Rough Set Based Analysis of Document Images

Ushasi Chaudhuri
Roll No: 15AT71P01

Supervisors
Prof. Partha Bhowmick
&
Prof. Jayanta Mukhopadhyay

A thesis submitted for the degree of
Master of Science (by Research)

Advanced Technology and Development Center
Indian Institute of Technology, Kharagpur
India

December 2017

Dedicated to my 'Dida'.

Declaration

I certify that

a. the work contained in this thesis is original and has been done by me under the guidance of my supervisors.

b. the work has not been submitted to any other Institute for any degree or diploma.

c. I have followed the guidelines provided by the Institute in preparing the thesis.

d. I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Ushasi Chaudhuri
Roll No.: 15AT71P01

Certificate

This is to certify that the thesis entitled ***Rough Set Based Analysis of Document Images*** submitted by **Ushasi Chaudhuri** to Indian Institute of Technology, Kharagpur, is a record of bonafide research work carried under our supervision and is worthy of consideration for the award of the degree of Master of Science (by Research) of the Institute.

Prof. Partha Bhowmick
(Supervisor)
Computer Science and Engineering Department
Indian Institute of Technology
Kharagpur 721302, INDIA

Prof. Jayanta Mukhopadhyay
(Co-Supervisor)
Computer Science and Engineering Department
Indian Institute of Technology
Kharagpur 721302, INDIA

Acknowledgements

First and foremost I thank the Almighty who kindly helped me to complete my thesis.

I am forever indebted to Prof. Partha Bhowmick and Prof. Jayanta Mukhopadhyay for their guidance, encouragement, help and support throughout during my studies here. They taught me how to read a technical paper, how to write one, and how to think about a research problem.

I am extremely grateful to all faculty members and my DSC members from the Departments of Computer Science and Engineering and Electronics and Electrical Communication Engineering, IIT Kharagpur.

I wish to thank my labmates Dipannita, Abantika-di, Shyantani-di, Barnalli-di, Anupam-da, Sai Ram Bhaiyya, Bijju Bhaiyya, Shashaank Bhaiyya, Jit da, Ankita, Piyush da, Jayashree di, Romil, Phani, Surajit, Aritra da, Sourya da, Srijoni di, Papia di, and Chhotu da for always cheering me up and providing a constant entertainment throughout my stay at Kharagpur. My special gratitude to Soumabha da and Soumyadeep da for providing me with technical helps and ideas throughout the project.

The acknowledgement would be incomplete if I do not express my gratitude to the support staff of the department for providing me the necessary help during this course of study.

My gratitude to my parents, grandparents, and my little brother is beyond words. Their immense support and love helped me to achieve my best towards the completion of this thesis. I thank them profusely for all their love, care, and support.

Ushasi Chaudhuri
December 2017, India

List of Abbreviations

OCR	–	Optical character recognition
CBIR	–	Content based image retrieval
BoVW	–	Bag of visual words
SVM	–	Support vector machine
UGB	–	Unit grid block
EN	–	Euler number
DC	–	Directional change
VDC	–	Vertical direction change
HDC	–	Horizontal direction change
PoH	–	Position of holes
RP	–	Relative position
VPC	–	Vertical perimeter component
HPC	–	Horizontal perimeter component
ER	–	Edge ratio
PoC	–	Position of concavity
TP	–	Total number of polygons
MP	–	Major polygons
HC	–	Hole containment
PC	–	Polygon containment
MAP	–	Mean average precision
VR	–	Vertical ratio
HR	–	Horizontal ratio

List of Symbols

O_i	–	i^{th} Object
A_i	–	i^{th} Attribute
$\mathbb{A}$	–	Approximation space
U	–	Universe
R	–	In-discernibility relation
x and y	–	Points
$\overline{Apr}_{\mathbb{A}}(X)$	–	Tight upper approximation
$\underline{Apr}_{\mathbb{A}}(X)$	–	Tight lower approximation
$\alpha_{\mathbb{G}}(S)$	–	Accuracy of the rough-set representation of X
S	–	2-D digital object
$\mathbb{G}$	–	Cellular grid
g	–	Grid size
$\overline{\mathcal{P}}_{\mathbb{G}}(S)$	–	Best tight upper approximation of S
$\underline{\mathcal{P}}_{\mathbb{G}}(S)$	–	Best tight lower approximation of S
$\mathbb{H}$	–	Horizontal grid lines
$\mathbb{V}$	–	Vertical grid lines
U_i	–	i^{th} Unit grid polygon
$\leq$	–	Less than or equal to
$-$	–	Subtraction
$=$	–	Equals to
$\neq$	–	Not equal to
$\emptyset$	–	Null set
$\cap$	–	Intersection operator
$\cup$	–	Union operator
(i_0, j_0)	–	Coordinates of top-left vertex of a component
(i_s, j_s)	–	Coordinates of first point of a polygon cover

V_0 – First vertex of polygon cover

C_i – Classes

T_i – Type of i^{th} vertex

d_i – Direction from i^{th} vertex

l_i – Length of polygon from i^{th} vertex

n – Number of polygons

L – Leftwards

B – Bottomwards

R – Rightwards

U – Upwards

$\mathcal{P}$ – Polygon vertices

M – Mean of the polygon perimeters

m_i – Perimeter of i^{th} polygon

D – Deviation

Abstract

Rough set is a well-studied subject with theoretical foundation and many applications. However, its usage in image processing has been very sparse. Most of the well-known algorithms for document image processing related to character recognition, character spotting, and logo retrieval resort to a supervised classification, causing the system to slow down in speed with increasing diversity in the documents, as well as the need to have a large training dataset. Hence, with an aim to resolve the tediousness and pitfalls of training, but without compromising on the efficiency, we introduce a rough-set-theoretic model. It is designed to perform an unsupervised classification of optical characters and logos with a small subset of attributes, called the semi-reduct. The semi-reduct attributes are mostly geometric and topological in nature, each having a small range of discrete values estimated from different combinatorial characteristics of rough-set approximations. This eventually leads to quick and easy discernibility of almost all the characters and logos.

In this thesis we first explain the basics of rough set theory. Subsequently, we propose various attributes that can be easily computed from the binary representation of the images. In subsequent chapters we show how one can select an appropriate subset of such attributes, known as semi-reduct, to perform a document processing task.

We demonstrate in this thesis that using the above attributes one can design a character recognition system which is both computationally and storage efficient. Using a different semi-reduct we show that one can also solve the very delicate task of character spotting in ancient inscriptions. Additionally we propose appropriate pre-processing steps to binarize the old and dilapidated inscriptions. Finally, we propose a novel technique for logo retrieval using a suitably prepared semi-reduct. Comparison with other existing techniques substantiates our claim that attributes from rough set are indeed good candidates for document image processing.

Keywords: *Document image processing, isothetic polygon, rough set, orthogonal hull, character spotting, logo retrieval, OCR.*

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Rough set is a well-developed theoretical subject [**?**]. In the standard version of rough set theory, a rough set is an approximation of a crisp set. Rough sets can be used to design and implement an image processing tool, especially for binary images. The reason is quite simple: binary images have no tonal variation, and hence its constituent objects possess crisp and well-defined boundaries, which comply with the input requirement in classical image processing algorithms. However, an exact representation of a boundary often leads to practical difficulties like excessive computation, sensitiveness to noise, sensitiveness to topological variations like holes within the object, etc. Hence, in order to process binary images in a faster and approximate approach without compromising much with the quality, we use the concepts of rough set.

In 1982, Zdislaw Pawlak introduced formally the concepts of *rough set approximations* [**?**]. These concepts are found to be very efficient in the analysis of binary valued objects. Under the rough set theory, the tight upper and lower approximations of any component of a binary valued object are computed. Based on these approximate covers, various properties can be designed and analyzed very efficiently. Quite naturally, the computations depend on the granularity at which the cover is obtained. Further, if the covers are obtained in an axis-parallel way, then computations are found to be even faster. In this thesis, we use the rough set theory as the basic concept on which we solve three specific and very demanding engineering problems, which are the following:

1. Character recognition

2. Logo retrieval

3. Character spotting

1.1 Motivation and Scope of the Work

Document image processing is one of the major areas of application of image processing since its early years of development. It still continues to be a major focus of researchers in this area. A document is a written, drawn, or a memorialized representation of thought. Documents play very important roles in our day to day lives. In the ancient times, people used stones, leaves, tablets, and papyrus to document their thoughts. In the present times, with the emergence of the computer age, a "document" primarily denotes a textual computer file, which includes additional information, like the structure of the document, its format, fonts, colors, and images. A modern-day document can either be a paper document or an electronic document.

Documents store various information and records. In the current world of digitization, almost every kind of physical form of document is stored in digital format, in order to avoid any loss due to aging, wear and tear, or any unforeseen hazards. In today's world, almost every company keeps digital copies of their documents, be it bank, hospital, government organization, or an insurance agency. Many countries have also taken up initiatives to keep a digital archival of all valuable documents like ancient manuscripts, literary books, magazines, journals, conference papers, and the like. Many challenging problems come up in the process of digitization, which has been explored by various experts and practitioners in document image processing. As the challenges go on undiminished with growth of data accumulation and diversification, quest for a better alternative is always in question. With this in mind, in this thesis, we show how concepts from rough set theory can be used to design and develop novel techniques for providing effective solutions to three very common engineering problems related to document image processing and analysis.

1.1.1 Recognition of Characters

Optical character recognition (OCR) is a critical task identified for document processing [**?**]. There has been a huge increase in the demand of OCR systems [**?** **?** **?**], and with a vast set of fonts and handwritings, designing an efficient OCR system has become challenging and cumbersome. Most of the existing OCR engines resort to supervised classification, which involves a tedious training process for every new font. Immense time and tenacity is required to selectively prepare a training set with all new upcoming fonts. Also, too much training slows down an OCR engine, even for recognizing simple printed fonts. Hence, with the increasing volume and diversity

of datasets in today's world, we need an OCR system with reasonable speed and with capability of handling different font styles. This motivates us to consider an approximate-yet-robust representation of characters using the concepts of rough set theory.

In this thesis, an attempt has been made to create a rough-set semi-reduct for an alphanumeric character set to design an efficient font-independent OCR pipeline. Toward this, a novel technique for unsupervised classification of optical characters has been proposed, which works on a sub-optimal attribute set (semi-reduct). The proposed technique circumvents the pitfalls of training in a supervised classification scheme by using rough sets. Fig 1.1 shows some examples of the character R, and one can clearly see the variations in the font. The proposed unsupervised method attempts to handle such variations in the font style.

Figure 1.1: Instances of character 'R' written with different fonts.

1.1.2 Retrieval of Logos

A logo bears the identity of an organization in a paper or a fully digital document. Organizations have their own personalized and trademarked logos. Searching for similar logos in the registered logo database is a very important and tedious task at the trademark office. In the modern world of digitization, organizations all over the world register their logos in digital formats. With a boom in the number of start-up companies emerging every year, designing a new logo has become a necessity. Whenever a new company comes up in the market, they want to get their logos registered in the trademark office. These companies have to go through a list of previously registered logo-database populated by the existing companies. Normally this entire process is carried out manually, and it is quite a tedious method. Even using a logo partially resembling with an existing one may lead to a complex litigation, a costly and time consuming event. As the data set for the trademarked logo increases, it is needed to have a faster and automated method, which can provide

similar logos to that of the chosen one. Speed and accuracy are two aspects that one must attend to while developing a system for retrieval of logos. This motivates us to choose this problem of retrieving similar logos from a database. As we need to consider also logos which are partially similar, we have used a representation using the concepts of rough set theory.

In this thesis, design, and implementation of an unsupervised algorithm has been considered, which uses a rough-set based method to quantify the structural information in a logo image that can be used to efficiently index an image. A novel representation scheme has been proposed, in which a logo is split into a number of polygons, and for each polygon, the tight upper and lower approximations are computed based on the principles of rough set. Fig 1.2 shows an example of a few logo instances and our motivation is to design an unsupervised algorithm to retrieve logos similar to a query logo.

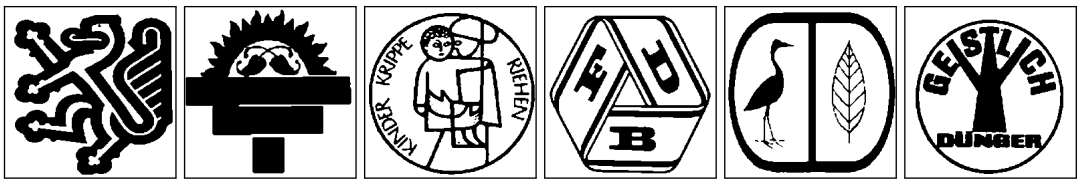


Figure 1.2: Instances of few logos from our database.

1.1.3 Spotting of Symbols

The early humans used to write on leaves, stones, papyrus scrolls, etc., in pictographic fonts. The ability to spot a few known characters or symbols allows the linguists and historians to get an idea about the era in which an inscription was made. With time, slowly these ancient inscriptions were found to be disintegrating, and on the verge of extinction due to various difficulties and challenges that are faced in their preservation. In order to keep records of these huge data, archiving them digitally can be useful for their conservation. For this, development of tools for their automated or semi-automated interpretation and indexing is required. Manual spotting of these characters proves to be very laborious and error-prone. Hence, automation in character spotting has evolved in recent time, which has its own challenges due to natural wear and tear of inscriptions through aging. In this problem too, a representation of inscription patterns is proposed by us using con-

cepts of rough set theory.

In this thesis, an effort has been made to develop a computationally efficient unsupervised technique for character spotting using concepts from rough set theory.

Figure 1.3: A sample Balinese inscription scroll.

To summarize, the main motivations for using rough set based analysis in document processing are as listed in the following:

- Rough set provides a computationally efficient way of analyzing a spatial binary data set, an image in this case.
- It is easy to define and compute the rough-set attributes.
- Character recognition, logo retrieval, and character spotting are chosen as typical problems to solve using the concepts of rough set theory, as each of them represents different aspects of document processing.
- Representations of characters derived from rough set theory provide font-independent models, which is a desirable feature in document processing.

1.2 Overview

This thesis presents three different methods in the field of document digitization, namely, character recognition, retrieval of logos, and spotting of characters. The following section briefs about the state-of-the-art techniques in these fields.

1.2.1 Literature Survey

In image processing, the document image processing is one of the classical fields of research and has been worked upon by many people over the years [**?**]; [**?**]; [**?**]; [**?**]. Still, this field has

many open challenges, as it is almost impossible to develop a technique which would work over a varied data set.

Optical character recognition (OCR) is one of the classic and oldest challenges of document digitization. With an increasing data size, it is required that we perform the recognition of characters requiring low computation. There exist several algorithms already implemented for the recognition purpose [**?** **?**]. An OCR allows us to store textual information in the images of documents in the form of electronic texts represented in ASCII, UNICODE, etc., requiring significantly less storage space [**?**]. The OCR systems can be designed for printed [**?**], [**?**], [**?**], [**?**] or handwritten scripts [**?**], [**?**], [**?**], [**?**], [**?**].

The Google Tesseract™, an open-sourced OCR [**?**], has remained a powerful tool in the public domain for a long time. Similar to the Tesseract™ there exists Abby™ [**?**], which is a licensed software. The performance of Abby™ is found to be better than that of Tesseract™. Tesseract™ uses various geometric features for detecting fonts. However, it requires a tedious training process to improve the efficiency of character recognition and the training also reduces the performance efficiency of Tesseract™ to a great extent. Beyond 32 training samples, Tesseract™ performs poorly.

For a retrieval system, the existing techniques of CBIR (Content Based Image Retrieval) retrieve images based on some of their features like color, texture, shape, bag of visual words (BoVW), spatial orientation, etc. Most researches that take place nowadays involve these kinds of feature sets [**?** **?** **?**]. However, still no fixed, best set of features has been recommended so as to have uniformly good results on varied data sets. Also, CBIR has two major problems, which are difficult to be resolved [**?**]. First, segmentation of an image to break it into meaningful parts based on the above low-level features is still a challenge. Then, there is a huge gap between the features and the semantic expressions embedded in an image [**?** **?**].

Shih and Piramuthu [**?**] have developed a technique of media retrieval by using rank SVM, and then augmented the data set for a large-scale text retrieval in books. Gagaudakis and Rosin [**?**] used a histogram-based scheme, as it maintains a lower complexity and high efficiency. They avoided segmentation of regions, as they considered it unreliable, and instead used a variety of schemes like incorporating shape feature and Delaunay triangulation as a performance enhancer.

In 2012, Bai et al.[**?**] developed an algorithm to fuse different similarity measures for robust shape retrieval through a semi-supervised learning framework, named co-transduction. When two

similarity measures and a query shape are given, their algorithm retrieves the most similar shape and assigns them to do a re-ranking. Fudos and Palios [**?**] also proposed a shape-based approach for image retrieval by gradually "fattening" the query shape until a best match is found. The accuracy is comparatively less and has a poly-logarithmic time behavior (for uniform distribution of shape vertices). A distance space densifying technique for shape and image retrieval was proposed by Yang et al. [**?**]. They presented an approach wherein they add synthetic ghost points in distance space directly. Using these ghost points they achieved an improved accuracy in shape, as well as image retrieval. A few other notable works in this area include logo retrieval using cascaded sparse color-localized matching by Pandey et al. [**?**], a tree-based shape descriptor for scalable logo detection by Wan et al. [**?**], and a modal analysis approach for the search of a reliable shape by Acqua and Gamba [**?**]. Swain and Ballard [**?**] used a technique of identifying objects in an image using colour histogram. This method became very popular since it was robust to changes in object's orientation, scale, and viewing position.

For the retrieval of logos, Jain [**?**] addressed some of their features of the issues like speed, accuracy and stability, based on colour and shapes of the images. Rajshekhar et al. [**?**] proposed a logo retrieval algorithm, which extracts shape features by calculating the morphological pattern spectrum for different structuring elements.

In this work, we propose an efficient method for binary logo retrieval using rough-set attributes. We create a reduct using these attributes and then retrieve images similar to the query images by using inverse Hough transform method [**?**]. For a faster searching algorithm, we use a k-d tree [**?**]. The proposed algorithm significantly gains in the average run time.

For designing a character spotting system, Aswatha et al. [**?**] proposed an algorithm for extraction of text from stone inscriptions using character-analytic elements like HoG features, vowel diacritics, and location-bounded scan lines. Word spotting in Latin alphabet has received the most attention [**?**], [**?**], [**?**], followed by Arabic text. The first Arabic system was developed by Srihari et al. [**?**]. They achieved a precision rate of 55% and a recall rate of 50% on 10 good-quality handwritten documents. Kchaou [**?**] in 2012 developed an algorithm for segmentation and word spotting of Arabic text.

Moghaddam and Cheriet [**?**] used Euclidean distance to measure the similarity between two connected components. The distance was enhanced by rotation and dynamic-time warping. Self-organizing maps were also used to cluster Arabic words, depending on the structure of their

shapes.

Srihari and Ball [**?**] proposed a language-independent word spotting system. The gradient features were first extracted, since it was a language-independent model. However, for Arabic language, it became necessary to apply manual word segmentation to circumvent the main problem that there exists no boundary between words.

Khayyat et al. [**?**] proposed a learning-based word spotting, yet again in Arabic handwritten documents. The system is designed to tackle the problem, in spite of the lack of boundary between the words in Arabic script. Language models are incorporated into the system to reconstruct words.

1.2.2 Limitation of the Existing Works

In the existing works, most of the algorithms use supervised techniques for image analysis. These algorithms require tedious training and their painstaking implementation. The efficiency of these algorithms drops drastically with an increased training data. Also, with a new dataset, the model has to be retrained again. More the training, more the efficiency of the algorithms get affected.

1.3 Contributions of the Thesis

Use of rough set analysis has proven to be a very popular approach in solving many difficult image analysis problems. However, the use of rough set analysis in various fields like character recognition, retrieval of logos, and spotting of characters has never been explored. Hence, the contributions of this thesis are as follows:

- **Optical Character Recognition:** It is investigated in detail the applicability of rough set reduct in OCR systems. Although there have been attempts [**?**] to solve OCR using a rough set approximation, a machine learning approach on the computed cover did not prove to be a good method for character recognition as the reported results have been very poor. An approximate set of features is designed from the computed covers (called the rough set semi-reduct) and define a decision tree which classifies each character based on its semi-reduct attributes. A Tesseract™ reinforced model has also been proposed which classifies a font-independent English character set.

- **Retrieval of Logos:** In this work, the previously designed attribute set for the OCR engine is

redesigned according to the requirements of the logo retrieval problem. Each logo is found to have multiple components and hence consists of multiple rough set polygon covers. Due to this multiple polygons, various containment relationships are preserved as attributes of the semi-reduct. Using an inverse Hough-transform-based retrieval technique, the logos are retrieved efficiently.

- **Spotting of Characters:** In this work, a leaf inscription in Balinese language is chosen. This problem is handled by first developing a binarization model for the images. While the character recognition problem is targeted at the recognition of the individual characters, here the characters are matched from a pool of other characters. Again, here an unsupervised spotting system is designed. Since the approach is not specific to the language script, with a modified semi-reduct set, a language-independent symbol spotting technique can also be re-designed.

1.4 Organization of the Thesis

The rest of the thesis is organized into six chapters, which are as follows:

- **Chapter 2:** contains a brief overview of the rough set theory, its application in our work, and various other existing methods that are used in our proposed methods.
- **Chapter 3:** presents the proposed optical character recognition technique for different binarized English fonts.
- **Chapter 4:** contains the proposed technique for the retrieval of logo images using a redesigned rough set semi-reduct.
- **Chapter 5:** presents a technique for the spotting of characters in an approximation space, with yet another re-designed rough set attributes for this purpose.
- **Chapter 6:** concludes the thesis by suggesting various possible extensions of the work and future direction of research work.

Chapter 2

Background

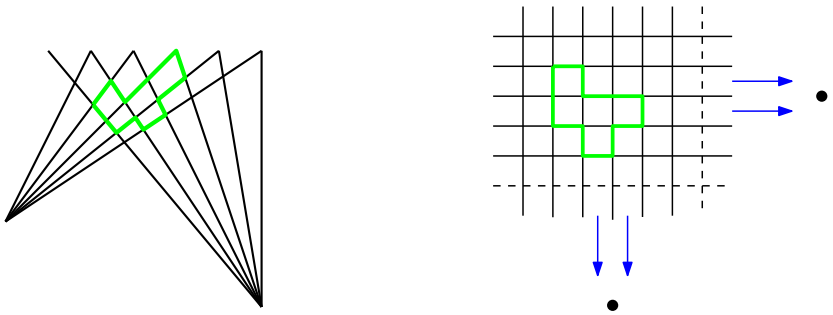
In this chapter, the concepts of rough set and its application in spatial domain and attribute domain have been briefly explained. The initial sections recall certain concepts of rough set and the subsequent sections describe few standard algorithms and their implementations in detail. Also, this chapter defines various notations that have been used throughout the thesis.

2.1 Rough Set

In computer science, Pawlak [**?** **?**] was the first person to introduce the fundamentals of rough set. Since then, the concepts of rough set have been very popularly used in the domain of pattern recognition, learning algorithms, inductive reasoning, etc. In the standard version of rough set theory, a *rough set* is an approximation of a crisp set. When the approximation set is considered to be a polygonal set, a rough set is defined as an approximation of a crisp set in terms of a pair of polygonal covers for lower and upper approximations of the original set. An object is treated as a digital object, laid on a cellular grid $\mathbb{G}$, and approximated by its tightest cover called rough-set cover.

A *rough-set polygon* is a polygon, which is constructed by two parametric families of straight lines with their centers at two points. Rectilinear polygons are a type of isothetic polygons with their centers lying at infinity. Fig 2.1a shows the basic mechanism of construction of a rough-set polygon. The diagram shows how diverging lines emerge from two pre-selected points. If these points are selected at infinity and the two families are orthogonal to each other, then a pair of axis-parallel polygons is obtained. For computational advantage, in this thesis, only axis-parallel polygons are considered, as shown in Fig 2.1b. Fig 2.2 shows two examples of rough-set covers with different crispness. Fig 2.2a does more approximation of the boundary of the object. Whereas, Fig 2.2b is

a more crisp example of the object. It is noticed that higher the approximation of an object, more is the loss in its intrinsic details. However, a higher approximation saves the computational time computational time (explained in the forthcoming chapters).



(a) A rough-set polygon with finite centers (b) A rough-set polygon with centers at infinity

Figure 2.1: Polygons in rough set.

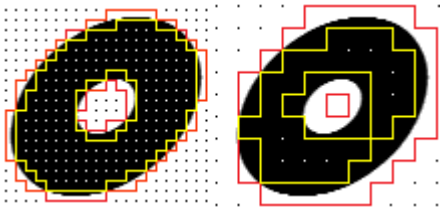


Figure 2.2: Rough-set approximations with different amounts of crispness, 6×6 and 12×12.

Sometimes a subset of attributes which can, by itself, fully characterize the knowledge in the database, is needed. Such an attribute set is called a *reduct*. A formal definition is as follows:

Definition 2.1 (Rough-set reduct)**.** *A* rough-set reduct *is considered to be a subset of definable attributes which is sufficient for a given purpose.*

The main objective of defining a reduct is that it is the minimal attribute set, which is enough for the discernibility of maximum number of objects. This is explained more elaborately in the following example. Let the following information table (Table 2.1) having objects O_i and its attributes A_i be considered. Let $A_i \in \mathbb{R}$. The table contains the value of each attribute for each object. There are a limited discretized values possible for an attribute A_i. Using the attributes, we are to construct a reduct, which makes the combined attribute set for each object so as to act like a primary key. In this example, it is noticed that the reduct can be constructed by $A = \{A_1, A_2, A_4\}$.

Table 2.1: Sample information table. Reduct = $\{A_1, A_2, A_4\}$

Object	A_1	A_2	A_3	A_4
O_1	0	0	0	0
O_2	2	1	0	0
O_3	0	0	1	1
O_4	1	0	1	0
O_5	2	0	0	1

This is the smallest set to discriminate any two objects. If a smaller set of attributes $A = \{A_1, A_2\}$ is considered, we get the following object classes which hold an equivalence relation : $\{\{O_1, O_3\}, \{O_2\}, \{O_4\}, \{O_5\}\}$. To distinguish between O_1 and O_3, at least one more attribute is needed, which is A_4. The remaining objects are all well-discernible amongst each other. If $A = \{A_1\}$, then a much coarser equivalence relation structure, i.e., $\{\{O_1, O_3\}, \{\{O_2\}, \{O_5\}\}, \{O_4\}\}$, is obtained.

A pair $\mathbb{A}$ = (U, R) is called an *approximation space*, where U is the *universe*, and R is an equivalence relation called *indiscernibility relation*. If any two points x and y in U are such that $(x, y) \in R$, then x and y are said to be indistinguishable in $\mathbb{A}$.

X is used as a subset of U. An empty set is denoted as 0, while an universal set U is denoted as 1. Every finite union of elementary sets is called a *composed set*. A finite and minimal composed set in $\mathbb{A}$ containing X is called the *best tight upper approximation* of X in $\mathbb{A}$, denoted by $\overline{Apr}_{\mathbb{A}}(X)$. The greatest composed set in $\mathbb{A}$ contained in X is called the *best tight lower approximation* of X in $\mathbb{A}$, denoted by $\underline{Apr}_{\mathbb{A}}(X)$. The following topological relations [**?**], [**?**] hold true:

$$\overline{Apr}_{\mathbb{A}}(X) \supset X \supset \underline{Apr}_{\mathbb{A}}(X) \tag{2.1}$$

$$\underline{Apr}_{\mathbb{A}}(1) = \overline{Apr}_{\mathbb{A}}(1) = 1 \tag{2.2}$$

$$\underline{Apr}_{\mathbb{A}}(0) = \overline{Apr}_{\mathbb{A}}(0) = 0 \tag{2.3}$$

$$\underline{Apr}_{\mathbb{A}}(\underline{Apr}_{\mathbb{A}}(X)) = \overline{Apr}_{\mathbb{A}}(\underline{Apr}_{\mathbb{A}}(X)) = \underline{Apr}_{\mathbb{A}}(X) \tag{2.4}$$

$$\overline{Apr}_{\mathbb{A}}(\overline{Apr}_{\mathbb{A}}(X)) = \underline{Apr}_{\mathbb{A}}(\overline{Apr}_{\mathbb{A}}(X)) = \overline{Apr}_{\mathbb{A}}(X) \tag{2.5}$$

$$\overline{Apr}_{\mathbb{A}}(X) = U - \underline{Apr}_{\mathbb{A}}(U - X) \tag{2.6}$$

$$\underline{Apr}_{\mathbb{A}}(X) = U - \overline{Apr}_{\mathbb{A}}(U - X) \tag{2.7}$$

In this thesis, the concepts of rough set are used in 2 domains:

i for object approximation in the spatial domain

ii for performing feature approximations

Fig 2.3 illustrates the usage of rough sets in the spatial domain. The connected component of the optical object in the image is referred to as X here. The red polygons show the tight upper approximation, while the yellow polygon shows the tight lower approximation.

A *boundary region* is defined as the difference between the upper approximation and the lower approximation, i.e., $\overline{Apr}_{\mathbb{A}}(X) - \underline{Apr}_{\mathbb{A}}(X)$. It consists of objects that can neither be ruled in or ruled out as members of the complete set X.

The *accuracy* of the rough set representation of X is given by

$$\alpha_{\mathbb{A}}(X) = \frac{\text{area}\left(\underline{\text{Apr}}_{\mathbb{A}}(\text{X})\right)}{\text{area}\left(\overline{\text{Apr}}_{\mathbb{A}}(\text{X})\right)} \tag{2.8}$$

Clearly, the value of $\alpha_{\mathbb{A}}(X)$ lies in between 0 and 1.

Moshkov et al. [**?**] stated that, let a family $\mathcal{Q}$ be called a *partial cover* for $(\mathbb{A}, X)$. Let $\alpha \in \mathbb{R}$ and $0 \leq \alpha < 1$. The family $\mathcal{Q}$ will be an α-*cover*. For example, 0.05-cover means that the cover contains at least 95% of the boundary elements of X. In our algorithm, we start by finding the optimum α value to set up an appropriate grid, $\mathbb{G}$. Fig 2.3 demonstrates the upper and the lower boundaries for an object X. It may be noted, without any loss of generality, the word upper approximation means the approximation of the outer boundary and the word lower approximation means the approximation of the inner boundary.

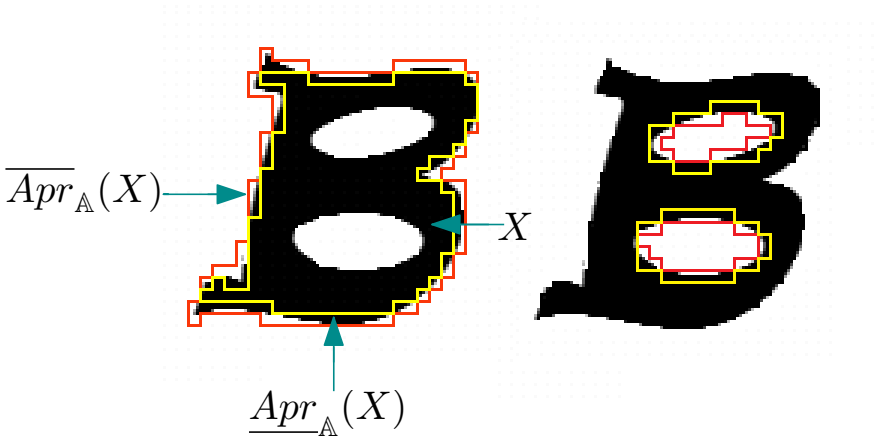


Figure 2.3: Upper and lower approximations of an object X

2.1.1 Requirements of Rough Set

We use rough sets because of many reasons, which are listed as follows:

- Computationally efficient for analyzing a spatial binary dataset.
- Ease of defining an optimal or sub-optimal set of attributes (reduct).
- Reduced computations.
- Reasonably accurate as it discards the effect of noise in the images.
- Avoiding tedious process of training and re-training, and useful for an unsupervised model.

In the following section (Sec. 2.2), the construction of the upper and the lower approximations of an object in the spatial domain has been explained.

2.2 Construction of Upper and Lower Approximations

Let S be a 2D digital object and $\mathbb{G}$ a cellular grid. Let the tight upper and lower approximations of S be denoted by $\overline{\mathcal{P}}_{\mathbb{G}}(S)$ and $\underline{\mathcal{P}}_{\mathbb{G}}(S)$, respectively. The concepts discussed in [**?**] are used for the construction of the rough-set polygon. In that algorithm, the authors define a digital grid as a set $\mathbb{H}$ and $\mathbb{V}$, of horizontal and vertical grid lines for a grid size of g, where $g \in \mathbb{Z}^+$. Here, we have $\mathbb{H} =$

$\{..., I_H(j-g), I_H(j), I_H(j+g), ...\} \subset \mathbb{Z}^2$ and $\mathbb{V} = \{..., I_V(i-g), I_V(i), I_V(i+g), ...\} \subset \mathbb{Z}^2$. Also, we define $I_H(j) = \{(i^{'}, j) : i^{'} \subset \mathbb{Z}\}$ and $I_V(i) = \{(i, j^{'})\} : j^{'} \subset \mathbb{Z}\}$, and here i and j are multiples of g.

A horizontal line I_H divides a digital plane $\mathbb{Z}^2$ into two halves (upper half and lower half). Similarly, the line I_V divides the digital plane $\mathbb{Z}^2$ into two lateral halves (left half and right half). The region enclosed by the lines I_H, I_{H+1}, I_V, and I_{V+1} is known as an *unit grid block* (UGB). Hence, each grid point g is contained in its four neighboring unit grid blocks, i.e, U_1 = UGB (i, j), U_2 = UGB $(i-g, j)$, U_3 = UGB $(i-g, j-g)$, and U_4 = UGB $(i, j-g)$. Fig 2.4 illustrates how four UGBs are incident at a point $p(i, j)$.

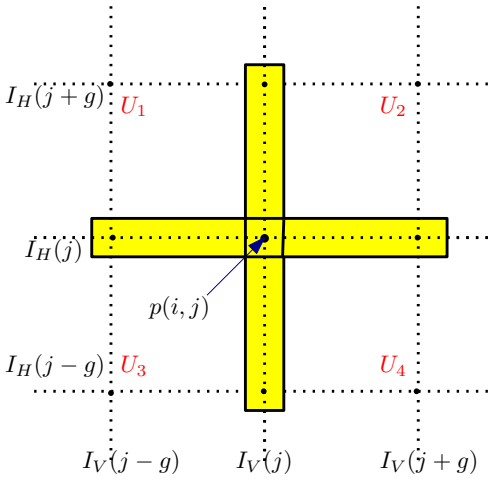
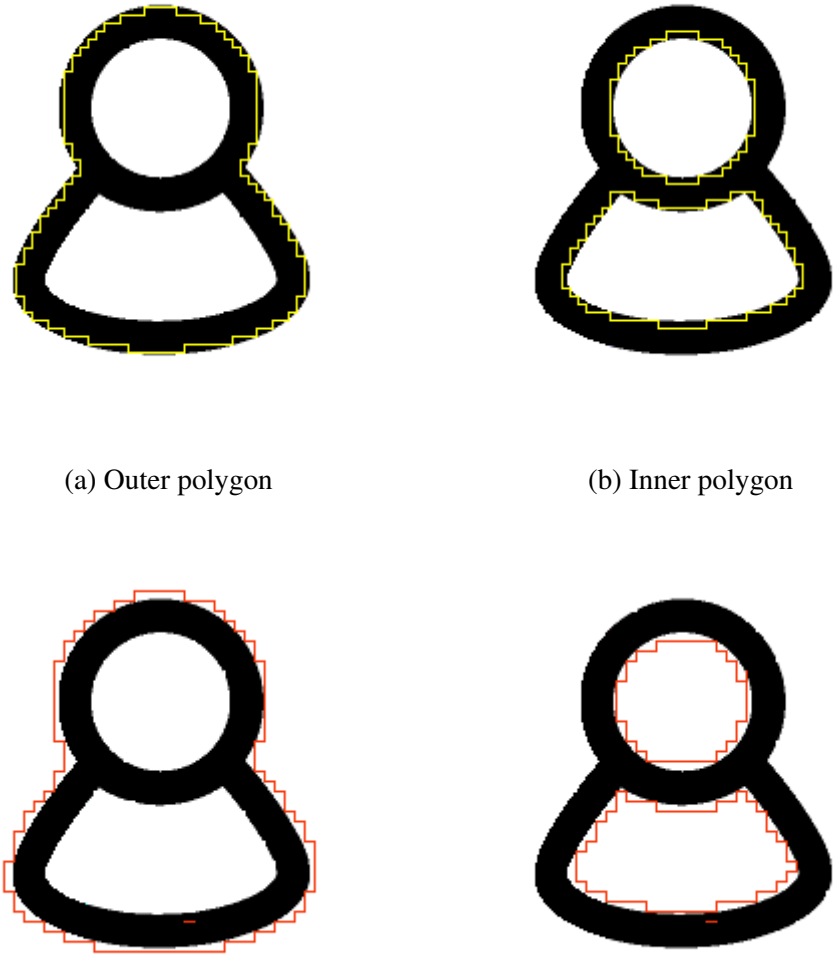


Figure 2.4: UGBs incident at a point $p(i, j)$

Usually, there are four types of rough-set polygons for an object S, which are as follows.

- *Outer polygon*, also called the tight upper approximation of S; (Fig 2.5(a))
- *Inner polygon*, also called the tight lower approximation of S; (Fig 2.5(b))
- *Outer hole polygon*, i.e., tight upper approximation of void; (Fig 2.5(c))
- *Inner hole polygon*, i.e., tight lower approximation of void; (Fig 2.5(d))

To construct the rough-set polygonal cover for an object S, the start vertex v_0 (i.e, the top-left vertex) is first found. Once this vertex is found, the top-left vertex of its unit grid block is



(a) Outer polygon

(b) Inner polygon

(c) Outer hole polygon

(d) Inner hole polygon

Figure 2.5: Different types of rough-set polygons.

considered. The coordinates of the point v_0 are given as follows.

$$i_s = \left(\left\lceil \frac{i_o}{g} \right\rceil - 1 \right) \cdot g, \tag{2.9}$$

$$j_s = \left(\left\lfloor \frac{j_o}{g} \right\rfloor + 1 \right) \cdot g. \tag{2.10}$$

Here, (I_0, J_0) is the top-left pixel in the set S. The unit grid block U_4 containing v_0 is considered. Fig 2.6 shows the selection of v_0 for the rough-set cover.

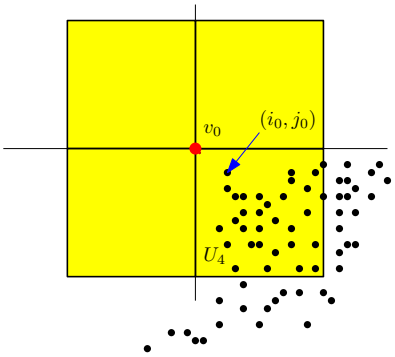


Figure 2.6: Top-left vertex.

After finding the start vertex, it is noticed that any grid point, which are surrounded by four unit grid blocks, can be categorized into one of the following five classes:

C_0: Invalid vertex point as none of the cells contains any object point.

C_1: Only one of the UGBs contains object point, making a 90°
vertex of the polygonal cover.

C_2: Two cells contain object points. It can be of two types:
(i) Object points lie in two edge-adjacent UGBs.
(ii) Object points lie in two vertex-adjacent UGBs.

C_3: Three cells contain object points. They make a 270° vertex.

C_4: All the cells contain object points, so it is not a valid vertex.

Fig 2.7 illustrates different classes. Since axis-parallel lines are used for construction of the polygon, any vertex can have one of the angles from 90°, 180°, and 270°. 'Type' values are assigned to different vertices based on these properties. A vertex having a 90° angle is assigned a 'Type' value of '$+1$', while a 270° vertex is assigned with a 'Type' value of '-1'. The 180° vertices are assigned with a value of '0'. Fig 2.8 illustrates the assigning of different Type values to different vertices based on the angle that they make.

Let d define the direction of traversal to be followed from a vertex, for the construction of the

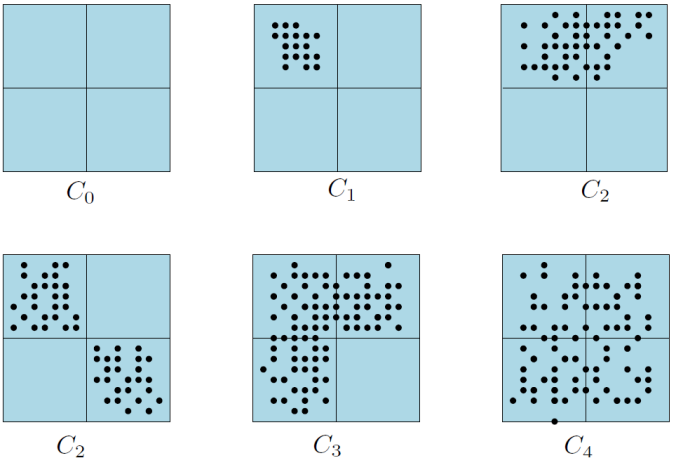


Figure 2.7: Types of classes.

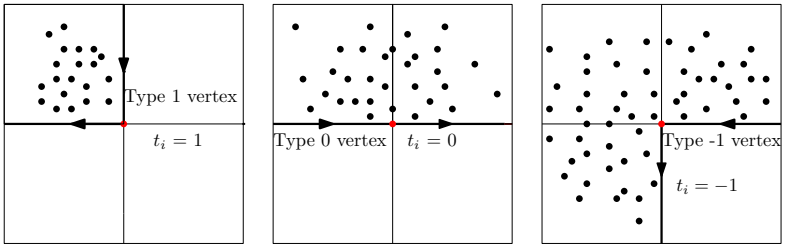


Figure 2.8: Types of vertices.

rough-set polygon. Four directions are defined, namely 0, 1, 2, and 3, for denoting the directions rightwards, upwards, leftwards, and downwards, respectively. Fig 2.9 illustrates the direction conventions used in this algorithm. From point v_0, the following equation is used for finding the direction of traversal, where d_s is the starting direction and t_s is the type of first vertex, which is 90° (Type '$+1$').

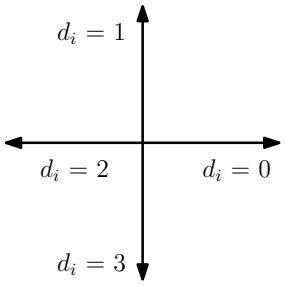


Figure 2.9: Direction conventions.

$$d_s = 2 + t_s \tag{2.11}$$

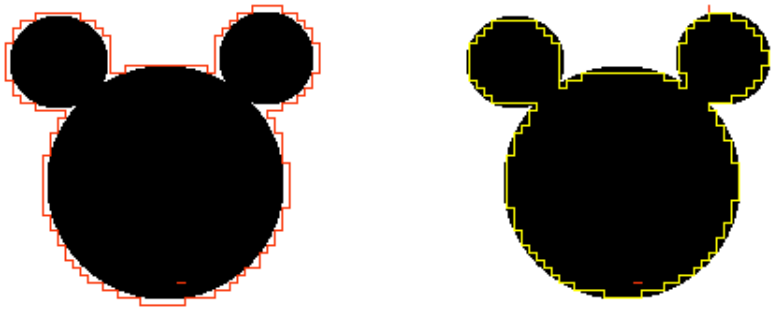
Once the value of d is obtained, the coordinates i and j of the next vertex are computed as:

$$d_s = 0\text{: } i = i + g; j = j$$
$$d_s = 1\text{: } i = i; j = j + g$$
$$d_s = 2\text{: } i = i - g; j = j$$
$$d_s = 3\text{: } i = i; j = j - g$$

After getting the updated values of i_x and j_y, these values are repeatedly updated by using the distance equation:

$$d_n = (d_c + t_n) \mod 4 \tag{2.12}$$

where, d_n is the n^{th} direction, d_c the current direction, and t_n the Type of the n^{th} vertex. During the construction, if the first vertex (i_s, j_s) is traversed back again, the computation ends, as continuation might lead to redundancies and result in a loop. Fig 2.10a shows an example of the final rough-set polygon created for a random object. The list L stores all the vertexs in it. Algorithm 1 shows the steps of construction of a rough-set polygons.



(a) Upper-approximation of a polygon.(b) Lower-approximation of a polygon.

Figure 2.10: Polygonal covers of an object S.

The above algorithm demonstrates the construction of the tight upper-approximation poly-

Algorithm 1: Construction of rough-set outer polygon

Input: Binary image
Input: Optical objects S
Output: Image segmentation using rough-set boundary $\overline{\mathcal{P}}_{\mathbb{G}}(S)$

$L \leftarrow \emptyset$
$i_s \leftarrow \left(\left\lceil \frac{i_o}{g} \right\rceil - 1\right) \cdot g$; $j_s \leftarrow \left(\left\lfloor \frac{j_o}{g} \right\rfloor + 1\right) \cdot g$
$v \leftarrow (i_s, j_s)$
$L \leftarrow \{v\}$; $t \leftarrow 1$; $d \leftarrow t + 2$;
$d \leftarrow (d_c + t_n) \mod 4$
do
    if $t \in \{1, -1\}$ **then**
        $L \leftarrow L \cup \{v\}$
        $q \leftarrow d \odot q$
        $t \leftarrow$ VertexType (S, v)
    end if
while $q \neq q_s$
return L

gon of an object. A similar approach has been used for the construction of the tight inner-approximation of an object. Whenever a top-left object point is encountered, instead of finding the top-left grid point, here the bottom right grid point is found, which is also another object point. To find this point, the following equations are used:

$$i_{s'} = \left(\left\lceil \frac{i_o}{g} \right\rceil\right) \cdot g, \tag{2.13}$$

$$j_{s'} = \left(\left\lfloor \frac{j_o}{g} \right\rfloor + 2\right) \cdot g. \tag{2.14}$$

Once the coordinates of the starting point is obtained, the direction is calculated using the direction equation given in eqn 2.12, but using opposite conventions for assigning the Type values, i.e, '-1' for the 90° vertices and '$+1$' for the 270° vertices. Fig 2.10b shows an example of a lower-approximation of a digital object X.

The basics of this algorithm has been used for the construction of the rough-set polygon. Using this algorithm for the construction of the polygon, and the algorithm in [**?**] an orthogonal hull is constructed. The algorithm for its construction is explained in the following section.

2.3 Orthogonal Hull

The algorithm discussed in [**?**] has been used for the construction of an approximate-orthogonal hull, inscribing the object X. The main reason of using this hull polygon is that it detects concavities and constructs a cover across it. Hence, by using this hull, we are also able to detect the direction of the concavities. The algorithm finds the hull of an object, without having any prior knowledge about its rough-set cover. It uses a retracing algorithm for the construction of hull..

A *concavity* is defined whenever two or more consecutive Type '-1' vertices occur consecutively. If more than two Type '-1' vertices occur consecutively, a nested concavity is obtained. Hence, concavities are classified in two types:

- Simple concavity—Pattern $(1, -1, -1, 1)$
- Nested concavity—Pattern $(1, -1, -1, -1)$

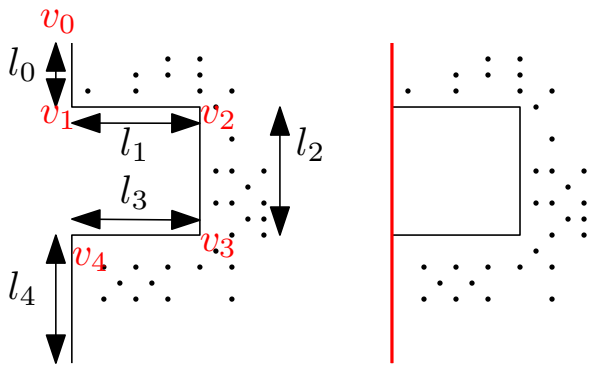
Considering four consecutive concave vertices together, two types of vertex pattern can occur, i.e, pattern '$(1, -1, -1, 1)$' and pattern '$(1, -1, -1, 1)$'. The construction for each cases have a few variations. Let v_0, v_1, v_2, v_3, and v_5 be five consecutive vertices in order, where v_4 is the most recently visited vertex. The value l_i gives the length of the line segment from v_i to v_{i+1}, d_i denotes the direction of traversal from v_i, and t_i denotes the Type of vertex of v_i.

2.3.1 Pattern $(1, -1, -1, 1)$

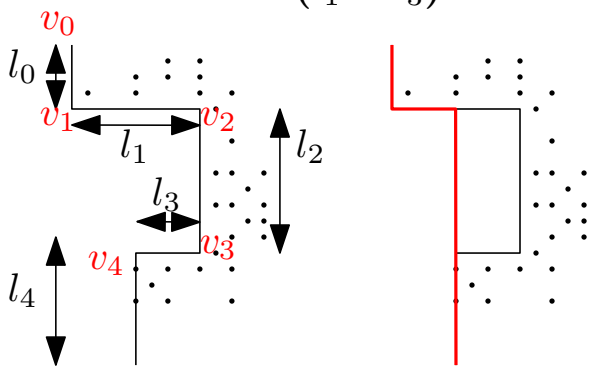
This pattern is a type of simple concavity, wherein two consecutive Type '-1' vertices are found to occur. The vertices of the upper-approximation of the polygon are stored in a list, containing the details about the l_i, t_i, and d_i corresponding to each vertex. There can arise three cases under these types, depending upon the lengths of l_1 and l_3.

- **Rule R11**— Used when $l_1 = l_3$. In this case, vertices v_1, v_2, v_3, and v_4 are removed from the list.
- **Rule R12**— Used when $l_1 > l_3$. In this case, vertices v_3 and v_4 are removed.
- **Rule R13**— Used when $l_1 < l_3$. In this case, vertices v_1 and v_2 are removed.

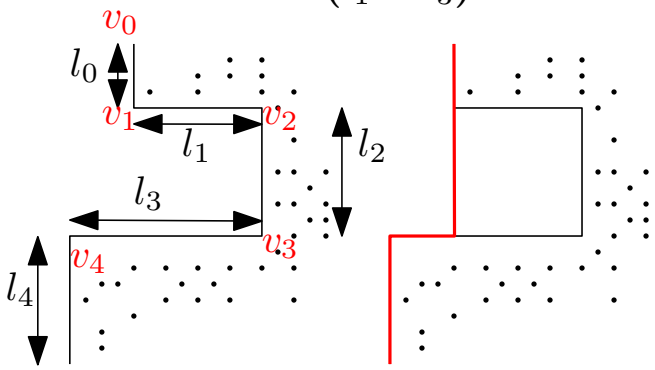
The lengths l_i get appended according to the requirements, so as to adjust with the new pattern of the cover. The Fig 2.11 illustrates the hull construction in all the three cases.



Rule R11: $(l_1 = l_3)$



Rule R12: $(l_1 > l_3)$



Rule R13: $(l_1 < l_3)$

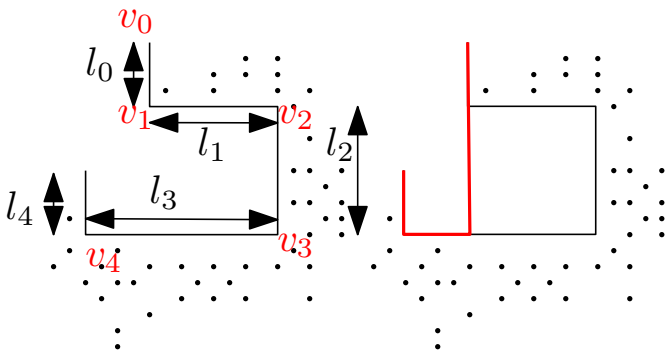
Figure 2.11: Rule R11, R12, and R13 illustration.

2.3.2 Pattern $(1, -1, -1, -1)$

This pattern is a type of nested concavity, wherein more than two consecutive Type '-1' vertices occur consecutively. There can arise three cases under this types, depending upon the lengths of l_1 and l_3.

- **Rule R21**— Used when $l_1 < l_3$. In this case, vertices v_1 and v_2 are removed.
- **Rule R22**— Used when $l_1 \geq l_3$ and $d = d_2$. In this case, vertices v_3 and v_4 are removed.
- **Rule R23**— Used when $l_1 \geq l_3$ and $d = d_3$. In this case, vertex v_4 is removed.

The Fig 2.12 to Fig 2.14 illustrate the construction of hulls in all the three cases. This algorithm of vertex reduction is implemented on the list of vertices multiple number of times, as long as the reduction stops. The end result is the orthogonal hull cover for the object.



Rule R21: $(l_1 < l_3)$

Figure 2.12: Rule 21

2.4 Attribute Domain

The concepts of the rough-set have also been made use in the attribute domain, wherein various approximations of the attributes have been performed, which are explained further in the next chapters. These approximated attributes are then used to construct a sub-optimal feature-set for our algorithms, also known as the *semi-reduct*.

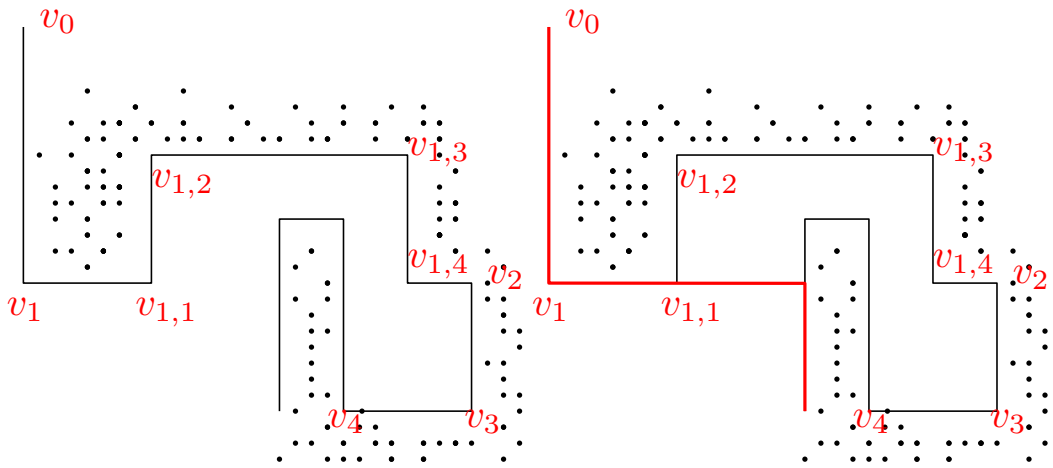


Figure 2.13: Rule 22; $((l_1 \geq l_3)$ and $d = d_2$

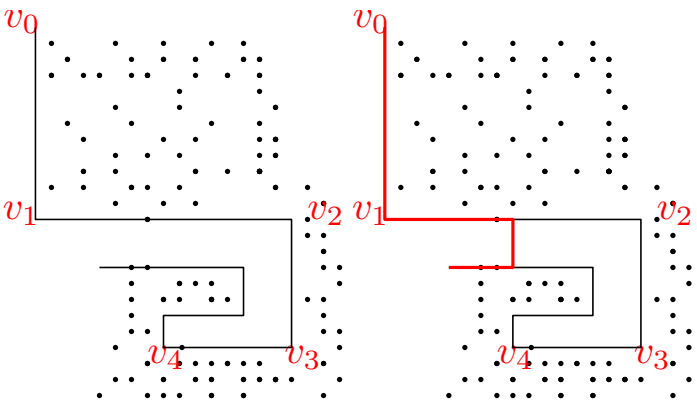


Figure 2.14: Rule 23; $((l_1 \geq l_3)$ and $d = d_3$

2.5 Summary

Various background techniques used in this work have been explained in this chapter. Effort has been taken to keep the background literature brief and concise and keep it in line with the proposed methods. The applications and constructions of the attribute are discussed in subsequent chapters.

Chapter 3

Character Recognition using Rough-Set Semi-Reduct

In today's world, information dissemination is a very important part of life. It can be through books, newspapers, letters, etc. Until a few decades ago, papers used to serve as the main medium for the purpose of storing documents. Computers were used to be very expensive and not easily available. With the advancement in the computer industry and its processing speed, we soon arrived in the era of digitization. Over the ages, as data keeps on accumulating, it becomes a necessity to store these data in a digitized format. To make a digitized copy of documents, initially manual methods were implied, which were quite painstaking and caused immense drudgery. Hence, the requirement of efficient OCR systems came into use.
Usually, documents can be of two forms:

- Printed documents
- Handwritten documents

Optical character recognition (OCR) continues to remain a major field of research in document digitization till date [**?**]. It has a multitude of connections with text- and graphics-related applications, like editing, searching, spotting, retrieval, and formatting, for developing a better recognition model [**? ? ?**]. With an increase in the demand of OCR systems and a vast set of different fonts and handwritings, designing an efficient OCR system has become more and more challenging and cumbersome. In recent time, efficient OCR engines resort to supervised classification techniques. This shoots up the challenge of an OCR system to an inordinate level with complex font styles. Varied complex font styles means undergoing a tedious training process for every new font. This also leads to a big droop in the speed of the OCR engine. Immense time and tenacity is required to selectively prepare a training set with all new upcoming fonts. Also, too

much training slows down an OCR engine, even for recognizing simple printed fonts. Hence, with the increasing volume and diversity of the datasets in today's world, a scheme for representing characters is required that can enable to perform the task of recognition with the least computational time complexity, and yet to be capable of handling a huge variations in font styles. We refer to [**?** **?**] and the bibliographies therein for their comparative study. Fig 3.1 shows a sample of a few fonts of the character G.

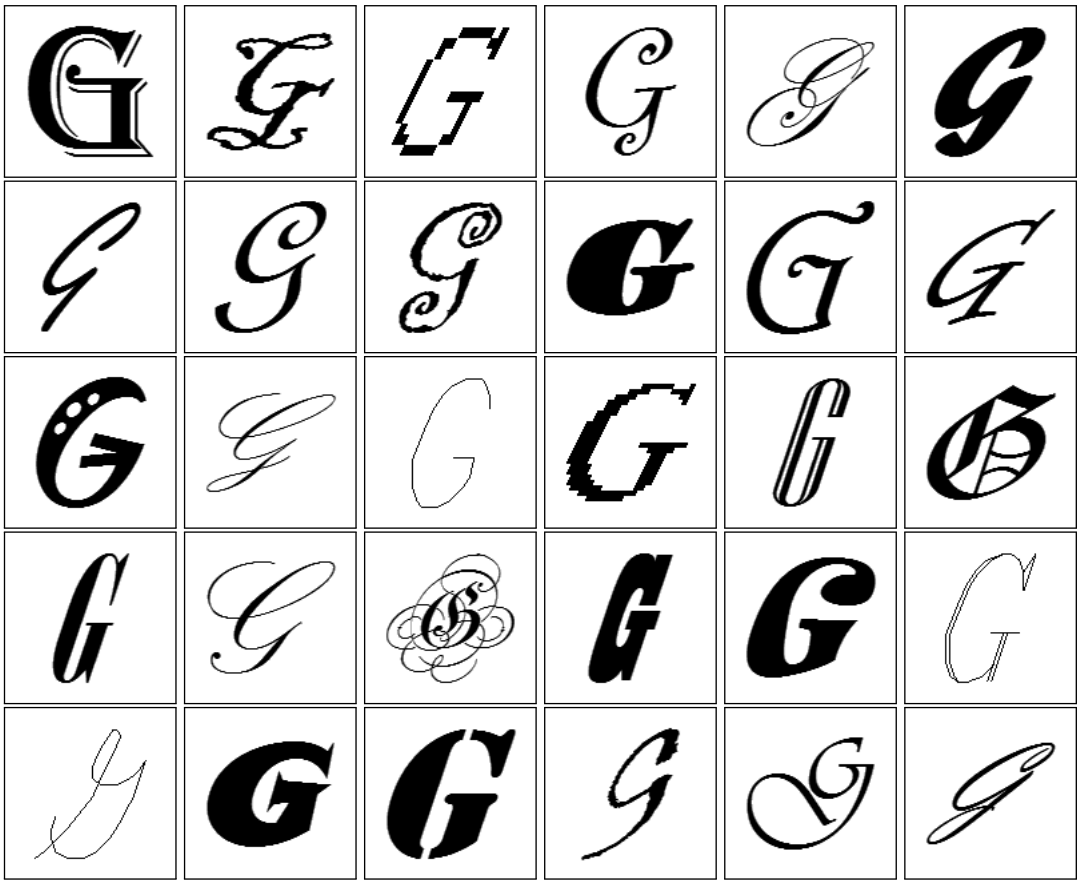


Figure 3.1: A typical example of character 'G' written with different fonts.

In this chapter, we aim to create a rough-set semi-reduct for an alphanumeric character set to design an efficient OCR pipeline. We propose a method for an unsupervised classification of optical characters with a suboptimal attribute set, which is referred to here as the semi-reduct. This work makes the following novel contributions:

- An unsupervised classification of optical characters with a suboptimal attribute set (semi-

reduct).

- A hybrid Google Tesseract™ reinforced technique for character recognition.

3.1 Methodology

Definition 3.1 (Semi-reduct)**.** *A suboptimal set of rough-set attributes, which is sufficient for classification of different classes is called a rough-set semi-reduct.*

An overview of the pipeline of the proposed approach is shown in Fig 3.2. We design some attributes, which are explained in Sec. 3.1.2, and find the semi-reduct for classifying the alphanumeric shapes. These characters are then classified as definite or indefinite classes. These classes are created based on how much discernible a character is from other characters. The indefinite class characters are the ones, which are difficult to recognize and sometimes yields erroneous results. These classes are therefore reinforced with the Tesseract™ for the final character recognition. The definite and indefinite class members have been explained in more detail in section 3.2. We perform a Tesseract™ reinforcement on the obtained indefinite class characters, and take the recognition from the definite class members directly from the proposed algorithm, and finally obtain the recognized character. The following section explains the attributes used in our algorithm in detail.

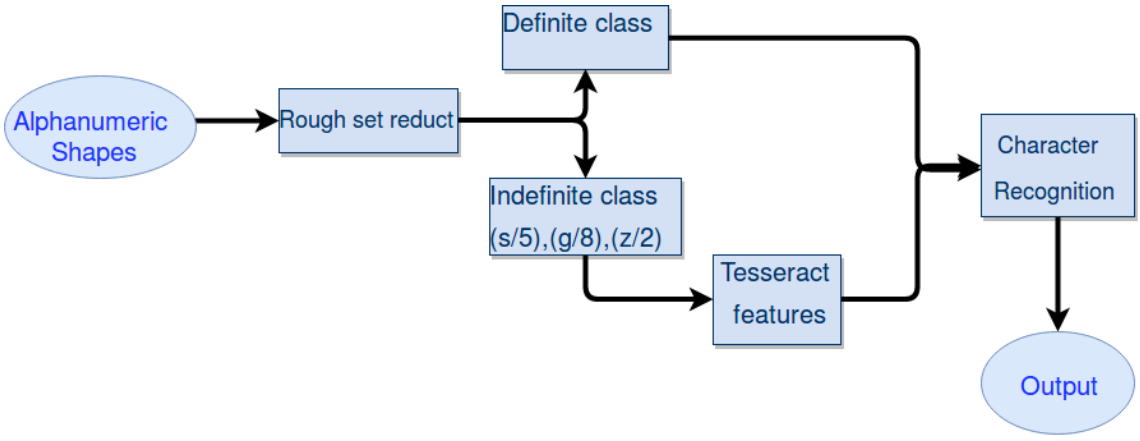


Figure 3.2: An input image with its processing blocks with colored bounding rectangles

3.1.1 Optimum Grid Size

Selection of an optimum grid size is very important for a model to be efficient, yet preserving on the accuracy of the system. With each input image of each character, if we consider the smaller between the length and breadth of the image and then scale down that dimension by $\frac{1}{8}$, then we get an initial value of grid size, g. After getting this grid size, we find the number of distinct polygonal covers (constructed by the algorithm explained in Sec. 2.2) detected by the system. We then keep dividing this grid size by 2 (i.e., double the resolution), until we see further increase in resolution does not change the number of distinct polygonal covers. The size that we finally obtain is set as the optimum grid size for that document. The steps are given in Algorithm. 2.

Algorithm 2: Optimum grid-size selection.

Input: Binary image
Input: Optical objects S
Output: Image segmentation using rough-set boundaries $\overline{\mathcal{P}}_{\mathbb{G}}(S)$

1: $N \leftarrow 0$
2: $g \leftarrow \min(\text{length, breadth})/8$
3: **do**
4: $\quad N' \leftarrow$RoughSetCover(g)
5: $\quad g \leftarrow g/2$
6: $\quad N \leftarrow$RoughSetCover(g)
7: **while** $N \geq N'$
8: **return** g

3.1.2 Semi-Reduct Attributes

We consider each character as a distinct optical object. Using the concepts discussed in Sec. 2.2, we construct the upper approximations of these objects and get their rough-set covers. The covers of each character are observed to have quite similar characteristics irrespective of the font style. These characteristics are extracted in form of rough-set attributes or features and are further used for classifying them. Extraction of these features is discussed from section 3.1.2.2 onwards.

3.1.2.1 Similar Characters

By studying the geometric structure of the upper-case characters, lower-case characters, and the numerals, we notice that there are several classes of characters where each class comprises two or more characters having close resemblance with each other. Sometimes, the only difference between two characters is the same class is their size. Since we are designing a context-free classification system, we group these similar characters into a single class, and name it as an *indefinite class*. This is a top-level labeling of the characters. Further sub-classification of characters is carried out by distinguishing similar characters within each of these classes. The characters that are in the same class are listed in Table. 3.1. The rest of the characters are all found to be distinct in their structures.

Table 3.1: Different classes of similar characters.

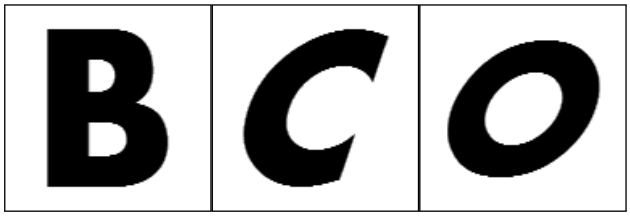
Class.	Characters
1	(C, c)
2	(J, j)
3	(K, k)
4	(M, m)
5	(P, p)
6	(S, s)
7	(U, u)
8	(V, v)
9	(W, w)
10	(X, x)
11	(Y, y)
12	(Z, z)
13	(O, o, 0)
14	(1, i, I, l)

3.1.2.2 Approximate Euler Number

Mathematically, the Euler number of a binary image is defined as $E = N - H$, where N is the number of connected components in the image and H is the number of holes in the image. Euler number is very useful in providing a topological description of an object. Since in this thesis we deal with rough set approximations, where an object is represented by its approximations, we

partly modify this definition of Euler number.

During the construction of the rough-set cover, we notice that some characters have no void or holes inside them, while the covers of the others contain contain void inside them. Fig 3.3 shows sample characters containing two, one, and no void enclosed inside them. If the void is big enough, then we have yet another hole polygon inscribed within the void. A few character even consists of more than one void inside them. Table 3.2 shows the number of voids contained by each character.



voids(B)=2 voids(C)=0 voids(O)=1

Figure 3.3: Illustration of voids.

Table 3.2: Information table for the number of voids contained by each character.

No. of voids	Characters	Numbers
0	CEFGHIJKLMNSTUVWXYZfhnrt	12357
1	ADOPQRabdeq	0469
2	Bg	8

Hence, we see that the upper-approximation $\overline{\mathcal{P}}_{\mathbb{G}}(S)$ can inscribe multiple polygons inside them. The largest of these polygons act as the *outer cover* and the rest as the *hole cover*. The outer cover tightly circumscribes the object S. Similarly, we notice that the hole polygons tightly inscribe the void or the cavity. To capture this information, we consider *approximate Euler number* (EN) as a discriminator.

Definition 3.2 (Approximate Euler number)**.** *The approximate Euler number of an object S imposed on a grid $\mathbb{G}$ is defined as*

$$EN = 2 - n$$

where n is the number of rough-set polygons in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$.

Hence, an object having two and three polygons will have an Euler number 0 and -1 respectively, since they have the value of n as 2 and 3, in order. Fig 3.4 shows the characters `A` and `B`. As we can see from the diagram, `A` has 2 polygons and hence has an EN 0, while `B` has EN -1, as it has $n = 3$, thus discriminating them. Fig 3.5 shows how different complex instances of the English optical character `B` get associated with the same value of (approximate) Euler number when its rough-set cover is considered. This is not feasible by a usual image analysis, wherein lies the importance of rough set. Without rough-set interpretation, the instance of `B` would have EN $= 1$ by conventional image processing, which would produce erroneous result in subsequent analysis. We classify the alphanumeric characters with their Euler number values as shown in Table. 3.3.

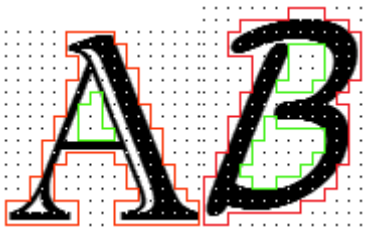


Figure 3.4: Illustration of approximate Euler number.

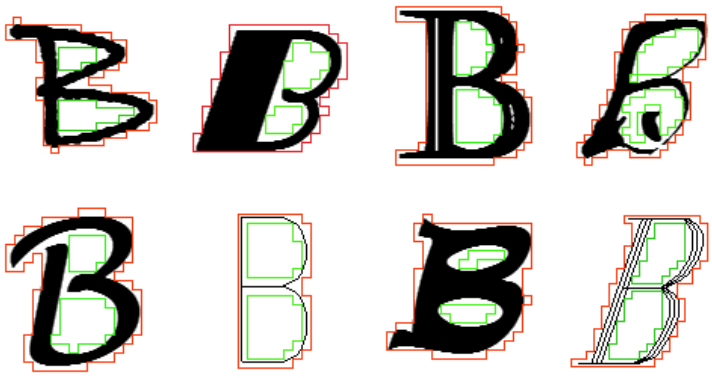


Figure 3.5: Different instances of `B` where (approximate) Euler number remains invariant as a semi-reduct attribute (red = outer polygon, green = hole polygons).

After finding the Euler number, we see that the number of characters in an equivalence class is quite large. This happens because EN has just three values: 1, 0, -1. We observe here from the

Table 3.3: Information table for the approximate Euler number of each character.

Approximate Euler number	Characters	Numbers
1	CEFGHIJKLMNSTUVWXYZfhnrt	12357
0	ADOPQRabdeq	0469
-1	Bg	8

geometric structure of the character, that different characters have different number of 'U-turns' (formally defined in Sec. 3.1.2.3) in its constructed polygon. Hence, we study these 'U-turns' in detail in the next subsection.

3.1.2.3 Directional Changes

Construction of outer covers of the characters involve 4 directions, as explained in Sec. 2.2, which are 0, 1, 2, and 3 (rightward, upward, leftward, and downward). Here, we consider the directions 1 and 3 (upward and downward) as the vertical directions, and 0 and 2 (rightward and leftward) as the horizontal directions. We introduce the following definition as a new rough-set attribute.

Definition 3.3 (U-turns)**.** *The number of times we encounter a change in the vertical direction or the horizontal direction is known as its U-turn count. In a U-turn, two consecutive vertices are of type* $\langle +1, +1\rangle$ *or* $\langle -1, -1\rangle$.

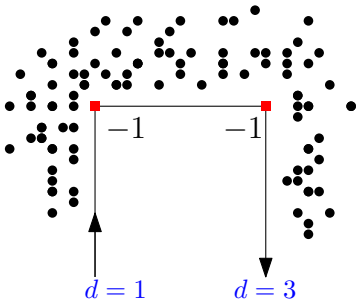


Figure 3.6: Illustration of a U-turn (VDC).

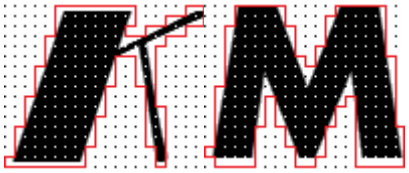
Fig 3.6 illustrates a U-turn in vertical direction. We notice that a few characters like D and O have less U-turn compared to characters like M and W.

A U-turn gives rise to a vertical direction change (VDC) when the polygon encounters a direction change from '1' to '3' or from '3' to '1'. Similarly, a character is said to undergo a horizontal

direction change (HDC) when the polygon encounters a direction change from '2' to '0' or '0' to '2'. Table 3.4 lists the number of vertical direction changes occurring in a character when we start the traversal from the top-left vertex.

Table 3.4: Information table for the VDC of each character.

VDC	Characters	Numbers
2	BDEFILOPQTZbdegq	01478
4	AJRUVXYafhrt	26
6	CGHKNSn	35
8	MW	



VDC(K)=6 VDC(M)=8

Figure 3.7: Instance of K and M for illustrating the VDC attribute.

Fig 3.7 shows how two characters are discriminated by the vertical direction change attribute, where we see that VDC(K)=6 and VDC(M)=8.

For some characters like p, q, b, etc., with different fonts, the curvatures result in different values of the VDC count. Hence, we keep a noise margin in computing the VDC count. On experimentation, it was found that approximately a directional change less than 10% the height of a character, were due to noise. So, if a directional change length is less than this amount, then we do not consider it as a valid directional change.

Another similar measure to that of the vertical direction change is horizontal direction change (HDC). The HDC is calculated along the horizontal direction, and the start point of the traversal is taken as the left-top vertex in this case. However, with experimentation, it was not found to be a strong discriminating attribute as VDC, as it varies a lot with different font styles. Hence, it is removed from the semi-reduct. However, HDC is used as an attribute of the semi-reduct in the algorithm for logo retrieval and character spotting (Chapters 4 and 5).

With the Euler number attribute followed by the VDC parameter, we get a finer classification, as shown in Fig 3.8. After breaking down the alphanumerics with the Euler number and the VDC

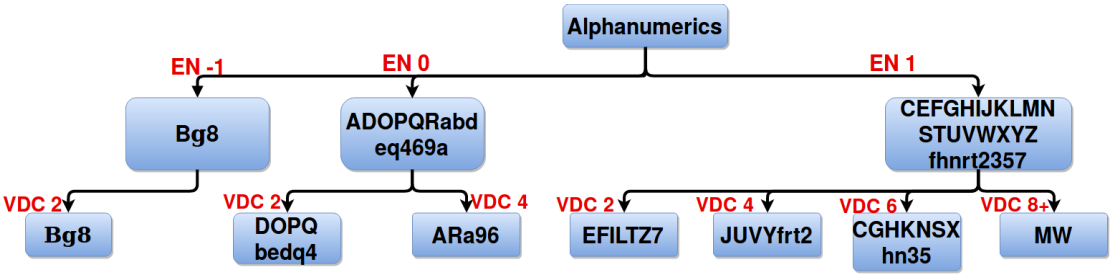


Figure 3.8: Classification using two attributes— Euler number and VDC.

attribute, a careful study of the sub-classes created show that just a few characters belong to the same class. `M` and `W` have the same geometric structure but just an inversion of one another. To preserve this characteristic, we study various characters having similar structures and find the concavity attribute as a strong discriminating feature. The concavity attribute is explained in section 3.1.2.4.

3.1.2.4 Concavity Tuple

Concavity serves as an important characteristic for the determination of any shape, also shown in [**?**]. It is formally defined as follows.

Definition 3.4 (Concavity)**.** *A concavity exists at the point of occurrence of two consecutive Type* $\langle -1, -1 \rangle$ *vertices in* $\overline{\mathcal{P}}_{\mathbb{G}}(S)$.

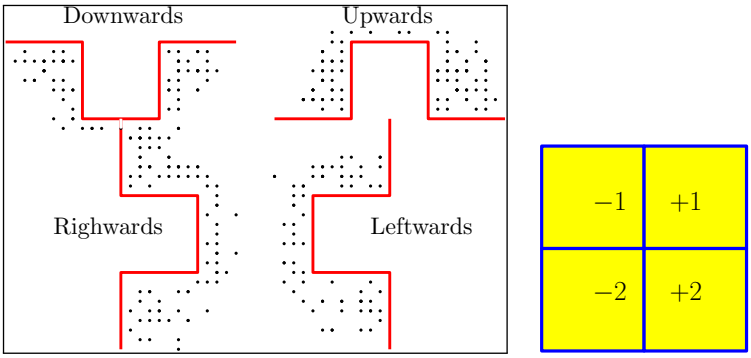
If we have more than two Type -1 vertices, then we get a nested concavity. We define a concavity in 3-tuple format, comprising of ⟨concavity direction, region, depth⟩.

1. Orientation

The first component of the concavity 3-tuple is its direction. A concavity can be assigned with the direction, based on its orientation. We classify the concavity direction into four types, namely, upward (U), downward (D), leftward (L), and rightward (R) concavity. Hence, we use concavity as an important attribute with much distinguishing capacity. Fig 3.9a illustrates four types of concavities.

2. Position

After getting the concavity, it is important to know the (approximate) position of the concavity in the character. This serves as the second component of the concavity 3-tuple. To find the position



(a) Concavity orientations (b) Concavity positions

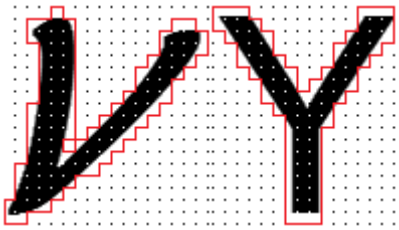
Figure 3.9: Attributes of Concavity 3-tuple.

of the concavity, we use the bounding box of each character along with its centroid. Considering this centroid as the reference point, we divide the bounding box into four parts. We assign $+$ and $-$ for the right and the left halves, and 1 and 2 for the upper and the lower halves, respectively. Hence, we get four possible positions of concavity: $+1$, $+2$, -1, and -2. Fig 3.9b shows the four different positions for the concavity attribute.

3. Depth

For differentiating one character from another, like `U`, `V`, and `Y`, we notice that the only primary difference between them is their depth of concavity. They all share the same direction of concavity and similar positions over a wide variety of fonts. Hence, to distinguish such characters, we use the concavity attribute. To use this attribute as a rough-theoretic measure, we discretize the relative depth of each concavity to the nearest value in $\{1, 2, 3\}$. In Fig 3.10, the characters `V` and `Y` have similar concavities (i.e., Upward), but have respective depths 2 and 1, and hence get discriminated.

Once we find characters with hole polygons inside them, after breaking down the class with the VDC and the concavity attribute, we use the positions of these holes inside the outer polygon to distinguish the characters. This is explained in Sec. 3.1.2.5.



(U,+1, 2) (U,+1, 1)

Figure 3.10: Illustration of Concavity 3-tuple.

3.1.2.5 Hole Positions

Once we get the Euler number, knowing the position of the hole polygon can be informative. The position of the hole polygon with respect to the outer polygon gives us more ability to distinguish between the characters. For example, the character `d` will have the centroid of the hole lying towards the left side of the top-left vertex of the outer cover, while the character `b` will have its hole centroid towards the right side of its top-left vertex.

Here again, we consider the centroid of the outer polygon as the reference point, and just like the concavity position, we define the position of the holes. The centroid of the outer polygon in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$ is compared with that of the centroid of the hole polygon in $\underline{\mathcal{P}}_{\mathbb{G}}(S)$. If the hole centroid, lies towards the left or right lateral half of the outer polygon centroid, then we denote it by $-$ or $+$, respectively. If the hole centroid lies towards the upper or the lower half region, we assign 1 or 2, respectively. Hence again, similar to the position obtained in the concavity attribute, we obtain the position of hole attribute. In Fig 3.11 and Table 3.5, we see how the characters `b` and `d` are differentiated by this attribute. Fig 3.12 shows the classification using these attributes up to this phase.

PoH $= +2$ PoH $= -2$

Figure 3.11: Illustration of position of holes.

Table 3.5: Attribute information table

	EN	PoH	VDC	Concavity
b	0	+2	8	-
d	0	−2	8	-

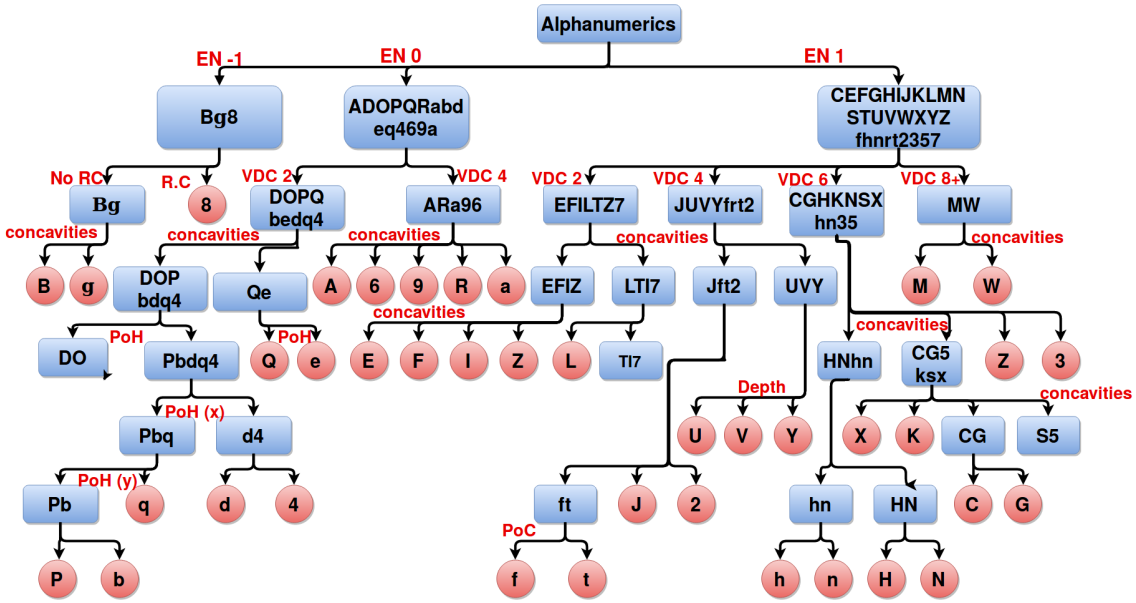


Figure 3.12: Classification using the attributes— Euler number, VDC, PoH, and concavity.

3.1.2.6 Edge ratio

Studying the level of classification obtained up to this stage, we notice that still a few characters like `D` and `0`, or `T` and `I`, are not discriminated. Carefully studying these classes result into the observation that the ratio of the vertical perimeter to the horizontal perimeter is different in these cases. To invoke this characteristic as an attribute, we define the following concepts:

Definition 3.5 (Vertical Perimeter Component (VPC))**.** *For each polygon in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$, we define vertical perimeter component as the sum of the lengths of its vertical edges.*

Definition 3.6 (Horizontal Perimeter Component (HPC))**.** *For each polygon in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$, we define horizontal perimeter component as the sum of the lengths of its horizontal edges.*

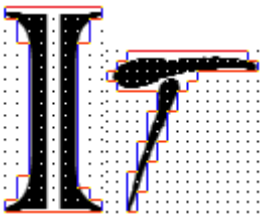
Definition 3.7 (Edge Ratio (ER))**.** *The ratio of VPC to HPC, discretized to the nearest value in*

Table 3.6: Attribute information table for the characters I and T.

	EN	PoH	VDC	Concavity	ER
I	+1	-	2	-	2
T	+1	-	2	-	1

$\left\{\frac{1}{2}, 1, 2\right\}$, *is called edge ratio (ER).*

The attribute ER is used for having a broad sense of idea of the overall geometric stretch of each glyph symbols. Fig 3.13 shows an example. Table 3.6 shows the differentiation of two characters T and I using this feature.



ER = 2 ER = 1

Figure 3.13: Illustration of edge ratio (red lines show the vertical perimeters and the blue lines show the horizontal perimeters).

With the above properties, we construct a character recognition pipeline, shown in Fig 3.14. In the initial stages of the pipeline, the average number of objects per equivalence class is more. This class gradually gets smaller in size down the pipeline until we reach the bottom, where each character is classified distinctly. We see that all the discriminations are made possible with these attributes. Using these attributes, we redesign a semi-reduct for the alphanumeric set (shown in partial in Table 3.7). We see that the characters are well-discernible amongst each other by this semi-reduct design, and this plays a key decisive role in the recognition part of our OCR model.
By varying the grid size, the attributes also vary. Keeping a higher grid size above optimum results in the loss of many information. All the polygons do not get detected, and many of the attributes also loose its form due to this coarser resolution. On reducing the grid size, while the finer details of the objects are preserved, many noise parameters are also detected along with the characters. Also, sometimes, too fine resolution of the grid results into spurious concavities (discussed already in Sec. 3.1.2.4). Hence, we see that, increasing too much or decreasing too much of the value of

grid size may produce incorrect results. Hence, an optimized grid size (found using Algorithm 2) is used.

Table 3.7: Sample partial information table containing the object properties against the semi-reduct.

Characters	EN	PoH	VDC	Concavity	ER
B	-1	$+1, +2$	2	(L,$+1, -$)	2
E	$+1$	-	2	(L,$+1, -$),(L,$+2, -$)	$\frac{1}{2}$
I	$+1$	-	2	-	2
M	$+1$	-	8	(D,$-2, -$),(D,$+2, -$),(U,$+1, -$)	1
T	$+1$	-	2	-	1
V	$+1$	-	8	(U,$+1, 2$)	2
Y	$+1$	-	8	(U,$+1, 1$)	2
b	0	$+2$	8	-	2
d	0	-2	8	-	2
3	$+1$	-	8	(R,$+1, -$),(R,$+2, -$)	1

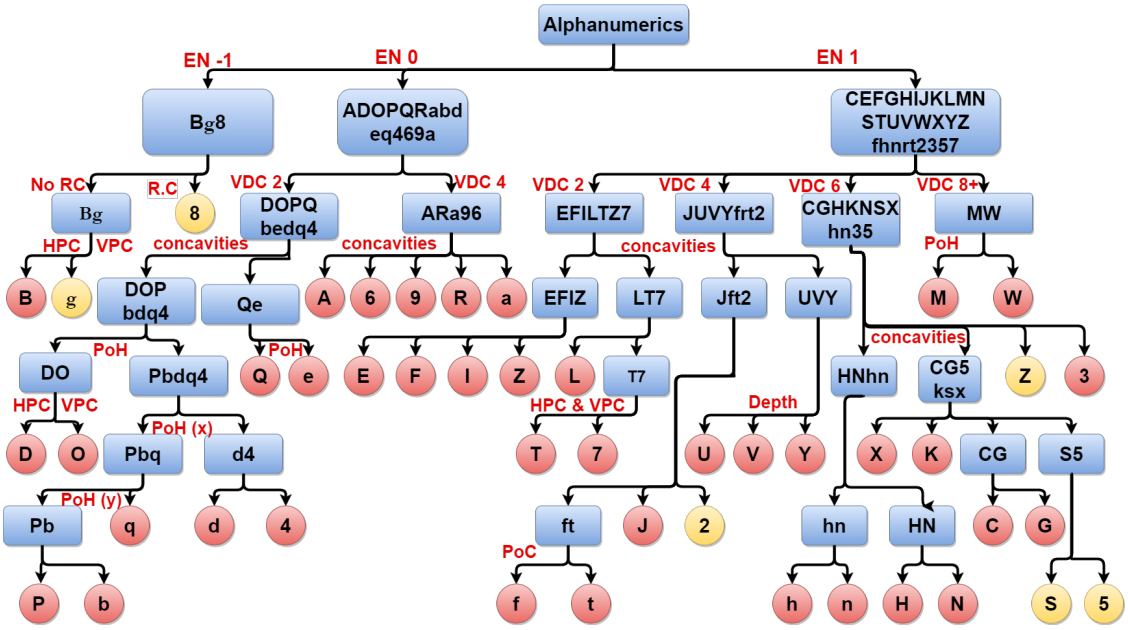


Figure 3.14: Pipeline of the Tesseract™ reinforced hybrid model.

3.1.3 Reinforcing Tesseract™ OCR

After implementing the OCR engine with all the attributes, it has been noticed that a few character classes, cannot still be distinguished from one another over varied font styles. These are referred to as the indefinite class characters. The characters belonging to an indefinite class have close resemblance with each other. When these indefinite class characters are encountered, the proposed model uses a Tesseract™ reinforcement for completion of the process of recognition. The indefinite class members are listed in Table 3.8.

Table 3.8: Indefinite class characters.

Character 1	Character 2	Character 3
Z	2	
S	5	
g	8	9

3.2 Experimental Results

For testing our OCR system, we checked various datasets and finally chose the `Chars74k` dataset [**?**], [**?**] to report our test results. The dataset was selected for its challenging scripting styles and various atypical or idiosyncratic font styles. This dataset contains 26 upper-case characters, 26 lower-case characters, and ten numeric digits in English. There are 1016 different font styles, including the bold and italicized fonts too. Each image has a resolution of 128×128 pixels. Fig 3.15 shows a sample set of the character `G` and its various instances. Since all the samples are of the same size, we use a constant grid size of 6 throughout our experiments.

We run our OCR engine on a platform of 64-bit Intel®, 2-Core™, i5 processor, with 4GB RAM, DELL machine. The average CPU runtime was found to be 0.051seconds for the recognition of a character, using our OCR engine. With a test on the same dataset, using the Google Tesseract™ engine [**?**], we got an average runtime of 0.203 seconds per character. Thus, we see that the designed OCR engine is computationally highly attractive. Table 3.9 gives the CPU time for both the engines.

After conducting the experiments, we get a result of 88.98% accuracy using our model. The Google Tesseract™, version 3.02.02, gives 64.79% with the standard `eng.trainneddata` train-

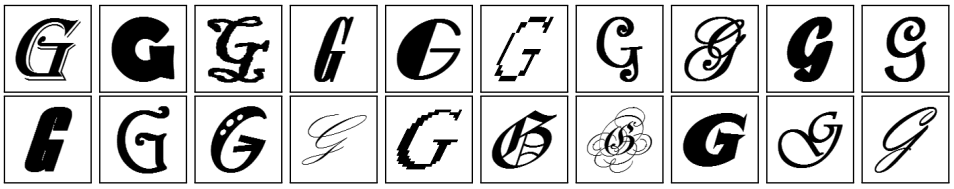


Figure 3.15: Samples of the character G from the Chars74k dataset.

ing set. The overall accuracy and the detailed accuracy of these models are shown in Table 3.10 and Table 3.11.

As we mentioned in Sec. 3.1.2.1, we are recognizing individual characters. The classification is a context-free model, and we assign certain characters with the same class label, like many of the upper-case and lower-case characters, such as C and c, since they are not mutually discernible. Also, other than these characters, there are some more characters which bear extremely close resemblance with each other over different font styles. These are the classes (Z/2), (S/5), and (g/8/9). We refer to these three classes as the *indefinite classes*. When the font style becomes complex, like the ones shown in Fig 3.19, we might get erroneous results, owing to the erratic mapping of the attribute values in the rough-set theoretic domain. With a larger dataset and a more detailed study and minute observation of the differences in these characters, the discernibility of these indefinite classes can be targeted. As shown in Fig 3.2, we reinforce our model with the Tesseract™ for these indefinite characters, and design a hybrid model, to achieve a higher accuracy and a faster OCR engine. Fig 3.14 shows the classification using the hybrid model. The yellow nodes are the ones that are classified using Tesseract™ reinforcement.

Figs 3.16 - Fig 3.19 show the experimental results in various cases. Fig 3.16 shows the case when the semi-reduct and Tesseract™ are both independently successful. Fig 3.17 shows the cases where the semi-reduct is successful, while the Tesseract™ fails. The cases where the semi-reduct combined with Tesseract™ is successful is shown in Fig 3.18. Both the methods fail in cases depicted in Fig 3.19.

3.3 Summary

In this chapter, we have shown how different geometric and topological attributes are defined in the spatial domain of rough set discretized values. A semi-reduct was designed for the classification

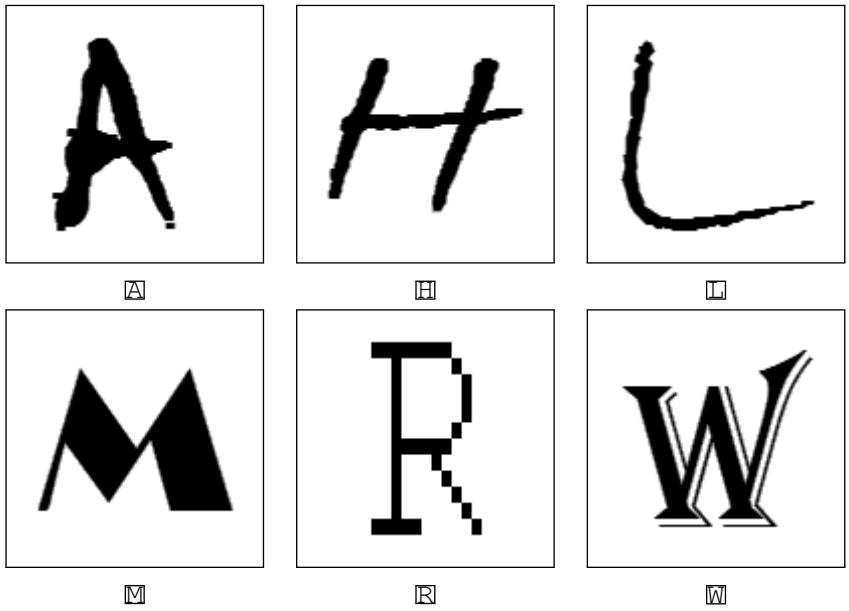


Figure 3.16: Reduct and Tesseract™ are independently successful.

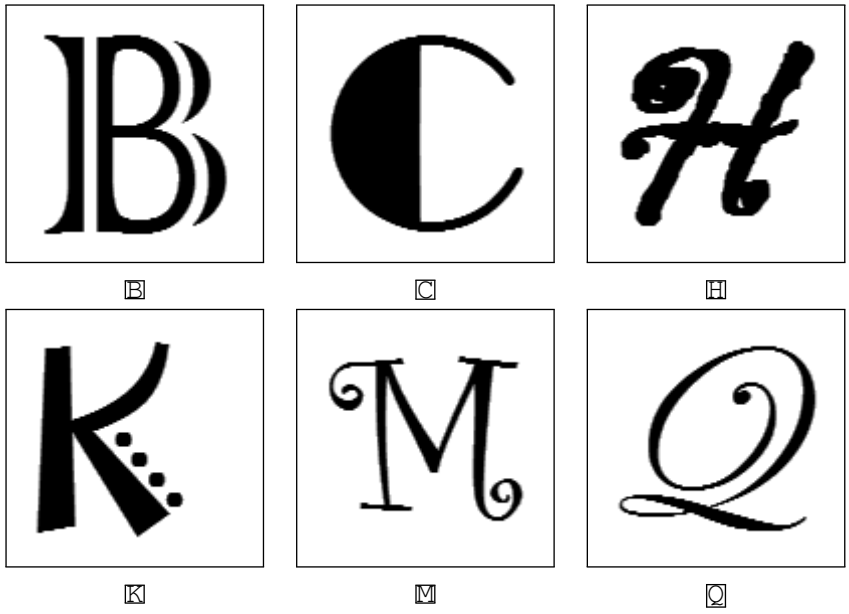


Figure 3.17: Reduct is successful; Tesseract™ fails.

of alphanumeric characters using these attributes. A rough-set model with a small-cardinality semi-reduct is found to be useful for a quick-yet-efficient discernibility of optical characters over

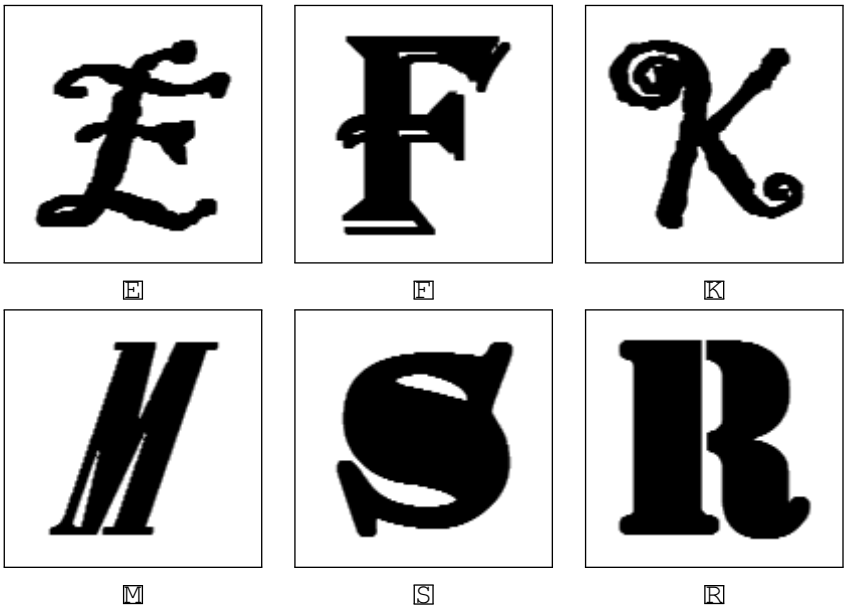


Figure 3.18: Reduct combined with Tesseract™ is successful.

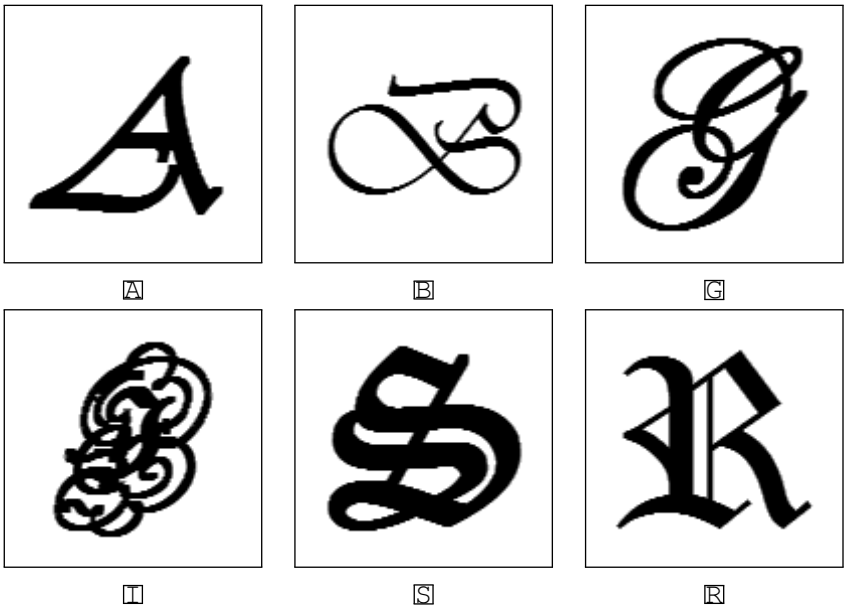


Figure 3.19: Both, the proposed method and the Tesseract method fail to recognize these characters.

a varied font style. Many a time, even an efficient algorithm may fail for character classification,

Table 3.9: Summary of experimental results.

Model	Runtime	Accuracy
Google Tesseract™ 3.02.02	0.203 secs	64.79%
Proposed Model	0.051 secs	78.29%
Hybrid model	0.079 sec	88.98%

Table 3.10: Comparison by accuracy

Rough set	Letters	Tesseract™
Above 90%	CEIJKLMSVXYZf83	75.49 - 90.84%
80 - 90%	ABDFHNOPQRTUW	4.90 - 87.00%
70 - 80%	Gbdem247	5.01 - 80.70%
60 - 70%	anqrt56	0.78 - 52.85%
50 - 60%	gh9	1.08 - 60.33%
Average 88.98%	-	64.79%

since it cannot accommodate the effect of noise to a large extent. The presence of noise is not always fully recognizable by a usual image analysis, wherein lies the importance of rough set. Without rough-set interpretation, it would be difficult to estimate a few attributes by conventional image processing, which would produce erroneous result in subsequent analysis. A multi-grid representation leading to multiple descriptors can also possibly be used to improve the recognition accuracy.

The proposed algorithm has a significant operational difference with the existing techniques, has a lower computational time, and can be designed to an efficient OCR with less runtime. The semi-reduct attributes used in our model are found to have strong discriminating power and can extend the concept further for OCR design in scripts other than English. It can also be used for efficient paragraph segmentation, word segmentation, and character segmentation from a PDF document. Additional attributes can be explored and tested in different combination with these attributes to downsize a suboptimal semi-reduct to an optimal reduct, especially when a script has a large alphabet size.

Table 3.11: Performance of the three engines in terms of accuracy over different characters.

Letter	Tesseract™	Rough-set Features	Tesseract™ reinforced Model (The dataset corresponds to upper case characters only)
A	82.57	79.52	86.81
B	4.90	85.53	86.51
C	83.90	90.25	93.20
D	75.59	85.82	87.89
E	87.00	54.92	90.64
F	86.70	58.17	89.96
G	80.70	55.51	77.06
H	80.20	79.82	86.22
I	79.20	93.30	93.89
J	81.00	91.73	92.70
K	82.97	83.60	91.53
L	82.87	90.64	91.43
M	76.77	94.09	94.29
N	77.90	89.46	89.46
O	55.90	84.44	84.54
P	78.05	85.33	85.33
Q	68.20	82.48	82.57
R	66.83	87.50	87.69
S	75.49	90.35	92.12
T	83.26	52.46	87.89
U	81.10	79.52	85.33
V	82.77	79.03	90.25
W	86.22	77.06	88.58
X	88.18	84.84	93.79
Y	83.75	85.72	90.15
Z	90.84	80.90	93.50
a	24.11	62.20	-
b	66.92	76.87	-
d	54.03	71.45	-
e	5.01	75.88	-
f	77.50	94.38	-
g	1.08	58.56	-
h	60.33	56.59	-
m	32.97	73.22	-
n	13.38	68.79	-
q	12.00	66.24	-
r	0.78	62.20	-
t	52.85	62.20	-
Average	64.79%	78.29%	88.98%

Chapter 4

Retrieval of Logos using Rough-Set Attributes

In official or private communication, a document may contain various types of information, such as, logos, official stamps, texts and graphics. A logo bears the identity of an organization in a paper or fully digital document. Organizations have their own personalized and trademarked logos. Searching for similar logos in the registered logo database is a very important and tedious task at the trademark office. Since logos are important parts of a document, automatic logo retrieval can be used in indexing and archiving documents. Fig 4.1 shows a sample set of few logos.

In the modern world of digitization, organizations all over the world register their logos in digital formats. With a boom in the number of start-up companies emerging every year, designing a new logo has become a necessity. Whenever a new company comes up in the market, they want to get their logo registered in the trademark office. These companies have to go through a list of previously registered logo-database by the existing companies. It is required to check this database to see whether the new logo bears any resemblance with existing logos, to get an approval. Normally this entire process is carried out manually, which is quite a tedious method. In case a logo resembles even partially with another, that may lead to a complex litigation, a costly and time-consuming event. As the data set for the trademarked logo increases, it is needed to have a faster and automated method, which can provide similar logos to that of the chosen one. Speed and accuracy are two aspects that one must attend to while developing a system for retrieval of logos. This chapter contains a proposed method for a fast and efficient retrieval of the logos.

Logo retrieval is still a major field of research in document image processing till date. Most existing techniques use CBIR (Content Based Image Retrieval) for the retrieval of images based on their features like color, texture, shape, bag of visual words (BoVW), spatial orientation, etc. Few examples involving this method are reported in [**? ? ?**]. Even after all these years of research, still no fixed, best set of features have been found so as to have uniformly good results on varied

data sets. Also, CBIR has two major problems, which are difficult to be resolved [**?**]. First, segmentation of an image to break into meaningful parts based on the above low-level features is still a challenge. Then, there is a huge gap between the feature descriptors and the semantic expressions embedded in an image.

Most of the earlier works on logo retrieval were based on shape features. Some methods like [**?**] have used edge-based shape recovery method. One problem with this method is its high sensitivity to noise, which causes a fall in its retrieval. Also, when there are multiple disjoint components, this method does not take into account their spatial relations. This problem is also seen in [**?**], which uses edge based features for developing a CBIR method. The method proposed by Bober [**?**] extracts the shape features at multiple scales. But it lacks the description of structural information of edges. The method reported in Rajashekhar et. al. [**?**] uses morphological operations to get the features of each logo image. While this method gives good retrieval results, its efficiency is not high, and it is computationally expensive. The rough set based method, being an approximate method, takes care of noise to a large extent, while preserving the spatial relations among various components. Also, it preserves the structural information about the edges, and provides an efficient retrieval system. In this work, we have provided a comparative study with these methods ([**? ? ? ?**]).

The proposed algorithm uses a rough-set technique to quantify the structural information in a logo image that can be used to efficiently index an image. A logo is split into a number of polygons, and for each polygon the tight upper and lower approximations are computed based on the principles of rough set. This representation is used for forming feature vectors for the retrieval of logos.

This work makes the following novel contributions:

- An unsupervised algorithm for the retrieval of logos.
- An efficient retrieval mechanism using inverse Hough transform.

4.1 Methodology

An overview of the proposed approach is shown in Fig 4.2. The logo database chosen for building this system consists of already binarized logos. For construction of the semi-reduct, some attributes designed for the OCR engine are used (Chapter 3) along with some more customized at-

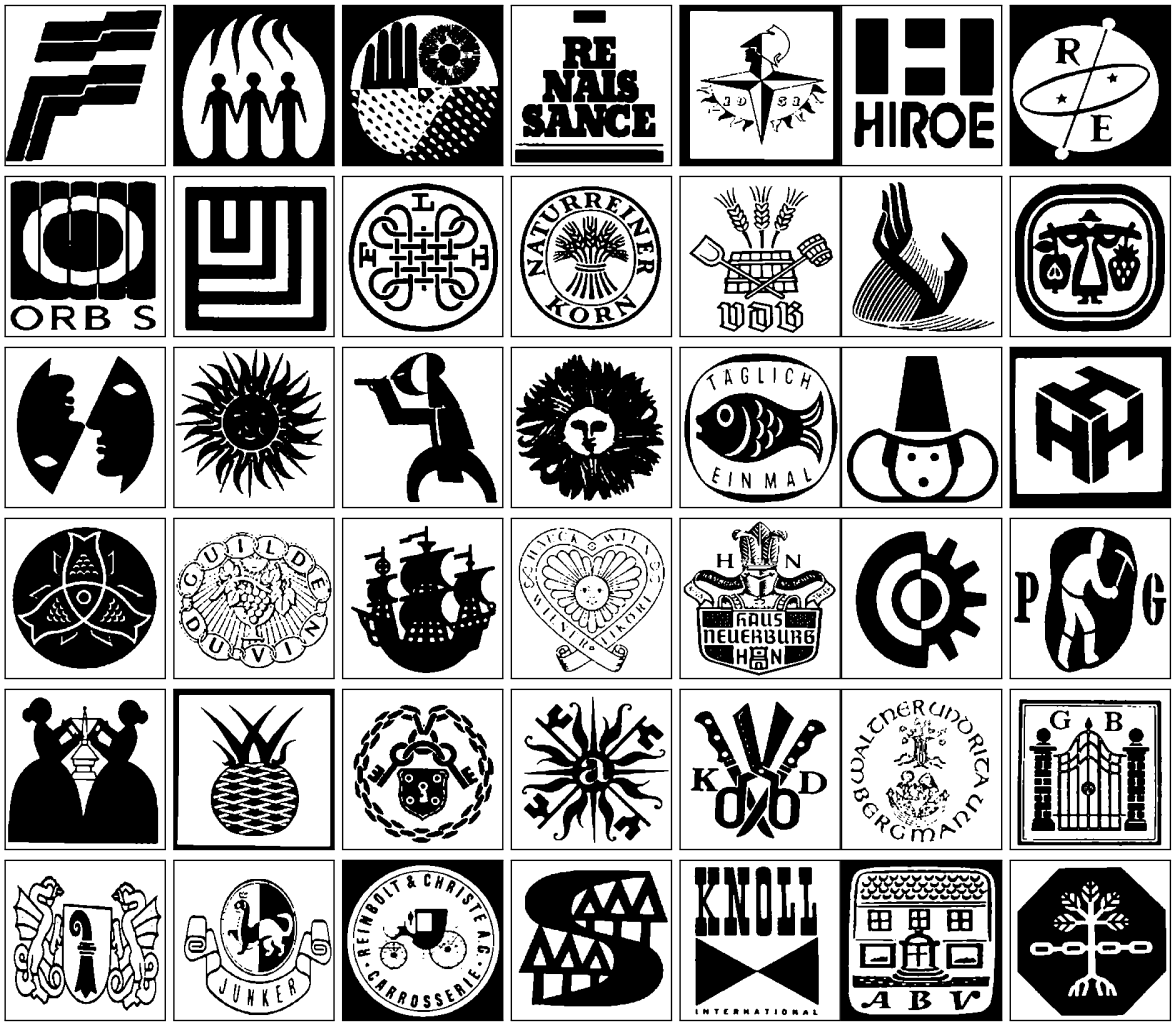


Figure 4.1: A typical set of logo images.

tributes. The database consists of N classes of logos. Each logo class may have multiple instances of digital images. The rough-set polygons comprise outer polygons and hole polygons. From these polygons, certain attributes are designed and used for semi-reduct construction. A database is then constructed for indexing purposes, containing the detailed attributes of each image.

For the retrieval part, a query image is taken and the rough-set boundaries of this logo image are computed. From these boundaries, a feature vector is created from the designed attributes. The attributes generated are then matched with the database, and the top K matches are found. The matching is done by inverse Hough-transform [**?**], and the searching from the database is done by

k-d tree [**?**]. The entire algorithm and its working are explained in the subsequent sections.

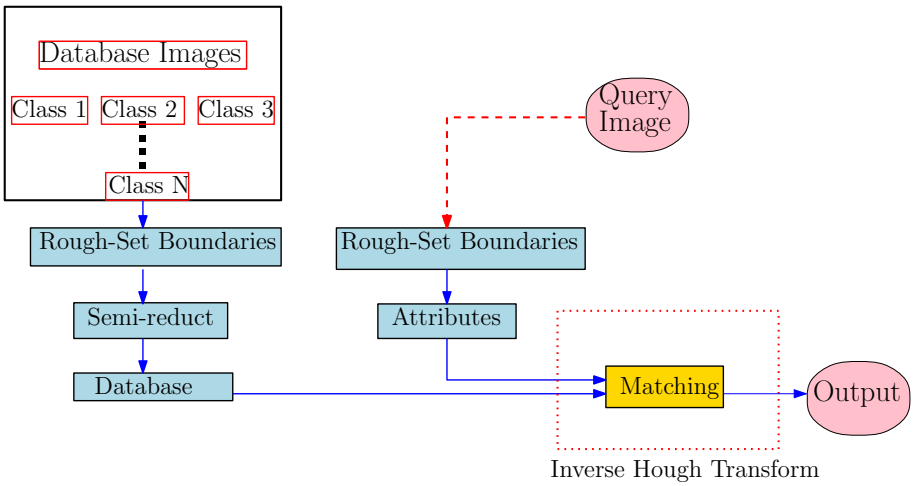


Figure 4.2: Block diagram of the overall working of the proposed algorithm.

Figure 4.3: Connected components in a logo image.

Fig 4.3 shows a logo image, containing 5 connected components, each shown in a different color. Evaluation of the retrieval system is slightly different from that of character recognition model, as the logos may contain multiple components in an image, and the polygon of each component has its own containment relation with another polygon. The containment relations and the other attributes used in this algorithm are explained in detail in the following section.

4.2 Semi-Reduct Attributes for Retrieval of Logos

Usually a logo image contains multiple connected components. Each individual component is separately considered in our algorithm. Segmentation of the components is done by creating tight upper approximation and tight lower approximation, $\overline{\mathcal{P}}_{\mathbb{G}}(S)$ and $\underline{\mathcal{P}}_{\mathbb{G}}(S)$. After creating these approximations, it is noticed that there are many components that are in turn enclosed in other components. In the retrieval of logo process, it is very important to store the information on containment relations. The attributes used are explained below.

4.2.1 Distinguishing Outer Polygon and Hole Polygon

On studying the various rough-set polygons obtained after the implementation of the rough-set covers of the components, two types of tight upper approximations are found:

- **Outer polygon:** The covers which inscribe the objects inside them.
- **Hole polygon:** The covers that are inscribed inside the voids in the objects.

Fig 4.4 shows examples of hole and outer polygons. Construction of these polygons are done

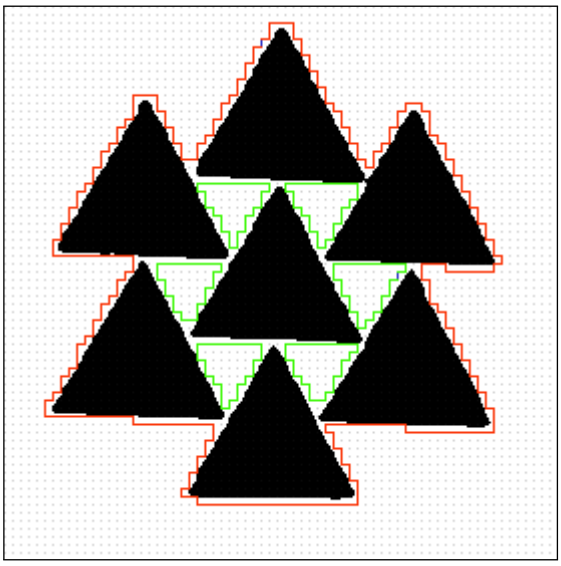


Figure 4.4: Hole polygons (shown in green) and outer polygons (red).

using the same algorithm, and so it is difficult to categorize each polygon as a hole polygon or an

outer polygon during construction. Hence, we provide a more specific definition of outer polygon as follows.

Definition 4.1 (Outer polygon)**.** An outer polygon *is defined as that polygon where the number of Type 1 vertices is four more than the number of Type* -1 *vertices.*

Similarly, a hole polygon can be defined as follows.

Definition 4.2 (Hole polygon)**.** *A* hole polygon *is that polygon where the number of Type 1 vertices is four less than the Type* -1 *vertices.*

Using this property, each polygon is labeled either as a hole polygon or as an outer polygon. Usually an outer polygon conntains one or more hole polygons inscribed inside them. A hole polygon, in turn, may enclose one or more outer polygons of separate components. The polygons and their interrelation can provide useful information in a logo image. We register initially the following attributes:

1. Number of outer polygons.
2. Number of hole polygons.

Fig 4.5 shows the values of these parameters for a sample logo image. In the given figure, the number of polygons is 7, the number of outer polygons is 2 and the number of hole polygons is 5. Here, polygon number '1' and '3' are the outer polygons, and '2', '4', '5', '6', and '7' are the hole polygons.
After getting the number of outer and hole polygons, it is noticed that for many logo images, the absolute number of polygons is not a good measure. Fig 4.6 shows some sample logos which display this problem. Many logos contain noisy polygons, which lead to an abrupt increase in the number. A similar instance of the logo with less noise would have a drastically different number of polygons. Hence, we have considered the number of "major" polygons in a logo image. This is explained in the next sub-section.

4.2.2 Number of Polygons (TP) and Major Polygons (MP)

The number of polygons for some cases is not a good measure (as evident from Fig 4.6). It is observed that the number of polygons for small and closely lying components can vary over different

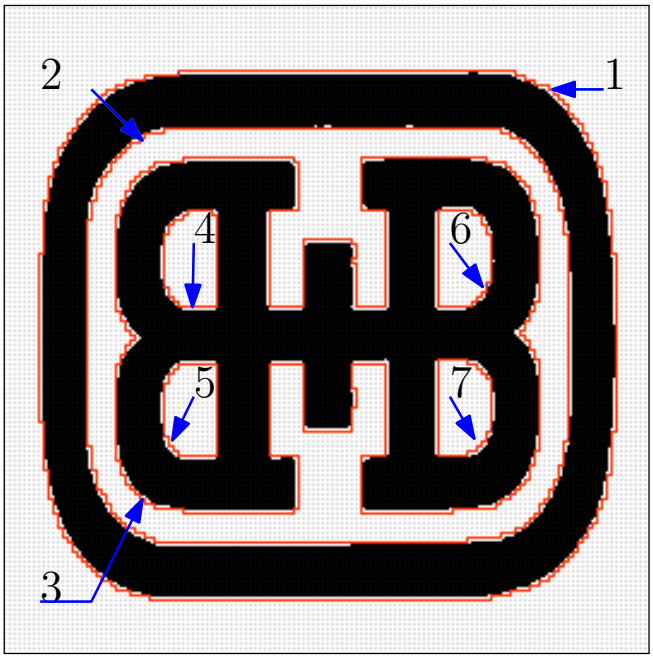


Figure 4.5: Number of polygons = 7; Number of outer polygons = 2; Number of hole polygons = 5.

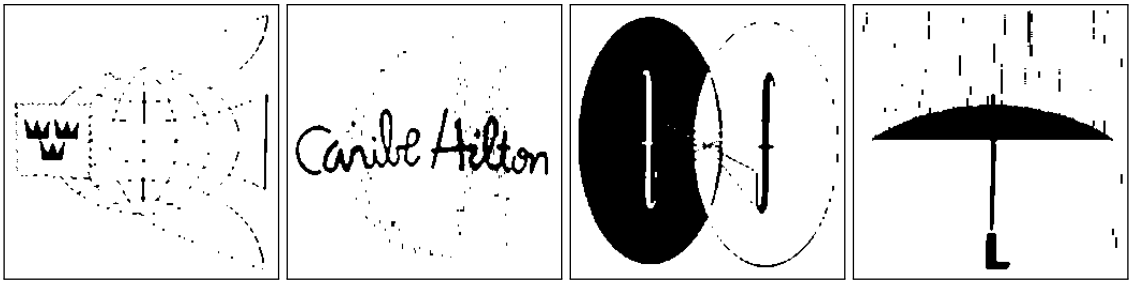


Figure 4.6: Uncertain number of polygons (presence of noise components).

image orientations. These polygons occur randomly and might vary from one logo instance to another. Hence, due to the randomness of their occurrence, we refer to these polygons as *random spurious polygons*. Changing the image orientation, grid size, thresholding errors, etc., can affect the number of such polygons. Hence, the number of *major polygons* is used as an attribute in the proposed method.

To identify the major polygons, the mean (M) of the perimeters (m_i) of all the outer polygons are found, using the equation 4.1. After finding the mean of the perimeters, the total deviation (D)

of the perimeter of the polygons is found out (using equation 4.2). From the total deviation, the individual deviations are subtracted (refer equation 4.3). If this measure is positive, we define the corresponding polygon to be a major one and if the result is found to be negative, the polygon is not a major one, and it is filtered out. Fig 4.6 shows two cases wherein, due to the high variance of size of the smaller outer polygons to that of the bigger polygons, they are ignored. Fig 4.7a contains 35 total polygons (TP), while only 10 major polygons. Fig 4.7b contains 41 total polygons, but only 10 major polygons.

$$M = \frac{1}{n}\sum_{i=1}^{n} m_i \tag{4.1}$$

$$D = \frac{1}{n}\sum_{i=1}^{n} |m_i - M| \tag{4.2}$$

$$D_i = m_i - D \tag{4.3}$$

(a) TP=35 & MP=10 (b) TP=41 & MP=10

Figure 4.7: Number of polygons and major polygons; Only major polygon count (MP) is taken as an attribute.

After getting the number of major polygons, it is noticed that containment relationships of polygons are important in logo images. The containment relationships can be applied on the hole polygons and the outer polygons differently. The following subsections elaborates the containment relationships of holes and outer polygons.

4.2.3 Hole Containment (HC)

Each hole polygon is enclosed within an outer polygon, since a hole polygon circumscribes a void. If each polygon is provided with an identification number, it is possible to encode their individual containment relationships. Each hole polygon is assigned with a containment number, which contains the identification number of the nearest outer polygon enclosing it.

To find the outer polygon number, all the grid points lying on the horizontal straight line, starting from the first vertex, (i_s, j_s), of the hole polygon are considered. If a grid point encounters a polygon number, which is different from its own polygon number, and is also an outer polygon, then it becomes the *parent polygon* for this hole polygon. A parent polygon is the polygon that inscribes another polygon. Once a parent polygon is found, the search along the horizontal direction is terminated to reduce unnecessary computations. All the hole polygons are bound to have a parent outer polygon. Fig 4.8 shows the algorithm of finding the hole containment. Algorithm 3 shows the steps of implementation of the hole containment relations. Fig 4.9 shows that the hole polygons '4', '5', '6', and '7' are contained within the polygon '3', while the hole polygon '2' is contained within the parent polygon '1'.

Algorithm 3: Finding the hole containment

Input: List of outer polygons and hole polygon H.
Output: Parent of H

```
Flag ←F
i ← i_s
do
    for each outer polygon P and Flag == F do
        if (i, j_s) ∈ P then
            Flag ←T
        end if
    end for
while flag == F
return P.id
```

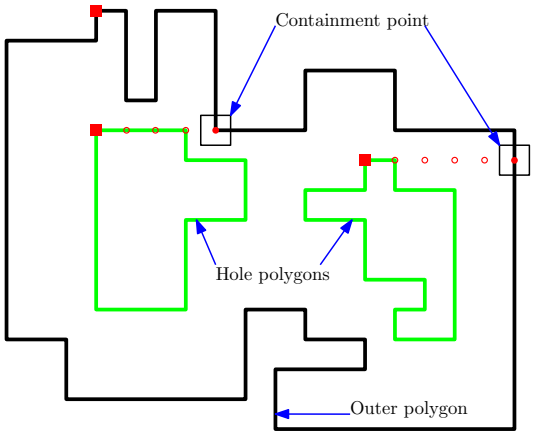


Figure 4.8: Illustration of hole containment algorithm.

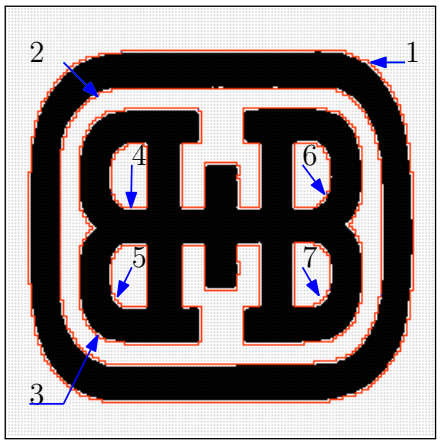


Figure 4.9: Hole containment.

4.2.4 Polygon Containment (PC)

Similar to the containment relation for holes, each outer polygon is also in turn sometimes contained in other polygons. The parent polygon of a primary polygon gives us relevant information about the relative containment of each polygon. If each polygon is provided with an identification number, it is possible to get their individual containment relationships. Each outer polygon is assigned with a containment number, which contains the identification number of the nearest outer polygon enclosing it.

To find the container polygon number, all the grid points lying on the horizontal straight line, starting from the first vertex, (i_s, j_s), of the concerned polygon are considered. A grid point is

searched, such that the polygon number of this new polygon is different. If this polygon is a hole polygon, and it is followed by another outer polygon with a separate identification number, then it becomes the parent polygon for this outer polygon. Once a parent polygon is found, the search along the horizontal direction is terminated to reduce unnecessary computations. Polygons, which are not enclosed in other parent polygon, are assigned with a random negative number, say -1, to denote that the particular polygon is its own parent polygons. Fig 4.10 shows the algorithm of finding the parent containment. Algorithm 4 shows the steps of implementation of the hole containment relations. Fig 4.9 illustrates the primary polygon containment relations. The primary polygon '3' is contained within the primary polygon '1'.

Algorithm 4: Finding the containment of an outer polygon

Input: List of outer polygons P and hole polygon H.
Output: Parent of P

```
Flag ←F
Count ←0
i ← i_s
do
    for each outer polygon P and Flag == F do
        if (i, j_s) ∈ H then
            Count= 1
        end if
        if (i, j_s) ∈ P and (Count ==1) then
            Flag ←T
        end if
    end for
while flag == F
return P.id
```

4.2.5 Approximate Euler Number

The Euler number (EN) has already been defined in Sec. 3.1.2.2. The logo images have various containment relationships, mainly for a hole polygon with an outer polygon and an outer polygon with another outer polygon. Hence, we use an approximation of Euler number in this case.

Definition 4.3 (Approximate Euler number for logos)**.** *The approximate Euler number for the*

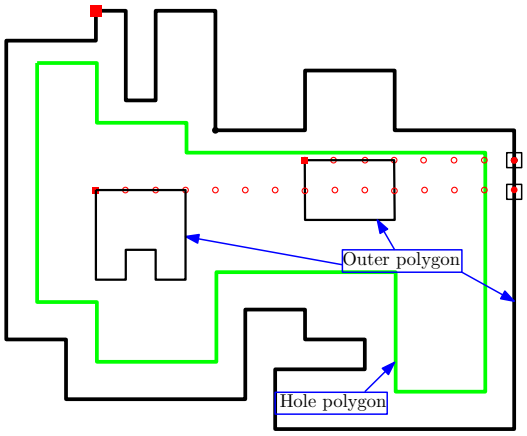


Figure 4.10: Illustration of parent containment algorithm.

logos are defined as follows:

$$EN = 1 - n \tag{4.4}$$

where n is the number of rough-set hole polygons, contained in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$.

Fig 4.11 shows a logo image, containing an outer polygon, '2'. The polygons '1', '3', '4', and '5' are the hole polygons, contained inside the outer polygon '2'. The polygon also inscribes another components, which would have its own outer polygon. But since only hole polygons are to be considered, the Euler number of the polygon '2' would remain -3 (i.e., $1 - 4$). The hole polygons do not have any Euler number associated with and we use INV (invalid) to their values for demonstration. In Fig 4.9, we have two primary polygons '1' and '3'. Here, n = 1 for polygon 1, and n = 4 for polygon 3, whereby EN = 0 and -3, respectively for them.

4.2.6 Black-to-White Ratio

It is noticed that some logos are constituted of multiple components, while some logos are plain and simple logos, consisting of minimal components. To make use of this property of logos, the following attribute is proposed.

Definition 4.4 (Black-to-White Ratio)**.** *The ratio of the number of black pixels to the number of*

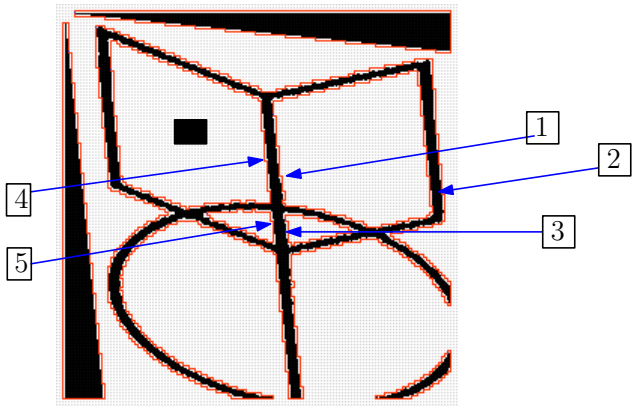
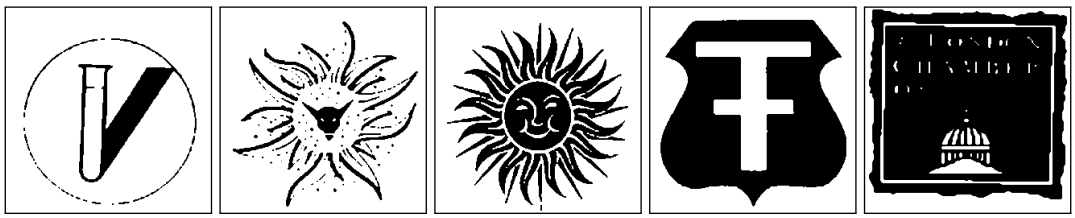


Figure 4.11: Illustration of containment and Euler number.

white pixels in a logo is referred to as the black-to-white ratio.

During the construction of the rough set approximations of a logo image, traversal along the boundary of a component takes place. The ratio of the number of black pixels to the number of white pixels encountered is referred to as the black-to-white ratio. This ratio gives an approximate idea about the composition of each individual logo image.

This attribute is quantized as 0.25, 0.5, 1, 2 and 4. While the value 0.25 suggests that a logo image has mostly white pixels, the other extreme 4 denotes that its pixels are mostly black. Fig 4.12 illustrates five examples with the black-to-white ratio (BW) as 0.25, 0.5, 1, 2 and 4, respectively. The difference in the ratio may clearly be noted visually.



(a) BW=0.25 (b) BW=0.5 (c) BW=1 (d) BW=2 (e) BW=4

Figure 4.12: Illustration of black-to-white ratio.

4.2.7 Relative Positions

Knowing the position of the holes can provide more insight to the structure of logo. This attribute has already been explained in the previous chapter, in Section 3.1.2.5. The major difference in using of this attribute in the retrieval of logos is that here an outer polygon encloses multiple hole polygons and outer polygons inside them. So we use relative position in two domains:

- Relative position of inscribed hole polygon.
- Relative position of inscribed outer polygon.

The inscribed polygon centroid c is found in the local coordinate with the top-left vertex of the polygon as the reference point. The relative position of each inscribed polygon's centroid is found with respect to the top-left vertex v_0 of the outer primary polygon i.e. $\overline{\mathcal{P}}_{\mathbb{G}}(S)$. In a rough-set approach, the relative position attribute is considered with respect to the c of the inscribed polygon to v_0 of the inscribing polygon. The sign conventions are similar to that of relative hole positions, explained in the previous chapter, Section 3.1.2.5. Negative ('$-$') and positive ('$+$') signs are assigned for the left and right lateral halves with respect to the point v_0 and '1' and '2' for the upper and lower halves. The polygons lying exactly in the center are assigned a position of 0. In Fig 4.13, it can be observed how the logos are differentiated by positions of their holes. The c of polygons 4 and 5 lies to the right of v_0 of the polygon 3 in Fig 4.13, and the center of polygons 6 and 7 lies to the left of v_0 of the polygon 3.

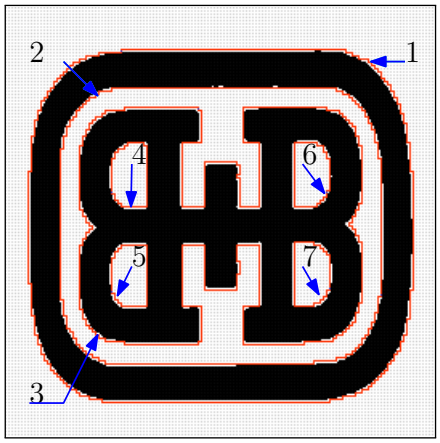


Figure 4.13: Illustration of relative position of polygons.

Table 4.1: Sample information table (shown partially) containing the object properties for the Fig 4.13.

No	EN	HC	PC	RP
1	0	inv	-1	inv
2	inv	1	-1	0
3	-3	inv	1	0
4	inv	3	inv	-1
5	inv	3	inv	-2
6	inv	3	inv	$+1$
7	inv	3	inv	$+2$

Table 4.1 illustrates the values of different attributes for the Fig 4.13. It can be seen that the Euler number for the holes are assigned with 'inv', denoting an invalid attribute case. The hole containment attribute stores the polygon number of the outer polygon, inscribing the hole (i.e, if the polygon under consideration is a hole polygon, else 'inv' for outer polygons). Similarly, the polygon containment attribute stores the polygon number of the inscribing polygon. It is assigned '-1' for the outer-most polygon. The relative position attribute stores the position attribute stores the position values for all the polygons, except for the outer-most polygon.

4.2.8 Concavity

A rough-set polygon is constructed by three types of vertices, 90°, 180°, and 270°. These vertices are defined as Type 1, 0, and -1 vertices, respectively. Concavity act as a very important characteristic to define the shape of a grid polygon [**?**],[**?**]. This attribute is defined exactly in the same way as defined in Chapter 3, section 3.1.2.4. The occurrence of at least two consecutive Type $\langle -1, -1 \rangle$ vertices is referred to as a concavity. If more than two Type -1 vertices occur consecutively, it is known as a nested concavity. The nested concavity cases are not considered in this algorithm.

Similar to the previous concavity declaration, a concavity is classified into 4 types, depending upon its orientation, namely left (L), right (R), upward (U), and downward (D). By preserving the order of these concavities, it is possible to calculate the structural similarities with each polygon using string edit distances.

Unlike the previous characterization of concavity attribute, the 3-tuple form is not used in this algorithm, as the depth and the position of the concavity attribute were not found to be informative enough to be included in a reduct. The position and the depth of a concavity are found to be important during the recognition of characters, as the fine intrinsic details were to be preserved. Whereas for logo retrieval, owing to multiple polygons and their own distinct varied characteristics, the intrinsic details of depth and position of concavities are not necessary. These attributes are not found to be robust enough for our target and hence can be dropped out of the semi-reduct. Fig 4.14 illustrates the concavity attribute in a logo image.

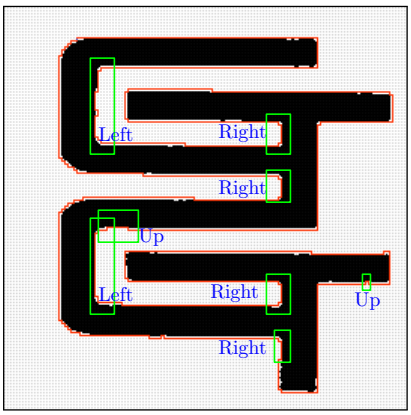


Figure 4.14: Illustration of the concavity orientation attribute.

4.2.9 Edge Ratio

The edge ratio is also defined exactly in the same way as in Chapter 3, Section 3.1.2.6. A grid polygon is comprised of horizontal and vertical line segments of fundamental unit length of $\mathbb{G}$ pixels together. So while traversing on the list of perimeter components, a rough idea about the vertical to the horizontal perimeter component can be obtained. From this, the summation of the horizontal perimeter component (HPC) and the summation of the vertical perimeter component (VPC) are calculated. Hence, for each outer polygon in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$, a horizontal perimeter component (HPC) is defined as the sum of lengths of its horizontal edges and vertical perimeter component (VPC) as that corresponding to its vertical edges. The ratio VPC:HPC is called the edge ratio (ER). It is discretized to the nearest value in $\{\frac{1}{2}, 1, 2\}$. The difference from the edge ratio in the previous chapter is that the edge ratio is calculated for both the outer polygon and the hole

polygon. For character recognition, the holes were extremely small, and were mostly observed as circular. Hence, it was not used for hole polygons. In the logo images, the hole polygons are also extremely significant and can be of different shapes and sizes. Hence, this attribute is considered for this problem. Fig 4.15 illustrates this feature with respect to a sample logo image.

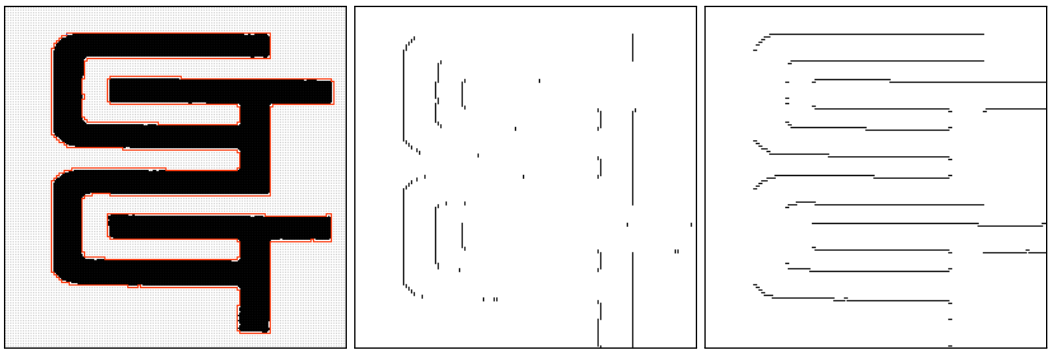


Figure 4.15: Illustration of the edge ratio attribute on logo images. ER = $\frac{1}{2}$. The first image shows the original image and its rough-set upper approximation. The second image shows the vertical perimeter components of the same, while the third image shows the horizontal components of the same image.

4.2.10 Directional Change

This attribute is defined in a similar way as that in Chapter 3, Section 3.1.2.3. While traversing along the boundary of the rough set cover of an object, starting from the top-left vertex, the number of times a change in vertical direction is encountered (i.e., d = '1' to d = '3' or the vice versa), a change in vertical direction is obtained. The number of times we encounter a vertical direction change, we increase our vertical direction change (VDC) count by 1. Similarly, in case of horizontal directional change (HDC), instead of the top-left vertex, the left-top vertex is considered as the start point, and the number of directional changes is counted. Each directional change is defined by a vertex sequence where two consecutive vertices are of Type $\langle +1, +1 \rangle$ or $\langle -1, -1 \rangle$.
Unlike character recognition, the horizontal direction change is also an important attribute. It is calculated as a unit value incrementation, when the polygon encounters a direction change from '2' to '0' or '0' to '2'. For the calculation of the HDC, the start vertex is considered as the left-top

vertex, and not the top-left vertex. Considering the left-top vertex eliminates the possibility of any additional extra directional change count, during termination. Fig 4.16 demonstrates the working of HDC and VDC, considering different v_0 points. Also, the directional change count is measured for the hole polygons in logo images. In Fig 4.17 we show an example of two images having a single polygon to demonstrate the values of VDC and HDC in them.

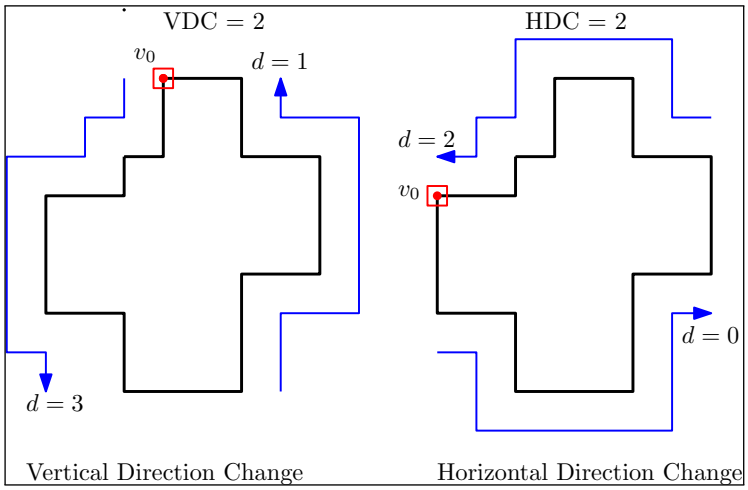
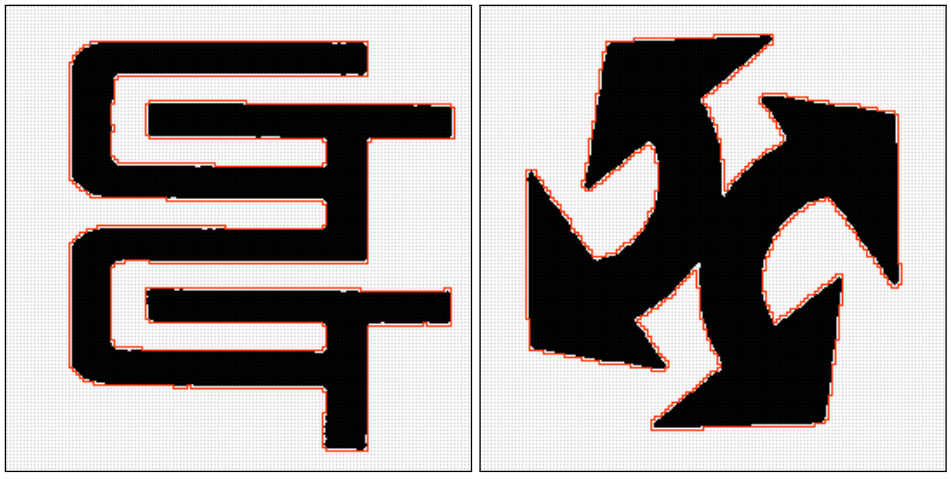


Figure 4.16: Illustration of the the directional change attribute.



(a) VDC=6 & HDC=12; (b) VDC=14 & HDC=14;

Figure 4.17: Example showing the difference in Edge Ratios and direction changes.

4.3 Retrieval using Inverse Hough Transform

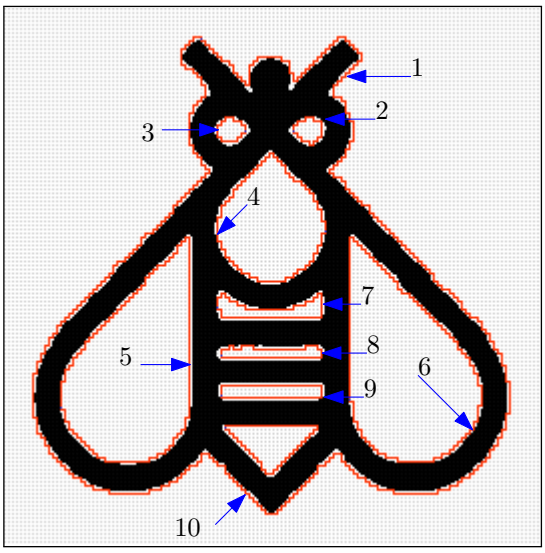


Figure 4.18: Polygons=10; TP=10; MP=10; Holes=9; Parents=1; BW=0.5

The attribute set is divided into two types:

- Global image attributes.
- Individual polygonal attributes.

The global attributes are the values which are used to denote the entire logo image. There are five global image attributes designed for this algorithm, which are:

- Number of polygons
- Number of major polygons
- Number of hole polygons
- Number of parent polygons
- black-to-white ratio

Other than the global attributes, there are a few polygon-specific attributes, derived from each polygon in a logo image. These attributes are as follows:

Table 4.2: Sample information table (shown partially) containing the object properties against the reduct.

No	EN	HC	PC	VDC	HDC	ER	PoH
1	-8	inv	-1	10	10	1	0
2	inv	1	inv	3	2	1	$+2$
3	inv	1	inv	3	2	1	$+2$
4	inv	1	inv	3	2	1	$+2$
5	inv	1	inv	3	2	2	-2
6	inv	1	inv	3	2	2	$+2$
7	inv	1	inv	5	3	$\frac{1}{2}$	$+2$
8	inv	1	inv	3	3	$\frac{1}{2}$	$+2$
9	inv	1	inv	3	3	$\frac{1}{2}$	$+2$
10	inv	1	inv	2	3	1	$+2$

- Euler number
- Hole containment
- Parent containment
- Vertical direction change
- Horizontal direction change
- Edge ratio
- Relative positions.

In Table 4.2, a subset of the rough set attributes for Fig 4.18 is shown for the visualization of how each logo image has some global image attributes (shown in the caption) and some polygon-specific attributes (shown in the table). We have 5 global attributes of an image (Number of polygons, major polygons, hole polygons, parent polygons, black and white ratio) and 7 polygon specific attributes (Euler number, hole containment, parent containment, vertical direction change, horizontal direction change, edge ratio and relative positions). Depending upon the number of polygons contained by each logo images, the attribute size varies. For example, an image having

'n' polygons will have $n \times 7$ local attributes plus 5 global image attributes. It can be seen that these attributes are well-discernible amongst each other and justifies their merit in the image retrieval system. To calculate the similarity of an image to a query image, Euclidean and string edit distance are used, whichever is applicable in each case. For example, to calculate the distance in the concavity direction attribute, string edit distance is used, while the distance between the Euler numbers are found using the Euclidean distance.

The data set on which the algorithm has been tested comprises 1034 binary logo images, which was used in [**?**]. The attribute databases are stored in two files, one for the global image features and the other for the local image features. When a query image is fed to the system, the rough-set covers for this query image is created and then the rough-set attributes for the image is found out. From the attributes found from the query image, a search is made on this database and the top 5 matches are found. The main challenge here is that each image has a different polygon number and hence has a varied size of feature vector for the polygon-specific attributes, e.g., $n1 \times 7, n2 \times 7$, and so on for images consisting of $n1$, $n2$, and so on number of polygons. So it is not possible to calculate the distance between any two images of different feature vector sizes, by simple distance measures. Hence, an inverse Hough transform based method is implemented, invoking a k-d tree based search operation.

For simplicity in designing, a two dimensional k-d tree is considered. In the global feature list for each image, the number of holes and number of parent polygons attributes is set as the two dimensions in a k-d tree, and it is seen that 306 distinct feature points are generated. So each feature point corresponds to around 4 to 5 images having the same attributes binned under them. Whenever a query image is taken, its attribute values are computed and the point is searched using this k-d tree.

This step makes the algorithm work in an $\mathcal{O}(\log n)$ time complexity. Once the point is retrieved, the image IDs and their features are found and stored in the form of a look-up table. The nearest 5 matches matches are found in this way. For finding the matches, each polygon feature prototype is matched with each polygon from the image bins, and if they are greater than a particular weighted Euclidean distance, a counter value is incremented to that image. The containment relations, Euler numbers and the directional change attributes are found to be more robust, and are hence given higher weights, compared to the remaining attributes. After each polygon-to-polygon matching, the image which has got the maximum counter value is found and hence the nearest

match logos are retrieved. Fig 4.19 shows the basic pipeline of the proposed algorithm.

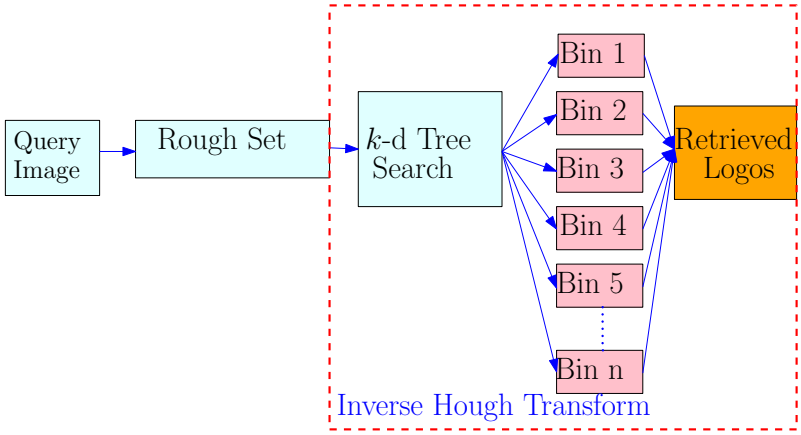


Figure 4.19: Illustration of the proposed algorithm using rough set semi-reduct.

4.4 Experimental Results

The data set that we used contains 1034 challenging binary logo images [**?**]. Each image has a dimension of 256×256 pixels. Since all the logo instances are of the same size, we use a constant grid size of 3 throughout the experimentations. A sample of the data set is shown in Fig 4.20. All experimentations were carried out in a computing platform with 64-bit Intel® Core ™ i5-4210U processor, 8GB RAM, under Windows 7.

For testing the accuracy of our model, we synthetically performed degradation and transformation on the data sets. These operations include rotation, affine transformation, addition of salt-and-pepper noise, erosion and dilation operations. The degradation model is applied on 75 test images. We used these images as query images, and found the top 5 matches using the proposed

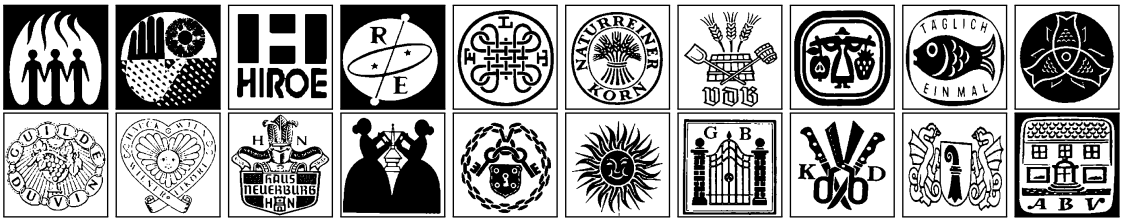


Figure 4.20: Samples from the logo dataset.

method.

For retrieving logo images, we get an average CPU time of 0.692 secs. We achieve a MAP value of 0.94 over 75 query images. The same database is also used by Rajashekhar et. al. in [**?**] for retrieval of logos, using morphological operations for feature extraction. This technique has been implemented by us. Fig 4.21 shows the sample query images used for the experimentations. Fig 4.22 gives the values of precision and recall of the two systems. We observe that the proposed algorithm has marginally less precision versus recall values compared to the algorithm in [**?**]. However, our algorithm has a significantly better runtime. The method in [**?**] takes 0.14 secs per image on average to extract its attributes. Our proposed algorithm takes 0.087 secs for the extraction of the attributes. This indicates that the proposed algorithm is computationally faster. Table. 4.3 shows the comparison of retrieval results and their respective CPU times. It can be observed that the proposed algorithm provides a reasonably good result as compared to some of the existing algorithms. It can be observed that [**?**] is fast but has a significantly poorer performance. Hence we can infer that the proposed algorithm provides a good trade-off between speed and accuracy.

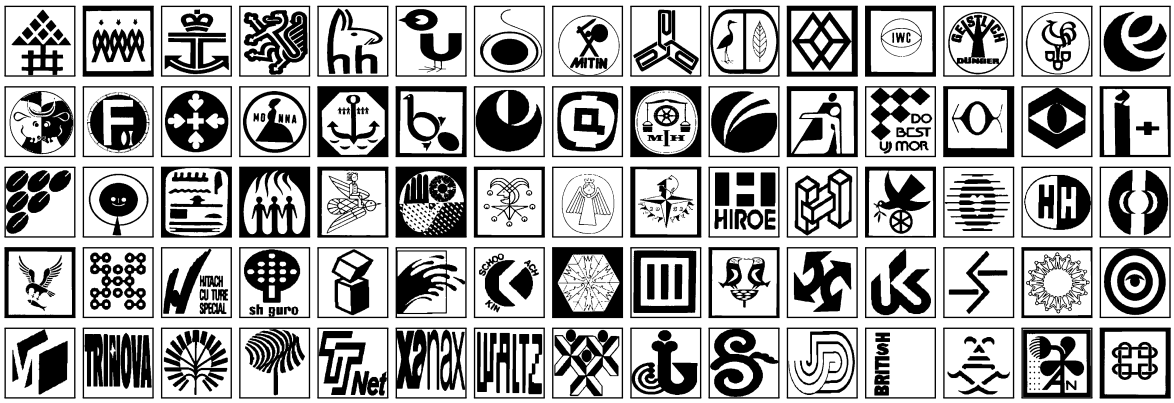


Figure 4.21: Set of logo-query images used for experimentation.

The images are failed to get recognized are the ones with high salt-and-pepper noise. These noise sometimes adds to the number of polygon value and hence affect the precision. We can achieve a further improved model by de-noising the images. On applying a median filter of block dimension 5×5 in the preprocessing stage, to reduce the effects of high levels of salt-and-pepper noise, we get a MAP value of 0.947. While there is an increase in the MAP value, the increase is

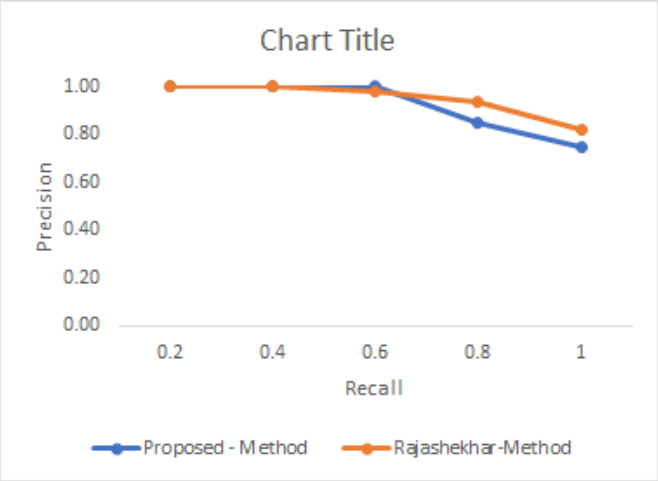


Figure 4.22: Precision-Recall curve for logo retrieval for the proposed algorithm and [**?**].

Table 4.3: Comparison of retrieval accuracy of different schemes and their time.

Model	Top 10 Recall value	Attribute generation time
Proposed method	0.8	0.087
Rajashekhar method [**?**]	0.82	0.14
Scheme in [**?**]	0.8	0.275
Scheme in [**?**]	0.75	0.233
Scheme in [**?**]	0.62	0.046

Table 4.4: Experimental results.

Model	MAP values
Increased salt & pepper noise	0.912
Increased rotation angle (upto 60°)	0.872
Increased rotation angle (beyond 60°)	0.690
Increased erosion	0.921
Increased dilation	0.935

not that significant due the size of the data set. We also try our experimentations by varying the amount of noise in the degraded model and getting the respective MAP values. Keeping the other degraded models constant, we study the variations in MAP value induced by the increased error in one of the models of the degradation model at a time. On increasing the salt-and-pepper noise,

we notice a partial fall in the MAP value (0.912). Increasing the rotation angles affects the MAP value drastically. While the value is not effected much if the angle of rotation is less than 60°, for angles greater than 60° it shows a significant fall in the MAP value. This is due to abrupt change in values of many attributes. The direction of concavities, the edge ratios, directional changes, etc. get effected with an increased skew. We also notice that changing the erosion and dilation amount by increasing the size of the structuring element does not effect the MAP value much, as long as the basic structure of the logo is preserved. Whether a logo is fat or has a single pixel outline, if the structure is the same, it would get classified properly using our algorithm. Table 4.4 shows the MAP values with different noise-induced models.

Fig 4.23 shows logo image classes which have a very good retrieval. We notice that logos can be of complex design, as long as the components have distinct boundary of their own, separating each other properly. Fig 4.24 show images which yield very poor retrieval using our algorithm. We can notice that the main reason for the failure of retrieval in these logo instances is their lack of any distinct boundary of the components, which separate each other. In case we increase the size of the grid, to avoid such discrepancies, the details of many small polygons would be lost. Hence, increasing the grid size would not help in this case. A possible approach can be to use multiple feature descriptors with different grid sizes in this case.

Fig 4.25 shows the result of two experimental cases. We tried generating a different looking logo, using the components of these logos and rearranging them. Since we did not use the relative positions of the parent polygons amongst themselves, the proposed model also retrieves these instances. The advantage is that the proposed model can even retrieve logo instances that resemble each other and yet are different. Fig 4.26 shows three sets of retrieved images. The first image of each row is given as the query image. We see that similar polygons are retrieved.

4.5 Summary

In this chapter, it is shown how the attributes were designed on the spatial domain of rough-set reduct and also discretized in the attribute domain. Global attributes and polygon-specific attributes were found corresponding to each logo image. Using these attributes, a semi-reduct has been designed for the retrieval of the logo images efficiently. The attributes chosen for the semi-

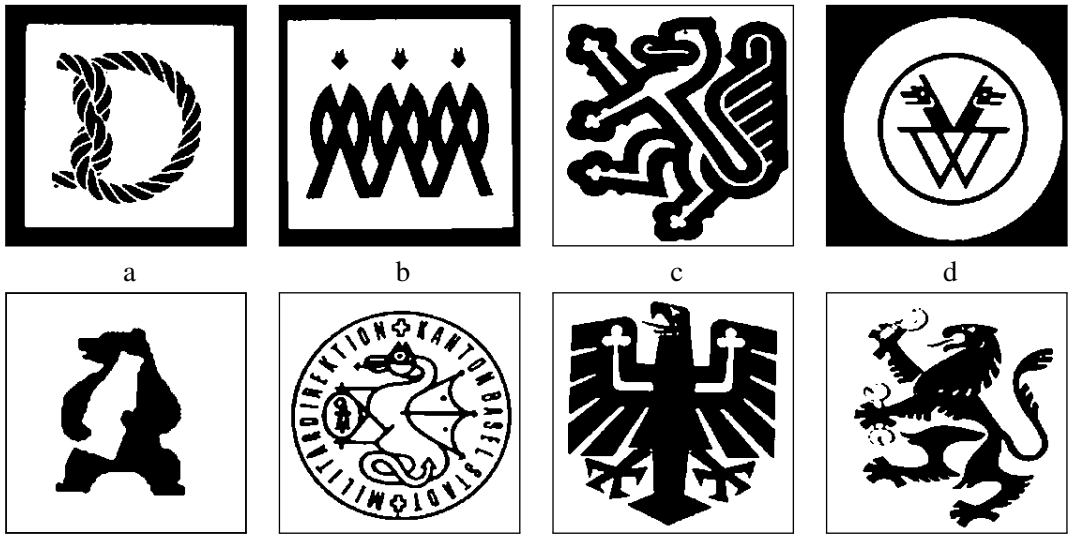


Figure 4.23: Logo image classes showing very good results.

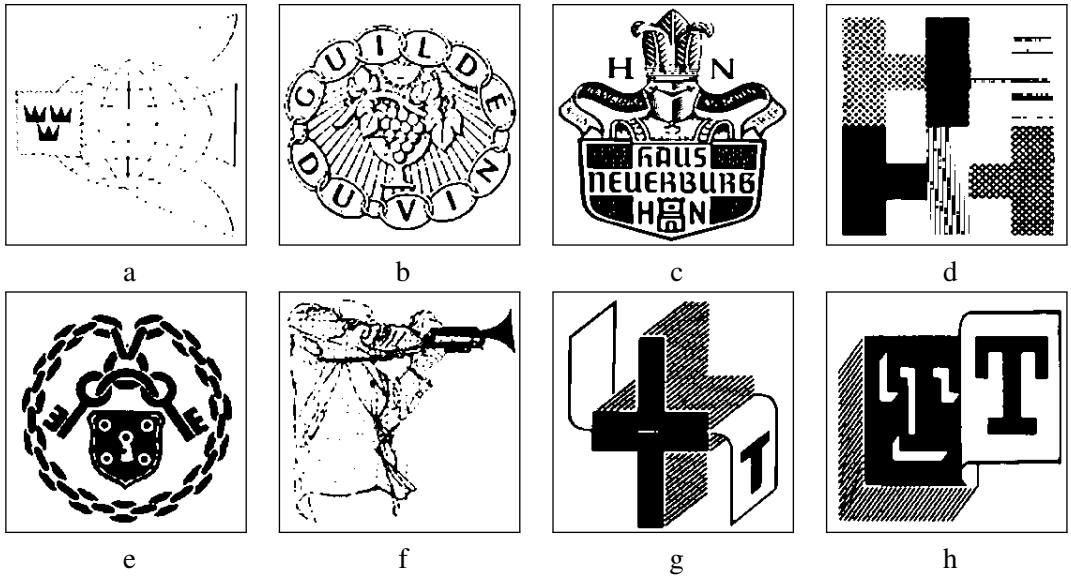
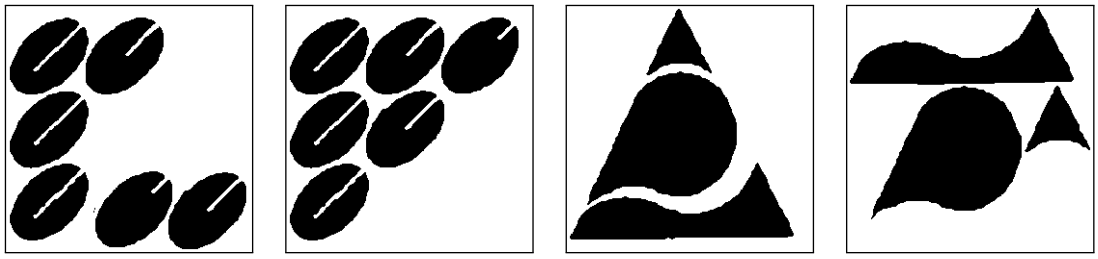


Figure 4.24: Logo image classes displaying poor retrieval results.

reduct have been found by careful study of the polygons of logo images. This small-cardinality semi-reduct has been found for a quick and efficient retrievable logo algorithm. As we have seen from the previous chapter, sometimes even efficient algorithm performs badly during retrieval, due



a b c d

Figure 4.25: Logo images displaying similar retrieval results.

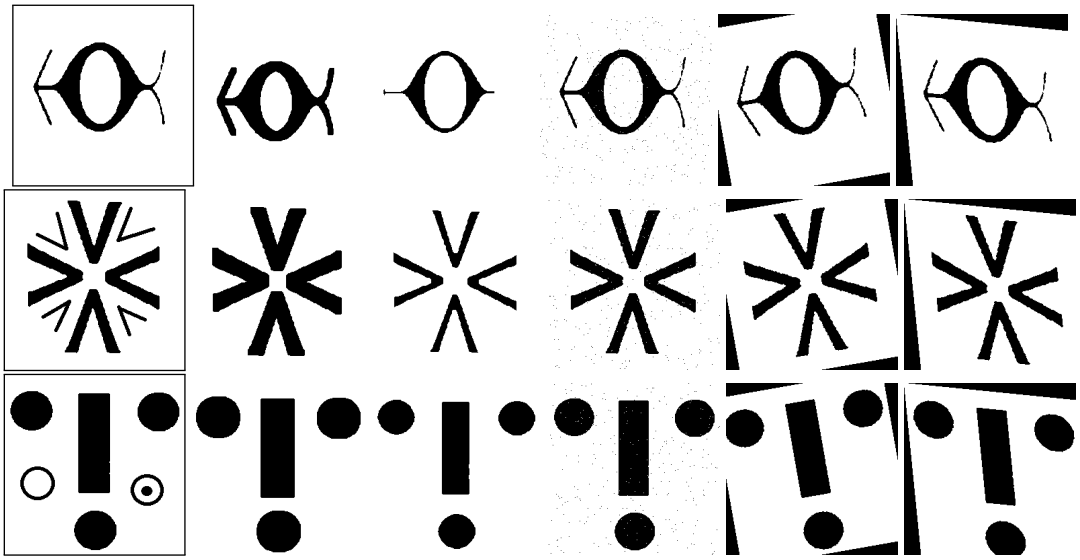


Figure 4.26: Three sets of retrieved images shown in three rows. The first image is given as query.

to its inability to incorporate the effects of noise to some extent. The standard image processing techniques first pre-process an image to get rid of all the noise and then the algorithms are implemented on the clean images. However, in some images, where noise is indispensable, these algorithm would fail. The proposed algorithm is designed in an approximation space and hence can accommodate noise to quite an extent. This is where lies the importance of rough sets.

Chapter 5

Applications in Character Spotting in Inscriptions

A document of historical, religious or other types of records, which may be cut, impressed, painted, or written on a surface of stone, brick, metal, leaf, etc., is called an *inscription*. Before the invention of papers (around 100 BC in China), people used to inscribe logo-graphic scripts for expressive communication. A very famous logo-graphic script that is pictographic in form, is also sometimes called "hieroglyphs". This was mainly found in the ancient Egyptian ruins. Soon, proper textual scripts also arrived. The ability to spot a few known characters or symbols allows the linguists and historians to guess the era in which an inscription was made. With time, slowly these ancient inscriptions were found to be disintegrating, and on the verge of extinction due to various difficulties and challenges that are faced in their preservation. Ever since the dawn of history, the Eurasian plain has been the originating ground for varied races and cultures, and only a few details of their traces can be found from these inscriptions, as deciphered by linguistic and paleographic experts, and historians. In order to keep records of these huge data, archiving them digitally can be useful for their conservation. For this, development of tools for their automated or semi-automated interpretation and indexing is required. Manual spotting of these characters proves to be very laborious and error-prone. Hence, automation in character spotting has evolved in recent time, which has its own challenges due to natural wear and tear of inscriptions through aging.

In this chapter, a computationally efficient technique has been proposed for character spotting using certain concepts from rough set theory. As a preprocessing step, image binarization has been performed. After image binarization, various attributes are found for the segmentation of isolated symbols within the ambit of rough set. In order to spot a symbol in the inscription, the corresponding attribute set for the query symbol is matched with that of the inscribed symbols. The

details of the method have been provided in this chapter to show that the symbols or characters can be quite accurately spotted in the inscriptions. This work makes the following novel contributions:

- An unsupervised algorithm for the retrieval of characters.
- An efficient spotting mechanism using inverse Hough transform.

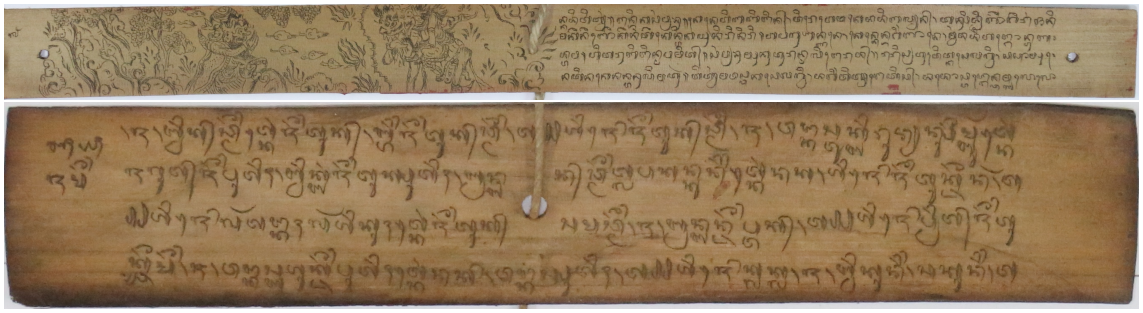


Figure 5.1: Sample palm leaf images from the data set.

5.1 Pre-processing—Binarization

The scrolls of the inscription images are in palm leaves, in Balinese language. A sample set of the palm leaf inscription images is shown in Fig 5.1. One of the major challenges in the analysis of this type of data is its binarization. Fig 5.2 shows the overall block diagram for the binarization for isolated noise removal process. The binarization process consists of the following steps:

- Bilateral filtering [**?**]
- Contrast enhancement
- Adaptive thresholding [**?**]
- Morphological closing operation [**?**]
- Median filtering [**?**]

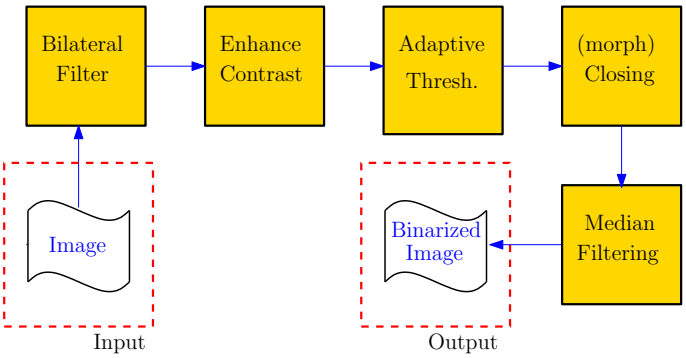


Figure 5.2: Block diagram for the binarization scheme.

5.1.1 Bilateral Filtering

As a basic binarization model, bilateral filters have been used for the initial smoothening of the image. A bilateral filter is a non-linear filter, which preserves the edges and reduces the noise in images. In this filter, the intensity of each pixel is replaced with a weighted average of intensity values from nearby pixels. These weights can be based on Gaussian distributions.

$$I^{\text{filtered}}(x) = \frac{1}{w_p(x)} \sum_{x_i \in \Omega} I(x_i) f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|), \tag{5.1}$$

where the normalization term is

$$w_p(x) = \sum_{x_i \in \Omega} f_r(\|I(x_i) - I(x)\|) g_s(\|x_i - x\|), \tag{5.2}$$

where,
$I^{\text{filtered}}(x)$ = filtered image;
I = input image;
x = coordinates of the current pixel to be filtered;
Ω = window centered in x;
f_r = Gaussian range kernel for smoothing differences in intensities;
g_s = Gaussian spatial kernel for smoothing differences in coordinates.
Fig 5.3 shows the result on two images, after applying a bilateral filter.

Figure 5.3: Applying bilateral filter on palm leaf images shown in Fig 5.1.

5.1.2 Contrast Enhancement.

After the smoothening of the image, it is observed that the contrast of the image needs to be increased. Contrast is created by the difference in the light/luminance reflected from two adjacent surfaces. In other words, contrast is the difference that makes objects differentiable from other objects and the background, and hence makes it better visible. The basic contrast and brightness adjustments are transformations of the form:

$$f(x) = \alpha x + \beta \tag{5.3}$$

where,
$\alpha =$ gain; (controls contrast);
$\beta =$ bias; (controls brightness);
$x =$ intensity at a point.
Fig 5.4 shows the result on two images, after increasing the contrast. The values of α and β have chosen to be 1.5 and 10. These values were obtained upon different experimentations.

5.1.3 Adaptive Thresholding

To binarize the image for rough set analysis, thresholding is required. Adaptive thresholding [**?**] takes this gray-scale image as an input and outputs a segmented binary image. In this technique, for each pixel, a threshold value is to be estimated. Since the palm leaves are observed to have a lot of gradient difference, an adaptive thresholding technique is used. In this algorithm the image

Figure 5.4: Increasing contrast on the pre-processed image shown in Fig 5.3.

is divided into multiple overlapping sub-images and then the optimum threshold is found for each sub-image, by calculating the mean of the pixels, shown in [**?**]. The overlapping block size of the sub-images is taken as 15, and the value of constant c is chosen as 5. This constant c is a weight used to subtract from the mean of these sub-image pixels. Fig 5.5 shows the result on two images, after applying a bilateral filter.

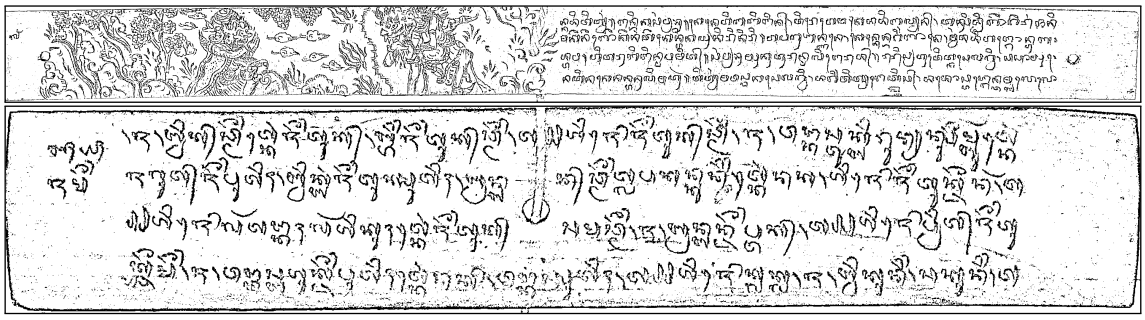


Figure 5.5: Result of adaptive thresholding on the pre-processed image shown in Fig 5.4.

5.1.4 Morphological Closing

After performing the adaptive thresholding of the image, it is noticed that a lot of gaps/ holes occurring in the characters on the leaf needs to be filled. To achieve this, the morphological closing operator is used. Technically, a mathematical closing of a binary image A, by a symmetric structuring element B, is defined as the erosion of the dilation of that image.

$$A \bullet B = (A \oplus B) \ominus B, \tag{5.4}$$

Fig 5.6 shows the result on two images, after performing a morphological closing operation. We have used B to be a circular disc of radius r (r = 2).

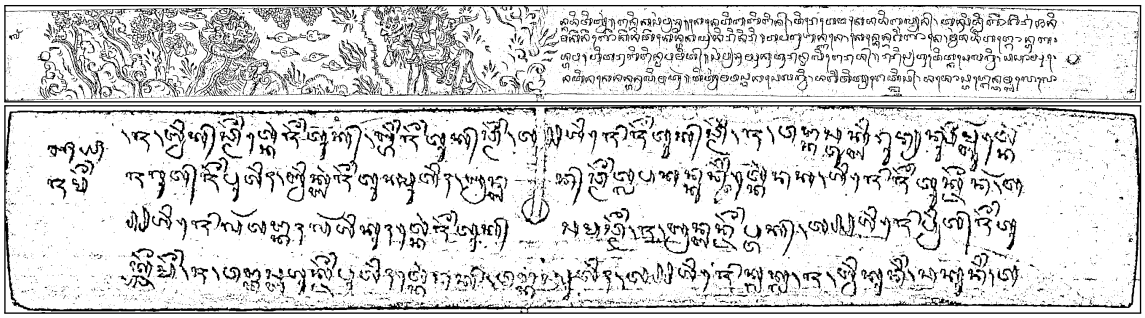


Figure 5.6: Result of morphological closing on the binarized image shown in Fig 5.5.

5.1.5 Median Filtering

After performing all the above pre-processing operations, it is noticed that there are some spurious noises in the data. To remove these extra spurious noises, a median filter is applied. A median filter is a non-linear digital filtering technique, to remove noise from an image. In this technique, the image is divided into multiple small blocks of overlapping sub-images. The median value of intensity of these pixels contained in the sub-image is found, and each entry is replaced by this median value. Hence, a median filter preserves the edges, while removing the noises, which is important for our technique. A median filter of maximum kernel size 6 has been used for the experimentations. Fig 5.7 shows the result on two images, after performing median filtering using a 5×5 block. Few more sample binarized inscription images are shown in Fig 5.8.

5.2 Segmentation using Rough-sets

After getting the binarized image, it is observed that each inscription contains multiple distinct characters inside them. In order to perform character spotting, each character needs to be segmented from another. This segmentation is performed by constructing the rough-set covers for

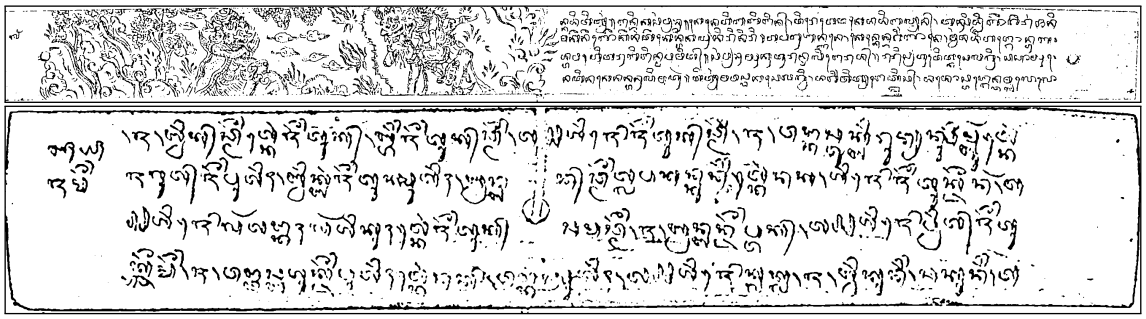


Figure 5.7: Median Filtering on the pre-processed image shown in Fig 5.6.

each character. The polygonal covers are very good for segmentation and less expensive than the standard connected-component segmentation. Fig 5.9 shows the inscription image after segmentation, using the upper-approximations of the rough-set covers.

5.3 Proposed Methodology

After obtaining the segmented inscription image, it is required to find the attributes of each of these characters for a powerful character spotting algorithm. A carefully selected and experimentally found attribute set has been chosen for the design of this algorithm. An overview of the working of the proposed approach is shown in Fig 5.10. Some of the attributes used in this algorithm have been explained in the previous two chapters, while some new attributes have also been used for this task. The Balinese data set [**?**] consists of 133 classes of different characters, including numerals, and diacritics. Each character image has multiple instances. The rough-set polygonal boundaries are found for each of the outer polygons and the hole polygons. However, it has been observed that in this script, most characters do not inscribe any void in between them, and hence, it lacks hole polygons. From these polygons, a database is created, which contains the attribute values of all the polygons in the inscription image.

For character spotting, a query character is taken and the rough-set boundaries are found. From these boundaries, a feature vector is created from the designed attributes. The attributes generated are then matched with the database and all matches above a selected threshold (explained in the sub-sections) are found. Similar to the previous algorithm, the matching is done by inverse Hough

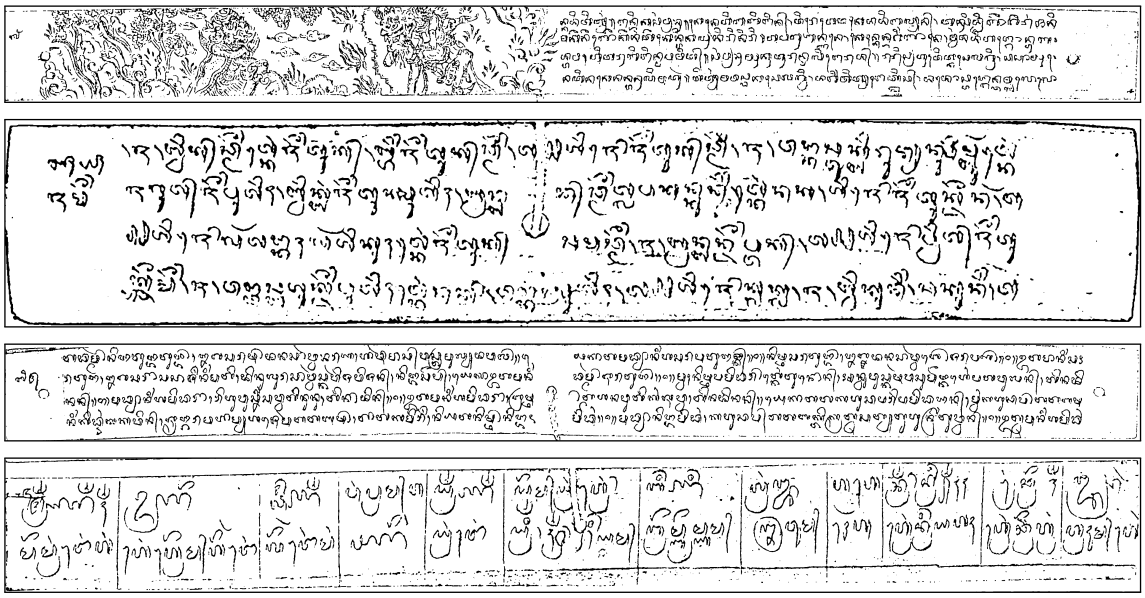


Figure 5.8: Results of binarization of various documents from Bali inscriptions.

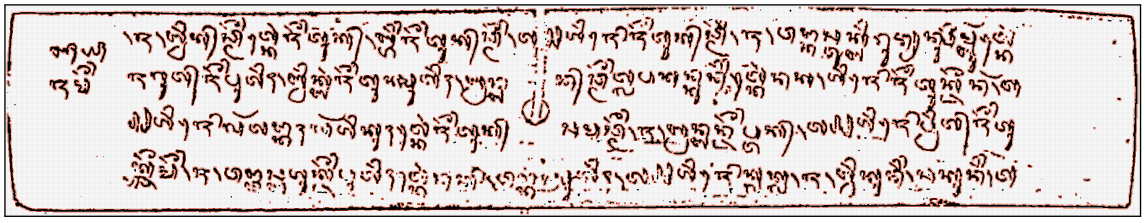


Figure 5.9: Results of character segmentation for the inscription image.

transform, and the searching is performed by a k-d tree approach. A k-d tree based search operation is performed for gaining on the average runtime further. By using this method, it is possible to keep the system average runtime low, as shown in Sec. 5.6. The detailed algorithm and its working are explained in the subsequent sections.

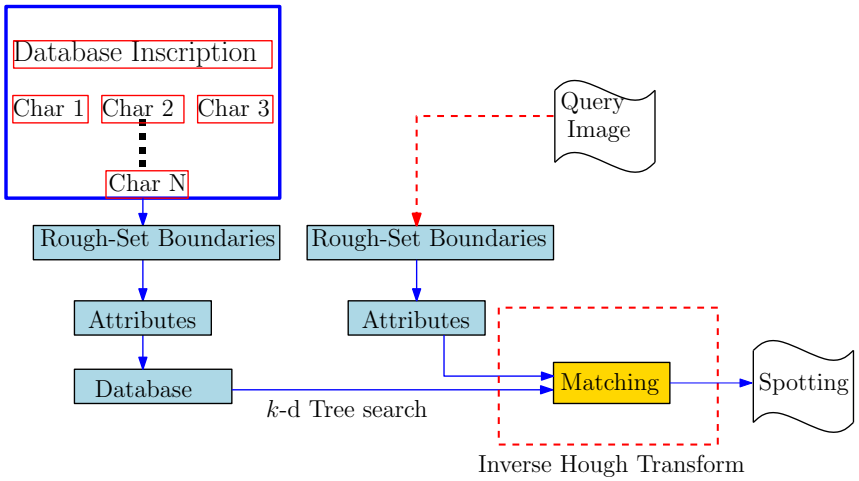


Figure 5.10: Pipeline of the proposed symbol spotting method.

5.4 Attributes for Spotting of Characters

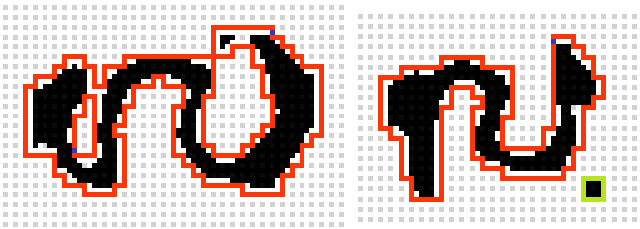
The proposed method for spotting of symbols or characters inscribed in palm leaves in Balinese language uses rough-set attributes. To achieve this, an attribute set is deigned based on observations drawn while designing the system. These attributes and the observations have been explained in the following sub-sections.

5.4.1 Number of polygons

The characters in Balinese usually consists of different *diacritics*. A diacritic is a sign, which, when written above or below a letter, indicates a difference in pronunciation of the same letter. It is noticed that each character individually consists of all singly connected components. But while cropping, for their usage as a query image, sometimes the character edges are chopped off or the curved segments are broken. In such cases, the continuity is broken, which results in the generation of multiple connected components. This results into an increased number of polygons. Also, sometimes due to improper binarization, the continuity of the glyph images are broken. In such cases, where the glyph images contain multiple number of polygons, only the polygon having the largest perimeter is considered as the primary polygon.

Fig 5.11 shows a few examples, where the continuity is broken. The red polygons shows the primary polygon, while the green polygon shows the minor polygons (Fig 5.11b). Also, due to a

rough-set representation, it is observed that all small discontinuities are ignored and treated as a single object (Fig 5.11a).



(a) (b)

Figure 5.11: Illustration of continuity breakage in glyph images.

5.4.2 Number of Holes

As mentioned in the previous sub-section, most characters do not have cavities inside them. Still, there are some characters, which have hole polygons (also sometimes caused due to binarization errors). In this algorithm, a grid size of 3 has been chosen to work on the entire data set. The concepts of a hole-polygon and its mathematical formulation have been already explained in Chapter 3, section 3.1.2.2.

Fig 5.12 shows the $\angle 90^\circ$ in yellow and the $\angle 270^\circ$ in blue, which are complementary for hole polygon and primary polygon. This differentiates an outer polygon from a hole polygon.

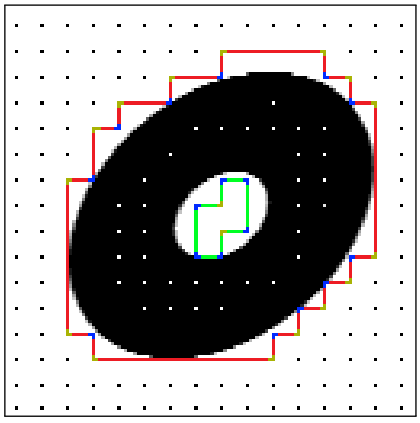
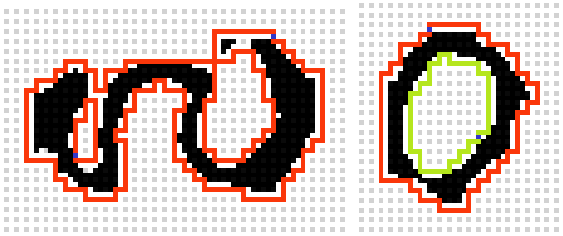


Figure 5.12: Illustration of $\angle 90^\circ$ and $\angle 270^\circ$ in holes and primary polygons.

5.4.3 Approximate Euler Number (EN)

As shown in the previous chapters, some polygons enclose one or more hole polygons. Euler number is used to represent this feature of the polygons. In this method, the Euler number is defined in a similar way as that of the definition in Chapter 3. It is given as $2 - n$, where n is the total number of polygons in $\overline{\mathcal{P}}_{\mathbb{G}}(S)$. So a polygon enclosing a hole polygon would have EN = 0, as $n = 2$, while one enclosing no hole polygon would have EN = 1, as $n = 1$ in this case. In Fig 5.13a, the EN of the polygon is observed to be 1, while that in Fig 5.13b it is 0. The primary polygons are shown in red, while the hole polygons are shown in green. In case of character spotting, none of the primary polygons is found to be further enclosed in any other polygon. Hence, storing the primary polygon containment relation is not important in this problem.



$n = 1$, EN=1 $N = 2$, EN=0

Figure 5.13: Illustration of number of polygons, holes, and Euler number.

5.4.4 Position of Holes (PoH)

Once the Euler number is obtained, the relative location of the hole polygon with respect to a reference point on the primary polygon is computed. The relative hole position is found in the same way as that explained in Chapter 3, Section 3.1.2.5. The centroid of the inscribed (hole) polygon is found, and its location with respect to the top-left vertex of the primary polygon is noted.

It is noticed that usually the horizontal center coincides with the centroid of the parental cover. The positions are denoted by the same earlier conventions. If the hole centroid, c, lies to the upper or the lower half of v_0, it is denoted by '2' and '1' signs, respectively. Similarly, the left and the right lateral halves with respect to v_0 are denoted by '$-$' and '$+$' signs, respectively.

5.4.5 Directional Changes

This attribute is also the same as the attribute, defined in Chapter 3, Section 3.1.2.3. While traversing along the boundary of $\overline{\mathcal{P}}_{\mathbb{G}}(S)$ of an object, the number of times of directional changes is recorded. These directional changes can be in different orientations. Depending on the orientations of directional changes, this attribute has been classified into two types, namely, the number of *vertical direction changes* (VDC) and the number of *horizontal direction changes* (HDC). Each directional change is defined as a consecutive occurrences of type $\langle +1, +1 \rangle$ or a $\langle -1, -1 \rangle$ vertices. These are further explained below.

5.4.5.1 Vertical Directional Changes (VDC)

During the traversal along the boundary, starting from the top-left vertex, whenever a directional change is encountered from '3' to '1' or from '1' to '3' [**?**], a vertical directional change is obtained. The summation of these directional changes along the vertical direction gives the total vertical directional changes for each polygon. However, this attribute is limited to only the primary polygons. Fig 5.14 shows the directional changes of a glyph image.

5.4.5.2 Horizontal Directional Changes (HDC)

Similarly, the summation of the number of directional changes encountered in the horizontal direction ('0' to '2' or '2' to '0'), while traversing from the leftmost vertex gives the attribute whose value is the number of horizontal directional changes. Fig 5.14 shows how two glyph symbols are discriminated by HDC and VDC.

5.4.6 Perimeter Components

This attribute has been slightly modified and broken into three sub-attributes in this method. The rough-set perimeter of an optical character is broadly decomposed into two components, i.e. the vertical perimeter component and the horizontal perimeter component. These attributes have been further explained in the following sub-sections.

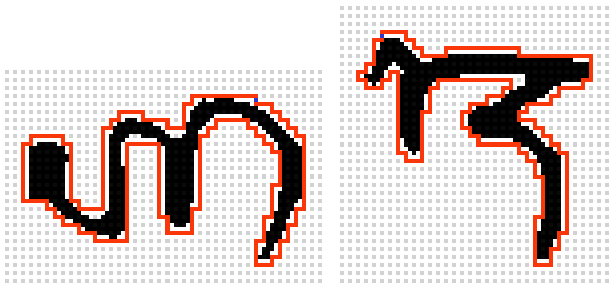


Figure 5.14: Illustration of VDC and HDC.

5.4.6.1 Vertical Perimeter Component (VPC)

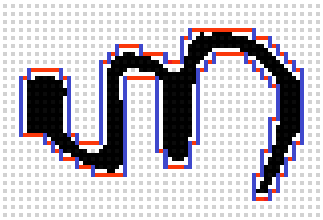
The summation of all the perimeters in the vertical direction gives the vertical perimeter component (VPC). The directions '1' and '3' are used to calculate the VPC. Also, the sum of the perimeter components along '1' and '3' are calculated separately, and their ratio is found. This ratio is referred to as the *vertical ratio* (VR). Fig 5.15 illustrates the value of VR on an image.

5.4.6.2 Horizontal Perimeter Component (HPC)

Similar to the VPC, the horizontal perimeter component (HPC) is calculated as the summation of all the fundamental unit lengths along the horizontal direction, i.e., along the directions '0' and '2'. Also, the ratio of the summation of the individual distances is calculated along '0' and '2'. This ratio is referred to as the *horizontal ratio* (HR). Fig 5.15 illustrates the value of HR on an image.

5.4.6.3 Edge Ratio (ER)

The ratio of VPC to HPC is called the *edge ratio* in this work. Its value is discretized and rounded to the nearest value in $\{\frac{1}{2}, 1, 2\}$. The horizontal ratios and the vertical ratios are also discretized to the nearest value in $\{\frac{1}{2}, 1, 2\}$. The attribute ER is used for having a broad sense of idea of the overall geometric stretch of each glyph symbol. Fig 5.15 shows the edge ratio, vertical ratio, and horizontal ratio in an example.



ER=2;VR=1; HR=1

Figure 5.15: Illustration of ER, VR and HR.

5.4.7 Concavities

Concavity is an important attribute for describing any shape, as shown in [**?**], and explained in Chapter 3, Section 3.1.2.4. Defining it in short, a concavity marked by as a consecutive occurrence of two Type '-1' vertices. For a sequence of more than two consecutive occurrences of '-1' vertices, a nested concavity is obtained. Similar to the definition of concavity as a 3-tuple, this attribute is used in this work as a 3-tuple format. The three components of this 3-tuple are:

1. Orientation
2. Position
3. Depth

The orientation of the concavity attribute is classified into four types, depending upon its spatial orientation, namely:

- leftward (L)
- downward (D)
- rightward (R)
- upward (U)

The positions of these concavities are coincidental with respect to the centroid of the primary polygon. The position is encoded in the same manner as that of the position of holes, in the form of:

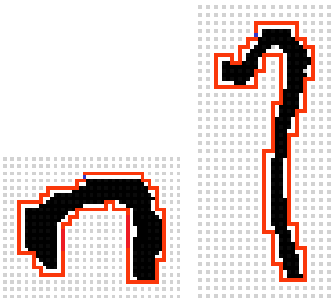
- -1

- -2
- $+1$
- $+2$

The concavities of hole polygons are not taken into analysis for this work. Also, other than the position and orientation of concavities, the depth of the concavity of considered as the third component of the 3-tuple. For the symbols having similar concavity orientations and positions, depth makes two different symbols discernible. The concavity depth is discretized in the following values, with respect to their total height :

- 1
- 2
- 3

Fig 5.16 illustrates the 3-tuple values of concavities of two different characters.



$\langle U,+2,3\rangle$ $\langle U,-1,1\rangle$

Figure 5.16: Concavity 3-tuple.

5.4.8 Hull Concavities

Sometimes, the presence of noise in the boundaries of the characters, due to binarization problems gives rise to a lot of unwanted erroneous small concavities. It is also observed that different glyphs produce different covers, and hence lead to different concavities [**?**][**?**]. Hence, to tackle this problem, an approximate orthogonal hull is used for finding the main direction of concavity.

This concavity suppresses all the concavities caused by noise, and also suppresses the nested concavities. To construct an orthogonal hull from a polygonal cover, we use a retracing algorithm, whose construction is explained in Chapter 2, Section 2.3. Wherever a concavity is obtained, the cover is retraced back and the hull is constructed. The hull generated by this algorithm is also a tight upper-approximation of a convex hull. Again we get four hull-concavity directions:

- leftward (L)
- downward (D)
- rightward (R)
- upward (U)

Fig 5.17 demonstrates a convex hull concavity.

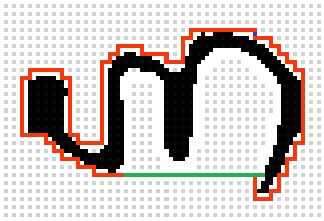


Figure 5.17: Upward hull-concavity.

5.5 Symbol Spotting

For an efficient spotting algorithm, the image of the palm-leaf is first binarized (shown in Fig 5.1). The rough-set approximations of the characters present in the page are then found, which are used for character segmentation. The values of the attributes are then stored for all the polygons. The advantage of the proposed algorithm is that many small discontinuities in the binarization step are disregarded by this algorithm, since it works on an approximate space. The features are stored in an 11-attribute form, corresponding to every primary polygon in the image. The features are $\langle$ number of polygons, number of holes, Euler number, position of holes, VDC, HDC, edge ratio, VR, HR, concavity tuple, hull-concavity orientation $\rangle$. The concavity 3-tuple in turn has 3 attributes under it. Hence, a total of 13 attributes are obtained.

Depending upon the number of polygons contained in the glyph images, the size of the attribute varies. For example, an image having n polygons will have $n \times 13$ number of attributes. The attributes are designed in such a way so as to obtain a good discernibility among the different symbols. The Balinese language contains 133 distinct glyph symbols in total (as obtained from the data set).

Most query images are comprised of a single polygon. Some query contains more than one polygons. In such cases, the polygons are arranged in the decreasing order of their perimeter. The biggest polygon from the query images is compared with each of the polygons in the segmented inscription image from the document file, created from the original leaf inscription. With the nearest matches found, the remaining polygons are compared, which lie in a window of $\pm\varepsilon$ pixels from the bounding box of the biggest polygon. For our experimentation, the value of ε is taken as 50.

For measuring dissimilarity, string edit distance is used for simple concavities and hull concavity features, while a weighted Euclidean distance for the rest of the attributes. The weights are chosen in a way so that the more robust attributes are given higher weights for the distance, while the relatively-fluctuating attributes are given lower weights. Since the distance between two images in the attribute space is taken as the Euclidean metric, a higher weight assigned to an attribute drastically increases the distance between two dissimilar objects when that attribute has a small difference of values for the two objects. Table 5.1 shows the values of the weights chosen for the experimentations. However, performing a linear search for the polygons from the page images is time intensive, and it slows down the process. Hence, a k-d tree [**?**] based searching mechanism is used. In this approach, we use a 2-dimensional k-d tree, wherein the edge ratios and VDC are kept as the two dimensions of the tree. The k-d tree based search mechanism has $\mathcal{O}(\log n)$ time complexity, which is a considerable reduction compared to a linear search.

All the symbols from the page which sums up to have a distance smaller than a pre-set threshold (threshold varies from page to page), are considered as the spotted symbols. Few attributes like directional changes, edge ratios, and concavity directions are found to be more robust, and are given more weights compared to the remaining attributes. Fig 5.18 shows the basic pipeline of the proposed algorithm. A sample spotting with a query image is shown in Fig 5.19.

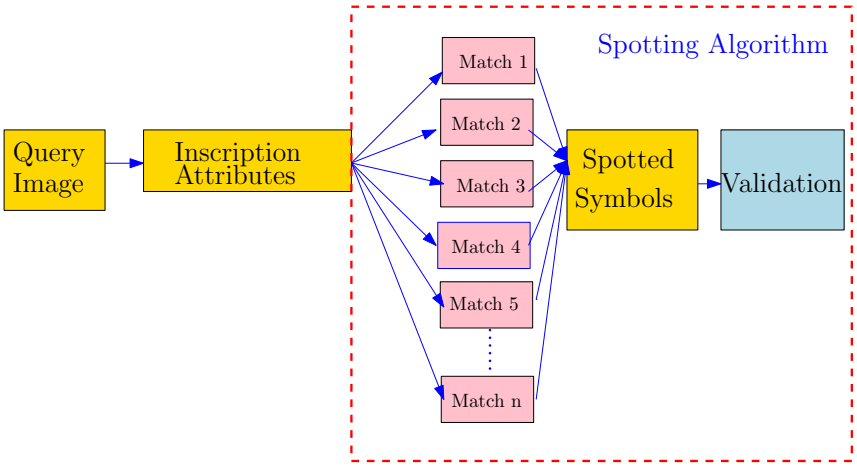


Figure 5.18: Block diagram of the proposed symbol spotting algorithm.

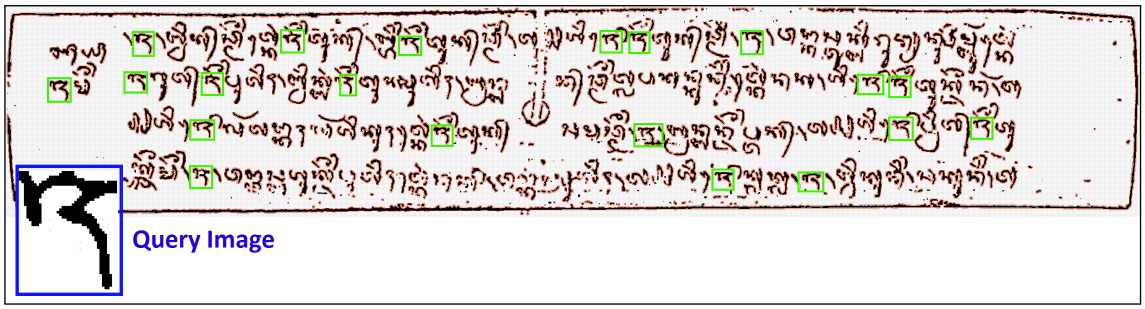


Figure 5.19: Result of a character spotting.

5.6 Experimental Results

The data set that is used for experimentation for this algorithm consists of 10 challenging Balinese inscription pages [**?**]. The inscription images are all of varied size. The approximate size of all are around 3000 $\times$ 500 pixels. Sample inscriptions from the data set have already been shown in Fig 5.1. The data set consists of 133 distinct symbols, which includes digits, characters, and diacritics. To validate the results obtained from the proposed algorithm, the ground truth XML file provided along with the data set is made to use. The XML files contains the details about the bounding box corners of each character appearing in the inscription scroll, along with its character annotation. While spotting the symbols using our algorithm, the symbols that have a minimum of 25% of the size of the query image as the overlapping area, are considered as the correct matches.

Using the proposed spotting algorithm, an average CPU time of 0.498 secs is achieved for

Table 5.1: Values of the weights used for the distance calculation.

Attributes	Weights
Number of polygons	1
Number of holes	1
Euler Number	1
position of holes	100
VDC	10
HDC	10
edge ratio	5
VR	10
HR	10
concavity tuple	1000
hull-concavities	10

the experimentations. It shows that this rough-set based technique is computationally efficient, considering the size of the images. An overall accuracy of 74.47% is achieved, when the algorithm is run on 20 query images. The CPU time is achieved on a computing platform having of 64-bit Intel ® Core ™ i5-4210U processor, with 8GB RAM. The performance of the system can be further improved by having an improved binarization technique. Most symbols that are failed to get spotted are observed to be the ones which lie mainly along the boundary edges. Also, a few symbols resemble each other so closely that they are difficult to distinguish. Some characters also loose their connectivity and have broken curvature due to errors in binarization and cropping of the query images.

5.7 Summary

In this chapter, the designing of rough-set attributes, aimed at an efficient character spotting algorithm has been shown in detail. Using these attributes, a detailed experimental analysis for the same has been shown. This chapter mainly shows how a small-cardinality rough-set attribute set has strong discernible power amongst various symbols.

The proposed algorithm requires no training, and hence is an unsupervised system. Nowhere in the algorithm have we trained the system with the Balinese script. Hence, this mechanism is partially language independent. Any script having discrete characters or symbols, should also

work with this spotting algorithm. The cases where this approach would fail is for scripts with no discrete boundary of the characters. For example, scripts like the Arabic and most Indic scripts have continuity amongst the characters. In such cases, a separate segmentation of these characters needs to be achieved first in order to use this algorithm.

Many a time, even an extremely efficient algorithm for the spotting of symbols fails, since it cannot accommodate the effects of noise. This is where lies the importance of rough sets. Without a rough-set interpretation, it would be difficult to estimate the values of some of the attributes. Conventional image processing is likely to produce erroneous result in subsequent analysis. Any error, owing to the discontinuities in a character, can lead to problems in the algorithms using the standard image processing techniques.

The main advantage for our rough-set method over other existing methods is its significant operational difference, lower computational complexity, and reduced runtime. If we try to preserve the relative changes of positions of the vertices of the cover, these features can be used for recognition in skewed orientations as well. Some additionally added attributes, such as convexities and perimeter of the lower boundaries of the rough-set covers as a ratio with the upper boundary, can be used to get the relative hole polygon size. Further, concavity ratio can be used as the ratio of the concave perimeter to the total perimeter. Adding new and improved attributes may also downsize the total required attribute set, even for scripts having large character class size.

Dissemination of Research Results

[1] Ushasi Chaudhuri, Partha Bhowmick, and Jayanta Mukhopadhyay, “A Novel OCR System based on Rough Set Semi-Reduct”. In *Pattern Recognition and Machine Intelligence*, LNCS vol 10597, pp-263–269, 2017.

[2] Ushasi Chaudhuri, Partha Bhowmick, and Jayanta Mukhopadhyay, “A Novel Rough Set based Technique for Character Spotting on Inscription Images”. In *9th International Conference on Advances in Pattern Recognition*. Bangalore, 2017.