౨            | ౩            | ౪            |
| va329.jpg    | va330.jpg    | va331.jpg    | va332.jpg    | va333.jpg    | va334.jpg    | va335.jpg    | va336.jpg    | va211311.jpg | va211312.jpg | va211313.jpg |
| క్           | కె           | కే           | కి           | క్రి         | కా           | ఖా           | ఖో           | ఖో           | ఖు           | ఖు           |
| va211314.jpg | va211315.jpg | va211316.jpg | va211317.jpg | va211318.jpg | va211319.jpg | va211320.jpg | va211321.jpg | va211322.jpg | va211323.jpg | va211324.jpg |
| ఖె           | ఖే           | ఖి           | ఖీ           | ఖౌ           | గ            | గా           | గొ           | గో           | గు           | గు           |
| va212317.jpg | va212318.jpg | va212319.jpg | va212320.jpg | va212321.jpg | va212322.jpg | va212323.jpg | va212324.jpg | va212325.jpg | va212326.jpg | va212327.jpg |
| గె           | గే           | గి           | గీ           | గౌ           | జా           | జొ           | జో           | జు           | జూ           | జే           |
| va213317.jpg | va213318.jpg | va213319.jpg | va213320.jpg | va213321.jpg | va213322.jpg | va213323.jpg | va213324.jpg | va213325.jpg | va213326.jpg | va213327.jpg |
| జే           | జి           | జీ           | జౌ           | చ            | చా           | చొ           | చో           | చె           | చే           | చి           |
| va215318.jpg | va215319.jpg | va215320.jpg | va215321.jpg | va215322.jpg | va215323.jpg | va215324.jpg | va215325.jpg | va215326.jpg | va215327.jpg | va215328.jpg |
| చీ           | చౌ           | జా           | జొ           | జో           | జు           | జూ           | జే           | జే           | జి           | జీ           |
| va216321.jpg | va216322.jpg | va216323.jpg | va216324.jpg | va216325.jpg | va216326.jpg | va216327.jpg | va216328.jpg | va216329.jpg | va216330.jpg | va216331.jpg |
| జౌ           | జౌ           | జౌ           | జౌ           | జౌ           | జౌ           | జౌ           | జౌ           | జౌ           | జౌ           | జౌ           |
| va218322.jpg | va220312.jpg | va220313.jpg | va220314.jpg | va220315.jpg | va220316.jpg | va220317.jpg | va220318.jpg | va220319.jpg | va220320.jpg | va220321.jpg |
| టో           | టు           | టూ           | టె           | టే           | టి           | టీ           | టౌ           | డ            | డా           | దొ           |
| va221314.jpg | va221315.jpg | va221316.jpg | va221317.jpg | va221318.jpg | va221319.jpg | va221320.jpg | va221321.jpg | va221322.jpg | va221323.jpg | va221324.jpg |
| దో           | దె           | దే           | డి           | డీ           | డౌ           | ణా           | ణొ           | ణో           | ణు           | ణు           |
| va223314.jpg | va223317.jpg | va223318.jpg | va223319.jpg | va223320.jpg | va223321.jpg | va223322.jpg | va223323.jpg | va223324.jpg | va223325.jpg | va223326.jpg |
| ణె           | ణే           | ణి           | ణీ           | ణౌ           | త            | తా           | తొ           | తో           | తె           | తే           |
| va225317.jpg | va225318.jpg | va225319.jpg | va225320.jpg | va225321.jpg | va225322.jpg | va225323.jpg | va225324.jpg | va225325.jpg | va225326.jpg | va225327.jpg |
| తి           | తీ           | తౌ           | ద            | దా           | దొ           | దో           | దె           | దే           | ది           | దీ           |
| va226320.jpg | va226321.jpg | va226322.jpg | va226323.jpg | va226324.jpg | va226325.jpg | va226326.jpg | va226327.jpg | va226328.jpg | va226329.jpg | va226330.jpg |

## Prototype Character Images VARTHA FONT (Contd.)

|                 |                 |                 |                 |                 |                 |                 |                 |                 |                 |                 |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| దా              | న               | నా              | నొ              | నో              | నె              | నే              | ని              | నీ              | నౌ              | పా              |
| va230322.jpg    | va230311.jpg    | va230312.jpg    | va230313.jpg    | va230314.jpg    | va230317.jpg    | va230318.jpg    | va230320.jpg    | va230321.jpg    | va230322.jpg    | va231312.jpg    |
| పొ              | పో              | పౌ              | బా              | బొ              | బో              | బు              | బూ              | బె              | బే              | బి              |
| va231313.jpg    | va231314.jpg    | va231322.jpg    | va233312.jpg    | va233313.jpg    | va233314.jpg    | va233315.jpg    | va233316.jpg    | va233317.jpg    | va233318.jpg    | va233320.jpg    |
| బీ              | బౌ              | భ               | భా              | భి              | భీ              | భౌ              | మ               | మొ              | మో              | మె              |
| va233321.jpg    | va233322.jpg    | va234311.jpg    | va234312.jpg    | va234320.jpg    | va234321.jpg    | va234322.jpg    | va235311.jpg    | va235312.jpg    | va235314.jpg    | va235317.jpg    |
| మే              | మి              | మీ              | య               | యో              | యె              | యే              | యి              | యొ              | యా              | ర               |
| va235318.jpg    | va235320.jpg    | va235321.jpg    | va236311.jpg    | va236314.jpg    | va236317.jpg    | va236318.jpg    | va236320.jpg    | va236321.jpg    | va236322.jpg    | va237311.jpg    |
| రా              | రొ              | రో              | రె              | రే              | రి              | రీ              | రౌ              | లా              | లొ              | లో              |
| va237312.jpg    | va237313.jpg    | va237314.jpg    | va237317.jpg    | va237318.jpg    | va237320.jpg    | va237321.jpg    | va237322.jpg    | va238312.jpg    | va238313.jpg    | va238314.jpg    |
| లు              | లూ              | లే              | లే              | లి              | లీ              | లౌ              | వ               | వా              | వొ              | వో              |
| va238315.jpg    | va238316.jpg    | va238317.jpg    | va238318.jpg    | va238320.jpg    | va238321.jpg    | va238322.jpg    | va239311.jpg    | va239312.jpg    | va239313.jpg    | va239314.jpg    |
| వె              | వే              | వి              | వీ              | వౌ              | శ               | శా              | శొ              | శో              | శె              | శే              |
| va239317.jpg    | va239318.jpg    | va239320.jpg    | va239321.jpg    | va239322.jpg    | va240311.jpg    | va240312.jpg    | va240313.jpg    | va240314.jpg    | va240317.jpg    | va240318.jpg    |
| శి              | శీ              | శౌ              | షా              | షొ              | షో              | షౌ              | సా              | సొ              | సో              | సౌ              |
| va240320.jpg    | va240321.jpg    | va240322.jpg    | va241312.jpg    | va241313.jpg    | va241314.jpg    | va241322.jpg    | va242312.jpg    | va242313.jpg    | va242314.jpg    | va242322.jpg    |
| ఱా              | ఱొ              | ఱో              | ఱు              | ఱూ              | ఱె              | ఱే              | ఱి              | ఱీ              | ఱౌ              | క               |
| va244312.jpg    | va244313.jpg    | va244314.jpg    | va244315.jpg    | va244316.jpg    | va244317.jpg    | va244318.jpg    | va244320.jpg    | va244321.jpg    | va244322.jpg    | va245311.jpg    |
| కా              | కొ              | కో              | కె              | కే              | కి              | కీ              | కౌ              | కు              | కూ              | చు              |
| va245312.jpg    | va245313.jpg    | va245314.jpg    | va245317.jpg    | va245318.jpg    | va245320.jpg    | va245321.jpg    | va245322.jpg    | va211311315.jpg | va211311316.jpg | va211311315.jpg |
| చూ              | ఝ               | ఞ               | ఞ               | డు              | డూ              | తు              | తూ              | దు              | దూ              | ను              |
| va216311316.jpg | va216311312.jpg | va219311315.jpg | va219311316.jpg | va223311315.jpg | va223311316.jpg | va226311315.jpg | va226311316.jpg | va228311315.jpg | va228311316.jpg | va230311315.jpg |
| నూ              | భు              | భూ              | మా              | ము              | మూ              | మౌ              | యా              | యు              | యూ              | రు              |
| va230311316.jpg | va234311315.jpg | va234311316.jpg | va235311312.jpg | va235311315.jpg | va235311316.jpg | va235311322.jpg | va236311312.jpg | va236311315.jpg | va236311316.jpg | va237311315.jpg |
| రూ              | వు              | వూ              | శు              | శూ              | కు              | కూ              | ౌ               | ౌ               | రూ              | హౌ              |
| va237311316.jpg | va238311315.jpg | va238311316.jpg | va240311315.jpg | va240311316.jpg | va245311315.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg |
| హౌ              | ఎ               | ట               | ౌ               | హౌ              | హౌ              | ద               | ద               | ౌ               | హౌ              | ౌ               |
| va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg |
| యా              | ౌ               | హౌ              | రు              | ౌ               | జ               | ట               | ణ               | మి              | ౌ               | బ               |
| va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg |
| ౌ               | ల               | న               | ని              | ద               | ౌ               | హౌ              | ద               | హౌ              | ౌ               | ౌ               |
| va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg |
| ౌ               | ౌ               | ౌ               | ౌ               | ౌ               | ౌ               | ౌ               | ౌ               | ౌ               | ౌ               | ౌ               |
| va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg | va245311316.jpg |



## Prototype Character Images VENNELA FONT

|             |             |             |             |             |             |             |             |             |             |             |
|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|-------------|
| అ           | ఆ           | ఇ           | ఈ           | ఉ           | ఊ           | ఋ           | ఎ           | ఐ           | ఒ           | ఓ           |
| ve001.jpg   | ve002.jpg   | ve003.jpg   | ve004.jpg   | ve005.jpg   | ve006.jpg   | ve007.jpg   | ve008.jpg   | ve010.jpg   | ve011.jpg   | ve012.jpg   |
| ఔ           | ౦           | ౧           | ఖ్          | గ్          | జ్          | చ్          | ట్          | డ్          | ణ్          | త్          |
| ve013.jpg   | ve014.jpg   | ve015.jpg   | ve016.jpg   | ve017.jpg   | ve018.jpg   | ve019.jpg   | ve021.jpg   | ve022.jpg   | ve023.jpg   | ve024.jpg   |
| ద్          | వ్          | బ్          | మ్          | య్          | ర్          | ల్          | ల్          | ల్          | శ్          | ఖ           |
| ve025.jpg   | ve026.jpg   | ve027.jpg   | ve028.jpg   | ve029.jpg   | ve030.jpg   | ve031.jpg   | ve032.jpg   | ve033.jpg   | ve034.jpg   | ve212.jpg   |
| ఙ           | ఞ           | ఞ్          | ట           | ణ           | బ           | ల           | ఱ           | ృ           | ౄ           | ూ           |
| ve215.jpg   | ve218.jpg   | ve220.jpg   | ve221.jpg   | ve225.jpg   | ve233.jpg   | ve238.jpg   | ve244.jpg   | ve311.jpg   | ve315.jpg   | ve316.jpg   |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |
| ve317.jpg   | ve318.jpg   | ve320.jpg   | ve321.jpg   | ve322.jpg   | ve323.jpg   | ve324.jpg   | ve325.jpg   | ve326.jpg   | ve327.jpg   | ve328.jpg   |
| ౄ           | ౄ           | ౄ           | ౄ           | ౄ           | ౄ           | ౄ           | ౄ           | ౄ           | ౄ           | ౄ           |
| ve329.jpg   | ve330.jpg   | ve331.jpg   | ve332.jpg   | ve333.jpg   | ve334.jpg   | ve335.jpg   | ve336.jpg   | ve337.jpg   | ve21311.jpg | ve21312.jpg |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |
| ve21313.jpg | ve21314.jpg | ve21317.jpg | ve21318.jpg | ve21320.jpg | ve21321.jpg | ve21322.jpg | ve21323.jpg | ve21324.jpg | ve21325.jpg | ve21326.jpg |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |
| ve21326.jpg | ve21327.jpg | ve21328.jpg | ve21329.jpg | ve21330.jpg | ve21331.jpg | ve21332.jpg | ve21333.jpg | ve21334.jpg | ve21335.jpg | ve21336.jpg |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |
| ve21338.jpg | ve21339.jpg | ve21340.jpg | ve21341.jpg | ve21342.jpg | ve21343.jpg | ve21344.jpg | ve21345.jpg | ve21346.jpg | ve21347.jpg | ve21348.jpg |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |
| ve21350.jpg | ve21351.jpg | ve21352.jpg | ve21353.jpg | ve21354.jpg | ve21355.jpg | ve21356.jpg | ve21357.jpg | ve21358.jpg | ve21359.jpg | ve21360.jpg |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |
| ve21362.jpg | ve21363.jpg | ve21364.jpg | ve21365.jpg | ve21366.jpg | ve21367.jpg | ve21368.jpg | ve21369.jpg | ve21370.jpg | ve21371.jpg | ve21372.jpg |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |
| ve22031.jpg | ve22032.jpg | ve22033.jpg | ve22034.jpg | ve22035.jpg | ve22036.jpg | ve22037.jpg | ve22038.jpg | ve22039.jpg | ve22040.jpg | ve22041.jpg |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |
| ve22131.jpg | ve22132.jpg | ve22133.jpg | ve22134.jpg | ve22135.jpg | ve22136.jpg | ve22137.jpg | ve22138.jpg | ve22139.jpg | ve22140.jpg | ve22141.jpg |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |
| ve22331.jpg | ve22332.jpg | ve22333.jpg | ve22334.jpg | ve22335.jpg | ve22336.jpg | ve22337.jpg | ve22338.jpg | ve22339.jpg | ve22340.jpg | ve22341.jpg |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |
| ve22531.jpg | ve22532.jpg | ve22533.jpg | ve22534.jpg | ve22535.jpg | ve22536.jpg | ve22537.jpg | ve22538.jpg | ve22539.jpg | ve22540.jpg | ve22541.jpg |
| ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           | ౄ           | ృ           |

## Prototype Character Images VENNELA FONT (Contd.)

|                 |                 |                 |                 |                 |                 |                 |                 |                 |                 |                 |
|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|-----------------|
| తీ              | తౌ              | ద               | దా              | దొ              | దో              | దె              | దే              | దీ              | దీ              | దౌ              |
| ve226321.jpg    | ve226322.jpg    | ve226311.jpg    | ve226312.jpg    | ve226313.jpg    | ve226314.jpg    | ve226317.jpg    | ve226318.jpg    | ve226320.jpg    | ve226321.jpg    | ve226322.jpg    |
| న               | నా              | నొ              | నో              | నె              | నే              | ని              | నీ              | నౌ              | పా              | పొ              |
| ve230311.jpg    | ve230312.jpg    | ve230313.jpg    | ve230314.jpg    | ve230317.jpg    | ve230318.jpg    | ve230320.jpg    | ve230321.jpg    | ve230322.jpg    | ve231312.jpg    | ve231313.jpg    |
| పో              | పొ              | బా              | బొ              | బో              | బు              | బూ              | బె              | బే              | బీ              | బౌ              |
| ve231314.jpg    | ve231322.jpg    | ve233312.jpg    | ve233313.jpg    | ve233314.jpg    | ve233315.jpg    | ve233316.jpg    | ve233317.jpg    | ve233318.jpg    | ve233320.jpg    | ve233321.jpg    |
| భా              | భ               | భా              | భీ              | భే              | భౌ              | మ               | మొ              | మౌ              | మె              | మౌ              |
| ve233322.jpg    | ve234311.jpg    | ve234312.jpg    | ve234320.jpg    | ve234321.jpg    | ve234322.jpg    | ve235311.jpg    | ve235312.jpg    | ve235314.jpg    | ve235317.jpg    | ve235318.jpg    |
| మీ              | మీ              | య               | యౌ              | యె              | యే              | యి              | యీ              | యౌ              | ర               | రా              |
| ve235320.jpg    | ve235321.jpg    | ve236311.jpg    | ve236314.jpg    | ve236317.jpg    | ve236318.jpg    | ve236320.jpg    | ve236321.jpg    | ve236322.jpg    | ve237311.jpg    | ve237312.jpg    |
| రా              | రో              | రె              | రే              | రి              | రీ              | రౌ              | లా              | లౌ              | లో              | లు              |
| ve237313.jpg    | ve237314.jpg    | ve237317.jpg    | ve237318.jpg    | ve237320.jpg    | ve237321.jpg    | ve237322.jpg    | ve238312.jpg    | ve238313.jpg    | ve238314.jpg    | ve238315.jpg    |
| లూ              | లె              | లే              | లి              | లీ              | లౌ              | వ               | వా              | వౌ              | వో              | వె              |
| ve239316.jpg    | ve239317.jpg    | ve239318.jpg    | ve239320.jpg    | ve239321.jpg    | ve239322.jpg    | ve239311.jpg    | ve239312.jpg    | ve239313.jpg    | ve239314.jpg    | ve239317.jpg    |
| వే              | వీ              | వీ              | వౌ              | శ               | శా              | శొ              | శో              | శె              | శే              | శి              |
| ve239318.jpg    | ve239320.jpg    | ve239321.jpg    | ve239322.jpg    | ve240311.jpg    | ve240312.jpg    | ve240313.jpg    | ve240314.jpg    | ve240317.jpg    | ve240318.jpg    | ve240320.jpg    |
| శీ              | శౌ              | షౌ              | షౌ              | షో              | షౌ              | సౌ              | సౌ              | సో              | సౌ              | ఱౌ              |
| ve240321.jpg    | ve240322.jpg    | ve241312.jpg    | ve241313.jpg    | ve241314.jpg    | ve241322.jpg    | ve242312.jpg    | ve242313.jpg    | ve242314.jpg    | ve242322.jpg    | ve244312.jpg    |
| ఱౌ              | ఱో              | ఱు              | ఱౌ              | ఱె              | ఱే              | ఱీ              | ఱీ              | ఱౌ              | ళ               | ళౌ              |
| ve244313.jpg    | ve244314.jpg    | ve244315.jpg    | ve244316.jpg    | ve244317.jpg    | ve244318.jpg    | ve244320.jpg    | ve244321.jpg    | ve244322.jpg    | ve245311.jpg    | ve245312.jpg    |
| ళౌ              | ళో              | ళె              | ళే              | ళి              | ళీ              | ళౌ              | చు              | చౌ              | డు              | డౌ              |
| ve245313.jpg    | ve245314.jpg    | ve245317.jpg    | ve245318.jpg    | ve245320.jpg    | ve245321.jpg    | ve245322.jpg    | ve216311315.jpg | ve216311316.jpg | ve223311315.jpg | ve223311316.jpg |
| తు              | తుౌ             | దు              | దౌ              | ను              | నౌ              | భు              | భౌ              | మౌ              | ము              | మౌ              |
| ve226311315.jpg | ve226311316.jpg | ve228311315.jpg | ve228311316.jpg | ve230311315.jpg | ve230311316.jpg | ve234311315.jpg | ve234311316.jpg | ve235311312.jpg | ve235311315.jpg | ve235311316.jpg |
| మౌ              | యౌ              | యు              | యౌ              | రు              | రౌ              | వు              | వౌ              | శు              | శౌ              | ళు              |
| ve235311322.jpg | ve236311312.jpg | ve236311315.jpg | ve236311316.jpg | ve237311315.jpg | ve237311316.jpg | ve239311315.jpg | ve239311316.jpg | ve240311315.jpg | ve240311316.jpg | ve245311315.jpg |
| ళౌ              | శ               | ని              | ని              | ని              | ఎ               | వౌ              | డ               | డ               | .               | ఎ               |
| ve245311316.jpg | veL11.jpg       | veL12.jpg       | veL13.jpg       | veL14.jpg       | veL15.jpg       | veL17.jpg       | veL21.jpg       | veL22.jpg       | veL23.jpg       | veL25.jpg       |
| 0               | 1               | 2               | 3               | .               | 4               | 5               | 6               | 7               | 8               | 9               |
| veL26.jpg       | veL27.jpg       | veL28.jpg       | veL29.jpg       | veL31.jpg       | veL41.jpg       | veL42.jpg       | veL43.jpg       | veL44.jpg       | veL45.jpg       | veL46.jpg       |

# DRISHTI Templates

## HEMALATHA FONT

|         |         |         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| అ       | ఆ       | ఇ       | ఈ       | ఉ       | ఊ       | ఋ       | ఎ       | ఐ       | ఒ       |
| 1.jpg   | 2.jpg   | 3.jpg   | 4.jpg   | 5.jpg   | 6.jpg   | 7.jpg   | 8.jpg   | 10.jpg  | 11.jpg  |
| ఓ       | ఔ       | ం       | క       | గ       | జ       | చ       | ట       | డ       | ణ       |
| 12.jpg  | 13.jpg  | 14.jpg  | 15.jpg  | 17.jpg  | 18.jpg  | 19.jpg  | 21.jpg  | 22.jpg  | 23.jpg  |
| త       | ద       | వ       | బ       | మ       | య       | ర       | ల్      | ల్      | ళ       |
| 24.jpg  | 25.jpg  | 26.jpg  | 27.jpg  | 28.jpg  | 29.jpg  | 30.jpg  | 31.jpg  | 32.jpg  | 33.jpg  |
| శ       | ష       | షో      | న్      | జ       | జ       | ష       | ట       | ణ       | బ       |
| 34.jpg  | 38.jpg  | 39.jpg  | 40.jpg  | 215.jpg | 218.jpg | 220.jpg | 221.jpg | 225.jpg | 233.jpg |
| ల       | ఱ       | ✓       | ృ       | ౌ       | ౓       | ౔       | ౕ       | ౖ       | ౗       |
| 238.jpg | 244.jpg | 311.jpg | 315.jpg | 316.jpg | 317.jpg | 318.jpg | 320.jpg | 321.jpg | 324.jpg |
| ఎ       | ఀ       | ఁ       | ం       | ః       | ఄ       | అ       | ఆ       | ఇ       | ఈ       |
| 325.jpg | 326.jpg | 327.jpg | 328.jpg | 329.jpg | 330.jpg | 331.jpg | 332.jpg | 333.jpg | 334.jpg |
| ఁ       | ం       | ః       | ఄ       | అ       | ఆ       | ఇ       | ఈ       | ఉ       | ఊ       |
| 335.jpg | 336.jpg | 337.jpg | 340.jpg | 343.jpg | 344.jpg | 345.jpg | 346.jpg | 347.jpg | 350.jpg |
| ఊ       | ఋ       | ౠ       | ౡ       | ౢ       | ౣ       | ౤       | ౥       | ౦       | ౠ       |
| 351.jpg | 352.jpg | 354.jpg | 355.jpg | 356.jpg | 357.jpg | 358.jpg | 359.jpg | 360.jpg | 361.jpg |
| ఋ       | ౠ       | ౡ       | ౢ       | ౣ       | ౤       | ౥       | ౦       | ౠ       | ౡ       |
| 362.jpg | 363.jpg | 370.jpg | 371.jpg | 372.jpg | 373.jpg | 374.jpg | 375.jpg | 376.jpg | 377.jpg |
| ౢ       | ౣ       | ౤       | ౥       | ౦       | ౠ       | ౡ       | ౢ       | ౣ       | ౤       |
| 378.jpg | 379.jpg | 380.jpg | 381.jpg | 382.jpg | 383.jpg | 384.jpg | 385.jpg | 386.jpg | 387.jpg |

# DRISHTI Templates

## HEMALATHA FONT (Contd.)

|            |            |            |            |            |            |            |            |            |            |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| రూ         | రీ         | రీ         | రు         | రూ         | రె         | రే         | రె         | రూ         | రూ         |
| 388.jpg    | 389.jpg    | 390.jpg    | 391.jpg    | 392.jpg    | 393.jpg    | 394.jpg    | 395.jpg    | 396.jpg    | 397.jpg    |
| ఎ          | ఎ          | ఎ          | ఎ          | ఎ          | బ          | బ          | బ          | యె         | యె         |
| 398.jpg    | 399.jpg    | 400.jpg    | 401.jpg    | 402.jpg    | 403.jpg    | 404.jpg    | 405.jpg    | 406.jpg    | 407.jpg    |
| ఎ          | ఎ          | ఎ          | రీ         | వ్         | క          | క          | క          | కో         | కె         |
| 408.jpg    | 409.jpg    | 410.jpg    | 411.jpg    | 412.jpg    | 211311.jpg | 211312.jpg | 211313.jpg | 211314.jpg | 211317.jpg |
| క          | క          | క          | క          | గ          | గా         | గ          | గ          | గు         | గూ         |
| 211318.jpg | 211320.jpg | 211321.jpg | 211322.jpg | 213311.jpg | 213312.jpg | 213313.jpg | 213314.jpg | 213315.jpg | 213316.jpg |
| గ          | గ          | గ          | గ          | గ          | జ          | జ          | జ          | జు         | జూ         |
| 213317.jpg | 213318.jpg | 213320.jpg | 213321.jpg | 213322.jpg | 215312.jpg | 215313.jpg | 215314.jpg | 215315.jpg | 215316.jpg |
| జ          | జ          | జ          | జ          | జ          | చ          | చ          | చ          | చ          | చ          |
| 215317.jpg | 215318.jpg | 215320.jpg | 215321.jpg | 215322.jpg | 216311.jpg | 216312.jpg | 216313.jpg | 216314.jpg | 216317.jpg |
| చ          | చ          | చ          | చ          | జ          | జ          | జ          | జ          | జ          | జ          |
| 216318.jpg | 216320.jpg | 216321.jpg | 216322.jpg | 218312.jpg | 218313.jpg | 218314.jpg | 218315.jpg | 218316.jpg | 218317.jpg |
| జ          | జ          | జ          | జ          | జ          | జ          | జ          | జ          | జ          | జ          |
| 218318.jpg | 218320.jpg | 218321.jpg | 218322.jpg | 220312.jpg | 220313.jpg | 220314.jpg | 220317.jpg | 220318.jpg | 220320.jpg |
| జ          | జ          | ట          | ట          | ట          | ట          | ట          | ట          | ట          | ట          |
| 220321.jpg | 220322.jpg | 221312.jpg | 221313.jpg | 221314.jpg | 221315.jpg | 221316.jpg | 221317.jpg | 221318.jpg | 221320.jpg |
| ట          | ట          | డ          | డ          | డ          | డ          | డ          | డ          | డి         | డి         |
| 221321.jpg | 221322.jpg | 223311.jpg | 223312.jpg | 223313.jpg | 223314.jpg | 223317.jpg | 223318.jpg | 223320.jpg | 223321.jpg |



# DRISHTI Templates

## HEMALATHA FONT (Contd.)

|            |            |            |            |            |             |            |            |            |            |
|------------|------------|------------|------------|------------|-------------|------------|------------|------------|------------|
| డా         | ణా         | ణో         | ణో         | ణు         | ణూ          | ణె         | ణే         | ణి         | ణీ         |
| 223322.jpg | 225312.jpg | 225313.jpg | 225314.jpg | 225315.jpg | 225316.jpg  | 225317.jpg | 225318.jpg | 225320.jpg | 225321.jpg |
| ణో         | త          | తా         | తో         | తో         | తె          | తే         | తి         | తీ         | తా         |
| 225322.jpg | 226311.jpg | 226312.jpg | 226313.jpg | 226314.jpg | 226317.jpg  | 226318.jpg | 226320.jpg | 226321.jpg | 226322.jpg |
| ద          | దా         | దో         | దో         | దె         | దే          | ది         | దీ         | దా         | న          |
| 228311.jpg | 228312.jpg | 228313.jpg | 228314.jpg | 228317.jpg | 228318.jpg  | 228320.jpg | 228321.jpg | 228322.jpg | 230311.jpg |
| నా         | నో         | నో         | నె         | నే         | ని          | నీ         | నా         | పా         | పా         |
| 230312.jpg | 230313.jpg | 230314.jpg | 230317.jpg | 230318.jpg | 230320.jpg  | 230321.jpg | 230322.jpg | 231312.jpg | 231313.jpg |
| పో         | పా         | బా         | బో         | బో         | బు          | బూ         | బె         | బే         | బి         |
| 231314.jpg | 231322.jpg | 233312.jpg | 233313.jpg | 233314.jpg | 233315.jpg  | 233316.jpg | 233317.jpg | 233318.jpg | 233320.jpg |
| బీ         | బా         | మ          | మె         | మె         | మె          | మే         | మి         | మీ         | మా         |
| 233321.jpg | 233322.jpg | 235311.jpg | 235313.jpg | 235314.jpg | 235317.jpg  | 235318.jpg | 235320.jpg | 235321.jpg | 235322.jpg |
| య          | యె         | యే         | యి         | యీ         | యా          | ర          | రా         | రో         | రో         |
| 236311.jpg | 236317.jpg | 236318.jpg | 236320.jpg | 236321.jpg | 236322.jpg  | 237311.jpg | 237312.jpg | 237313.jpg | 237314.jpg |
| రె         | రే         | రి         | రీ         | రా         | లా          | లో         | లో         | లు         | లూ         |
| 237317.jpg | 237318.jpg | 237320.jpg | 237321.jpg | 237322.jpg | 238312.jpg  | 238313.jpg | 238314.jpg | 238315.jpg | 238316.jpg |
| తె         | తే         | లి         | లీ         | లా         | వ           | వా         | వో         | వో         | వ          |
| 238317.jpg | 238318.jpg | 238320.jpg | 238321.jpg | 238322.jpg | 239311r.jpg | 239312.jpg | 239313.jpg | 239314.jpg | 239317.jpg |
| చే         | వి         | వీ         | వా         | శ          | శా          | శో         | శో         | శె         | శే         |
| 239318.jpg | 239320.jpg | 239321.jpg | 239322.jpg | 240311.jpg | 240312.jpg  | 240313.jpg | 240314.jpg | 240317.jpg | 240318.jpg |

**DRISHTI Templates**  
**HEMALATHA FONT (Contd.)**

|               |               |               |               |               |               |               |               |               |               |
|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| శి            | శీ            | శౌ            | షా            | షి            | షో            | షౌ            | సా            | సీ            | సో            |
| 240320.jpg    | 240321.jpg    | 240322.jpg    | 241312.jpg    | 241313.jpg    | 241314.jpg    | 241322.jpg    | 242312.jpg    | 242313.jpg    | 242314.jpg    |
| సౌ            | టా            | టి            | టో            | టు            | టూ            | టె            | టే            | టి            | టీ            |
| 242322.jpg    | 244312.jpg    | 244313.jpg    | 244314.jpg    | 244315.jpg    | 244316.jpg    | 244317.jpg    | 244318.jpg    | 244320.jpg    | 244321.jpg    |
| టౌ            | ళ             | ళా            | ళి            | ళో            | ళె            | ళే            | ళి            | ళీ            | ళో            |
| 244322.jpg    | 245311.jpg    | 245312.jpg    | 245313.jpg    | 245314.jpg    | 245317.jpg    | 245318.jpg    | 245320.jpg    | 245321.jpg    | 245322.jpg    |
| కు            | కూ            | చు            | చూ            | డు            | డూ            | తు            | తూ            | దు            | దూ            |
| 211311315.jpg | 211311316.jpg | 216311315.jpg | 216311316.jpg | 223311315.jpg | 223311316.jpg | 226311315.jpg | 226311316.jpg | 228311315.jpg | 228311316.jpg |
| ను            | నూ            | మా            | ము            | మూ            | యా            | యు            | యూ            | రు            | రూ            |
| 230311315.jpg | 230311316.jpg | 235311312.jpg | 235311315.jpg | 235311316.jpg | 236311312.jpg | 236311315.jpg | 236311316.jpg | 237311315.jpg | 237311316.jpg |
| వు            | వూ            | శు            | శూ            | ళు            | ళూ            |               |               |               |               |
| 239311315.jpg | 239311316.jpg | 240311315.jpg | 240311316.jpg | 245311315.jpg | 245311316.jpg |               |               |               |               |

# DRISHTI Templates

## SRILIPi FONT

|         |         |         |         |         |         |         |         |         |         |
|---------|---------|---------|---------|---------|---------|---------|---------|---------|---------|
| అ       | ఆ       | ఇ       | ఈ       | ఉ       | ఊ       | ఋ       | ఎ       | ఐ       | ఒ       |
| 1.jpg   | 2.jpg   | 3.jpg   | 4.jpg   | 5.jpg   | 6.jpg   | 7.jpg   | 8.jpg   | 10.jpg  | 11.jpg  |
| ఓ       | ఔ       | ం       | క్ష     | ఖ్      | గ్      | చ్      | ట్      | డ్      | ణ్      |
| 12.jpg  | 13.jpg  | 14.jpg  | 15.jpg  | 16.jpg  | 17.jpg  | 19.jpg  | 21.jpg  | 22.jpg  | 23.jpg  |
| త్      | ద్      | వ్      | బ్      | మ్      | య్      | ర్      | ల్      | ళ్      | శ్      |
| 24.jpg  | 25.jpg  | 26.jpg  | 27.jpg  | 28.jpg  | 29.jpg  | 30.jpg  | 32.jpg  | 33.jpg  | 34.jpg  |
| జ్      | న్      | ఛ       | ఛా      | ఛి      | ఛీ      | ఛు      | ఛూ      | ఛే      | ఛే      |
| 38.jpg  | 40.jpg  | 99.jpg  | 100.jpg | 101.jpg | 102.jpg | 103.jpg | 104.jpg | 105.jpg | 106.jpg |
| ఛా      | ఛో      | ఛౌ      | ఛ్      | ఛ       | ఛా      | ఛి      | ఛీ      | ఛు      | ఛూ      |
| 107.jpg | 108.jpg | 109.jpg | 110.jpg | 159.jpg | 160.jpg | 161.jpg | 162.jpg | 163.jpg | 164.jpg |
| ఛే      | ఛై      | ఛౌ      | ఛో      | ఛౌ      | ఛ్      | ఛ       | ఛా      | ఛూ      | ఛ       |
| 165.jpg | 166.jpg | 167.jpg | 168.jpg | 169.jpg | 170.jpg | 200.jpg | 201.jpg | 206.jpg | 207.jpg |
| ఛా      | ఛి      | ఛీ      | ఛు      | ఛ       | ఛే      | ఛ       | ఛ       | ఛో      | ఛౌ      |
| 208.jpg | 209.jpg | 210.jpg | 211.jpg | 212.jpg | 213.jpg | 214.jpg | 215.jpg | 216.jpg | 217.jpg |
| జ       | జా      | ట       | ణ       | బ       | ల       | ఱ       | ఫా      | ఫా      | ఫా      |
| 218.jpg | 220.jpg | 221.jpg | 225.jpg | 233.jpg | 238.jpg | 244.jpg | 250.jpg | 251.jpg | 252.jpg |
| ఫా      | భ్      | ఁ       | ా       | ఱ       | ఱ       | ఁ       | ా       | ా       | ఁ       |
| 253.jpg | 278.jpg | 311.jpg | 312.jpg | 313.jpg | 314.jpg | 315.jpg | 316.jpg | 317.jpg | 318.jpg |
| ౧       | ౨       | ౩       | ౪       | ౫       | ౬       | ౭       | ౮       | ౯       | ౧౦      |
| 320.jpg | 321.jpg | 322.jpg | 323.jpg | 324.jpg | 325.jpg | 326.jpg | 327.jpg | 328.jpg | 329.jpg |



# DRISHTI Templates

## SRILIPi FONT (Contd.)

|            |            |            |            |            |            |            |            |            |            |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| ఎ          | ఁ          | ఏ          | వ          | ఁ          | ఏ          | ఎ          | ఎ          | ల          | ఁ          |
| 330.jpg    | 331.jpg    | 333.jpg    | 334.jpg    | 335.jpg    | 337.jpg    | 338.jpg    | 340.jpg    | 343.jpg    | 344.jpg    |
| ఁ          | న          | ఎ          | ఁ          | ఁ          | హా         | హా         | హా         | హా         | హా         |
| 345.jpg    | 346.jpg    | 347.jpg    | 351.jpg    | 352.jpg    | 354.jpg    | 355.jpg    | 356.jpg    | 359.jpg    | 360.jpg    |
| నూ         | ను         | న          | మూ         | ఁ          | మూ         | ఎ          | లు         | లూ         | లు         |
| 361.jpg    | 362.jpg    | 363.jpg    | 382.jpg    | 385.jpg    | 386.jpg    | 398.jpg    | 399.jpg    | 400.jpg    | 401.jpg    |
| మ          | యె         | యో         | ఎ          | మ          | ఁ          | ఁ          | ఁ          | ఁ          | ఁ          |
| 403.jpg    | 406.jpg    | 407.jpg    | 408.jpg    | 409.jpg    | 420.jpg    | 468.jpg    | 472.jpg    | 475.jpg    | 478.jpg    |
| ఎ          | .          | ,          | !          | -          | ?          | !          | ఖై         | గై         | ఖై         |
| 480.jpg    | 600.jpg    | 601.jpg    | 602.jpg    | 603.jpg    | 604.jpg    | 605.jpg    | 900.jpg    | 901.jpg    | 902.jpg    |
| ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         |
| 904.jpg    | 905.jpg    | 906.jpg    | 907.jpg    | 908.jpg    | 921.jpg    | 922.jpg    | 923.jpg    | 924.jpg    | 925.jpg    |
| ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         |
| 926.jpg    | 927.jpg    | 928.jpg    | 929.jpg    | 930.jpg    | 931.jpg    | 932.jpg    | 211311.jpg | 211312.jpg | 211313.jpg |
| ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         |
| 211314.jpg | 211317.jpg | 211318.jpg | 211320.jpg | 211321.jpg | 211322.jpg | 212312.jpg | 212313.jpg | 212314.jpg | 212315.jpg |
| ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         |
| 212316.jpg | 212317.jpg | 212318.jpg | 212320.jpg | 212321.jpg | 212322.jpg | 213311.jpg | 213312.jpg | 213313.jpg | 213314.jpg |
| ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         | ఖై         |
| 213315.jpg | 213316.jpg | 213317.jpg | 213318.jpg | 213320.jpg | 213321.jpg | 213322.jpg | 216311.jpg | 216312.jpg | 216313.jpg |

## DRISHTI Templates

### SRILIPI FONT (Contd.)

|            |            |            |            |            |            |            |            |            |            |
|------------|------------|------------|------------|------------|------------|------------|------------|------------|------------|
| చో         | చె         | చే         | చి         | చీ         | చౌ         | జా         | జౌ         | జో         | జు         |
| 216314.jpg | 216317.jpg | 216318.jpg | 216320.jpg | 216321.jpg | 216322.jpg | 218312.jpg | 218313.jpg | 218314.jpg | 218315.jpg |
| జూ         | జె         | జే         | జి         | జీ         | జౌ         | రు         | ఝ          | ఞ          | ఞు         |
| 218316.jpg | 218317.jpg | 218318.jpg | 218320.jpg | 218321.jpg | 218322.jpg | 219311.jpg | 219313.jpg | 219314.jpg | 219317.jpg |
| ఝ          | ఞ          | ఞు         | ఞు         | టా         | టౌ         | టో         | టు         | టూ         | టె         |
| 219318.jpg | 219320.jpg | 219321.jpg | 219322.jpg | 221312.jpg | 221313.jpg | 221314.jpg | 221315.jpg | 221316.jpg | 221317.jpg |
| టే         | టి         | టీ         | టౌ         | డ          | డా         | డౌ         | డో         | డె         | డే         |
| 221318.jpg | 221320.jpg | 221321.jpg | 221322.jpg | 223311.jpg | 223312.jpg | 223313.jpg | 223314.jpg | 223317.jpg | 223318.jpg |
| డి         | డీ         | డౌ         | ణా         | ణౌ         | ణో         | ణు         | ణూ         | ణె         | ణే         |
| 223320.jpg | 223321.jpg | 223322.jpg | 225312.jpg | 225313.jpg | 225314.jpg | 225315.jpg | 225316.jpg | 225317.jpg | 225318.jpg |
| ణి         | ణీ         | ణౌ         | త          | తా         | తౌ         | తో         | తె         | తే         | తి         |
| 225320.jpg | 225321.jpg | 225322.jpg | 226311.jpg | 226312.jpg | 226313.jpg | 226314.jpg | 226317.jpg | 226318.jpg | 226320.jpg |
| తీ         | తౌ         | ద          | దా         | దౌ         | దో         | దె         | దే         | ది         | దీ         |
| 226321.jpg | 226322.jpg | 228311.jpg | 228312.jpg | 228313.jpg | 228314.jpg | 228317.jpg | 228318.jpg | 228320.jpg | 228321.jpg |
| దౌ         | న          | నా         | నౌ         | నో         | నె         | నే         | నీ         | నౌ         | పా         |
| 228322.jpg | 230311.jpg | 230312.jpg | 230313.jpg | 230314.jpg | 230317.jpg | 230318.jpg | 230321.jpg | 230322.jpg | 231312.jpg |
| పా         | పొ         | పౌ         | బా         | బౌ         | బో         | బు         | బూ         | బె         | బే         |
| 231313.jpg | 231314.jpg | 231322.jpg | 233312.jpg | 233313.jpg | 233314.jpg | 233315.jpg | 233316.jpg | 233317.jpg | 233318.jpg |
| బి         | బీ         | బౌ         | భ          | భా         | భౌ         | భో         | భె         | భే         | భి         |
| 233320.jpg | 233321.jpg | 233322.jpg | 234311.jpg | 234312.jpg | 234313.jpg | 234314.jpg | 234317.jpg | 234318.jpg | 234320.jpg |

## DRISHTI Templates

### SRILIPi FONT (Contd.)

|               |               |               |               |               |               |               |               |               |               |
|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|---------------|
| భీ            | భౌ            | మ             | మొ            | మో            | మె            | మే            | మి            | మీ            | మౌ            |
| 234321.jpg    | 234322.jpg    | 235311.jpg    | 235313.jpg    | 235314.jpg    | 235317.jpg    | 235318.jpg    | 235320.jpg    | 235321.jpg    | 235322.jpg    |
| య             | యె            | యే            | యి            | యా            | యౌ            | ర             | రా            | రొ            | రో            |
| 236311.jpg    | 236317.jpg    | 236318.jpg    | 236320.jpg    | 236321.jpg    | 236322.jpg    | 237311.jpg    | 237312.jpg    | 237313.jpg    | 237314.jpg    |
| రె            | రే            | రి            | రీ            | రొ            | లా            | లొ            | లో            | లు            | లూ            |
| 237317.jpg    | 237318.jpg    | 237320.jpg    | 237321.jpg    | 237322.jpg    | 238312.jpg    | 238313.jpg    | 238314.jpg    | 238315.jpg    | 238316.jpg    |
| లె            | లే            | లి            | లీ            | లొ            | వా            | వొ            | వో            | వె            | వే            |
| 238317.jpg    | 238318.jpg    | 238320.jpg    | 238321.jpg    | 238322.jpg    | 239312.jpg    | 239313.jpg    | 239314.jpg    | 239317.jpg    | 239318.jpg    |
| వీ            | వౌ            | శ             | శొ            | శౌ            | శో            | శె            | శే            | శి            | శీ            |
| 239321.jpg    | 239322.jpg    | 240311.jpg    | 240312.jpg    | 240313.jpg    | 240314.jpg    | 240317.jpg    | 240318.jpg    | 240320.jpg    | 240321.jpg    |
| శౌ            | షొ            | షొ            | షో            | షౌ            | సొ            | సొ            | సో            | సౌ            | టొ            |
| 240322.jpg    | 241312.jpg    | 241313.jpg    | 241314.jpg    | 241322.jpg    | 242312.jpg    | 242313.jpg    | 242314.jpg    | 242322.jpg    | 244312.jpg    |
| టొ            | టో            | ఱు            | ఱూ            | టె            | టే            | టి            | టీ            | టౌ            | ళ             |
| 244313.jpg    | 244314.jpg    | 244315.jpg    | 244316.jpg    | 244317.jpg    | 244318.jpg    | 244320.jpg    | 244321.jpg    | 244322.jpg    | 245311.jpg    |
| ళా            | ళొ            | ళొ            | ళె            | ళే            | ళి            | ళీ            | ళౌ            | కు            | కూ            |
| 245312.jpg    | 245313.jpg    | 245314.jpg    | 245317.jpg    | 245318.jpg    | 245320.jpg    | 245321.jpg    | 245322.jpg    | 211311315.jpg | 211311316.jpg |
| చు            | చూ            | ఝ             | ఝు            | ఝూ            | డు            | డూ            | తు            | తూ            | దు            |
| 216311315.jpg | 216311316.jpg | 219311312.jpg | 219311315.jpg | 219311316.jpg | 223311315.jpg | 223311316.jpg | 226311315.jpg | 226311316.jpg | 228311315.jpg |
| దూ            | ను            | నూ            | భు            | భూ            | మా            | ము            | మూ            | యా            | యు            |
| 228311316.jpg | 230311315.jpg | 230311316.jpg | 234311315.jpg | 234311316.jpg | 235311312.jpg | 235311315.jpg | 235311316.jpg | 236311312.jpg | 236311315.jpg |
| యా            | రు            | రూ            | వు            | వూ            | శు            | శూ            | ళు            | ళూ            |               |
| 236311316.jpg | 237311315.jpg | 237311316.jpg | 239311315.jpg | 239311316.jpg | 240311315.jpg | 240311316.jpg | 245311315.jpg | 245311316.jpg |               |

## APPENDIX - B

### Distinct Prototype Characters

| S.No. | CharCode | Prototype Image |
|-------|----------|-----------------|
| 1     | 1        | ၁               |
| 2     | 2        | ၂               |
| 3     | 3        | ၃               |
| 4     | 4        | ၄               |
| 5     | 5        | ၅               |
| 6     | 6        | ၆               |
| 7     | 7        | ၇               |
| 8     | 8        | ၈               |
| 9     | 10       | ၉               |
| 10    | 11       | ၁၀              |
| 11    | 12       | ၁၁              |
| 12    | 13       | ၁၂              |
| 13    | 14       | ၁၃              |
| 14    | 15       | ၁၄              |
| 15    | 16       | ၁၅              |
| 16    | 17       | ၁၆              |
| 17    | 18       | ၁၇              |
| 18    | 19       | ၁၈              |
| 19    | 20       | ၁၉              |
| 20    | 21       | ၂၀              |
| 21    | 22       | ၂၁              |
| 22    | 23       | ၂၂              |

|    |    |    |
|----|----|----|
| 23 | 24 | ၂၃ |
| 24 | 25 | ၂၄ |
| 25 | 26 | ၂၅ |
| 26 | 27 | ၂၆ |
| 27 | 28 | ၂၇ |
| 28 | 29 | ၂၈ |
| 29 | 30 | ၂၉ |
| 30 | 31 | ၃၀ |
| 31 | 32 | ၃၁ |
| 32 | 33 | ၃၂ |
| 33 | 34 | ၃၃ |
| 34 | 35 | ၃၄ |
| 35 | 38 | ၃၅ |
| 36 | 39 | ၃၆ |
| 37 | 41 | ၁  |
| 38 | 42 | ၂  |
| 39 | 43 | ၃  |
| 40 | 44 | ၄  |
| 41 | 45 | ၅  |
| 42 | 46 | ၆  |

|    |     |     |
|----|-----|-----|
| 43 | 47  | ၇   |
| 44 | 48  | ၈   |
| 45 | 49  | ၉   |
| 46 | 99  | ၁၀၀ |
| 47 | 100 | ၁၀၁ |
| 48 | 101 | ၁၀၂ |
| 49 | 102 | ၁၀၃ |
| 50 | 103 | ၁၀၄ |
| 51 | 104 | ၁၀၅ |
| 52 | 105 | ၁၀၆ |
| 53 | 106 | ၁၀၇ |
| 54 | 107 | ၁၀၈ |
| 55 | 108 | ၁၀၉ |
| 56 | 109 | ၁၁၀ |
| 57 | 110 | ၁၁၁ |
| 58 | 138 | ၁၁၂ |
| 59 | 159 | ၁၁၃ |
| 60 | 160 | ၁၁၄ |
| 61 | 161 | ၁၁၅ |
| 62 | 162 | ၁၁၆ |
| 63 | 163 | ၁၁၇ |

|    |     |     |
|----|-----|-----|
| 64 | 164 | ၁၁၈ |
| 65 | 165 | ၁၁၉ |
| 66 | 166 | ၁၂၀ |
| 67 | 167 | ၁၂၁ |
| 68 | 168 | ၁၂၂ |
| 69 | 169 | ၁၂၃ |
| 70 | 170 | ၁၂၄ |
| 71 | 200 | ၁၂၅ |
| 72 | 201 | ၁၂၆ |
| 73 | 206 | ၁၂၇ |
| 74 | 207 | ၁၂၈ |
| 75 | 208 | ၁၂၉ |
| 76 | 209 | ၁၃၀ |
| 77 | 210 | ၁၃၁ |
| 78 | 211 | ၁၃၂ |
| 79 | 212 | ၁၃၃ |
| 80 | 213 | ၁၃၄ |
| 81 | 214 | ၁၃၅ |
| 82 | 215 | ၁၃၆ |
| 83 | 216 | ၁၃၇ |
| 84 | 217 | ၁၃၈ |
| 85 | 218 | ၁၃၉ |

|     |     |      |
|-----|-----|------|
| 86  | 220 | မိုး |
| 87  | 221 | မိုး |
| 88  | 225 | မိုး |
| 89  | 233 | မိုး |
| 90  | 238 | မိုး |
| 91  | 244 | မိုး |
| 92  | 250 | မိုး |
| 93  | 251 | မိုး |
| 94  | 252 | မိုး |
| 95  | 253 | မိုး |
| 96  | 278 | မိုး |
| 97  | 311 | မိုး |
| 98  | 312 | မိုး |
| 99  | 313 | မိုး |
| 100 | 314 | မိုး |
| 101 | 315 | မိုး |
| 102 | 316 | မိုး |
| 103 | 317 | မိုး |
| 104 | 318 | မိုး |
| 105 | 320 | မိုး |
| 106 | 321 | မိုး |
| 107 | 322 | မိုး |
| 108 | 323 | မိုး |
| 109 | 324 | မိုး |
| 110 | 325 | မိုး |
| 111 | 326 | မိုး |
| 112 | 327 | မိုး |

|     |     |      |
|-----|-----|------|
| 113 | 328 | မိုး |
| 114 | 329 | မိုး |
| 115 | 330 | မိုး |
| 116 | 331 | မိုး |
| 117 | 332 | မိုး |
| 118 | 333 | မိုး |
| 119 | 334 | မိုး |
| 120 | 335 | မိုး |
| 121 | 336 | မိုး |
| 122 | 337 | မိုး |
| 123 | 338 | မိုး |
| 124 | 339 | မိုး |
| 125 | 340 | မိုး |
| 126 | 343 | မိုး |
| 127 | 344 | မိုး |
| 128 | 345 | မိုး |
| 129 | 346 | မိုး |
| 130 | 347 | မိုး |
| 131 | 350 | မိုး |
| 132 | 351 | မိုး |
| 133 | 352 | မိုး |
| 134 | 354 | မိုး |
| 135 | 355 | မိုး |
| 136 | 356 | မိုး |
| 137 | 357 | မိုး |
| 138 | 358 | မိုး |
| 139 | 359 | မိုး |
| 140 | 360 | မိုး |



|     |     |    |
|-----|-----|----|
| 141 | 361 | హం |
| 142 | 362 | బి |
| 143 | 363 | బి |
| 144 | 371 | బి |
| 145 | 372 | బి |
| 146 | 373 | బి |
| 147 | 374 | బి |
| 148 | 375 | బి |
| 149 | 376 | బి |
| 150 | 377 | బి |
| 151 | 378 | బి |
| 152 | 379 | బి |
| 153 | 380 | బి |
| 154 | 381 | బి |
| 155 | 382 | బి |
| 156 | 383 | బి |
| 157 | 384 | బి |
| 158 | 385 | (  |
| 159 | 386 | బి |
| 160 | 386 | )  |
| 161 | 387 | రు |
| 162 | 388 | రు |
| 163 | 389 | రు |
| 164 | 390 | రు |
| 165 | 391 | రు |
| 166 | 392 | రు |
| 167 | 393 | రు |
| 168 | 394 | రు |

|     |     |    |
|-----|-----|----|
| 169 | 395 | రు |
| 170 | 396 | రు |
| 171 | 397 | రు |
| 172 | 398 | రు |
| 173 | 399 | రు |
| 174 | 400 | రు |
| 175 | 401 | రు |
| 176 | 402 | రు |
| 177 | 403 | రు |
| 178 | 404 | రు |
| 179 | 405 | రు |
| 180 | 406 | రు |
| 181 | 407 | రు |
| 182 | 409 | రు |
| 183 | 410 | రు |
| 184 | 411 | రు |
| 185 | 420 | రు |
| 186 | 468 | రు |
| 187 | 472 | రు |
| 188 | 475 | రు |
| 189 | 478 | రు |
| 190 | 480 | రు |
| 191 | 600 | •  |
| 192 | 601 | ,  |
| 193 | 602 | !  |
| 194 | 604 | ?  |



|     |        |         |
|-----|--------|---------|
| 195 | 605    | မိုးဝါး |
| 196 | 900    | မိုးဝါး |
| 197 | 901    | မိုးဝါး |
| 198 | 902    | မိုးဝါး |
| 199 | 904    | မိုးဝါး |
| 200 | 905    | မိုးဝါး |
| 201 | 906    | မိုးဝါး |
| 202 | 907    | မိုးဝါး |
| 203 | 908    | မိုးဝါး |
| 204 | 921    | မိုးဝါး |
| 205 | 922    | မိုးဝါး |
| 206 | 923    | မိုးဝါး |
| 207 | 924    | မိုးဝါး |
| 208 | 925    | မိုးဝါး |
| 209 | 926    | မိုးဝါး |
| 210 | 927    | မိုးဝါး |
| 211 | 928    | မိုးဝါး |
| 212 | 929    | မိုးဝါး |
| 213 | 930    | မိုးဝါး |
| 214 | 931    | မိုးဝါး |
| 215 | 932    | မိုးဝါး |
| 216 | 211311 | မိုးဝါး |
| 217 | 211312 | မိုးဝါး |
| 218 | 211313 | မိုးဝါး |

|     |        |         |
|-----|--------|---------|
| 219 | 211314 | မိုးဝါး |
| 220 | 211317 | မိုးဝါး |
| 221 | 211318 | မိုးဝါး |
| 222 | 211320 | မိုးဝါး |
| 223 | 211321 | မိုးဝါး |
| 224 | 211322 | မိုးဝါး |
| 225 | 212312 | မိုးဝါး |
| 226 | 212313 | မိုးဝါး |
| 227 | 212314 | မိုးဝါး |
| 228 | 212315 | မိုးဝါး |
| 229 | 212316 | မိုးဝါး |
| 230 | 212317 | မိုးဝါး |
| 231 | 212318 | မိုးဝါး |
| 232 | 212320 | မိုးဝါး |
| 233 | 212321 | မိုးဝါး |
| 234 | 212322 | မိုးဝါး |
| 235 | 213311 | မိုးဝါး |
| 236 | 213312 | မိုးဝါး |
| 237 | 213313 | မိုးဝါး |
| 238 | 213314 | မိုးဝါး |
| 239 | 213315 | မိုးဝါး |
| 240 | 213316 | မိုးဝါး |
| 241 | 213317 | မိုးဝါး |
| 242 | 213318 | မိုးဝါး |
| 243 | 213320 | မိုးဝါး |
| 244 | 213321 | မိုးဝါး |
| 245 | 213322 | မိုးဝါး |
| 246 | 215312 | မိုးဝါး |
| 247 | 215313 | မိုးဝါး |

|     |        |      |
|-----|--------|------|
| 248 | 215314 | ಜೊ   |
| 249 | 215315 | ಜು   |
| 250 | 215316 | ಜ್ಞಾ |
| 251 | 215317 | ಜ್ಞಾ |
| 252 | 215318 | ಜ್ಞಾ |
| 253 | 215320 | ಜ್ಞಾ |
| 254 | 215321 | ಜ್ಞಾ |
| 255 | 215322 | ಜ್ಞಾ |
| 256 | 216311 | ಜ್ಞಾ |
| 257 | 216312 | ಜ್ಞಾ |
| 258 | 216313 | ಜ್ಞಾ |
| 259 | 216314 | ಜ್ಞಾ |
| 260 | 216317 | ಜ್ಞಾ |
| 261 | 216318 | ಜ್ಞಾ |
| 262 | 216320 | ಜ್ಞಾ |
| 263 | 216321 | ಜ್ಞಾ |
| 264 | 216322 | ಜ್ಞಾ |
| 265 | 218312 | ಜ್ಞಾ |
| 266 | 218313 | ಜ್ಞಾ |
| 267 | 218314 | ಜ್ಞಾ |
| 268 | 218315 | ಜ್ಞಾ |
| 269 | 218316 | ಜ್ಞಾ |
| 270 | 218317 | ಜ್ಞಾ |
| 271 | 218318 | ಜ್ಞಾ |
| 272 | 218320 | ಜ್ಞಾ |

|     |        |      |
|-----|--------|------|
| 273 | 218321 | ಜ್ಞಾ |
| 274 | 218322 | ಜ್ಞಾ |
| 275 | 219311 | ಜ್ಞಾ |
| 276 | 219313 | ಜ್ಞಾ |
| 277 | 219314 | ಜ್ಞಾ |
| 278 | 219317 | ಜ್ಞಾ |
| 279 | 219318 | ಜ್ಞಾ |
| 280 | 219320 | ಜ್ಞಾ |
| 281 | 219321 | ಜ್ಞಾ |
| 282 | 219322 | ಜ್ಞಾ |
| 283 | 220312 | ಜ್ಞಾ |
| 284 | 220313 | ಜ್ಞಾ |
| 285 | 220314 | ಜ್ಞಾ |
| 286 | 220317 | ಜ್ಞಾ |
| 287 | 220318 | ಜ್ಞಾ |
| 288 | 220320 | ಜ್ಞಾ |
| 289 | 220321 | ಜ್ಞಾ |
| 290 | 220322 | ಜ್ಞಾ |
| 291 | 221312 | ಜ್ಞಾ |
| 292 | 221313 | ಜ್ಞಾ |
| 293 | 221314 | ಜ್ಞಾ |
| 294 | 221315 | ಜ್ಞಾ |
| 295 | 221316 | ಜ್ಞಾ |
| 296 | 221317 | ಜ್ಞಾ |
| 297 | 221318 | ಜ್ಞಾ |
| 298 | 221320 | ಜ್ಞಾ |
| 299 | 221321 | ಜ್ಞಾ |

|     |        |       |
|-----|--------|-------|
| 300 | 221322 | ဗျူဟာ |
| 301 | 223311 | အောက် |
| 302 | 223312 | အောက် |
| 303 | 223313 | အောက် |
| 304 | 223314 | အောက် |
| 305 | 223317 | အောက် |
| 306 | 223318 | အောက် |
| 307 | 223320 | အောက် |
| 308 | 223321 | အောက် |
| 309 | 223322 | အောက် |
| 310 | 225312 | အောက် |
| 311 | 225313 | အောက် |
| 312 | 225314 | အောက် |
| 313 | 225315 | အောက် |
| 314 | 225316 | အောက် |
| 315 | 225317 | အောက် |
| 316 | 225318 | အောက် |
| 317 | 225320 | အောက် |
| 318 | 225321 | အောက် |
| 319 | 225322 | အောက် |
| 320 | 226311 | အောက် |
| 321 | 226312 | အောက် |
| 322 | 226313 | အောက် |

|     |        |       |
|-----|--------|-------|
| 323 | 226314 | အောက် |
| 324 | 226317 | အောက် |
| 325 | 226318 | အောက် |
| 326 | 226320 | အောက် |
| 327 | 226321 | အောက် |
| 328 | 226322 | အောက် |
| 329 | 228311 | အောက် |
| 330 | 228312 | အောက် |
| 331 | 228313 | အောက် |
| 332 | 228314 | အောက် |
| 333 | 228317 | အောက် |
| 334 | 228318 | အောက် |
| 335 | 228320 | အောက် |
| 336 | 228321 | အောက် |
| 337 | 228322 | အောက် |
| 338 | 230311 | အောက် |
| 339 | 230312 | အောက် |
| 340 | 230313 | အောက် |
| 341 | 230314 | အောက် |
| 342 | 230317 | အောက် |
| 343 | 230318 | အောက် |
| 344 | 230320 | အောက် |

|     |        |    |
|-----|--------|----|
| 345 | 230321 | నీ |
| 346 | 230322 | నో |
| 347 | 231312 | లం |
| 348 | 231313 | లం |
| 349 | 231314 | లం |
| 350 | 231315 | లు |
| 351 | 231316 | లు |
| 352 | 231322 | లం |
| 353 | 233312 | లం |
| 354 | 233313 | లం |
| 355 | 233314 | లం |
| 356 | 233315 | లం |
| 357 | 233316 | లం |
| 358 | 233317 | లం |
| 359 | 233318 | లం |
| 360 | 233320 | లం |
| 361 | 233321 | లం |
| 362 | 233322 | లం |
| 363 | 234311 | లం |
| 364 | 234312 | లం |
| 365 | 234313 | లం |

|     |        |    |
|-----|--------|----|
| 366 | 234314 | లం |
| 367 | 234317 | లం |
| 368 | 234318 | లం |
| 369 | 234320 | లం |
| 370 | 234321 | లం |
| 371 | 234322 | లం |
| 372 | 235311 | లం |
| 373 | 235313 | లం |
| 374 | 235314 | లం |
| 375 | 235317 | లం |
| 376 | 235318 | లం |
| 377 | 235320 | లం |
| 378 | 235321 | లం |
| 379 | 236311 | లం |
| 380 | 236314 | లం |
| 381 | 236317 | లం |
| 382 | 236318 | లం |
| 383 | 236320 | లం |
| 384 | 236321 | లం |
| 385 | 236322 | లం |
| 386 | 237311 | లం |
| 387 | 237312 | లం |
| 388 | 237313 | లం |

|     |        |        |
|-----|--------|--------|
| 389 | 237314 | 237314 |
| 390 | 237317 | 237317 |
| 391 | 237318 | 237318 |
| 392 | 237320 | 237320 |
| 393 | 237321 | 237321 |
| 394 | 237322 | 237322 |
| 395 | 238312 | 238312 |
| 396 | 238313 | 238313 |
| 397 | 238314 | 238314 |
| 398 | 238315 | 238315 |
| 399 | 238316 | 238316 |
| 400 | 238317 | 238317 |
| 401 | 238318 | 238318 |
| 402 | 238320 | 238320 |
| 403 | 238321 | 238321 |
| 404 | 238322 | 238322 |
| 405 | 239311 | 239311 |
| 406 | 239312 | 239312 |
| 407 | 239313 | 239313 |
| 408 | 239314 | 239314 |
| 409 | 239317 | 239317 |
| 410 | 239318 | 239318 |
| 411 | 239320 | 239320 |

|     |        |        |
|-----|--------|--------|
| 412 | 239321 | 239321 |
| 413 | 239322 | 239322 |
| 414 | 240311 | 240311 |
| 415 | 240312 | 240312 |
| 416 | 240313 | 240313 |
| 417 | 240314 | 240314 |
| 418 | 240317 | 240317 |
| 419 | 240318 | 240318 |
| 420 | 240320 | 240320 |
| 421 | 240321 | 240321 |
| 422 | 240322 | 240322 |
| 423 | 241312 | 241312 |
| 424 | 241313 | 241313 |
| 425 | 241314 | 241314 |
| 426 | 241322 | 241322 |
| 427 | 242312 | 242312 |
| 428 | 242313 | 242313 |
| 429 | 242314 | 242314 |
| 430 | 242322 | 242322 |
| 431 | 244312 | 244312 |
| 432 | 244313 | 244313 |
| 433 | 244314 | 244314 |
| 434 | 244315 | 244315 |
| 435 | 244316 | 244316 |
| 436 | 244317 | 244317 |

|     |           |    |
|-----|-----------|----|
| 437 | 244318    | తూ |
| 438 | 244320    | దూ |
| 439 | 244321    | దూ |
| 440 | 244322    | దూ |
| 441 | 245311    | దూ |
| 442 | 245312    | దూ |
| 443 | 245313    | దూ |
| 444 | 245314    | దూ |
| 445 | 245317    | దూ |
| 446 | 245318    | దూ |
| 447 | 245320    | దూ |
| 448 | 245321    | దూ |
| 449 | 245322    | దూ |
| 450 | 211311315 | దూ |
| 451 | 211311316 | దూ |
| 452 | 216311315 | దూ |
| 453 | 216311316 | దూ |
| 454 | 219311312 | దూ |
| 455 | 219311315 | దూ |
| 456 | 219311316 | దూ |
| 457 | 223311315 | దూ |
| 458 | 223311316 | దూ |
| 459 | 226311315 | దూ |

|     |           |    |
|-----|-----------|----|
| 460 | 226311316 | దూ |
| 461 | 228311315 | దూ |
| 462 | 228311316 | దూ |
| 463 | 230311315 | దూ |
| 464 | 230311316 | దూ |
| 465 | 234311315 | దూ |
| 466 | 234311316 | దూ |
| 467 | 235311312 | దూ |
| 468 | 235311315 | దూ |
| 469 | 235311316 | దూ |
| 470 | 235311322 | దూ |
| 471 | 236311312 | దూ |
| 472 | 236311315 | దూ |
| 473 | 236311316 | దూ |
| 474 | 237311315 | దూ |
| 475 | 237311316 | దూ |
| 476 | 239311315 | దూ |
| 477 | 239311316 | దూ |
| 478 | 240311315 | దూ |
| 479 | 240311316 | దూ |
| 480 | 245311315 | దూ |
| 481 | 245311316 | దూ |



## APPENDIX – C

### TELUGU FONT NAMES AND FONT CODES

| S.No | Font name                  | Font code | Example Character |
|------|----------------------------|-----------|-------------------|
| 1    | Amma                       | 1         | అ                 |
| 2    | Gowthami                   | 2         | అ                 |
| 3    | Hemalatha                  | 3         | అ                 |
| 4    | Srilipi-G series           | 4         | అ                 |
| 5    | Vartha                     | 5         | అ                 |
| 6    | Vennela                    | 6         | అ                 |
| 7    | Srilipi-9 series (DRISHTI) | 7         | అ                 |
| 8    | Hemalatha (DRISHTI)        | 8         | అ                 |

.....



## References

1. [Abuhaiba, 2005] Ibrahim S. I. Abuhaiba, “Arabic Font Recognition using Decision Trees Built from Common Words”, *Journal of Computing and Information Technology - CIT 13*, pp. 211–223, 2005.
2. [Allier, & Emptoz, 2003] Bénédicte Allier, Hubert Emptoz, “Font Type Extraction and Character Prototyping Using Gabor Filters” *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR’03)* pp.799-803, 2003.
3. [Anbumani & Subramanian, 2000] Anbumani and Subramanian, “Optical Character Recognition of Printed Tamil Characters” Department of Electrical and computer Engineering, Virginia Tech, Blacksburg, December 2000.
4. [Amin & Fischer, 2000] A.Amin and S. Fischer “A Document SkewDetection Method Using the Hough Transform”, *Pattern Analysis and Applications*, Springer-Verlag London Limited, 3, pp 243-253, 2000.
5. [Antani & Agnihotri, 1999] Antani Sameer, Lalitha Agnihotri, “Gujarati Character Recognition”, *Fifth International Conference on Document Analysis and Recognition (ICDAR’99)*, pp. 418, 1999.
6. [Aparna & Chakravarthy, 2002] Aparna K G, V.S.Chakravarthy, 2002 “A complete Tamil Optical Character Recognition System”, *Tamil Internet*, Chennai, pp. 45-51, 2003.
7. [Aparna & Ramakrishnan, 2001] Aparna K G ,Ramakrishnan A G, “Tamil Gnani - an OCR on Windows”, *Proc. Tamil Internet 2001*, Kuala Lumpur, pp. 60-63, 2001.
8. [Ashwin & Sastry, 2002] Ashwin T. V. And P. S. Sastry 2002 “A font and size-independent OCR system for printed Kannada documents using support vector machines”, *Saadhanaa*, Vol. 27, Part 1, pp. 35–58, February 2002.
9. [Bansal, 1999] Veena Bansal, “Integrating knowledge sources in Devnagari text recognition”, Ph.D. Thesis, IIT Kanpur, March, 1999.
10. [Bayes , 1763] Thomas Bayes. “An essay towards solving a problem in the doctrine of chances” *Philosophical Transactions of the Royal Society*, 53:370–418, 1763.

11. [Bazzi et al.,1997] I.Bazzi,C. Lapre, R. Schwartz, and J. Makhoul, "Omnifont and unlimited vocabulary OCR system for English and Arabic" *Proceedings of international Conference on Document Analysis and Recognition*, Ulm, Germany, Vol. 2, pp.842-846, 1997.
12. [Belaid, 1997 ] Abdel Belaid, "OCR Print - An overview" chapter 2 in "Survey of the state of the art in human language technology" 1997, pp: 71 – 74.  
[http://www.loria.fr/~abelaid/publi\\_ps/ocr\\_art.ps](http://www.loria.fr/~abelaid/publi_ps/ocr_art.ps)
13. [Bellman, 1957] R. Bellman, "Dynamic Programming", Princeton, N.J.: Princeton University Press.
14. [Bhagvati et. al., 2003] Chakravarthy Bhagvati, T.Ravi, S.M.Kumar, and Atul Negi. "On Developing High Accuracy OCR Systems for Telugu and other Indian Scripts", *Proc. of Language Engineering Conference*, Pp 18-23, Hyderabad, IEEE computer society Press, 2003.
15. [Brieman et al., 1984] L. Brieman, J.H.Friedman, R.A.Olshen, and C.J. Stone, "Classification and regression trees", Wadsworth, California, 1984.
16. [Brown, 1994] Brown R.L. "The fringe distance measure: an easily calculated image distance measure with recognition results comparable to Gaussian blurring", *IEEE Trans.System Man and Cybernetics*, 24(1):111-116, 1994.
17. [Bunke & Wang, 1997] H.Bunke and P.S.P.Wang (Eds) "Handbook of character recognition and document image analysis", pp35, World scientific, 1997.
18. [Busse,1995] J.W. Grzymala-Busse: "Rough Sets" *Advances in Imaging and Electron Physics* 94, pp.151–195, 1995.
19. [Chaudhuri & Pal, 1997] B.B. Chaudhuri and U. Pal "Skew Angle Detection of Digitized Indian Script Documents" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, VOL. 19, NO. 2, February 1997.
20. [Chaudhuri, & Pal, 1998] Chaudhuri, B.B. and Pal, U. 1998 "A complete printed Bangla OCR system", *Pattern Recognition*, Vol. 31, pp. 531-549, 1998.
21. [Chaudhuri, Pal, & Mitra, 2002] B B Chaudhuri, U Pal and M Mitra, "Automatic recognition of printed Oriya script", *Saadhnnaa*, Vol. 27, Part 1, pp. 23–34, February 2002.

22. [Chen & Haralick, 1994] S. Chen and R. M. Haralick, "An automatic algorithm for text skew estimation in document images using recursive morphological transforms," *ICIP- 94*, Austin, Texas, November, pp.139-143, 1994.
23. [Chernoff & Moses , 1959] Herman Chernoff and Lincoln E. Moses. "Elementary Decision Theory" John Wiley, New York, NY, 1959.
24. [Chinnuswamy & Krishnamoorthy, 1980] Chinnuswamy P., S.G. Krishnamoorthy, 1980 "Recognition of hand-printed Tamil characters" *Pattern Recognition*, Vol 12, issue 3, 141–152.
25. [Chow, 1957] Chao K. Chow. "An optimum character recognition system using decision functions" *IRE Transactions*, pages 247–254, 1957.
26. [Dash & Liu, 1997] Dash, M., & Liu, H., "Feature Selection for classification" *Intelligent Data Analysis*, Vol. 1, No. 3, pp. 131-156, 1997.
27. [Duda & Hart, 1982] Duda & Hart, 1982 "Pattern Classification and Scene Analysis". J. Wiley & Sons, New York, 1982.
28. [Duda, Hart & Stork ,2000], Richard O Duda, Peter E Hart, David G Stork "Pattern Classification" , - 2Ed - *Wiley-Interscience*, 2000.
29. [Farlex, 2003] Farlex, "The Free Dictionary by Farlex", from the *American Heritage® Science Dictionary* by Houghton Mifflin Company, 2003.  
<http://www.thefreedictionary.com/cognition>
30. [Fisher, 1936] Ronald A. Fisher. "The use of multiple measurements in taxonomic problems" *Annals of Eugenics*, 7 Part II: pp. 179–188, 1936.
31. [Flusser & Suk , 1994] Flusser and Suk "Affine moment invariants: A new tool for character recognition" , *Pattern Recognition Letters* vol 15, pp433-436, 1994.
32. [Fodor, 2002] Fodor I. K., "A survey of dimension reduction techniques," LLNL technical report, UCRL-ID-148494, Lawrence Livermore National Laboratory Technical Information Department's Digital Library, June 2002.
33. [Fu & Rosenfeld, 1976] King-Sun Fu, Azriel Rosenfeld, "Pattern Recognition and Image Processing", *IEEE Transactions On Computers*, Vol. C-25, NO. 12, pp.1336-1346, December 1976.

34. [Fukunaga, 1990] Keinosuke Fukunaga. "Introduction to Statistical Pattern Recognition", Academic Press, New York, NY, 2nd edition, 1990.
35. [Ganesan & Rao, 2006] G.Ganesan, and C. Raghavendra Rao "Feature Selection using Fuzzy Decision Attributes", *International Journal of Information*. Vol.9, No.3, May, pp. 381-384, 2006.
36. [Garain & Chaudhuri, 2002] Garain U. and B. B. Chaudhuri, "Segmentation of Touching Characters in Printed Devnagari and Bangla Scripts using Fuzzy Multifactorial Analysis", *IEEE Transactions on Systems, Man and Cybernetics*, Part C, Vol. 32, No. 4, pp. 449-459, 2002.
37. [Gonzalez & Woods, 2002] R.C. Gonzalez, and R.E. Woods, "Digital Image Processing", Prentice-Hall, New Jersey, pp.541, 2002.
38. [Hashizume et al., 1986] "A method of detecting the orientation of aligned components" *Pattern Recognition Letters* 4, pp. 125-132, 1986.
39. [Hilditch, C.J., 1969] Hilditch, C.J., "Linear skeletons from square cupboards" Meltzer, B., Mitchie, D. (Eds.), *Machine Intelligence*, Vol. 4. Edinburgh University Press, pp. 404-420, 1969.
40. [Hu, 1962], M.K. Hu. "Visual pattern recognition by moment invariants". *IRE Tras. Inform. Theory* IT, 8(2), pp179-187, 1962.
41. [Hull, 1998] Jonathan J. Hull "Document Image Skew Detection: Survey And Annotated Bibliography" *Document Analysis Systems II*, J..J.Hull, S.L.Taylor, Eds., World Scientific, 1998, pp. 40-64.
42. [Iyer et al., 2005] Parvati Iyer, Abhipsita Singh, S.Sanyal "Optical Character Recognition System for Noisy Images in Devnagari Script", *UDL Workshop on Optical Character Recognition with Workflow and Document Summarization* , 2005.
43. [Jain, Duin, & Mao,2000] A. Jain, P. Duin, and J. Mao, "Statistical pattern recognition: A review," *IEEE Transactions on PAMI*, Vol 22(1), pp. 4-37, 2000.
44. [Jawahar et al., 2003] Jawahar C. V., M. N. S. S. K. Pavan Kumar, S. S. Ravi Kiran "A Bilingual OCR for Hindi-Telugu Documents and its Applications" *Proceedings of the 7<sup>th</sup> International Conference on Document Analysis and Recognition*, 2003.

45. [Jensen & Shen, 2007] R. Jensen, Q. Shen (2007), "Rough set based feature selection: A review," *Rough Computing: Theories, Technologies and applications*, Idea Group Inc., Global, pp.70-107, 2007.
46. [JLT, April 2000] Journal of Language Technology, *Vishwabharat@tdil*, July2000.
47. [JLT, July 2003] Journal of Language Technology, *Vishwabharat@tdil*, July2003.
48. [JLT, October 2003] Journal of Language Technology, *Vishwabharat@tdil*, pp.89-99, October, 2003.
49. [JLT, July 2004]Journal of Language Technology, *Vishwabharat@tdil*. July 2004.  
<http://www.tdil.mit.gov.in/July05/extract/indian%20language%20technology%20testing%20reports.pdf>
50. [Jung ,Shin & Srihari,1999] Min-Chul Jung, Yong-Chul Shin and Sargur N. Srihari "Multifont Classification using Typographical Attributes" *Proceedings of the Fifth International Conference on Document Analysis and Recognition, ICDAR*, pp. 353-356, 1999.
51. [Kasturi et al, 2002] Rangachar Kasturi, Lawrence O’Gorman and Venu Govindaraju, "Document image analysis: A primer", *Saadhanaa* Vol. 27, Part 1, pp. 3–22, February 2002.
52. [Khoubyari & Hull, 1996] Siamak Khoubyari and Jonathan j. Hull, 'Font and Function Word Identification' In *Computer Vision and Image Understanding*,Vol. 63, no. 1, pp. 66–74, January 1996.
53. [Khorsheed & Clocksin, 2000] Mohammad S Khorsheed and William F Clocksin, "Multi-Font Arabic Word Recognition Using Spectral Features", 15th *International Conference on Pattern Recognition (ICPR'00)* - Volume 4, pp. 4543, 2000.
54. [Kirkby & Frank, 2006] Richard Kirkby , Eibe Frank, "WEKA Explorer User Guide for Version 3-5-3", University of Waikato, 2006. [www.cs.waikato.ac.nz/ml/weka](http://www.cs.waikato.ac.nz/ml/weka)
55. [Komorowski , Polkowski & Skowron,1999] Jan Komorowski, Lech Polkowski , and Andrzej Skowron , "Rough Sets: A Tutorial",1999.  
<http://folli.loria.fr/cds/1999/library/pdf/skowron.pdf>
56. [Lakshmi & Patvardhan, 2002] Lakshmi C V, C Patvardhan, "A multi-font OCR system for printed Telugu text", *Proc. Of Language engineering conference LEC*, Hyderabad.pgs.7-17, 2002.



57. [Lakshmi & Patvardhan, 2003] Lakshmi C V , C Patvardhan, "Optical Character Recognition of Basic Symbols in Printed Telugu Text", *IE(I)Journal-CP* Vol 84, pgs.66-71, 2003.
58. [Leedham,et al., 2002] Graham Leedham, Saket Varma, Anish Patankar and Venu Govindaraju, "Separating text and background in degraded document Images – a comparison of global thresholding techniques For multi-stage thresholding" *Proceedings of the Eighth International Workshop on Frontiers in Handwriting Recognition* , 2002.
59. [Lehal & Singh, 2000] Lehal G S and Chandan Singh "A Gurmukhi Script Recognition System", *15th International Conference on Pattern Recognition (ICPR'00)* - Volume 2 p. 2557, 2000.
60. [Lehal & Singh, 2002] Lehal G S and Chandan singh, "A post-processor for Gurmukhi OCR", *Saadhana* Vol. 27, Part 1, pp. 99–111, February, 2002.
61. [Line, 1993] Eikvil Line, "OCR-Optical character recognition system" , Report No. 876, *Document Image Analysis Publications*, Norwegian computing center., 1993, [http://www.nr.no/documents/samba/research\\_areas/BAMG/Publications/OCR.ps.gz](http://www.nr.no/documents/samba/research_areas/BAMG/Publications/OCR.ps.gz)
62. [Liu & Fujisawa, 2005] Cheng-Lin Liu, Hiromichi Fujisawa "Classification and Learning for Character Recognition: Comparison of Methods and Remaining Problems" *Proc. of Neural Networks and Learning in Document Analysis and Recognition*, Seoul, Korea, 2005.
63. [Liu et al.,2004] Cheng-Lin Liu, Kazuki Nakashima, Hiroshi Sako, Hiromichi Fujisawa, "Handwritten digit recognition: investigation of normalization and feature extraction techniques" , *Pattern Recognition* 37 , pp.265 – 279, 2004.
64. [Lu &Tan, 2003] Yue Lu, Chew Lim, Tan , "A nearest-neighbor chain based approach to skew estimation in document images", *Pattern Recognition Letters* 24, pp. 2315–2323, 2003.
65. [Mahmud, et al., 2003] Jalal Uddin Mahmud, Mohammed Feroz Raihan and Chowdhury Mofizur Rahman , "A Complete OCR System for Continuous Bengali Characters" , *IEEE*, 2003.

66. [Mantas, 1986] J. Mantas, "An Overview of Character Recognition Methodologies" *Pattern Recognition*, Vol. 19, No 6, pp. 425-430, 1986.
67. [Mohanty & Behera, 2004] Mohanty S., H.K. Behera, "A Complete OCR Development System for Oriya Script" *Proceeding of symposium on Indian Morphology, phonology and Language Engineering*, IIT Kharagpur, 2004.
68. [Mori, Nishida & Yamada, 1999] S. Mori, H. Nishida, H. Yamada, "Optical Character Recognition", John Wiley & Sons, 1999.
69. [Nagy, Nartker & Rice, 2000] George Nagy, Thomas A. Nartker, Stephen V. Rice, "Optical Character Recognition: An illustrated guide to the frontier", *Proceedings of Document Recognition and Retrieval VII*, SPIE Vol. 3967, Sanjose, pp. 58-69, January 2000.
70. [Nagy, Seth & Vishwanathan, 1992] Nagy G, S. Seth, and M. Vishwanathan "A prototype document image analysis system for technical journals". *Computer*, Vol 25, No. 7, 1992.
71. [Nagy & Seth, 1984] G. Nagy and S. Seth. "Hierarchical representation of optically scanned documents" *Proceedings of 7th ICPR*, pp.347-349, 1984.
72. [Negi , Bhagvati & Krishna, 2001] Atul Negi, Chakravarthy Bhagvati and Krishna B, "An OCR system for Telugu" *Proc. Of 6th Int. Conf. on Document Analysis and Recognition*, IEEE Comp. Soc. Press, USA,. Pp. 1110-1114, 2001.
73. [Niblack , 1986]W. Niblack, "An Introduction to Digital Image Processing", pp. 115-116, Prentice Hall, 1986.
74. [Øhrn, 1999] Øhrn, A., "Discernibility and Rough Sets in Medicine: Tools and Applications" Department of Computer and Information Science. Trondheim, Norway, Norwegian University of Science and Technology. pp 239, 1999.
75. [Ohrn & Rowland, 2000] Ohrn and T. Rowland, "Rough Sets: A Knowledge Discovery Technique for Multifactorial Medical Outcomes" *American Journal of Physical Medicine and Rehabilitation*, Vol 79, no. 1, pp. 100-108, Jan/Feb 2000.
76. [Otsu, 1979] N. Otsu. "A threshold selection method from gray-level histograms". *IEEE Transactions. Systems., Man., Cyber.* vol. 9: 62–66, 1979.



77. [Pal & Chaudhuri, 1996] Pal, U., Chaudhuri, B.B., “An improved document skew angle estimation technique” *Pattern Recognition Letters* 17, pp. 899–904, 1996.
78. [Pal & Chaudhuri, 1997] Pal U. , B. B. Chaudhuri, “Printed Devnagari Script OCR System” *Vivek*, vol.10, pgs.12-24, 1997.
79. [Pal & Chaudhuri, 2004] U. Pal, B.B. Chaudhuri “Indian script character recognition: a survey” *Pattern Recognition* 37, 1887 – 1899, 2004.
80. [Pavlidis, 1977] T. Pavlidis. “Structural Pattern Recognition” Springer Verlag, Berlin Heidelberg New York, 1977.
81. [Pavlidis, 2000] Theo Pavlidis “36years on the Pattern Recognition Front”, ICPR 2000, Barcelona, Spain.
82. [Pawlak, 1982] Zdzislaw Pawlak. “Rough sets.” *International Journal of Information and Computer Science*, 11(5), pp341-356, 1982.
83. [Platt , 1998] John C. Platt, “Sequential Minimal Optimization: A Fast Algorithm for Training Support Vector Machines”, Microsoft Research Technical Report MSR-TR-98-14, April 21, 1998.
84. [Postl ,1986] W. Postl, “Detection of linear oblique structures and skew scan in digitized documents,” *Proceedings of the 8th International Conference on Pattern Recognition, Paris, France*, pp.687-689, October 1986.
85. [Pujari, Naidu & Jinaga 2002] Pujari Arun K , C. Dhanunjaya Naidu & B. C. Jinaga 2002 “An Adaptive Character Recognizer for Telugu Scripts using Multiresolution Analysis and Associative Memory” *Proceedings Of ICVGIP*, Ahmedabad, 2002.
86. [Rajasekaran & Deekshatulu, 1977] Rajasekaran S.N.S. Deekshatulu B.L. “Recognition of printed Telugu characters”, *Computer Graphics Image Processing*, Vol 6, pgs.335–360, 1977.
87. [Ramadevi, Rao & Reddy, 2007] Y. Ramadevi, C. R. Rao, Vivekchan Reddy, “Decision Tree Induction Using Rough Set Theory- Comparative Study”, *Journal of Theoretical and Applied Information Technology*, Vol.3, 2007.

88. [Raudys, 1998] S. Raudys, ‘Evolution and generalization of single neuron; single layer perceptron as seven statistical classifiers’, *Neural networks*, vol. 11, n0.2, pp.283-296, 1998.
89. [Rao & Ajitha, 1995] Rao P. V. S. and T. M. Ajitha “Telugu Script Recognition - a Feature Based Approach”, *Proce.of ICDAR, IEEE* ,pp.323-326, 1995.
90. [Rawat, et al, 2006] S.Rawat, K.S.Kumar, M.Meshesha, A.Balasubramanian, I.D. Sikdar, and C.V. Jawahar, “A Semi-Automatic Adaptive OCR for Digital Libraries”, *Proc. of 7th IAPR Workshop on Document Analysis Systems*, H. Bunke and A.L. Spitz (Eds.): *LNCS 3872*, pp. 13–24, 2006.
91. [Ray & Chatterjee, 1984] K. Ray, B. Chatterjee, “Design of a nearest neighbor classifier system for Bengali character recognition”, *Journal of Inst. Electronics and Telecom. Eng.* 30, 226–229, 1984.
92. [Roberts, 2005] Andrew Roberts “AI32 — Guide to Weka”, 2005.  
<http://www.comp.leeds.ac.uk/andyr>
93. [Rosenfeld & Kak, 1976] A. Rosenfeld and A.C. Kak, “Digital picture processing”, Academic press, 1976.
94. [Rubinstein , 1988] R. Rubinstein, “Digital Typography: An Introduction to Type and Composition for Computer System Design” Addison-Wesley, 1988.
95. [Sauvola & PietikaKinen, 2000] J. Sauvola, M. PietikaKinen, “Adaptive document image Binarization” *Pattern Recognition* Vo.33 , pp.225-236, 2000.
96. [Schalkoff, 1992] RJ Schalkoff “Pattern Recognition: Statistical, Structural and Neural Approaches”, John Wiley & Sons, pp.2, 1992.
97. [Shen & Chouchoulas, 2001] Qiang Shen and Alexios Chouchoulas. “Rough set-based dimensionality reduction for supervised and unsupervised learning”, *Int. Journal of Applied Mathematics and Computer Science*, Special Issue on Rough Sets and their Applications, Volume 11(3):101–119, 2001.
98. [Shi, & Pavlidis, 1997] Hongwei Shi, Theo Pavlidis, “Font Recognition and Contextual Processing for More Accurate Text Recognition”, 4th *International Conference on Document Analysis and Recognition (ICDAR '97)*, *IEEE Computer Society* , pages. 39-44, Ulm, Germany, August 1997 .

99. [Shivakumara et al., 2003] P.Shivakumara, G. Hemantha Kumar, D. S Guru, P. Nagabhushan, "Skew Estimation of Binary Document Images Using Static and Dynamic Thresholds Useful for Document Image Mosaicing" *Proceedings of National Workshop on IT Services and Applications (WITSA2003)* Feb 27-28, 2003.
100. [Sinha & Mahabala , 1979] Sinha R.K, Mahabala "Machine recognition of Devnagari script" *IEEE Trans. Systems Man Cybern.* pp. 435–441, 1979.
101. [Siromony, Chandrasekaran & Chandrasekaran, 1978] Siromony, G. Chandrasekaran R., Chandrasekaran M. "Computer recognition of printed Tamil characters" *Pattern Recognition* vol.10, pp.243–247, 1978.
102. [Srihari, & Govindraju, 1989] Srihari, S.N., Govindraju, V., "Analysis of textual image using the Hough transform", *Machine Vision Applications*, Vol.2, pp.141–153, 1989.
103. [Sukhaswami, Seetharamulu., & Pujari,1995] Sukhaswami R, Seetharamulu P., Pujari A.K. "Recognition of Telugu characters using Neural networks", *Int. Journal of Neural Systems*, Vol.6 pp..317–357, 1995.
104. [Swiniarski , 2001] Roman W. Swiniarski,"Rough sets methods in feature reduction and classification" *Int. Journal of Applied. Maths and Computer Science*, Vol.11, No.3, pp 565-582, 2001.
105. [Tohka, 2006] Jussi Tohka, "Introduction to Pattern Recognition", Institute of Signal Processing, Tampere University of Technology, [www.cs.tut.fi/sgn/m2obsi/courses/IPR/Lectures/IPR\\_Lecture\\_1.pdf](http://www.cs.tut.fi/sgn/m2obsi/courses/IPR/Lectures/IPR_Lecture_1.pdf)
106. [Toussaint, 1997], Godfried Toussaint, "Introduction To Pattern Recognition", *Pattern recognition on the web*, chapter 1. <http://www-cgri.cs.mcgill.ca/%7Egodfried/teaching/pr-notes/introduction.ps>
107. [Trier, Jain, & Taxtt ,1996] O.D.Trier, Anil.K.Jain,Torfinn Taxtt "Feature extraction methods for character recognition- A survey" , *Pattern Recognition*, vol. 29, no.4, pp.641-662,1996.
108. [Vapnik , 1995] Vladimir Vapnik. "The Nature of Statistical Learning Theory" Springer-Verlag,New York, NY, 1995.

109. [Venkat Rao et al, 2006] N.Venkat Rao, I.Santi Prabha, L. Pratap Reddy, "A study on corpus based analysis of characters in the present Telugu script", *Proceedings of ICSCI*, Hyderabad, pp.761-765, 2006.
110. [Wajahat, 2008] Mohammed Wajahat Ali Siddiqui, "EM-OCR: An OCR System using EMAlgorithm", Technical report, Department of CIS, University of Hyderabad, 2008.
111. [Wu & Amin, 2003] Sue Wu, Adnan Amin, "Automatic Thresholding of Gray-level Using Multi-stage Approach" *Proceedings of the Seventh International Conference on Document Analysis and Recognition*, 2003.
112. [Wikipedia, 2009] [http://en.wikipedia.org/wiki/Support\\_vector\\_machine](http://en.wikipedia.org/wiki/Support_vector_machine)
113. [Witten & Frank, 2000] Ian H. Witten, and Eibe Frank, "WEKA- Machine Learning Algorithms in Java", Chapter 8 of "Data Mining: Practical Machine Learning Tools and Techniques with Java Implementations", *Morgan Kaufmann Publishers*, 2000.
114. [Wong, Casey, &Wahl, 1982 ] Wong K., Casey R., and Wahl F. "Document analysis system", *IBM Journal of Research and Development*, Vol 26, No.6, 1982.
115. [Yan, 1993] Yan, H. "Skew correction of document images using interline cross-correlation", *CVGIP: Graphical Models and Image Processing*, Vol 55 No. 6, pp.538-543, 1993.
116. [Zadeh, 1965] L. Zadeh, "Fuzzy sets", *Information and Control*, 8, pp.338-353, 1965.
117. [Zhu, Tan & Wang, 2001] Yong Zhu, Tieniu Tan, and Yunhong Wang, "Font Recognition Based on Global Texture Analysis" *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 23, no. 10, 2001.
118. [Zramdini & Ingold., 1993] Abdelwahab Zramdini and Rolf Ingold. "Optical Font Recognition from projection profiles". *Electronic publishing*, vol. 6(3), pp. 249-260, September 1993.
119. [Zramdini & Ingold, 1998] Abdelwahab Zramdini and Rolf Ingold "Optical Font Recognition Using Typographical Features", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 20, No. 8, 1998.