DISSERTATION

submitted

to the

Combined Faculty for the Natural Sciences and Mathematics

of

Heidelberg University, Germany

for the degree of

Doctor of Natural Sciences

Put forward by

MSc:	Chamroeun Khim
Born in:	Phnom Penh, Cambodia
Oral examination:	.....

3D Image Processing, Analysis, and Software Development of Khmer Inscriptions

Advisor: Prof. Dr. Dr. h.c. mult. Willi Jäger
Prof. Dr. Dr. h.c. mult. Hans Georg Bock

Zusammenfassung

Khmer-Inschriften sind primäre Quellen für Informationen zur turbulenten Geschichte Kambodschas. Verwitterung, Umweltverschmutzung, mutwillige Beschädigungen oder Zerstörungen in kriegerischen Auseinandersetzungen verursachten einen teilweise schlimmen Zustand der Inschriften. Geschichtswissenschaften und die Linguistik nutzen bereits intensiv die Khmer-Inschriften, um das Wissen über die Khmer-Zivilisation alter Zeiten zu verbessern.

Allerdings gibt es bisher keine Untersuchungen, bei denen moderne Methoden des Wissenschaftlichen Rechnens entwickelt und eingesetzt werden, um

- Khmer-Inschriften adäquat digital zu erfassen,
- diese Daten mit Methoden der Bildverarbeitung aufzuarbeiten und in 2 dimensionale Schriftbilder zu transformieren,
- digitale Verfahren der Schrifterkennung für Khmer-Schriften zu liefern,
- beschädigte oder unvollständige Schriftdaten soweit als möglich zu reparieren oder wenn möglich zu ergänzen,
- Werkzeuge für eine digitale Verarbeitung von Texten in historischen Khmer-Schriften zu liefern,
- und so zur Rekonstruktion, Lesbarkeit und der Erhaltung dieser wertvollen historischen Dokumente beizutragen.

Dabei stellen sich Herausforderungen, die nur durch neue Konzepte und Methoden bewältigt werden können. Diese werden durch folgende Fakten verursacht:

- die Inschriften liegen in der Regel als 3 dim Daten vor,
- sie sind sowohl in der Größe und als auch im Zustand teilweise problematisch,
- die Anzahl als auch die Komplexität der Schriftzeichen sind groß und, was die möglichen Kombinationen von Konsonanten, Vokalen, Sub-konsonanten und abhängigen Vokalen angeht, vergleichsweise sogar riesig.

Ziel dieser Untersuchung ist es, diese Herausforderungen zu bewältigen und dazu beizutragen, die angeführten Lücken zu schließen. Dazu werden zunächst die Originaldaten durch 3D-

Scans erfasst, mit Bildverarbeitungsmethoden verarbeitet und transformiert in eine 2D Darstellung, die dann weiter analysiert werden kann. Einzelne Schriftzeichen werden isoliert und Verfahren zu deren Identifizierung werden entwickelt. Zur Charakterisierung werden deren topologische und geometrische Strukturen genutzt, die entsprechend durch einfach zu berechnende Indizes erfasst werden. Es gelingt durch schrittweises Filtern vorgegebene Schriftzeichen zu identifizieren. Das Verfahren wird anhand der Khmer-Schrift der prä-Angkor Periode entwickelt, lässt sich jedoch auch auf Schriften der anderen Epochen erweitern. Da bisher kein Unicode vorlag, wurde für die prä-Angkor Periode auch der notwendige digitale Schrift Code entwickelt. Anhand der entworfenen mathematischen Methoden und Algorithmen wird ein Software-Werkzeug entwickelt, mit dem historische Khmer-Zeichen und Inschriften verarbeitet und analysiert werden können. Dabei handelt es sich um eine erste, experimentelle Version, die in einem weiterführenden Projekt noch für die allgemeine Nutzung auszubauen ist.

Abstract

Khmer inscriptions are primary sources of information on the history of Cambodia. Weathering, pollution, intentional damage or destruction and the acts of war have caused the inscriptions in a bad condition. Historians and Linguists have already intensively used the Khmer inscriptions to gain a better understanding of the ancient Khmer civilization.

Still, there has so far not been any research project using modern methods of scientific computing in order to

- appropriately capture Khmer inscriptions digitally,
- enhance these data using image processing techniques and transform them into 2D representations,
- optimally repair or if possible supplement damaged or incomplete script,
- provide tools for digital image processing and analysis of texts in ancient Khmer inscriptions,
- and thereby contribute to the reconstruction, readability and preservation of these valuable historic documents.

In doing so, challenges arise that can only be addressed by the use of new concepts and methods. This is caused by the following facts:

- The inscriptions usually exist as 3D data.
- Their size or condition is sometimes problematic.
- The number as well as the complexity of the characters is large. In regard to the possible combinations of consonants, sub-consonants and dependent vowels, they are comparatively huge.

The aim of this research project is to overcome these challenges and to contribute to the filling the above mentioned gaps. For this purpose, the original data of the inscriptions was acquired by a 3D scanning, then processed with image processing methods and transformed into a 2D representation, which could then be further analyzed. Single characters were isolated, and methods for their identification were developed. To identify the characters, their topological

and geometrical structures were used. The features of the characters were captured by indices that are easy to compute. The identification of a given character was done by repeated filtering. The methods were developed by the example of the Khmer script of pre-Angkorian period but it can be extended to be used on Khmer scripts of other periods. As it does not exist Unicode for pre-Angkorian period script, a digital code was developed. Based on the designed mathematical methods and algorithms, a software tool was developed with which ancient Khmer characters and inscriptions can be processed and analyzed. This software tool is a first experimental version, which will be expanded in a further project for general use.

Acknowledgments

To start with, I would like to express my heartfelt thanks to my supervisors, Prof. Dr. Dr. h.c. mult. Willi Jäger and Prof. Dr. Dr. h.c. mult. Hans Georg Bock, for giving me an opportunity to do my PhD research project in Interdisciplinary Center for Scientific Computing (IWR), Faculty of Mathematics and Computer Science, Heidelberg University. Most importantly, I express my deep gratitude to Prof. Willi Jäger for his advice and guidance to me to develop the mathematical models in my PhD research. Without his supervision and his putting full trust in me, the accomplishment of my PhD research would not be feasible.

This research was supported by Erasmus Mundus Mobility with Asia (EMMA) Program and Gerda Henkel Foundation's Lisa Maskell Fellowship Program. I am very thankful to all sponsors for their grants to my PhD research.

I very much thank Ministry of Culture and Fine Arts of Cambodia for giving the permission to scan Khmer inscriptions at Angkor Conservation Center (ACC), Siem Reap, Cambodia and thank the colleagues at ACC for coordinating the scanning operation.

A special thanks to Dr. Michael J. Winckler for his suggested topic of this research and discussion and also deep thanks to Dr. Susanne Krömker for her discussions and providing me a privilege to access the facilities in computer graphics laboratory, IWR. Additionally, I would like to thank Dr. Hubert Mara for his Gigamesh software framework.

Last but not least, a sincere thanks also goes to Dr. Pheakdey Nguonphan (Phady) for his discussions and recommendations to me during my PhD study and Prof. Phal Des for encouraging me to do my PhD in Heidelberg University. Many thanks to my unforgettable dearest friend, Pinyo Yonthanthum, for sharing his time to discuss all kinds of topics with me and for all the fun we had in Heidelberg. I also very much thank all of my friends in IWR for their help.

Finally, this thesis is dedicated to my father CHHE Bunsreng and my mother TEA Cheangin.

List of Abbreviations

Abbreviation	Meaning
2D	Two dimensional space
3D	Three dimensional space
ACC	Angkor Conservation Center
DOF	Depth of Field
FOV	Field of View
ICP	Iterative Closest Point
IWR	Interdisciplinary Center for Scientific Computing
KIAST	Khmer Inscription Analysis Software Tool
OCR	Optical Character Recognition
RGB	Red- Green- Blue
UNESCO	United Nations Educational, Scientific and Cultural Organization

List of Notations

Symbol	Meaning	Defined in
a	Agent	section 3.5
$C_{1,s}$	Transition cost of the agent from state s to initial state	section 3.5
$C_{n,s}$	Transition cost of the agent from state s to desired state	section 3.5
C_s	Total transition cost of the agent traversing across state s	section 3.5
CON	A set of consonants	section 4.2
$domain_{h_1}^{L_u}$	Domain between h_1 and L_u	section 4.2
$domain_{L_u}^{L_l}$	Domain between L_u and L_l	section 4.2
$domain_{L_1}^{h_2}$	Domain between L_1 and h_2	section 4.2
ds	Arc length	section 4.1.4
D	Domain	section 3.2
DV	A set of dependent vowels	section 4.2
$E[X^2]$	Expected value	section 3.3
$f[p,r]$	Function representing the curvature distribution of the curved edge	secion 4.1.4
$f^*[p,r]$	Smoothing function of $f[p,r]$	secion 4.1.4
$F(x,y)$	Gray-scale image	section 3.3
$G(V,E)$	Graph of a set of nodes V and a set of edges E	section 4.1.2
$G(x,y)$	Segmented image/ Binary image	section 3.3
$I_D^r(p)$	Integral invariant at point p in domain D with respect to a ball B_r	section 3.2
l	Minor arc of the disk B_r	section 4.1.4

L_l	Lower bound line segment	section 4.2
L_u	Upper bound line segment	section 4.2
$M(i, j)$	Matrix	section 3.4
$n(p)$	Normal vector at point p	section 4.1.4
pa^a	Path of agent a	section 3.5
pa^*	Optimal path	section 3.5
P	Pixel	section 3.4
P_b	Black pixel	section 3.5
Pr_i	Probability of image intensity at level i	section 3.3
r	Radius of disk/ball B_r	section 4.1.4
s_1	Initial state	section 3.5
s_n	Desired state	section 3.5
s_i	State i	section 3.5
sc	Parameter for smoothing the edge segment γ	section 4.1.4
scv	Parameter for smoothing the value of function $f(p, r)$	section 4.1.4
SUB	A set of sub-consonants	section 4.2
$t(p)$	Normed tangent at point p	section 4.1.4
th^*	Optimal threshold	section 3.3
$th(x, y)$	Threshold of a pixel(x,y)	section 3.3
TFC	Topological feature of the character	section 4.1.4
$\mathbf{V}_{TFC}$	Topological feature tuple	section 4.1.4
$\gamma(s_p)$	Curve of the character edge	section 4.1.4
ϵ	Parameter to decompose value of function f to obtain the sequence	section 4.1.4
θ	Central angle subtended by minor arc l	section 4.1.4
κ	Curvature value	section 3.2
μ	Mean value	section 3.3
σ	Standard deviation	section 3.3
σ^2	Variance	section 3.3
$\chi_D(x)$	Characteristic function	section 3.2
ω	Class probability	section 3.3
$\mathcal{L}_h$	A set of horizontal blank space line	section 4.2
$\mathcal{L}_v$	A set of vertical blank space line	section 4.2
$\mathfrak{R}$	Feature vector	section 3.2

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The main aim of this investigation is the development of mathematical and computational methods to process and analyse Khmer inscriptions which have remained more or less preserved throughout the turbulent history of Cambodia. The inscriptions date back to different periods of the Khmer culture and are important historical documents. Quite often the inscriptions are in bad condition due to adverse influence of the weathering, pollution, deliberate damage or acts of war.

As always, in the analysis of data, the quality of the data and the questions posed are important factors in choosing the methodical approach. Here we are not dealing with text documents written in the modern Khmer font on paper, which in many situations of modern daily life have to be recorded and analyzed with tools of IT, but we are concerned with historical documents on stones, where all available information, also the spatial one, has to be used to get the best out of noisy and often incomplete data. This fact is requiring IT methods to retrieve from this data the information on the original inscriptions, to provide methods of image processing to derive 2D representations of the original 3D engravings, representing letters and symbols. Filtering techniques identifying characters and derived combinations have to be designed. A basis for text analysis has to be developed.

Following the very positive experiences obtained in research at the Interdisciplinary Center for Scientific Computing, Heidelberg University, for cuneiform tablets and inscriptions on Jewish tombstones[45] [1], we started this project, which we are going to describe in this section.

1.1 General Description of the Problems

Khmer inscriptions are the main sources of Cambodia's ancient history [82]. The studies of the information in Khmer inscriptions and the inferences from this information offer a wealthy

knowledge of certain facets of ancient Khmer civilization including literature, art, religion, the way of life and society (social and political organization), etc. in the ancient times [25] [26].

Khmer inscriptions are well studied in history and linguistics. However, there is no Khmer inscription research in scientific computing. Computational methods can provide useful tools to restore and analyze the inscriptions, available as digital versions of stone-rubbing or as 3D scans.

Quite often the state of the inscriptions is bad due to the exposure to weathering, pollution, deliberate damage or acts of war. The size, eroded surfaces, the large number of characters and their combinations are causing difficulties in the analysis of the inscriptions.

- *Size:* the fact is that most of Khmer inscriptions are engraved on big stones. The preservation of the whole inscriptions is difficult because (1) the limitation of camera lenses to capture the small script when we take photograph and (2) the size of paper when we do stone-rubbing. Often, script data may be lost during stone-rubbing. 3D scanning can be an approach to solve this difficulty.

- *Eroded surfaces:* they lead to incomplete data of script and to noisy data. The geometric structures of the script may also be destroyed. Digital restoration of the inscriptions and making them readable again, at least for experts, are challenges to mathematics and computer science. The eroded surfaces of two inscriptions are shown in figure 1.1 and 1.2 (p.3).

- *Script of ancient Khmer:* it is classified into three periods– pre-Angkor, Angkor and post-Angkor. In each period, the total number of characters are quite different while the shapes of characters quite similar. In general, the script in each period consists of four different sets of characters: consonants, sub-consonants, independent vowels and dependent vowels. Additionally, the derived combinations of consonants, sub-consonants and dependent vowels have complex structures which are difficult to be identified.

- *The styles of engraving:* they produce the variety of shapes and sizes of characters.

Figure 1.1: The eroded surface of the inscription at the door jamb of the Preah Vihear temple[53]

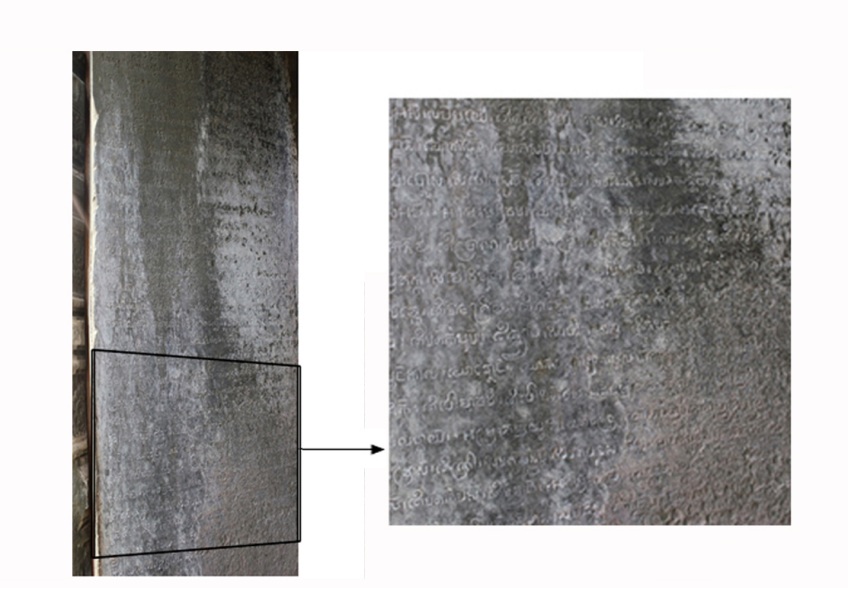


Figure 1.2: The eroded surface of the inscription engraved at the door jamb of the Banteay Srei temple (photo by prof. W. Jäger, 2012)

1.2 State of the Art

1.2.1 3D Scanned Data

3D scanning is an accurate method of measuring the geometry of physical objects. The 3D scan data produced by 3D scanners is a point cloud of geometric samples on the surface of the object. For 3D scan data, the dimensions of the objects such as length, width, height, volume, surface area, etc. can be determined. So 3D scan data outweighs 2D data in preserving more geometric information of the objects. 3D data models are used to solve cultural heritage problems, for example, script visualization of both cuneiform tablets and inscriptions on Jewish tombstones [1] [45], a partial reconstruction of the Banteay Chhmar temple [68] and the reconstruction of Shiva statues located in Kor Ker temple [2]. Figure 1.3 shows reassembled stones of the Banteay Chhmar temple, using 3D data models and figure 1.4 (p.5) shows the script visualization of cuneiform tablets.

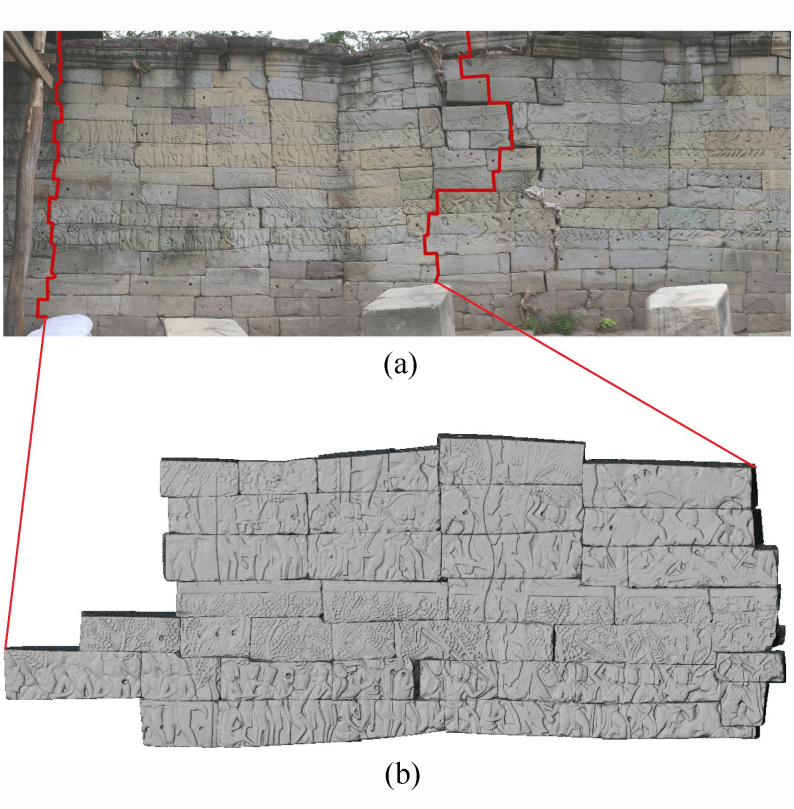
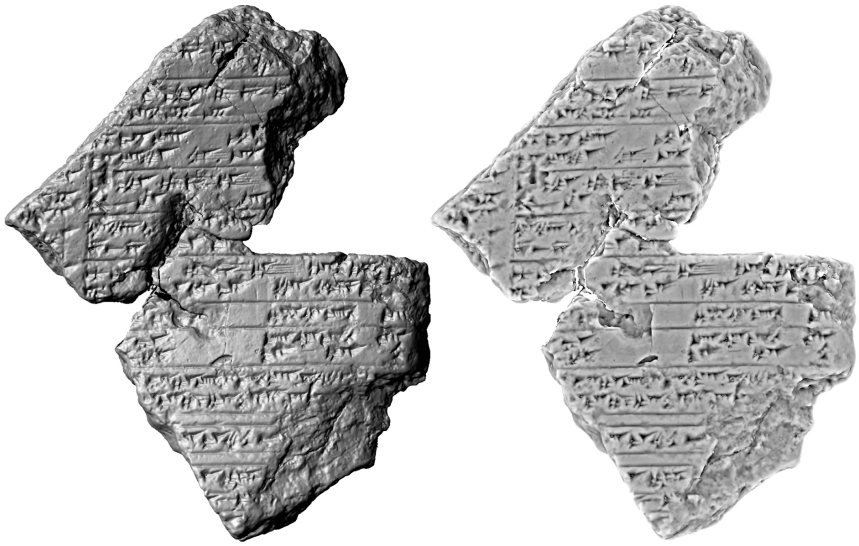


Figure 1.3: (a) The photograph of a wall at the Banteay Chhmar temple. (b) The reassembled stones of the temple wall bounded in red, using 3D data models [68].



(a) (b)

Figure 1.4: 3D models of conjoined cuneiform tablets VAT no. 9927 and 9970 (a) with virtual illumination. (b) with a gray-scale texture map computed by using multi-scale integral invariants [45].

1.2.2 Modern and Ancient Character Recognitions

The character recognition is one domain of object recognition. The increasing demand of paperless office spurs optical character recognition (OCR) systems to development. Digit and signature recognition systems are widely used in bank industry.

Character recognitions consist in recognition of characters from document images and conversion of them into editable text format, which make documentation and information distribution easier. To recognize the characters, we need to capture the distinctive characteristics of the characters. It is called feature extraction. It is required that an individual character has to be represented by a unique feature.

Methods such as moment, histogram, Fourier and wavelet descriptors, Hough transform, topological and differential geometric concepts [17][28] are used to extract features for various scripts.

Generally, in object recognition, a shape of a physical object is represented by regions. The boundary and skeleton of the region can also represent the shape. Features can be extracted from external representation when we are interest in shape characteristics of the region or from internal representation when we are interest in regional properties such as color, texture, brightness, etc. [5] [17] [28]. In OCR, objects mean characters. Characters can be represented by regions, the boundaries of regions, and/or the skeletons of regions.

Once the characters are characterized by mathematical models, known as feature vectors, the final decision is to classify these features into classes based on pre-defined labels. Decision-theoretic and structural approaches can be applied (see more details in chapter 12 [17]). The decision-theoretic approach is concerned with statistical and machine learning methods [21] [22] [85] while the structural approach is concerned with attributed string matching or edit distance [4] [44] [47] [77] [78] [84].

Over the last decades, the work on character recognition system has become very mature [5] [49] [50] [52] [64] [69]. A lot of methods have been applied to various modern scripts such as Latin [9] [10] [15] [19] [63] [71] [76] [87], Japanese [32] [66] [72], Chinese [29] [35] [36][41] [76] and some kinds of Indian scripts. Also, some other scripts in Asia such as Khmer and Thai are found in [6] [48] [75] [80]. Nevertheless, open problems still exist in recognitions of handwritten characters because of the variety of hand writing.

Other problems still exist as it comes to the recognition of ancient scripts like cuneiform and calligraphy, which appear in tablets, stones, fan-palm leaves and wood. The major problems are the distorted and damaged geometric shapes and the destruction of texture and color of the materials. Research work on digital restoration of ancient document images and feature extractions of various ancient scripts are presented in [16] [31] [45] [46] [55] [88] [90] [91] . However, with the demand for digital preservation of historical documents, the research on further ancient script recognition is still necessary.

1.2.3 Khmer Inscriptions

The background of Khmer inscriptions and the study of the Khmer inscription in the area of history and linguistics are described in this section.

The ancient Khmer started to record information on the stones in the 1^{st} century AD. Approximately 1300 inscriptions were officially registered in the inventory of Khmer inscriptions. The discovery of new inscriptions accounts for the increasing statistics every year [27]. Khmer inscriptions are found in French museum and in some countries which used to be Khmer empire's territory– Thailand, Lao and South Vietnam [82]. In general, the inscriptions were placed in the temple compounds such as walls, doorjambs, the bases of statues, stelas or pillars of the temples. Figures 1.5 and 1.6 (p.7) show the inscription engraved in the doorjamb of the Preah Vihear temple and figure 1.7 (p.8) displays the inscription at the Benteay Srei temple in Siem Reap. The inscriptions, nevertheless, have not been in situ because of preservation in the museum or looting. Figure 1.8 (p.8) displays well-preserved Khmer inscriptions in the Angkor Conservation Center (ACC), Siem Reap, Cambodia. The center is the oldest center established in the 1900s to store the cultural heritage of the Khmer empire.

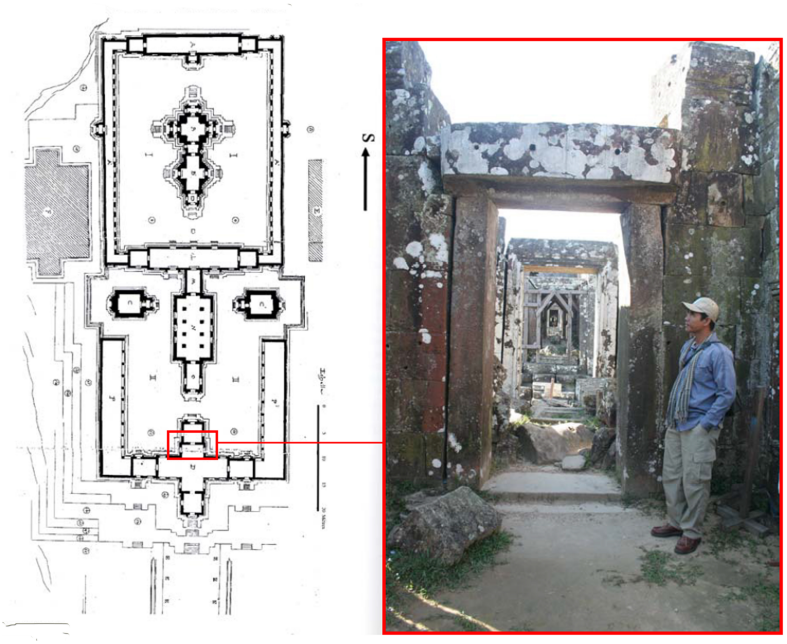


Figure 1.5: Map of inscription K380 at the doorjamb of the Phrea Vihear temple[53].

Figure 1.6: The campaign of 3D scanning of inscription K380 at the doorjamb of the Phrea Vihear temple[53].

Figure 1.7: The inscription at the doorjamb of the Banteay Srei temple in Siem Reap, Cambodia (photo by prof. W. Jäger, 2012).

Figure 1.8: The inscriptions are well-preserved in Angkor Conservation Center(ACC), Siem Reap, Cambodia.

Khmer epigraphy was initiated in 1890 by the Dutch and French. Louis Finot (1864-1935) published six volumes of Khmer inscription books [13]; Georges Coedès (1886-1969) five volumes [8]. Furthermore, they are followed by Pos Saveros [37] [38] [39] [40], Long Seam [42] [43], Jenner N. Philip [57] [59], Vong Sotheara [81] [82] [83] and Horm Chhaily [7]. The United Nations Educational, Scientific and Cultural Organization (UNESCO) also published four volumes of Khmer inscriptions [79].

However, these publications are in the disciplines of history, literature and linguistics.

1.3 Project Aims and Solution Approach

In our PhD research project, we mainly focus on developing methods from mathematics and computer science to process and analyze Khmer inscriptions.

Due to the problems as described in section 1.1, they influence choosing the appropriate approach to develop the methods. We thus confine our study to pre-Angkorian inscriptions of ancient Khmer. We focus on the pre-Angkorian inscriptions in which the script data can be readable. We also consider a certain variation of script shapes. However, this variety of shapes have to maintain topological and geometrical structures with regard to the standard shapes which are defined by history and linguistic scientists. The approaches to our confined study are described as follows.

(A) Image Processing Methods:

- Starting from 3D scanning of original Khmer inscriptions at Angkor Conservation Center (ACC), Siem Reap, Cambodia, we apply a curvature-based method, Gigamesh, which were developed at IWR. This method extracts the information on the inscriptions. From the Gigamesh software framework, we derive 2D representations of the original 3D engravings.
- The following step is to reduce the obtained information from script extraction to essential script data, using global and local thresholding methods of binarization. In some circumstances, image enhancement techniques are implemented to improve the geometric shapes of the obtained script data and to reduce noise from the representations.
- We then separate the script data in the representations into a set of isolated characters. Based on the characteristics of Khmer writing system, the script apparently stays in lines. Between each line, there exists curvilinear blank space. We find a path with

optimal traverse cost of two points through blank space to segment the script data into a set of isolated characters.

- We take skeletons to denote the topological and geometrical structures of the characters. To obtain the skeletons of the characters, Zhang and Suen's method of thinning operation from image processing is applied.

(B) Image Analysis Methods:

- We develop new methods to identify the characters. Concepts of topology and geometry are used to extract features of the characters.
- Ancient Khmer characters are considered as compositions of line and curve segments. These sets of line and curve segments can be characterized by their topologies. Curve segments are characterized by their positions and curvatures.
- Undirected connected graphs are used to describe the topologies of the characters. We extract topological features of the characters from the degrees of nodes. Based on the topological features, the ancient Khmer characters can be identified and grouped into classes.
- For each topological feature-based class, we define another group of features called geometric features to differentiate the characters in the class if there exist more than one character. The geometric features are extracted from curvatures of the character curve segments. We do not need the precise values of curvatures. We are only interested in the signs of the curvatures denoted by +1 (convex), 0 (flat) or -1 (concave). Differentiability is too strong for this analysis since the curves arising in our applications are not smooth. In this purpose, concepts of integral geometry are used. For a point p of a curve segment γ and a disk of the radius r, centered at p, there exists a sub-curve of γ passing through p and separating the disk into two parts. Determination of two separated parts of the disk is good indicator for convex, concave and flat. As a result, the geometric features of the characters are the finite sequences of the permutations of +1, 0 and -1, respectively.
- We develop an algorithm to replace the sub-sequences such as (0,-1), (0, 1), (0, 0), (-1, 0), (1, 0), (-1, -1), (1, 1) with -1, 1, 0, -1, 1, -1, 1, respectively, in the sequences. We finally obtain the unique sequence which acts as the filter for each character.
- We design an algorithm to decompose the combinations of consonants, sub-consonants and/or dependent vowels. The algorithm uses two predefined lines– lower and upper bound lines– to separate the region of consonants, sub-consonants and dependent

vowels in the text region. Based on lower and upper bound lines, we can track the information about the region of consonants, sub-consonants and dependent vowels in the combinations. Then, the combinations are decomposed according to the tracking region information.

1.4 Summary of Research Results and Contributions

The approaches described in section 1.3 give us significant results. The summary of the results and contributions of our PhD research project are provided below.

Summary of Research Results

From the image processing methods defined in our thesis, we can obtain the compact topological and geometrical structure of the characters from the inscriptions. As for image analysis methods defined in our thesis to identify the characters, the results show that the filtering technique based on the topological features can classify the characters into 15 classes. Among 15 classes, 4 classes are unique, that is, there exists only one character in the class. Then, for the geometric features, they can identify the characters which have the same topological structures. However, the accuracy of character identification also depends on a certain variation of character shapes. In our methods, we use the sign of the curvature to identify the characters. Therefore, for some characters having the same topological structures, if the curvature is very small or missing in some parts of the character structures, they will be mixed up with other characters.

Research Contributions

Our research aim is to develop mathematical and computational methods to process and analyze Khmer inscriptions in the pre-Angkorian period. The research is concerned with mathematics, computer science and cultural heritage. Our contributions are listed as follows:

(1) We provide the curvature-based method to retrieve the information on the original 3D Khmer inscriptions and the methods of binarization and skeletonization to extract the topological and geometrical structures of ancient Khmer characters.

(2) We also provide the line segmentation method to separate the script data on the inscriptions to a set of isolated characters.

(3) We prove that concepts of topology and integral geometry can be used to discriminate the characters. We define two types of features– topological and geometric features– for the

characters. Topological features are extracted from topological structures of the characters, whereas geometric features are extracted from the sign of curvatures of character curve segments.

(4) We give a simple robust algorithm to separate the derived combinations of consonants, sub-consonants and dependent vowels. This algorithm uses two pre-defined lines to separate the region of consonants, sub-consonants and dependent vowels in the text region. Then the combination is decomposed based on the information of these regions.

(5) we develop vector graphics font of ancient Khmer characters in pre-Angkorian period. This font can be used to support the functionality of converting the results of character identification into editable file format.

(6) As the final result, the designed mathematical methods and algorithms are used to process and analyze Khmer inscriptions.

(7) The developed software based on the designed mathematical methods and algorithms is replacing the preservation of the inscriptions by stone-rubbing and the transcription and transliteration of the inscriptions by word processors.

In the following chapters, the details of the work on 3D image processing, analysis and software development of Khmer inscriptions are described.

1.5 Structure of Thesis

The structure of this thesis is arranged as follows:

Chapter 2 gives the description of the ancient Khmer writing system. The chapter starts with the origin of Khmer inscriptions and primitive script, used by the ancient Khmer. Next, the direction of writing in the ancient Khmer writing system is explained. At last, chapter 2 mentions the ancient Khmer characters in linguistics. We discuss linguistic method to use ancient Khmer characters to record the sounds of three languages– Sanskrit, Pali and Khmer. This is the principle to assist in discovering mathematical methods to characterize ancient Khmer characters.

Chapter 3 goes into the details of image processing methods of Khmer inscriptions. It covers the process of the 3D data scanning of Khmer inscriptions and the introduction of a curvature-based method to retrieve the information on the inscriptions. Then, the methods of thresholding are given, reducing the obtained information to essential script data. We discuss two methods

of thresholding for Khmer inscriptions. It is followed by a thinning operation. The thinning operation reduces script data to characteristics of topological and geometrical structures of the script. We also present the character line segmentation method to segment script data in 2D representations of the inscriptions into a set of isolated characters. Finally, in the last part of this chapter, we show image enhancement method which needs to further improve the structures of the characters.

Chapter 4 presents the image analysis methods of Khmer inscriptions. In this chapter, we explain novel mathematical methods to characterize the ancient Khmer script. Concepts of connected graphs, topology and integral geometry are used for character identification. Subsequently, we thoroughly discuss the development of features of the characters. We also present a new algorithm to decompose the combinations of consonants, sub-consonants and/or dependent vowels.

Chapter 5 gives the examples of applying the methods described in chapter 3 and 4 to the ancient Khmer script.

Chapter 6 describes a software framework to implement the methods from chapter 3 and 4. The software is called *Khmer Inscriptions Analysis Software Tool* (KIAST). KIAST is based in Matlab. It consists of four important parts– image processing, image analysis, visualization and image enhancement. The interface of KIAST is also designed for user-friendly interactions.

Chapter 7 provides the conclusion and the outlook of our research work.

Chapter 2

Basic Facts in Khmer Epigraphy

The aim of chapter 2 is basically to discuss the ancient Khmer writing system such as the script, the direction of writing and how to use the script to represent speech sounds of three languages– Khmer, Pali and Sanskrit. In the first section, we start with a presentation of the script used in the ancient Khmer time.

2.1 Ancient Khmer Script

The primitive of Khmer script was descended from the Brahmi script, which was used in southern India and south-east Asia during the 5th and 6th centuries AD [30]. The Brahmi script in the inscription of Asokar and standard Brahmi alphabet are shown in figure 2.1 and 2.2 below.

Figure 2.1: The Brahmi script in the Asokar inscription.

Figure 2.2: The standard Brahmi alphabet.

The inscription K600 dated 611 AD is considered as the oldest dated Khmer inscription using ancient Khmer alphabet for writing Khmer language. It was found at Angkor Borei in Takeo Province south of Phnom Penh[7] [82]. The inscription is shown in figure 2.3.

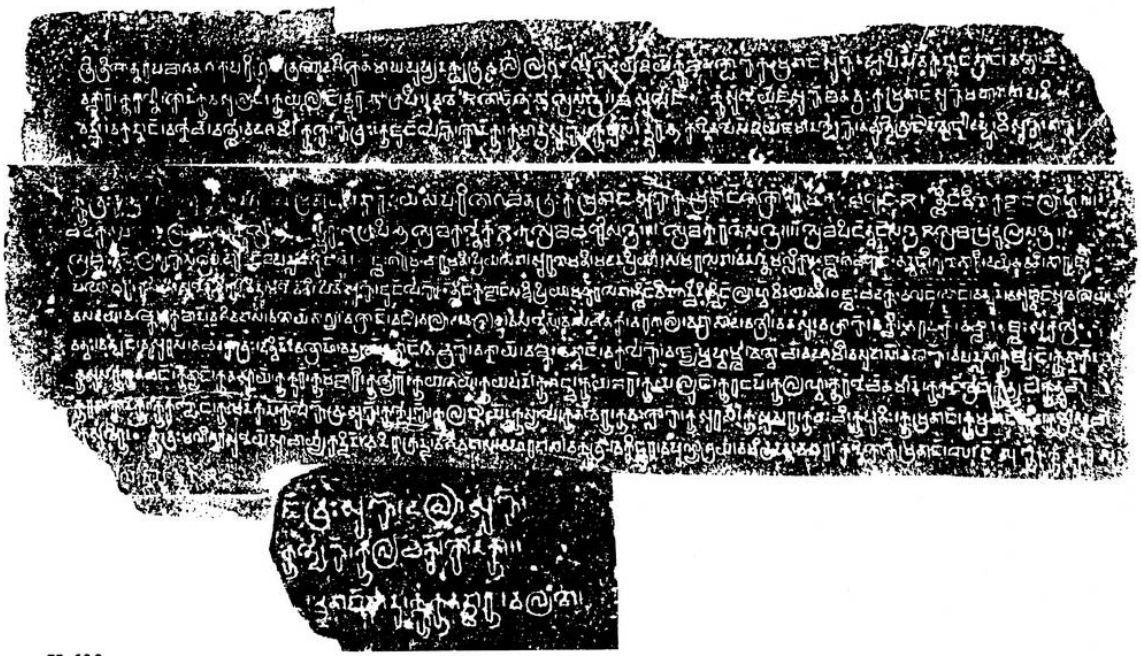


Figure 2.3: The oldest dated inscription K600 [58] [82]

The Khmer alphabet developments are grouped into three main periods– pre-Angkorian (1st century-802), Angkorian (802-1432) and post-Angkorian period (1432-1863) [79] [82]. During these three periods, several geometric shapes of the characters changed and more characters were added to the script. According to [79] [82], the ancient Khmer alphabet consists of five sets– consonants, subscripts or sub-consonants, dependent vowels, independent vowels and symbols to represent numbers. The alphabet in pre-Angkorian period is composed of 33 consonants, 33 sub-consonants, 12 independent vowels, 13 dependent vowels and symbols for representing numbers. The ancient Khmer alphabet in each period is depicted in appendix A.

2.2 The Direction of Writing

Traditionally, the direction of ancient Khmer writing is from right to left and from top to bottom. The text stays in row or horizontal direction. One could write text in more than one columns per row. The text lines in columns and rows are separated from each other by blank space. However, characters were still read from right to left and from top to bottom. The Khmer inscriptions in figure 2.4, 2.5 and 2.6 display the text written in rows and columns.

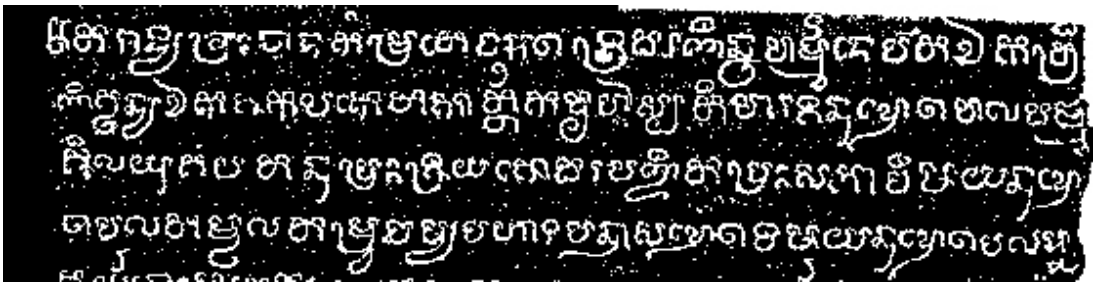


Figure 2.4: Text is written in one column per row.

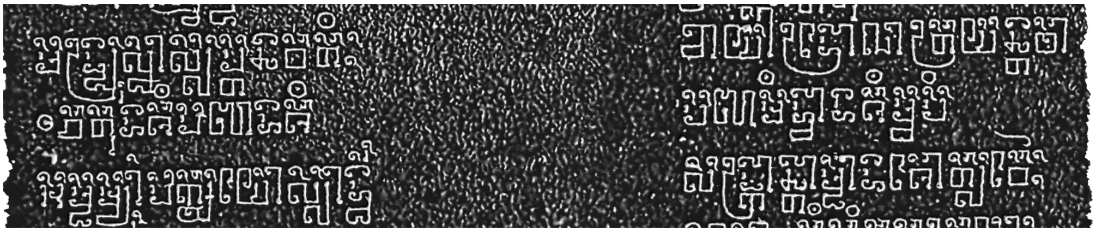


Figure 2.5: Text is written in two columns per row.

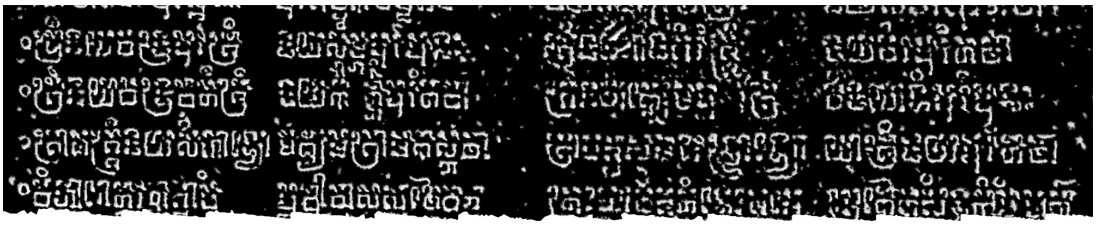


Figure 2.6: Text is written in four columns per row.

2.3 How to Use the Script

As mentioned in section 2.1, the Khmer ancient alphabet is classified into five groups– consonants, sub-consonants, independent vowels, dependent vowels and some symbols.

The alphabet was used to write three languages– Khmer, Pali and Sanskrit. The speech sounds of these languages can be represented by one or two consonants, independent vowels or the combinations of consonants, sub-consonants and/or dependent vowels. For the combinations, they must have a consonant connected with dependent vowels and/or sub-consonants. It is impossible to combine dependent vowels and sub-consonants without the presence of any consonant. In the ancient Khmer writing system, it exist the rules to place dependent vowels and sub-consonants in combinations. The positions of dependent vowels can be above, below and on the left and right sides of the consonants. As for sub-consonants, their positions are the same as dependent vowels' positions, except the position being above the consonants. The sample text of inscription K49 is displayed in figure 2.7 and the combinations are shown in figure 2.8.

We use some abbreviations as given in table 2.1 to explain the combinations. The combinations of consonants, sub-consonants and/or dependent vowels have complex geometric structures. There is no space that visually separate between consonants, dependent vowels and sub-consonants.

Abbreviation	Meaning
C	Consonant
INV	Independent vowel
D	Dependent vowel
S	Sub-consonant

Table 2.1: The abbreviation for the types of characters.

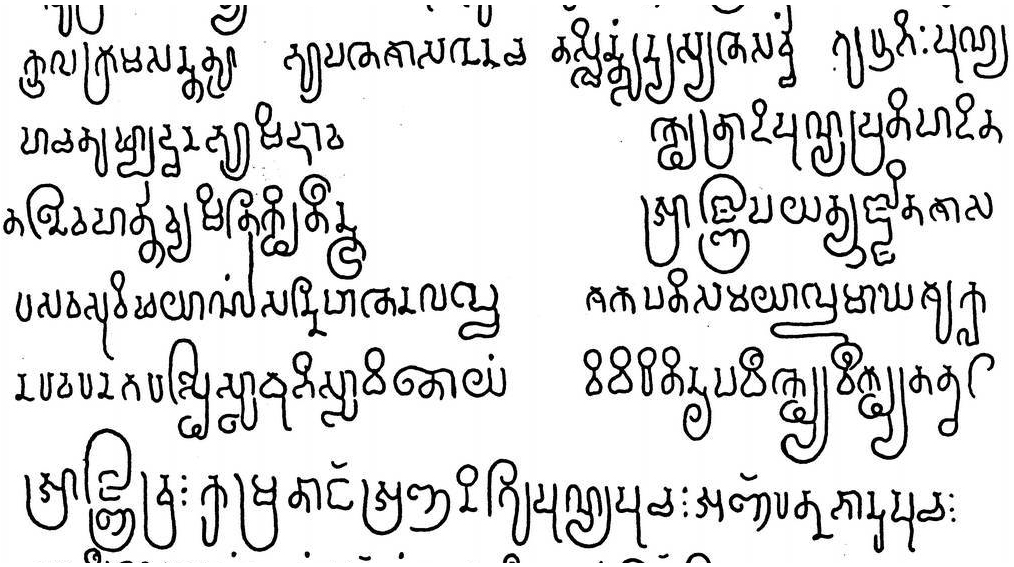


Figure 2.7: The sample text of inscription K49.

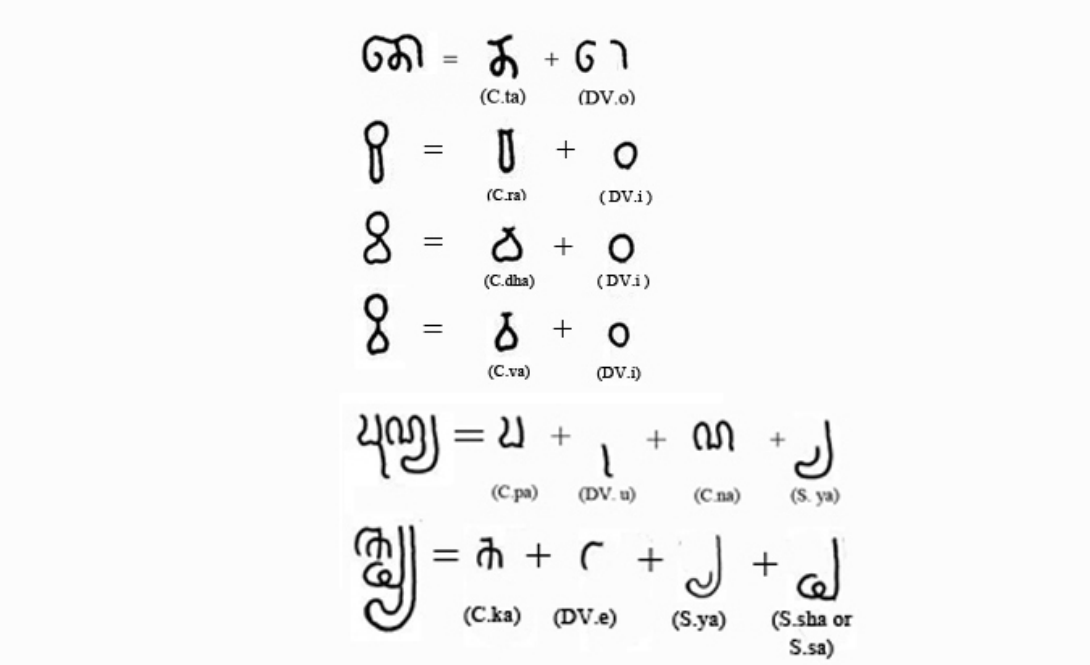


Figure 2.8: The combinations of consonants, sub-consonants and/or dependent vowels to represent the speech sounds in Khmer, Pali and Sanskrit languages.

Therefore, it is difficult to characterize the combinations. The first approach to solve this difficulty is to develop an algorithm which can separate dependent vowels and sub-consonants from consonants. Another possible approach is to treat this kind of the combinations as new characters. However, the latter produces a large set of features for characters because of the

union and intersection operations of three sets– consonants, sub-consonants and dependent vowels. In this dissertation, the first approach is considered. We are explaining the algorithm to separate the combinations in the chapter 4, section 4.2.

2.4 Summary

To sum up, chapter 2 gave discussion on the ancient Khmer writing system. We described the ancient Khmer script and the rules of using them to write Khmer, Pali and Sanskrit languages. We also introduced how to write and read the text in the inscriptions. What we presented in this chapter is important for character segmentation and characterization of ancient Khmer characters by mathematical methods. We consider the script in the pre-Angkorian period because it is the oldest script and our application is also the first application of Khmer inscriptions. In chapter 3, the image processing methods of Khmer inscriptions are presented.

Chapter 3

Image Processing Methods – Transfer of 3D Inscription Data to 2D Representations

In chapter 3, we present methods to retrieve topological and geometrical structures of the script from the inscriptions. In the first section, we present 3D scanning of Khmer inscriptions and the method to transfer 3D inscription data to 2D representations. We then give thresholding methods to reduce data in 2D representations to essential script data and a thinning operation to reduce script data to the characteristics of topological and geometrical structures of the script. We also give character segmentation to segment the script data in the representations to a set of isolated characters. At last, we present the image enhancement techniques to improve the structures of the script.

3.1 3D Scanning of Khmer Inscriptions

We use the optical 3D scanner to capture 3D models of Khmer inscriptions like [45] [68] [86]. In principle, optical 3D scanner produces range images of the object, having pixel value from [0-255] that correspond to the distance to the object surface in the specific *field of view (FOV)*. The range images are then stitched together to form a 3D-model of the object. The stitching process is known as registration. Iterative Closest Point(ICP) method, which minimizes the least square error between range images, is the most prominent technique for registration [45].

In scanning, working distance of optics is defined by the *depth of field (DOF)* and the *focal plan*. The working distance is usually fixed when fixed focal length optics are used. The fixed focal length optics can give more accuracy than zoom optics. The working distance is between

50 centimeters and 2 meters [45]. Characteristics of optical 3D scanner is shown figure 3.1.

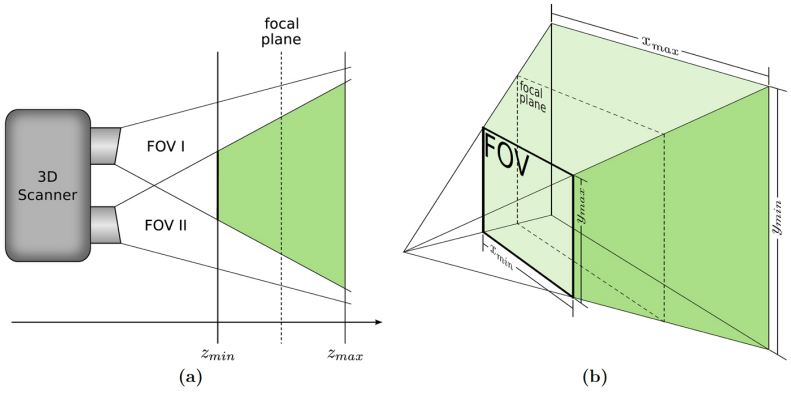


Figure 3.1: (a) The side view of 3D scanner and its two *FOVs*. (b) Perspective drawing of the *FOV* and its dimensions [45] (p.17).

From the figure 3.1, the optical 3D scanner consists of two cameras accounting for two *FOVs– FOV I and FOV II*. *FOV I* is *FOV* of camera I and FOV II is either the FOV of camera II (steros) or the projection of structured light. The frustum in green color is the space in which an object can be captured. The space is defined by:

$$x_{min}, y_{min}, z_{min}, x_{max}, y_{max}, z_{max}$$

The optical 3D scanner– *Breuckmann smartSCAN HE* with *FOV-475mm* is selected to acquire 3D models of Khmer inscriptions. The selection of the *Breuckmann smartSCAN HE* is based on the experiments in [45] where the various optical 3D scanners from several manufacturers were evaluated by criteria such as time for scanning process, accuracy measurement and the price in the market. The results showed that the *Breuckmann smartSCAN HE* is the best solution because the 3D-model of the *Breuckmann smartSCAN HE* displays the detailed information of the script much more distinct. The results of the experiments from [45] are shown in table 3.1 and figure 3.2 (p.23).

The ***Breuckmann smartSCAN HE*** is the structured-light and stereo optical scanner that uses fringe projection to measure precise 3D coordinates. It has two cameras– left and right cameras and one projector to produce the 3D-model. *Breuckmann smartSCAN HE* provides various types of lenses or *field of view (FOV)*– 60, 150, 475, 825 and 1000*mm*. The lenses have to be adjusted to the size of the objects. The smaller the objects are the smaller lenses are used. For example, for the cuneiforms and fingerprints, 60 and 150*mm* are recommended [45] and for the large objects such as statues or temples, 825 and 1.000*mm* are suggested [62]. The *Breuckmann smartSCAN HE* is also shown in figure 3.3 (p.23).

3D Scanner		Resolution *[DPI]*			
		Average	Maximum	Quantile	
Manufacturer	Model			1%	99%
Breuckmann	smartSCAN	305,8	501,0	80,2	829,1
Minolta	RANGE5	261,9	198,4	125,6	254,3
Perceptron	Infinite SC	229,0	191,0	95,7	232,0

Table 3.1: Estimated resolutions of 3D Scanners in *DPI* assuming a flat surface with equidistant measuring points

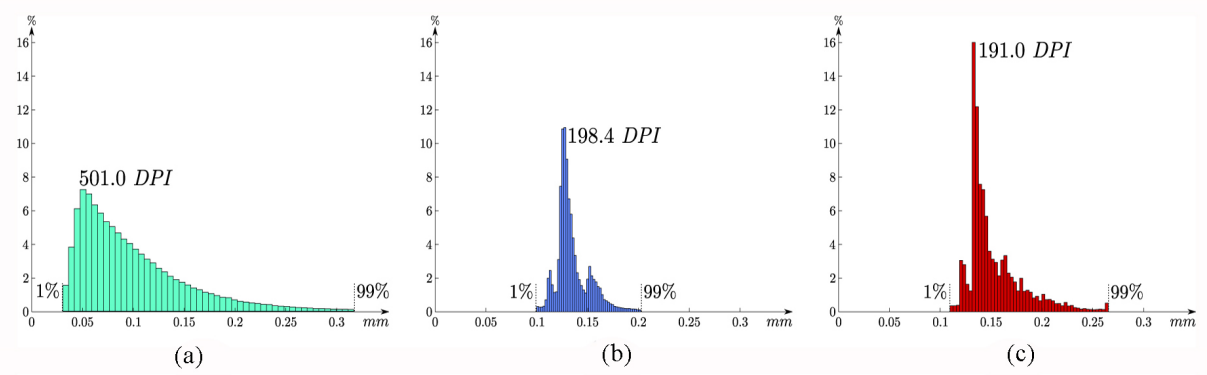


Figure 3.2: Distribution of distances of adjacent measuring points in *mm* for the same object acquired with (a) Breuckmann smartSCAN, (b) Minolta RANGE5 and (c) Perceptron Infinit *SC* 3D scanner. Smaller distance means higher resolution.

Figure 3.3: The structured-light 3D scanner– *Breuckmann smartSCAN HE*.

The investigation into *FOV* during scanning campaign in Angkor Conservation Center (ACC), Siem Reap, Cambodia, 2013 shows that *FOV 475mm* is an optimal choice to capture the script in the inscriptions in terms of resolutions and time of scanning. It takes 30 minutes to 2 hours to scan an inscription. Namely, for each inscription, ten to forty scans are required. The time of scanning relies on the size of each inscription. The scanning campaign of Khmer inscriptions is illustrated in figure 3.4.

Figure 3.4: The 3D scanning operation of Khmer inscriptions in Angkor Conversation Center, Siem Reap, Cambodia, March 2013.

3.2 Transfer of 3D Inscription Data to 2D Representations

Transfer of 3D inscriptions data to 2D representations is to visualize the script data from the inscriptions and represent the script data in 2D representations. To achieve the transformation, we compute the feature vectors of 3D models of the inscriptions. Before the computation of feature vectors, the 3D models of the inscriptions obtained from the scanner need to be cleaned to remove unnecessary parts of the models, align the mesh and optimize the number of vertices to represent the geometric shapes of the inscriptions. It can be done by Optocat which is licensed software together with the *Breuckmann smartSCAN HE 3D Scanner*.

There are further processes to clean up solo vertices and singularities vertices as well as non-manifolds faces as described in [34]. These processes can be manipulated by Gigamesh framework developed in [45], which was developed at Interdisciplinary Center for Scientific Computing (IWR), Heidelberg University.

A curvature-based method is then used to compute the feature vectors of the inscriptions. In this method, the features of 3D inscription model is characterized by the curvature of surface of the model. The curvature of the surface at a point p_i is estimated by a volume integral invariant. The detail of the method is described in [45]. Here, we introduce the concept of integral invariants [60][61] to approximate the curvature of curve in $\mathbb{R}^2$ or surface $\mathbb{R}^3$.

Assume that a $2D$ curve c or $3D$ surface Φ is the boundary ∂D of a domain D in $\mathbb{R}^2$ or $\mathbb{R}^3$. Integral invariant can be computed by

$$I_D^r(p) = \int_{B_r(p)} \chi_D(x)dx \tag{3.1}$$

where $\chi_D(x)$ is the characteristic function. $\chi_D(x)$ is 1 if $x \in$ domain D and 0 otherwise. $B_r(p)$ is the ball of radius r centered at $p \in D$ and Φ. The ball $B_r(p)$ is a disk in $\mathbb{R}^2$ and a sphere in $\mathbb{R}^3$. From the equation 3.1, the volume below the intersection of the surface $\partial\Phi$ and the sphere $B_r^3(p)$ is computed with respect to radius r. Giving a point p_i, the volume $V_r(\partial D := \partial\Phi \cap B_r(p_i))$ indicates the curvature κ_i of surface at p_i. If half sphere is defined as a threshold of a flat segment, curvature κ_i can be mapped to convex (+1), concave (-1) or flat(0).

The curvature κ_i of surface at p_i indicates the elevation of the surface at p_i. The feature vectors $\Re = (\kappa_1, \kappa_2, ..., \kappa_n)$ represent the elevation of the surface of 3D model. The heightmap is used to visualize the surface elevation with respect to the curvature of surface.

From curvature-based method, we retrieve 2D representations of the original Khmer inscriptions as shown in figure 3.5 (p.26). We then reduce the information on the 2D representations to essential script data. The process is explained below.

3.3 Reduction in Information on 2D Representations to Essential Script Data

The 2D representations of the original inscriptions contain various information including essential script data and noise. In this section, we describe the global and local thresholding methods to reduce the information to essential script data.

Thresholding

Thresholding method segments pixels of a gray-scale image $F(x, y)$ into two distinct classes $c = \{c_0, c_1\}$ which $c_1 = \{(x, y), G(x, y) = 1\}$, $c_0 = 1 - c_1$, and $G(x, y)$ is a segmented image.

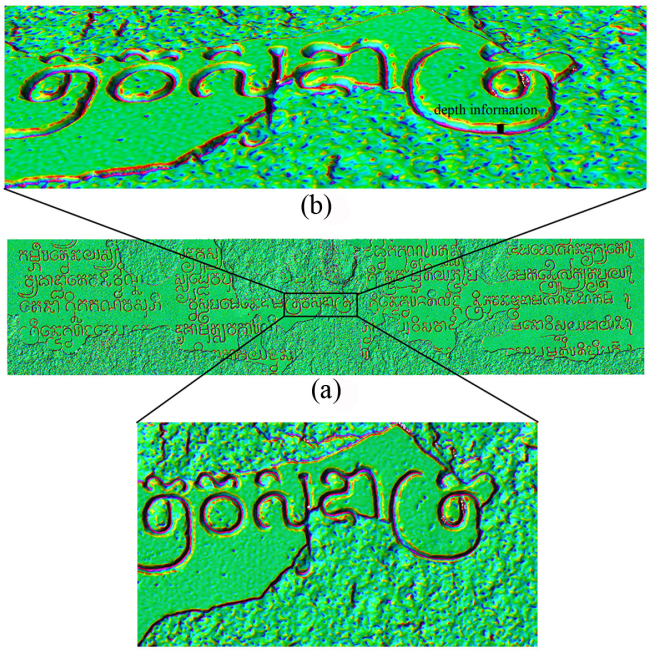


Figure 3.5: (a) The 2D representation of inscription K826 retrieved from curvature-based method– Gigamesh. (b) The visualization of elevation of the surface of inscription K826. (c) A part of script visualization from (a).

The segmented image $G(x, y)$ is defined by

$$G(x,y) = \begin{cases} 1 & if \quad F(x,y) > T \\ 0 & if \quad F(x,y) \leq T \end{cases} \tag{3.2}$$

where $F(x, y)$ is the gray-scale intensity of any point (x, y) in an image. When T is a constant applied over an entire image, the equation 3.2 is referred to as global thresholding. When values of T change over an image, the equation 3.2 referred to as local thresholding in which the value of T at any point(x,y) depends on the properties of its neighborhood [18].

In our research, we adapted Otsu's method for global thresholding[56] and Sauvola's method for local thresholding [67]. Otsu's global thresholding is designed for 2D representations in which pixel intensities of script data are homogeneous and Sauvola's local thresholding is designed for 2D representations in which pixel intensities of script data are inhomogeneous.

Otsu's Global Thresholding Method

Otsu's method– known as histogram-based method find *'goodness'* of threshold and then automatically select an optimal threshold th^*.

In a gray-scale image of the inscription, we consider script data in dark intensity against white background, so the image gray-level pixels $[0, 1, 2, ..., L-1]$ are partitioned into two classes, c_0 denoting $[0, 1, 2, ..., t]$ belonging to script data and c_1 denoting $[t+1, t+2, t+3, ..., L-1]$ belonging to background, at optimal level th^*. To determine th^*, the criteria for class separability in discriminant analysis of statistics [14] is used .

Define σ_W^2,σ_B^2 and σ_T^2 as within-class variance, the between-class variance and the total-class variance, respectively. The objective functions of σ_W^2,σ_B^2 and σ_T^2 are defined as follows.

$$\lambda = \frac{\sigma_B^2}{\sigma_W^2} \quad , \quad \eta = \frac{\sigma_B^2}{\sigma_T^2} \quad , \quad \beta = \frac{\sigma_T^2}{\sigma_W^2} \tag{3.3}$$

where

$$\sigma_T^2 = \sigma_W^2 + \sigma_B^2 \tag{3.4}$$

$$\sigma_W^2 = \omega_0\sigma_0^2 + \omega_1\sigma_1^2 \tag{3.5}$$

$$\sigma_B^2 = \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \tag{3.6}$$

$$= \omega_0\omega_1(\mu_1 - \mu_0)^2 \tag{3.7}$$

$$\sigma_T^2 = \sum_{i=0}^{L-1}(i - \mu_T)^2 Pr_i \tag{3.8}$$

The optimal threshold th^* can be determined by maximizing one of the object functions in (3.3). Taking η, the optimal threshold th^* is determined as

$$th^* = \arg\max_{t\in[0,L-1]}(\eta) \tag{3.9}$$

where

$$Pr_i = \frac{n_i}{n}, \quad Pr_i > 0, \quad \sum_{i=0}^{L-1} = 1 \tag{3.10}$$

$$\omega_o = \sum_{i=1}^{t} Pr_i, \tag{3.11}$$

$$\omega_1 = \sum_{i=t+1}^{L-1} Pr_i = 1 - \omega_0 \tag{3.12}$$

$$\mu_T = \sum_{i=0}^{L-1} iPr_i, \quad \mu_t = \sum_{i=0}^{t} iPr_i, \tag{3.13}$$

$$\mu_0 = \frac{\mu_t}{\omega_0}, \quad \mu_1 = \frac{\mu_T - \mu_t}{1 - \omega_0} \tag{3.14}$$

Here n_i denotes the number of pixels at level i, n is the total number of pixels in input image (2D representation) by $n = n_0 + n_1 + n_2, ..., +n_{L-1}$. ω_0 is class probability of object (script data) c_0 and ω_1 class probability of background c_1.

We formulate the algorithm 3.3.1 to compute the threshold t^* of Otsu's method below.

Algorithm 3.3.1 (Otsu's Algorithm)

1. *Read character input image*
2. *If input image is color image then*
   gray-level image = Round(aR + bG + cB)
   which a, b, c is coefficient of R(Red), G(Green), B(Blue).
3. *Compute histogram and probabilities of each intensity level.*
4. *Set up initial* $\omega_i(0)$ *and* $\mu_i(0)$.
5. *Step through all possible threshold t =1..., maximum intensity.*
   (a) *Update* ω_i *and* μ_i.
   (b) *Compute* $\sigma_B^2(t)$.
6. *Optimal threshold* th^* *corresponds to the maximum* $\sigma_B^2(t)$.

The extractions of essential script data from 2D gray-scale representations (images) using Otsu's method are shown in figure 3.6 (p.29).

Sauvola's Local Thresholding Method

Unlike global thresholding approach, local thresholding approach determines threshold locally, e.g. pixel by pixel, or region by region. Each pixel or region has individual threshold. Sauvola's method defines the threshold $t(x, y)$ of each pixel by computing the local mean $m(x, y)$, standard deviation $\sigma(x, y)$ of the pixel in a small neighborhood– $n \times n$ window. The equation of the threshold $t(x, y)$ is defined by

$$th(x, y) = m(x, y)\left[1 + k\left(\frac{\sigma(x, y)}{R}\right) - 1\right] \tag{3.15}$$

Where $m(x, y)$ and $\sigma(x, y)$ are mean and standard deviation of the pixel intensities in a $n \times n$ window centered around the pixel (x, y). R is the maximum value of the standard deviation (R=128 for a gray-scale document). k is a parameter which takes positive values in the range [0.2, 0.5]. $k = 0.5$ is recommended [70]. The Sauvola's algorithm is formulated below.

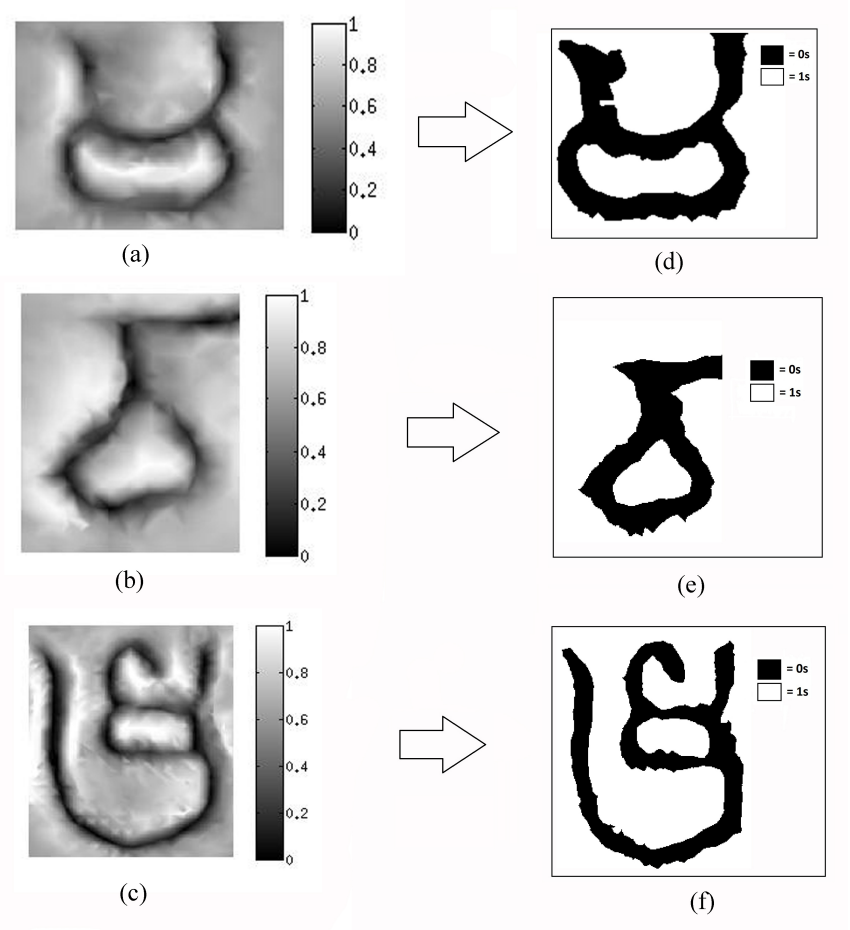


Figure 3.6: The extraction of script data from 2D representations using Otsu's method. script data in (d), (e) and (f) is extracted from (a) with $t^* = 0.5529$, (b) with $t^* = 0.6000$ and (c) with $t^* = 0.0.5216$, accordingly.

Algorithm 3.3.2 (Sauvola's Algorithm)

1. *Read gray-level image $F(x, y)$ which $0 <= F(x, y) < t$ belonging to characters and $t + 1 < F(x, y) < L - 1$ belonging to background.*

2. *Define n (e.g. $n = 10, 20,...$) for a small window $g(x, y)$ which n is a number of row and column of $g(x, y)$*

3. *Define $k = 0.5$ ($k = [0.2, 0.5]$)*

4. *Compute mean values of image $F'(x, y)$ of image $F(x, y)$*

   (a) *Define mean value for a small window $g(x, y)$*

   (b) *$F'(x, y) = (F * g)(x, y)$ where $(*)$ is convolution*

5. *Compute* $\sigma(x, y)$ *standard deviation of image* $F(x, y)$ *by* $\sigma(x, y) = \sqrt{E[X^2] - \mu^2}$ *which* $X = F(x, y), \quad \mu = F'(x, y)$

   (a) *Compute expected value* $E[X^2] = F^2 * g(x, y)$ *where* $(*)$ *is convolution*

   (b) *Compute* $\mu^2(x, y)$ *mean square of image* $F(x, y)$ *by* $\mu^2(x, y) = F'^2(x, y)$

   (c) $\sigma(x, y) = \left(E(X^2) - F'^2(x, y)\right)^{0.5}$

6. *Define* $R = max\,(\sigma(x, y))$

7. *Compute threshold* $th(x, y) = F'(x, y) \times \left[1 + k\left(\frac{\sigma(x,y)}{R} - 1\right)\right]$

8. *Computed segmented image* $G(x, y)$ *by* $F(x, y) > th(x, y)$

The quality of script data, e.g script structures connectivity and noise reduction from 2D representation can be achieved depending on the contrast of 2D representation and predefined window $n \times n$. We observed that noise can be removed from the representation when the 2D representation has low contrast. However, the pixel connectivity in script structure is lost. We also observed that there is no instant way of defining a well-predefined window. It is the process of trial and error. Therefore, the satisfactory results from sauvola's method have to satisfy criteria such as:

- Preservation of the connectivity of the script structures.
- No hole in the script structures.
- Noise reduction achieved to a certain extent.

We illustrate 2D representation of inscription K193 in figure 3.7a, the gray scale of K193 in figure 3.7b and the high-contrast gray scale of K193 in figure 3.7c. Figure 3.8 shows the results of two predefined small window– 15×15 and 65×65– for local neighborhood of each pixel. The results comparatively obtained by methods– Otsu and Sauvola– are depicted in figure 3.9 (p.32) .

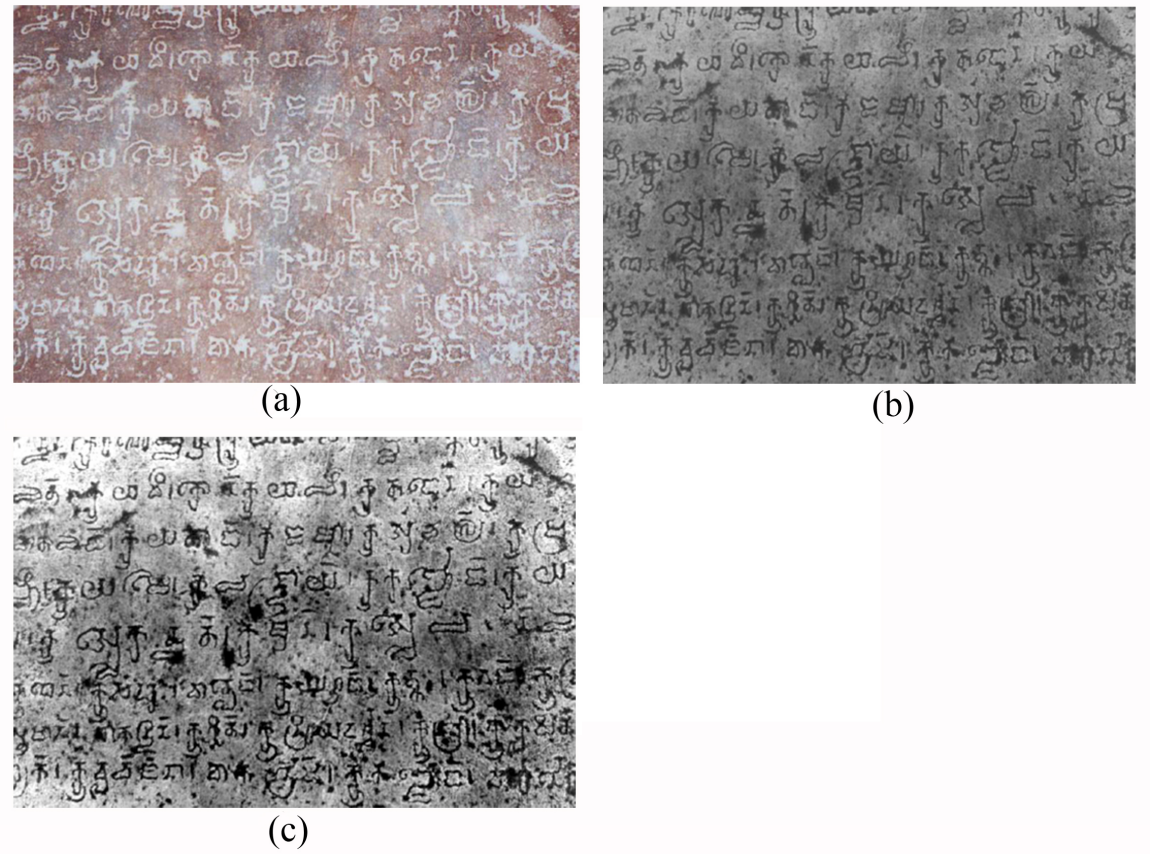


Figure 3.7: (a) The original Khmer inscription K193. (b) The gray-scale image of inscription K193.(c) The gray-scale image of inscription K193 with the high contrast by saturating 1 percent at low and high intensities of gray-level inscription K193.

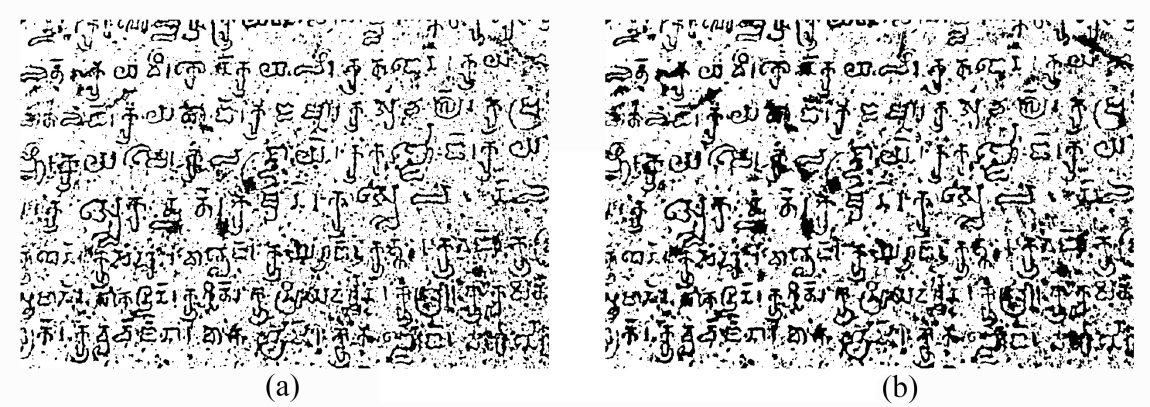


Figure 3.8: The results of applying Sauvola's method to gray-scale image of inscription K193 with 1-percent saturation at low and high intensity of gray level using (a) small window 15×15 and (b) small window 65×65.

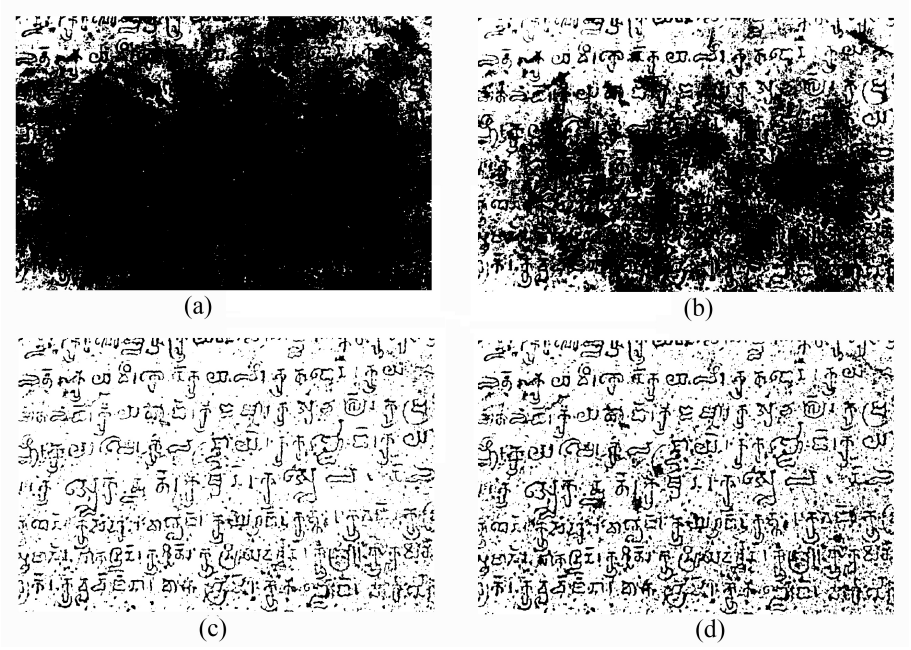


Figure 3.9: The results of applying (a) Otsu's method to gray-scale image of inscription K193, (b) Otsu's method to gray-scale image of inscription K193 with 1-percent saturation at low and high intensity, (c) Sauvola's method to gray-scale image of inscription K193 using small window 15×15 and (d) Sauvola's method to gray-scale image of inscription K193 with 1-percent saturation at low and high intensity using small window 15×15.

3.4 Reduction in Script Data to Characteristics of Topological and Geometrical Structures of the Script

Our concept is based on topological and geometric features. So, we reduce the script data to characteristics of topological and geometrical structures of the script (characters). In this thesis, we take skeletons to denote the topological and geometrical structures of the script. The skeleton of the script can be retrieved from image data by a thinning operation.

The thinning– also known as skeletonization– is the operation to peel off pixels as many as possible without changing the structures of patterns or objects. After this operation, the topological and geometrical properties such as connectivity, topology, length, direction and width of the patterns are in the focus of investigation.

The final result of thinning operation is the skeleton representing the shapes of the script. The skeleton should have the properties: (1) as thin as possible (thinness), (2) connected (connectivity) and (3) centered (position). One-by-one connected pixels are required for the skeletons. We show the thinning operation in figure 3.10 and the skeleton of ancient Khmer script in figure 3.11.

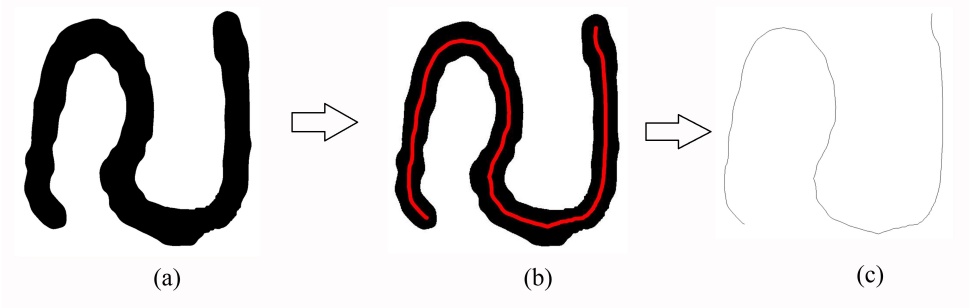


Figure 3.10: The thinning operation to reduce original structure of consonant *"la"* to the skeleton representing topological and geometrical structure of the consonant. (a) original structure of the consonant. (b) the skeleton in red located in the center of original structure of the consonant. (c) The skeleton obtained from the thinning operation.

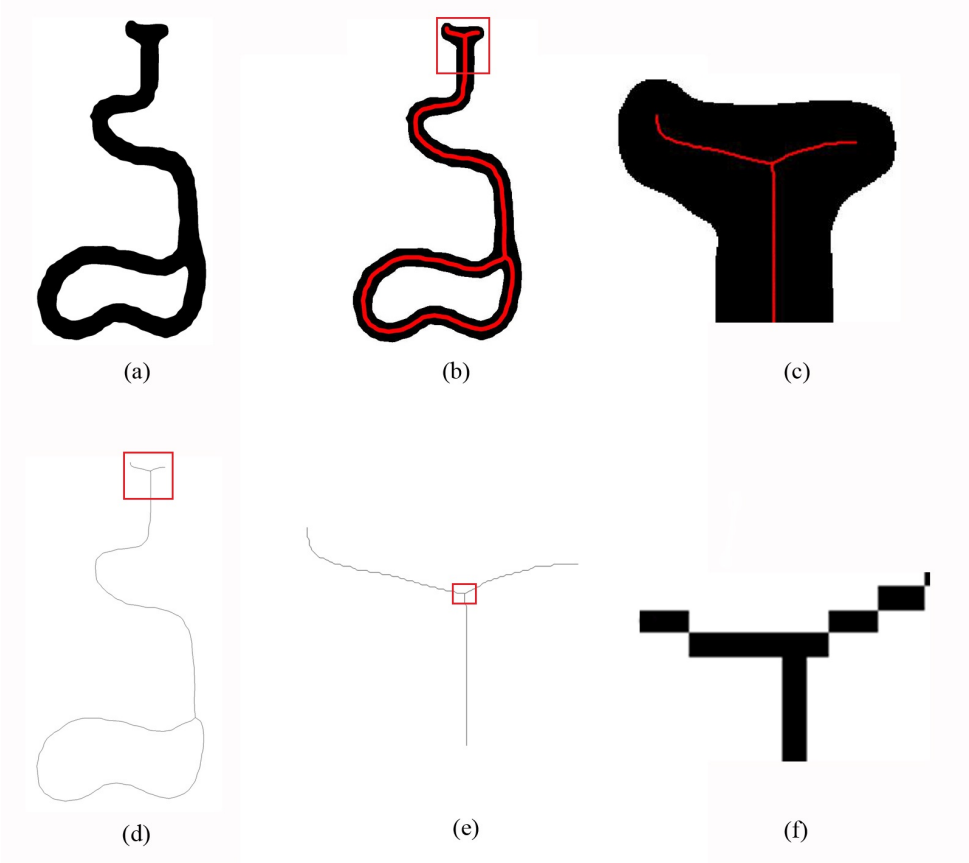


Figure 3.11: (a) The combination of consonant "*da*"and sub-consonant "*dha*". (b) The skeleton structures in red. (c) The skeleton from red rectangle in (b) stays in the center of original structure. (d) The skeleton of the combination after thinning. (e) A part of the combination zoomed in red rectangle in (d). (f) The one-by-one connected pixels of the combination zoomed in from junction part in red rectangle in (e).

We adapt Zhang and Suen's method [89] for the thinning operation. It is considered a prominent method in comparison with other methods suggested in [24][51][65][73]. The following is Zhang and Suen's method.

Zhang and Suen's Method

Let matrix $M(i, j)$ be 2D representation of script data which $1 \leq i \leq n$ and $1 \leq j \leq m$. n and m are row and column of input image, accordingly. So,

$$M(i, j) = \begin{cases} 1 \\ 0 \end{cases} \tag{3.16}$$

$M(i, j) = 1$ denotes script shapes and $M(i, j) = 0$ background. The script shapes consist of more than one pixels having value 1(1s). Hence, the pixels having value 1 are deleted or transformed to value 0 according to their neighbors at the n^{th} iteration .

Assume that the neighbors of pixel $P(i, j)$ in 3×3 window are like in figure 3.12 below.

	j-1	j	j+1
i-1	P_9	P_2	P_3
i	P_8	P_1	P_4
i+1	P_7	P_6	P_5

Figure 3.12: The neighbor pixels ($P_2, P_3, P_4, P_5, P_6, P_7, P_8, P_9$) of P_1 in 3×3 window.

We denote:

- $B(P_1)$ as the number of nonzero neighbors of P_1. It is defined by $B(P_1)$=$P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8 + P_9$, that are the eight neighbors of P_1 as shown in figure 3.12. The computation of $B(P_1)$ is depicted in figure 3.13.
- $A(P_1)$ as the number of 01 patterns in the ordered set $P_2, P_3, P_4, ..., P_9$ which are also demonstrated in figure 3.13.

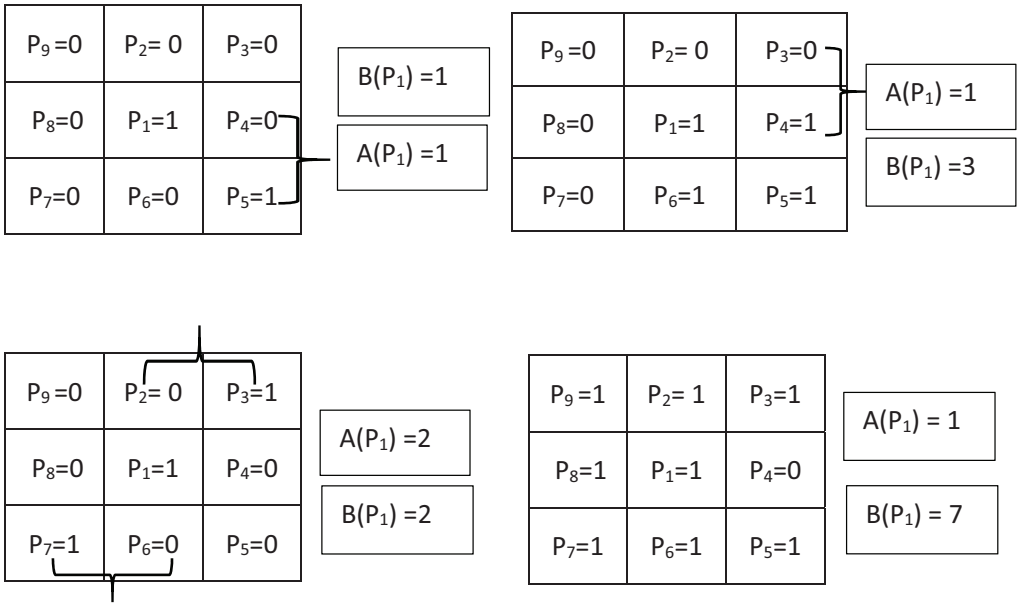


Figure 3.13: The computation of counting the 01 patterns $A(P_1)$ and $B(P_1) = P_2 + P_3 + P_4 + P_5 + P_6 + P_7 + P_8 + P_9$.

Zhang and Suen's method is to remove all contour pixels(1s) of the script shapes except those pixels belonging to the skeleton. A pixel P_1 is removed if it satisfies the following conditions.

1. P_1 is deleted if it satisfies the four conditions below:

   (a) $2 \leq B(P_1) \leq 6$

   (b) $A(P_1) = 1$

   (c) $P_2 * P_4 * P_6 = 0$

   (d) $P_4 * P_6 * P_8 = 0$

2. P_1 is deleted if it satisfies the four conditions below:

   (a) $2 \leq B(P_1) \leq 6$

   (b) $A(P_1) = 1$

   (c) $P_2 * P_4 * P_8 = 0$

   (d) $P_2 * P_6 * P_8 = 0$

The condition (a) is to preserve the endpoints of a skeleton line and condition (b) is to prevent the deletion of those points that lie between the endpoints of a skeleton line as displayed in figure 3.14.

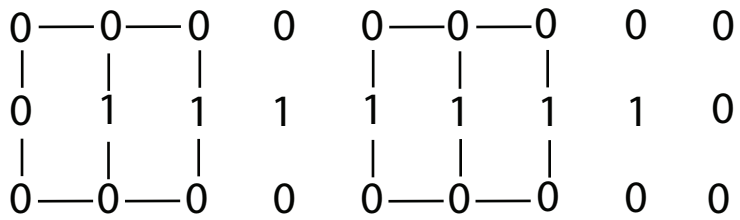


Figure 3.14: Preventing the deletion of the endpoints.

The Zhang and Suen's algorithm is formulated below:

Algorithm 3.4.1 (Zhang and Suen's Algorithm)

1. *Read binary character input image* $M(i, j)$
2. *Define* $S = 1$, R *=total rows of input image,* C *=total columns of input image, and new matrix* $F(i, j)$ *having the same size as input image with its elements having value 1.*
3. *While loop S==1*
   (a) *S==0*
   (b) *For loop* $i = 2$ *till* $R - 1$ *and for loop* $j = 2$ *till* $C - 1$
   - *if* $P(i, j)$*==1*
     i. *compute* $A(P(i, j))$ *and* $B(P(i, j))$
     ii. *compute* $P_2 * P_4 * P_6$ *and* $P_4 * P_6 * P_8$
     A. *if* $A(P(i, j)) == 1$ *and* $2 \leq B(P(i, j)) \leq 6$ *and* $P_2 * P_4 * P_6 == 0$ *and* $P_4 * P_6 * P_8 == 0$
     – *update F(i,j)=0;*
     – *update S=1;*

   (c) *update input image* $M(i, j) = M(i, j) * F(i, j)$
   (d) *For loop* $i = 2$ *till* $R - 1$ *and for loop* $j = 2$ *till* $C - 1$
   - *if* $P(i, j)$*==1*
     i. *compute* $A(P(i, j))$ *and* $B(P(i, j))$
     ii. *compute* $P_2 * P_4 * P_8$ *and* $P_2 * P_6 * P_8$

A. *if* $A(P(i,j)) == 1$ *and* $2 \leq B(P(i,j)) \leq 6$ *and* $P_2 * P_4 * P_8 == 0$ *and* $P_2 * P_6 * P_8 == 0$

- *update F(i,j)=0;*
- *update S=1;*

(e) *Update input image* $M(i,j) = M(i,j) * F(i,j)$

(f) *The skeleton* $G(i,j) = M(i,j)$

3.5 Character Segmentation

In this section, we describe character segmentation to segment the script data (text) on 2D representations into a set of isolated characters.

Character segmentation is subjective. The process to segment the text into the sentence and then into individual words or characters depends on the complexity of the text, i.e. how much the text can be segmented. It also depends on an individual's definition.

In our definition, we define consonants, independent vowels and the combinations of consonants, sub-consonants and dependent vowels as characters. So, the character segmentation to our definition is to segment the script data (text) into a set of consonants, independent vowels and the combinations with regard to ancient Khmer writing system.

As described in chapter 2 (section 2.2), in ancient Khmer writing system characters were written rowwise and read from right to left and from top to bottom. There can be more than one column in each row.

Based on the characteristics of Khmer writing system, the script apparently stays in line- the so-called text line. Between each text line, there exists curvilinear blank space. Therefore, we design line segmentation to segment the script data into sets of isolated characters.

In line segmentation of 2D representations of the inscriptions, blank space acts as delimiter to separate script data into text line and into isolated characters. It requires two processes– horizontal and vertical line segmentation. The horizontal line separates script data into text lines while vertical line separates each text line into isolated characters: consonants, independent vowels and the combinations. Basically, horizontal and vertical line segmentation can be achieved by projection-based methods. The vertical projection profile is obtained by summing pixel values belonging to the script along the horizontal axis for each y value and horizontal projection profile can also be obtained by summing pixel values belonging to the script along the vertical axis for each x value. They are defined by

$$profile(y) = \sum_{1 \le x \le M} G(x, y) \tag{3.17}$$

$$profile(x) = \sum_{1 \le y \le N} G(x, y) \tag{3.18}$$

where $profile(y)$ is the vertical projection profile, $profile(x)$ is horizontal projection profile and $G(x, y)$ is 2D representations of the inscriptions.

for given thresholds a and b, any $profile(y) = b$ and $profile(x) = a$ are considered as blank space lines to separate the script data into text lines and into a set of isolated characters, respectively. However, this method cannot be applied to curvilinear white space. The projection-based methods make some parts of the characters missing. Figure 3.15 demonstrates the project-based methods applied to inscription K49.

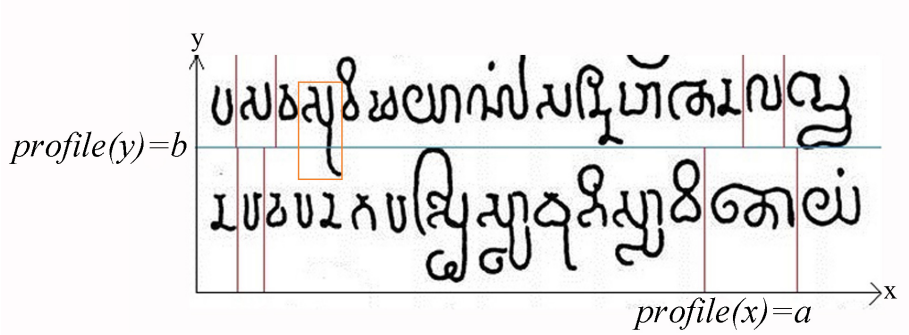


Figure 3.15: An example of applying line segmentation to K49 and the failure of $profile(y) = b$ at segmenting the character in orange box.

Therefore, we investigate an optimal path function to adjust the blank space line according to the direction of writing and size of characters as displayed in figure 3.16 (p.39).

In our thesis, we used that a path planning method [12] [23][54][74] to segment 2D representations of Khmer inscriptions into sets of consonants, independent vowels and the combinations.

The path planning method is used in artificial intelligence, especially in robotic system to find shortest path that allows agents travel successfully from initial position (initial state) to desired position (desired state). In path planning, the states are agent locations and transition between states are actions that the agent takes. Each action has associated transition cost or edge cost.

We consider a 2D representation $F(x, y)$ of an inscription as a plane for the agent to traverse. All the pixels are divided into two groups– states and obstacles. The pixels belonging to script data (black pixels) are obstacles that the agent has to avoid traversing, whereas white pixels are

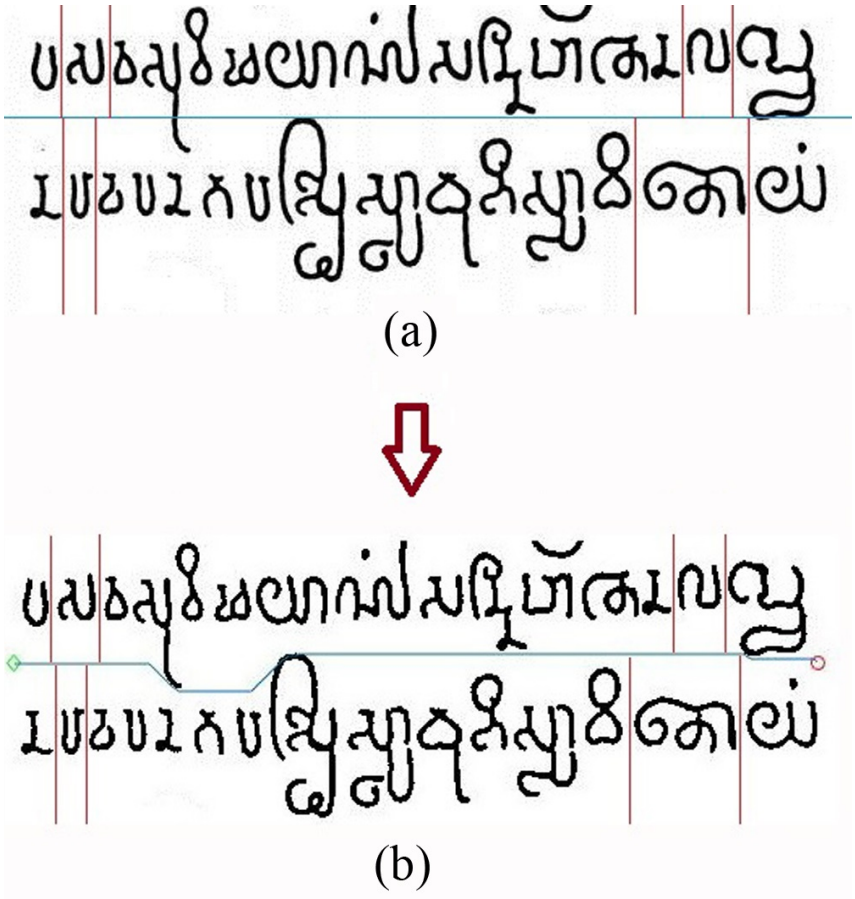


Figure 3.16: (a) Text line segmentation of inscription K49 with $profile(y) = b$ and $profile(x) = a$. (b) Text line segmentation of inscription K49 with adjusted horizontal line in blue color.

states that the agent can traverse. To optimize the space and time of computation, the predefined sub plane $M(x, y)|M(x, y) \subset F(x, y)$ to let the agent traverse may be required. The method and algorithm of path planning– the so-called A star (A^*)– are described as follows.

A^* Path Planning Method

A^* path planning method is to minimize the sum of transition costs through all possible paths from initial state s_1 to desired state s_n.

Denote $s_1^a, s_2^a, ..., s_n^a$ for $1 \leq i < n$ as the sequence of states traversed by agent a across path pa^a. The optimal path pa^* is defined:

$$pa^* = \arg\min_{pa^a} \sum_{i=1}^{n-1} C_{i,i+1} \tag{3.19}$$

The transition cost at state s_i is defined by:

$$C_i = C_{1,i} + C_{n,i} \tag{3.20}$$

where $C_{1,i}$ is the cost from state s_i to initial state s_1 and $C_{n,i}$ is the cost from state s_i to goal state s_n. Euclidean distance is used to compute $C_{1,i}$ and $C_{n,i}$.

$$C_{1,i} = \sqrt{(x_{s_i} - x_{s_1})^2 + (y_{s_i} - y_{s_1})^2} \tag{3.21}$$

$$C_{n,i} = \sqrt{(x_{s_n} - x_{s_i})^2 + (y_{s_n} - y_{s_i})^2} \tag{3.22}$$

where $(x_{s_i}, y_{s_i}), (x_{s_1}, y_{s_1}), (x_{s_n}, y_{s_n})$ are the position of the state s_i, s_1, s_n,respectively, in the defined plane.

The $\mathbf{A}^*$ Path Planning Algorithm is formulated below.

Algorithm 3.5.1 ($\mathbf{A}^*$ Path Planning Algorithm)

1. *Read binary inscription image $G(x, y)$.*
2. *Find obstacles (black pixels) $P_b(x, y) := G(x, y) == 0$.*
3. *Create openlist $list_{open}$ consisting of attributes such as*
   - *$status = 0 or 1$ for removing or adding to openlist accordingly.*
   - *$s(x, y)$: store the position of each state.*
   - *$PP(x, y)$: store the parent's position of each state.*
   - *$C_{1,s}$: store cost from each state to initial state.*
   - *$C_{n,s}$: store cost from each state to goal state.*
   - *C_s : store total cost from each state: $C_s = C_{1,s} + C_{n,s}$.*
4. *Create closelist $list_{close}(x, y)$ for storing visted states and obstacles.*
5. *Insert all obstacles (black pixels) $P_b(x, y)$ into closelist $list_{close}(x, y)$.*
   - *$list_{close}(x, y) = P_b(x, y)$*
6. *Define initial state s_1 and desired state s_n.*
7. *Define current state $s_{current} = s_1$*
8. *Compute all costs of $s_{current}$*
   - *$C_{1,s} = 0$*

- $C_{n,s} = \sqrt{(x_{s_n} - x_{s_1})^2 + (y_{s_n} - y_{s_1})^2}$
- $C_s = C_{n,s}$

9. *Insert* $s_{current}$ *and its attributes:*
   $status = 1,\ PP(x, y) = s_{current},\ C_{1,s},\ C_{n,s},\ C_s$ *into openlist* $list_{open}$.

10. *Insert* $s_{current}$ *into closelist* $list_{close}$

    - $list_{close}(x, y) = s_{current}(x, y)$.

11. *Update attribute* $status = 0$ *of* $s_{current}$ *in openlist* $list_{open}$.

12. *Do while loop until* $s_{current} = s_n$

    (a) *Find 8 neighboring states of* $s_{current}$ *in* 3×3 *window and compute their attributes:* $PP(x, y), C_{1,s}, C_{n,s}, C_s$.

    - $C_{1,s} = \sqrt{(x_{s_i} - x_{s_1})^2 + (y_{s_i} - y_{s_1})^2}$
    - $C_{n,s} = \sqrt{(x_{s_i} - x_{s_n})^2 + (y_{s_i} - y_{s_n})^2}$
    - $C_s = C_{1,s} + C_{n,s}$

    (b) *Check 8 neighboring states of* $s_{current}$ *in closelist* $list_{close}(x, y)$.

    (c) *Check each neighboring state (not in closelist) of* $s_{current}$ *in openlist* $list_{open}$ *if the current total cost is smaller than previous total cost, update that state and its attribute with smaller cost.*

    (d) *Insert the remaining neighboring states of state* $s_{current}$ *and their attributes:* $status = 1, PP(x, y), C_{1,s}, C_{n,s}, C_s$ *into openlist* $list_{open}$.

    (e) *Find the state* s_{min} *in openlist* $list_{open}(x, y)$ *that have the smallest total cost* C_s.

    (f) *Assign* $s_{current} = s_{min}$.

    (g) *Insert* $s_{current}$ *into closelist* $list_{close}$

    - $list_{close}(x, y) = s_{current}(x, y)$.

    (h) *Update attribute* $status = 0$ *of* $s_{current}$ *in openlist* $list_{open}$.

13. *Find the optimal path if* $s_{current} = s_n$ *through attribute* $PP(x, y)$ *in openlist* $list_{open}$.

3.6 The Improvement of Topological and Geometrical Structure of the Script

The original inscriptions contain noise because of some factors as mentioned in chapter 1 (section 1.1). Applying the mathematical methods described in section 3.2, 3.3 and 3.4 we cannot remove all noise and obtain full topological and geometrical shapes of the script.

Therefore, we need to improve the structures of the obtained script such as connecting the missing parts of the script, size normalization and denoising. We improve the script structure with respect to the standard script structures. The improvement can be done by existing image enhancement techniques and graphics editors.

We also need the knowledge of Khmer epigraphy to connect the missing parts of script and denoise the images of the inscriptions. In the following section, we describe the size normalization of the script.

Character Size Normalization

Character size normalization is a transformation of the coordinates from the original character plane to normalized plane. It maps character images onto a standard plane with predefined size[5].

Character size normalization is useful for compute feature vectors of the characters which are described in chapter 4. Errors and bias occur if the small and big scales are measured by the same parameter m.

To normalize the size, we firstly estimate the frame binding an object image (character) and then crop the image by the frame. Finally, resize the image in a certain scale and add some additional free space $m \times n$ to the resized frame of the image. The size normalization of two shapes of a character is shown in figure 3.17 and 3.18 below (p.43).

The original frames of images in the figures 3.17a and 3.18a are in turquoise. They are cropped into red frames by removal of some free space $m \times n$. Finally, they are resized to a standard scale as shown figure 3.17b and 3.18b by a equation below.

$$row_{new} = (row_{old} * column_{new})/column_{old} \tag{3.23}$$

where $row_{new}, column_{new}, row_{old}, column_{old}$ are the rows and columns of the new resized image and old image accordingly and $column_{new}$ has to be predefined.

An algorithm to crop the image is formulated below.

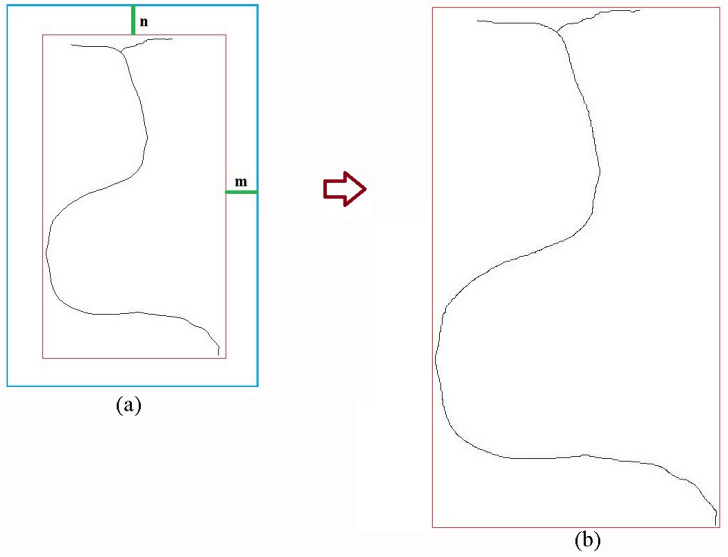


Figure 3.17: (a) The original frame of the character in turquoise. (b) The resized frame of character in red.

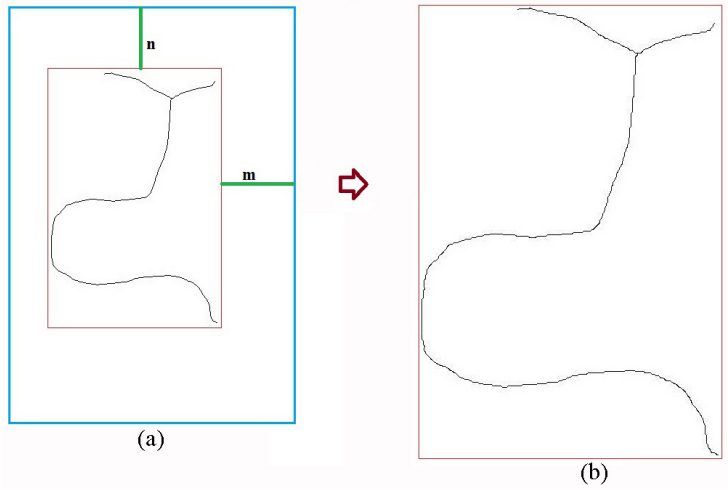


Figure 3.18: (a) The original frame of the character in turquoise. (b) The resized frame of character in red.

Algorithm 3.6.1 (Crop Image Algorithm)

1. *Read binary image of a character* $G(x, y)$ *(1s=background, 0s= object)*
2. *Find the coordinates* (x, y) *of all pixels* $P(x, y)$ *with value 0.*
3. *Find the maximum, minimum values* X_{min}, X_{max}, *and* Y_{min}, Y_{max} *in array coordinate* $P(x, y)$.
4. *Croped image* $H(x, y) = G(X_{min} : X_{max}, Y_{min} : Y_{max})$ *where* $x = [X_{min}, X_{max}]$ *and* $y = [Y_{min}, Y_{max}]$.

3.7 Summary

In conclusion, chapter 3 provided the methods to extract the script data from original of Khmer inscriptions. Applying the curvature-based method– Gigamesh, we can transfer the original inscriptions to 2D representation. We explained the thresholding methods of Otsu and Sauvola to reduce the information in 2D representations to essential script data and skeletonization operation– Zhang and Suen's method– to reduce essential script data to characteristics of topological and geometrical structures of the script. We also described character segmentation to segment script data in 2D representations into sets of isolated characters. A^* Path Planning is used to define the paths of the segmentation. At the end of the chapter, we provided methods to improve the topological and geometrical structures of the script.

In the next chapter, we are describing the mathematical methods to characterize ancient Khmer script and methods to define the feature vectors of ancient Khmer script.

Chapter 4

Image Analysis Methods – Definition of Feature Vectors for Characters of Ancient Khmer Fonts

Chapter 4 describes the mathematical methods to characterize the ancient Khmer characters. We explain the mathematical concepts of topology and integral geometry to extract the features of the characters. In the first part of the chapter, the shape analysis of the characters is discussed. From the shape analysis, we give the principles of graph theory to describe the shapes structures of the characters. We then explain the methods to extract the features– topological and geometric– to represent the character structures.

4.1 Characterization of Ancient Khmer Characters

We consider the script in the pre-Angkorian period. The script consists of 33 consonants, 33 sub consonants, 12 independent vowels, 13 dependent vowels and symbols for representing numbers. Regarding a large number of the characters, it is challenging to identify not only the handwritten characters but also the characters of the computer font. In the following section, we are describing the shape analysis of ancient Khmer characters.

4.1.1 Shape Analysis of Ancient Khmer Characters

The characters are the connections of line and/or curve segments. We illustrate the structure of three standard characters in figure 4.1 below. We also give more illustrations of the structures of other characters in figure 4.2.

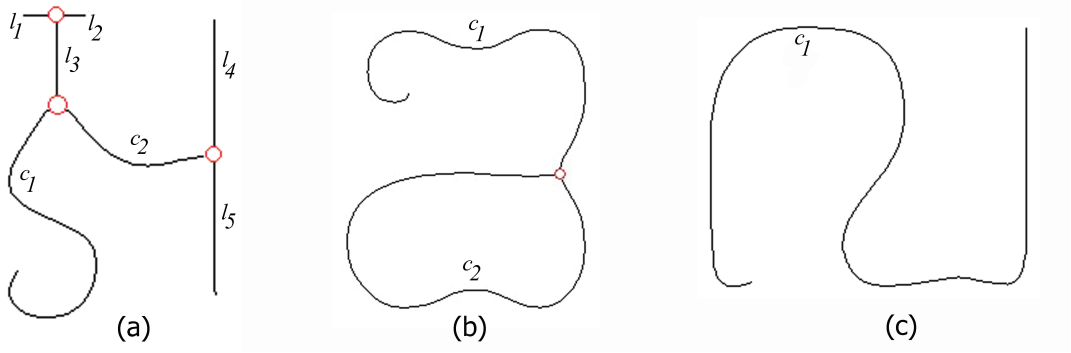


Figure 4.1: (a) Consonant *"a"*. (b) Consonant *"kha"*. (c) Consonant *"la"*

In figure 4.1, we denote the structure of each character by a set of line and curve segments by:

- consonant *"a"* =: $C_{la} = \{l_1, l_2, l_3, l_4, l_5, c_1, c_2\}$.
- consonant *"kha"* =: $C_{kha} = \{c_1, c_2\}$.
- consonant *"la"* =: $C_{la} = \{c_1\}$.

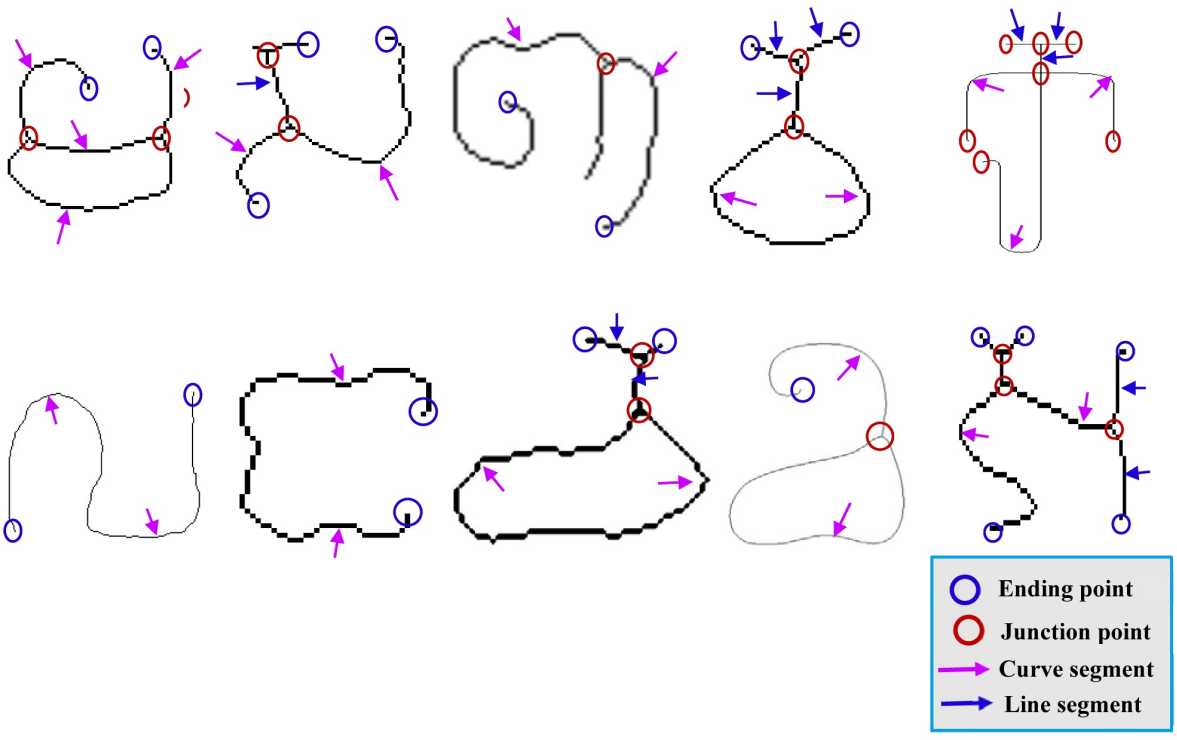


Figure 4.2: The shapes of the characters are the connections of line and curve segments.

From the shape analysis, concepts of topology and integral geometry are the most appropriate approach to identify the characters. Topology is applied to the connectivity of line and curve segments and integral geometry to the curvature distributions of the curve segments.

Undirected connected graphs are used to represent the structures of the characters. Topological connectivity and curvatures of the curve segments can be computed from the nodes and edges of the graphs.

We are presenting the principles of graph theory and the characterization of the characters by undirected connected graphs in the following subsections.

4.1.2 Graph Theory Principles

We are recalling basic definitions of undirected graph, connected graph, adjacency matrix and the degree of node [3] [11] [20].

Definition 4.1.1 (Graph)
A graph $G = (V, E)$ consists of two sets V and E. The elements of V are called nodes. The elements of E are called edges.

Definition 4.1.2 (Undirected graph)
An undirected graph $G(V, E)$ consists of a finite, nonempty set of nodes V and a set of edges E. Each edge is a set $\{v, w\}$ of nodes. Simply, in undirected graph there is no orientation of edges. The edge $E(v_1, v_2)$ is identical to the edge of $E(v_2, v_1)$.

Definition 4.1.3 (Connected graph)
A graph is a connected graph if there exists a path between every pair of its nodes.

Definition 4.1.4 (The degree of a node)
The degree of a node v in a graph G is the number of proper edges incident on v plus twice the number of self-loops.

Definition 4.1.5 (Adjacency matrix)
Adjacency matrix for a graph $G = (V, E)$ is the matrix to represent the connectivity between the nodes in the graph. The adjacency matrix of the ordered nodes $(v_1, v_2, ..., v_n)$ of graph $G = (V, E)$ is the $n \times n$ matrix M_G such that

$$M_G(i, j) = \begin{cases} 1 & if \quad v_i \quad and \quad v_j \quad are \quad adjacent \\ 0 & otherwise \end{cases} \tag{4.1}$$

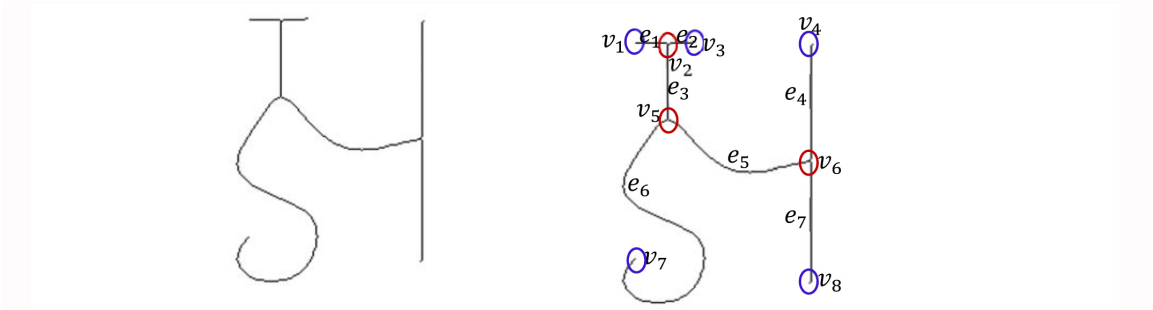
4.1.3 Undirected Connected Graph of Ancient Khmer Character

As mentioned in section 4.1.1, the characters are the compositions of line and/or curve segments intersecting to each other. The topological object is typical of graph or network. We define the character as a graph.

Definition 4.1.6 (Ancient Khmer characters)
A character is an undirected connected graph $C = (L_v, E_v)$. A finite set L_v of nodes are the endpoints and the junction points of line and curve segments. A finite set of edges E_v of adjacent nodes and self-connected nodes $\subseteq L_v$ of the graph C are typical of line and curve segments.

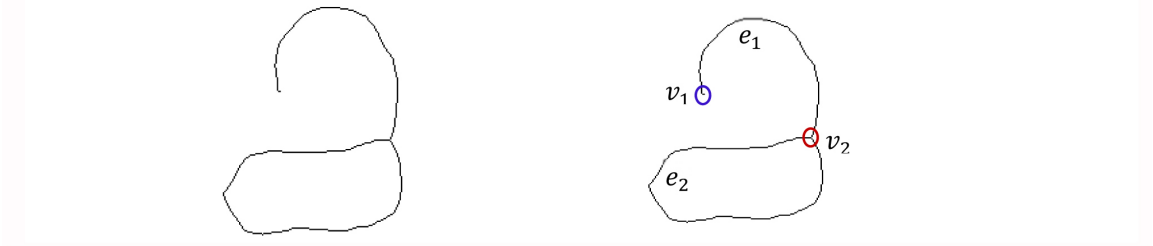
A character $C = (L_v, E_v)$, where $L_v = \{v_1, v_2, ..., v_n\} | v_1, v_2, ..., v_n \in C$ are the endpoints and the junction points of line and curve segments and $E_v = \{e_1, e_2, ..., e_n\} | e_n = \{v_i, v_j\} \in C$ if v_i and v_j are adjacent nodes or $e_n = \{v_i, v_i\} \in C$ if v_i is self-connected. $i, j = 1, 2, 3, ..., n$.

In figure 4.3, 4.4 and 4.5, the characters *"a"*, *"kha"* and *"la"* are represented in undirected connected graphs, accordingly.



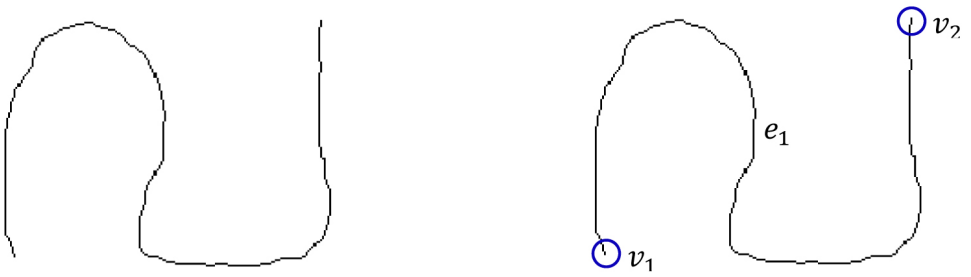
Character "a" C_a=: $(\mathcal{L}_v, E_v) \mid \mathcal{L}_v = \{v_1, v_2, v_3, v_3, v_4, v_5, v_6, v_7, v_8\}$ & $E_v = \{e_1, e_2, e_3, e_4, e_5, e_6, e_7\}$ *where* $e_1 = \{v_1, v_2\}, e_2 = \{v_2, v_3\}, e_3 = \{v_2, v_5\}, e_4 = \{v_4, v_6\}, e_5 = \{v_5, v_6\}, e_6 = \{v_5, v_7\}, e_7 = \{v_6, v_8\}$

Figure 4.3: The consonant *"a"* is represented in a undirected connected graph.



Character "Kha" C_{kha}=: $(\mathcal{L}_v, E_v) \mid \mathcal{L}_v = \{v_1, v_2\}$ & $E_v = \{e_1, e_2\}$ *where* $e_1 = \{v_1, v_2\}, e_2 = \{v_2, v_2\}$

Figure 4.4: The consonant *"kha"* is represented in a undirected connected graph.



Character "la" C_{la}=: $(\mathcal{L}_v, E_v) \mid \mathcal{L}_v = \{v_1, v_2\}$ & $E_v = \{e_1\}$ *where* $e_1 = \{v_1, v_2\}$

Figure 4.5: The consonant *"la"* is represented in a directed connected graph.

We also code the connectivity of the character graph in the adjacency matrix which is presented below.

Adjacency Matrix of Character Graph

The connectivity of the nodes is coded in adjacency matrix $M_{ij} = (v_i, v_J)$. M_{ij} is defined by

$$M_{ij} = (v_i, v_j) = \begin{cases} 1 & if \quad v_i \quad connects \quad v_j \\ 2 & if \quad v_i \quad connects \quad v_i \\ 0 & otherwise \end{cases} \tag{4.2}$$

where the rows v_i and columns v_j of the adjacency matrix are the nodes and $i, j = 1, 2, 3, ..., n$.

The adjacency matrix is defined uniquely up to the permutation of the nodes. The adjacency matrices of the consonant *"a"*, *"kha"* and *"la"* are illustrated in figure 4.6, 4.7 and 4.8.

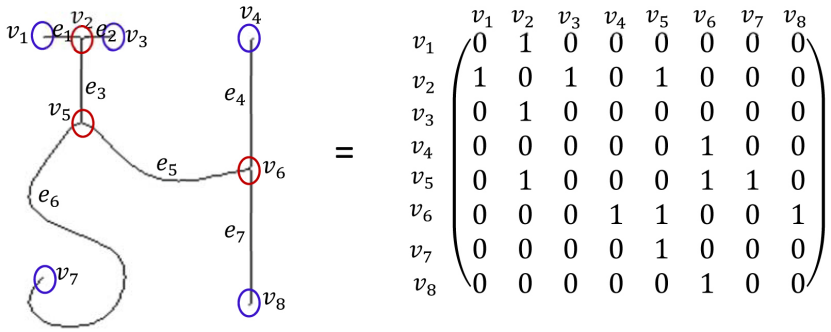


Figure 4.6: The connected graph of consonant *"a"* is coded in the adjacency matrix.

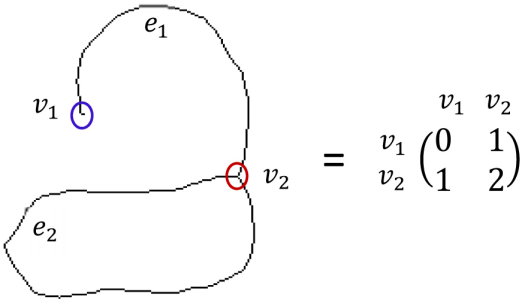


Figure 4.7: The connected graph of consonant *"kha"* is coded in the adjacency matrix.

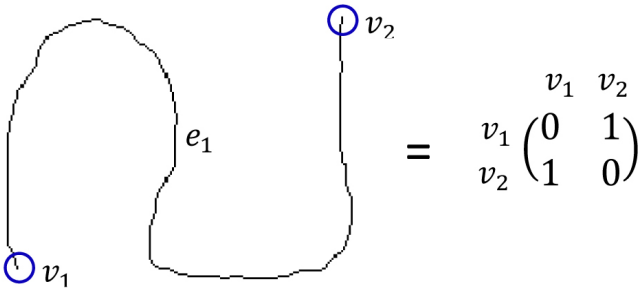


Figure 4.8: The connected graph of consonant *"la"* is coded in the adjacency matrix.

4.1.4 Feature Extraction of Ancient Khmer Characters

Two types of features are extracted from the character structures. Topological features are extracted from the connectivity of line and curve segments, whereas geometric features are extracted from the curvature distributions of curve segments.

The methods to extract these two types of features of the characters are formulated.

Topological Feature Extraction

In the character graphs, the connectivity of the graphs– nodes and degrees of the nodes– represents the topological connectivity of line and curve segments of the characters.

Definition 4.1.7 (Topological features of the characters)
The topological features of the characters are the total number of nodes of each degree in the character graphs. The features are defined by:

$$TFC(kc) = (n_{d_1}, n_{d_3}, n_{d_4}, ..., n_{d_n}) \tag{4.3}$$

where $TFC(kc)$ *is the topological features of the character* kc *and* $n_{d_1}, n_{d_3}, n_{d_4}, ..., n_{d_n}$ *are the total*

number of nodes of degree $1, 3, 4, ..., n$ *in the graph of the character* kc*. In this case, the node of the degree 2 is avoided.*

The topological features can be computed from the adjacency matrix of the character graph by the summation of the connectivity of each node in rowwise and then sum up the nodes which have the same degree as displayed in figure 4.9, 4.10 and 4.11.

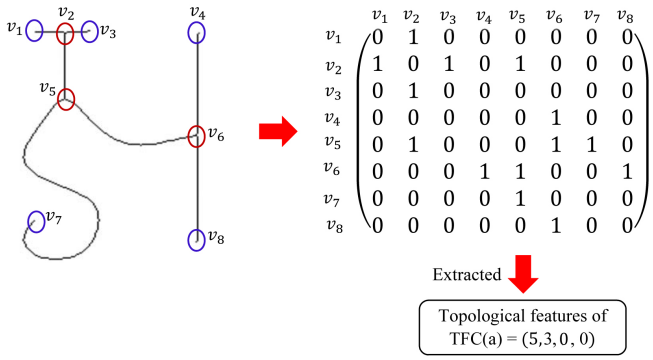


Figure 4.9: The topological feature extraction of consonant *"a"* from its adjacency matrix.

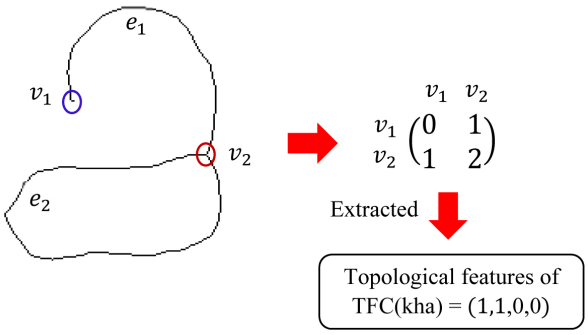


Figure 4.10: The topological feature extraction of consonant *"kha"* from its adjacency matrix.

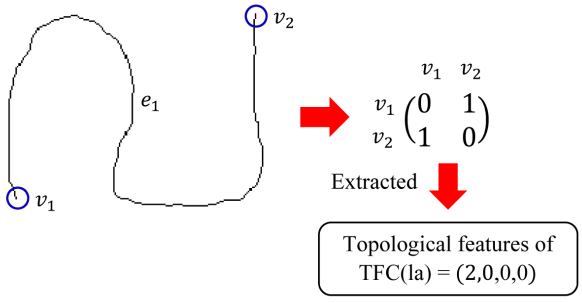


Figure 4.11: The topological feature extraction of consonant *"la"* from its adjacency matrix.

The nodes of degree 1 refer to endpoints and the nodes of degree $3, 4, ..., n$ are junction points. An algorithm to compute the endpoints and junction points of the characters is formulated.

Algorithm 4.1.1 (Computation of endpoints and junction points of the characters)
All points (pixels) $P_i = 1$ if P_i belongs to the character structures and 0 otherwise. We test all the pixels of the character structures by considering the tested pixel P_i as the center pixel within a 3×3 neighbor as displayed 4.12a. The operation is designed to detect endpoints and junction points as shown in figure 4.12b.

P_1	P_6	P_7
P_2	P_5	P_8
P_3	P_4	P_9

(a)

$$A = \begin{bmatrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_9 \\ P_8 \\ P_7 \\ P_6 \end{bmatrix} - \begin{bmatrix} P_2 \\ P_3 \\ P_4 \\ P_9 \\ P_8 \\ P_7 \\ P_6 \\ P_1 \end{bmatrix}$$

(b)

Figure 4.12: (a) 3×3 neighbor pixels. P_5 is the tested pixel. (b) The operation for detecting the endpoints and junction points [33].

From the operation in figure 4.12b, The endpoints and junction points can be detected by

- *P_5 is the endpoint if $P_5 == 1$ and $sum(abs(A)) == 2$.*
- *P_5 is the junction point if $P_5 == 1$ and $sum(abs(A)) \geq 6$.*
- *To get the degree of the junction point P_i, we use the circle of radius r centered at P_i and we count the locus of points with value 1. If there exists 3 points of the locus with value 1, P_i is the junction point of degree 3.*

Topological Feature Tuple

Topological features tuples of the characters refer to the total number of nodes of degree $1, 3, 4, ..., n$ except degree 2. The topological feature tuples are defined by

$$\mathbf{V}_{TFC(kc)} = (n_{d_1}, n_{d_3}, ..., n_{d_n}) \tag{4.4}$$

where $n_{d_1}, n_{d_3}, ..., n_{d_n}$ are the total number of nodes of degree $1, 3, ..., n$. The nodes of degree 1 refer to endpoints and the nodes of degree $3, ..., n$ are junction points. According to the graphs of the characters, the maximum degree is 4. Therefore, the dimensions of the feature vectors are reduced to 4, defined by

$$\mathbf{V}_{TFC(kc)} = (n_{d_1}, n_{d_3}, n_{d_4}, n_{d_5}) \tag{4.5}$$

where n_{d_1} are total number of endpoints and n_{d_3}, n_{d_4} and n_{d_5} are total number of junction points of degree 3, 4 and 5.

Geometric Feature Extraction

We provide the methods to extract and compute the geometric features of the characters in this subsection. We use geometric features to differentiate the characters which have the same topological structures as shown in figure 4.13.

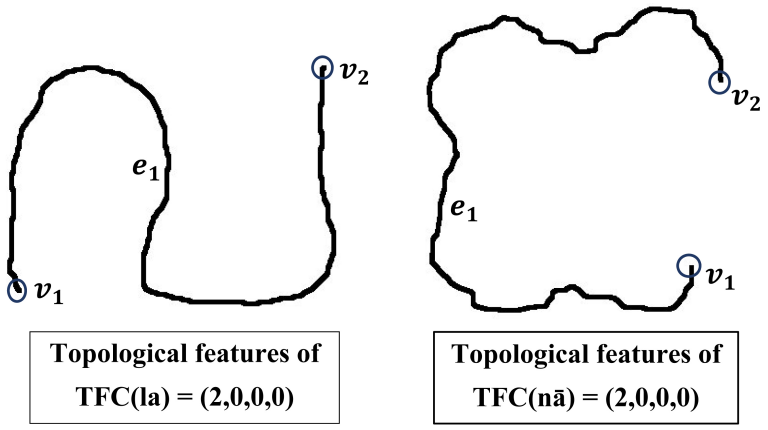


Figure 4.13: The consonants *"la"* and *"nā"* have the same topological features.

The characters *"la"* and *"nā"* in figure 4.13 are further characterized by geometric features.

The geometric features are extracted by the curvatures of the character edges. The precise values of the curvatures are not necessary. We only need the properties of the curvatures denoted by concave (+1), flat (0), and convex (-1).

Curvature of Edge as Index in Features of the Characters

In order to distinguish the characters, we may have to take into account the shapes of their edges, in particular "curvatures". To obtain distinguishing geometric indices, it is not necessary to determine curvatures numerically precisely, but it will be sufficient to decompose the edges in convex, concave and flat subsections, in a properly chosen scale.

Curvatures measure how a curve changes its direction in space. In case of a plain smooth curve $\gamma\colon [0, L] \to \mathbb{R}^2$, parametrized with respect to arc length, the curvature $\kappa(p)$ is in a point $p = \gamma(s_p)$ is determined by

$$\frac{d\mathbf{t}(p)}{ds} = \kappa(p)\mathbf{n}(p) \tag{4.6}$$

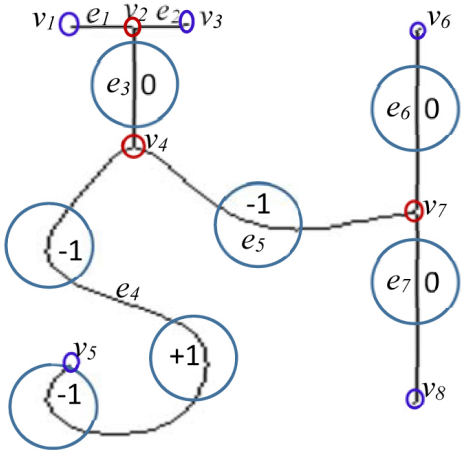
$\mathbf{t}(p) = (\mathbf{t_1}, \mathbf{t_2})(p)$ is the normed tangent to γ in p and $\mathbf{n}(p) = (-\mathbf{t_2}, \mathbf{t_1})(p)$ the normal to γ.

For our purposes we are only interested in

$$sign\ \kappa_p = \begin{cases} +1 & convex \\ 0 & flat \\ -1 & concave \end{cases} \tag{4.7}$$

Definition 4.1.8 (Geometric features of the characters)
The differential geometric features of the smooth edge $e(v_i, v_j) \in E_v$ are the sequences of the permutations of +1, 0 and/or -1.

We show an example of the differential geometric features extraction of the consonant *"a"* in figure 4.14.

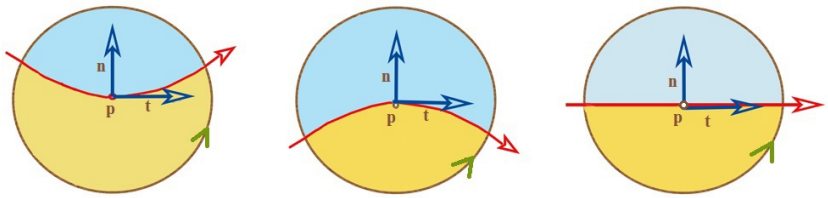


$$E_{v(a)} = \begin{cases} e_1 = \{v_1, v_2, \kappa_1 = (0)\} \\ e_2 = \{v_2, v_3, \kappa_2 = (0)\} \\ e_3 = \{v_2, v_4, \kappa_3 = (0)\} \\ e_4 = \{v_4, v_5, \kappa_4 = (-1, +1, -1)\} \\ e_5 = \{v_4, v_7, \kappa_5 = (-1)\} \\ e_6 = \{v_6, v_7, \kappa_6 = (0)\} \\ e_7 = \{v_7, v_8, \kappa_7 = (0)\} \end{cases}$$

Figure 4.14: The geometric features of consonant *"a"*.

Differentiability is too strong for this analysis since the curve arising in our applications are not smooth. Therefore, we change to approximations of curvatures using concepts of integral geometry.

To this purpose, we consider a point $p = \gamma(s_p),\ 0 < s_p < L$ and a disk $B_r(p)$. There exists a r_p such that for all $r < r_p$ there is a sub-curve $\gamma_{p,r}$ of γ passing through p and separating $B_r(p)$ into two parts. Using the orientation of the curve and the counter-clockwise orientation of the boundary of $B_r(p)$, we get $B_r(p) = \gamma_{p,r} \bigcup M_{p,r}^{right} \bigcup M_{p,r}^{left}$.



Let $|M|$ be the area of a measurable set M. Then

$$f(p,r) = \frac{|M_{p,r}|}{\frac{1}{2}|B_r(p)|} - 1 \tag{4.8}$$

At least for smooth curves and small r, the f is a good indicator for convex, concave and flat. The area can be approximated by counting pixels lying in the set $M_{p,r}$ or by the area of the sector of the disk $B_r(p)$ which is stated in algorithm 4.1.2.

We compute $f[p,r]$ in a properly chosen subset of point p, smoothen the results and interpolate obtaining a function $f^*[p,r,scv]$, representing the curvature distribution of the curved edge. We attribute to the considered edge a pattern using f^* and a threshold parameter ϵ constructed in the following way:

Define

$$\varphi_\epsilon(s) = \begin{cases} +1 & f^*(s) > \epsilon \quad (\text{“}convex\text{”}) \\ 0 & |f^*(s)| \leqslant \epsilon \quad (\text{“}flat\text{”}) \\ -1 & f^*(s) < -\epsilon \quad (\text{“}concave\text{”}) \end{cases} \tag{4.9}$$

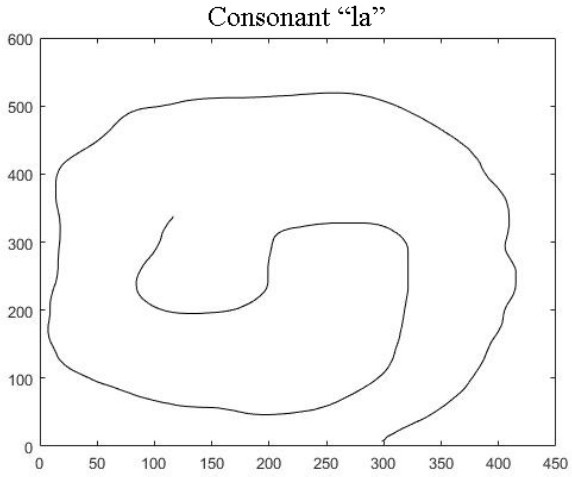
where the dependence on the parameters different from ϵ is dropped. The interval I can be decomposed in a ordered finite sequence of sub-intervals where φ_ϵ^* is a constant. We take the maximal intervals. Thus we obtain a finite sequence values $c_j, j = 1, ..., k$, of numbers $0,\ +1,\ -1$ which we reduce just in a sequence of $+1$ and -1 in the following.

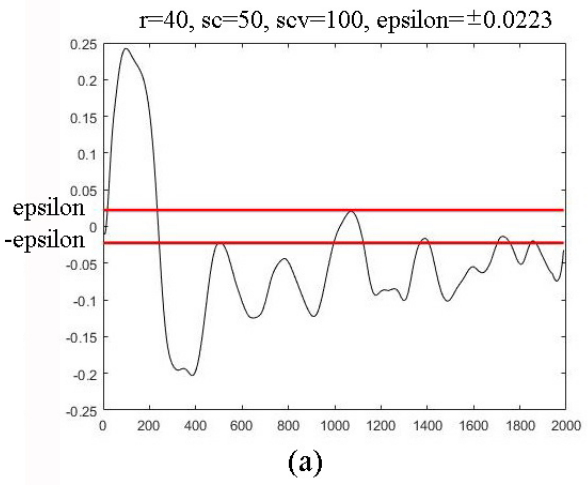
Skipping the number 0 and reducing “multiple ”+1 respectively -1 just by a single +1 respectively -1. More precise we define a sequence κ_l as follows:

- Select $j_i, i = 1, ..., l$ such that
  - Choose j_1 such that $c_j = c_1$ for all $j \leqslant j_1$ and $c_{j_1+1} \neq c_1$.
  - Define inductively j_{i+1} such that $c_j \neq c_{j_i}$ for all $j_i < j \leqslant j_{i+1}$ and $c_{j_i+1} = c_j$.
- Define $\kappa_i = c_{j_i}$ for $1 \leqslant i \leqslant l$.

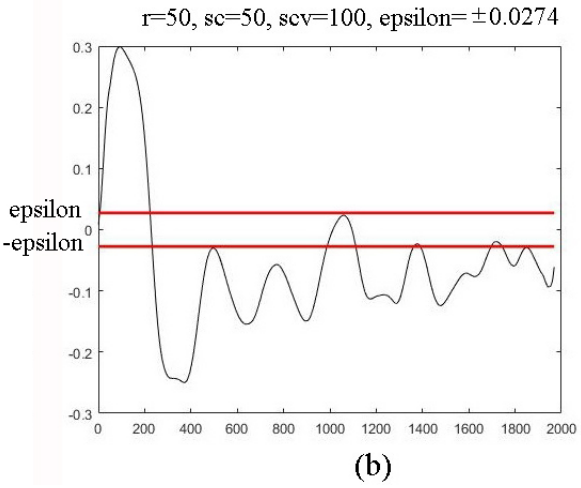
We are using sequence $\kappa_1, ..., \kappa_l$ to describe the geometric property of segment, necessary to discriminate characters in the classes with the same topological features.

Before we discuss the dependence of this geometric features on the parameters and listing the different steps in the algorithm, we consider the following examples.





(a)



(b)

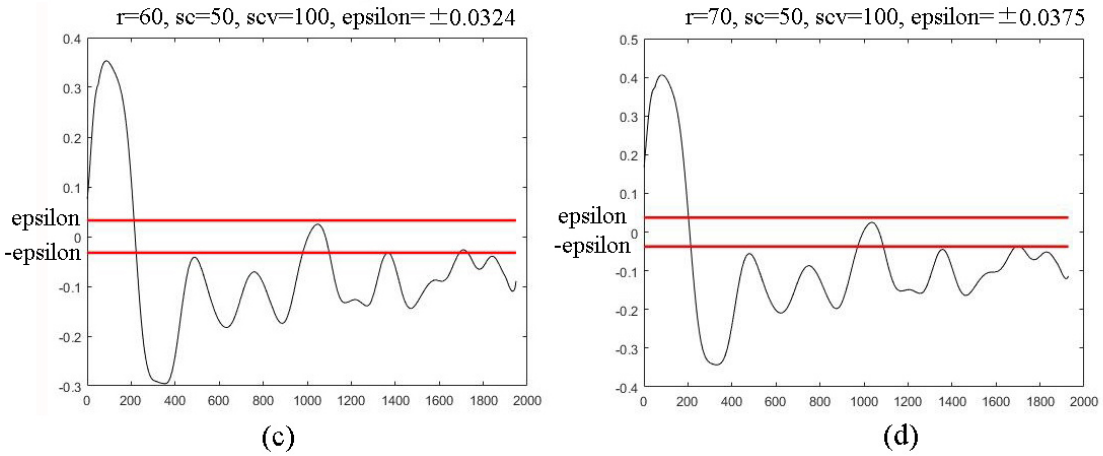


Figure 4.15: The graph of the curvature computations with respect to r, sc, scv, ϵ.

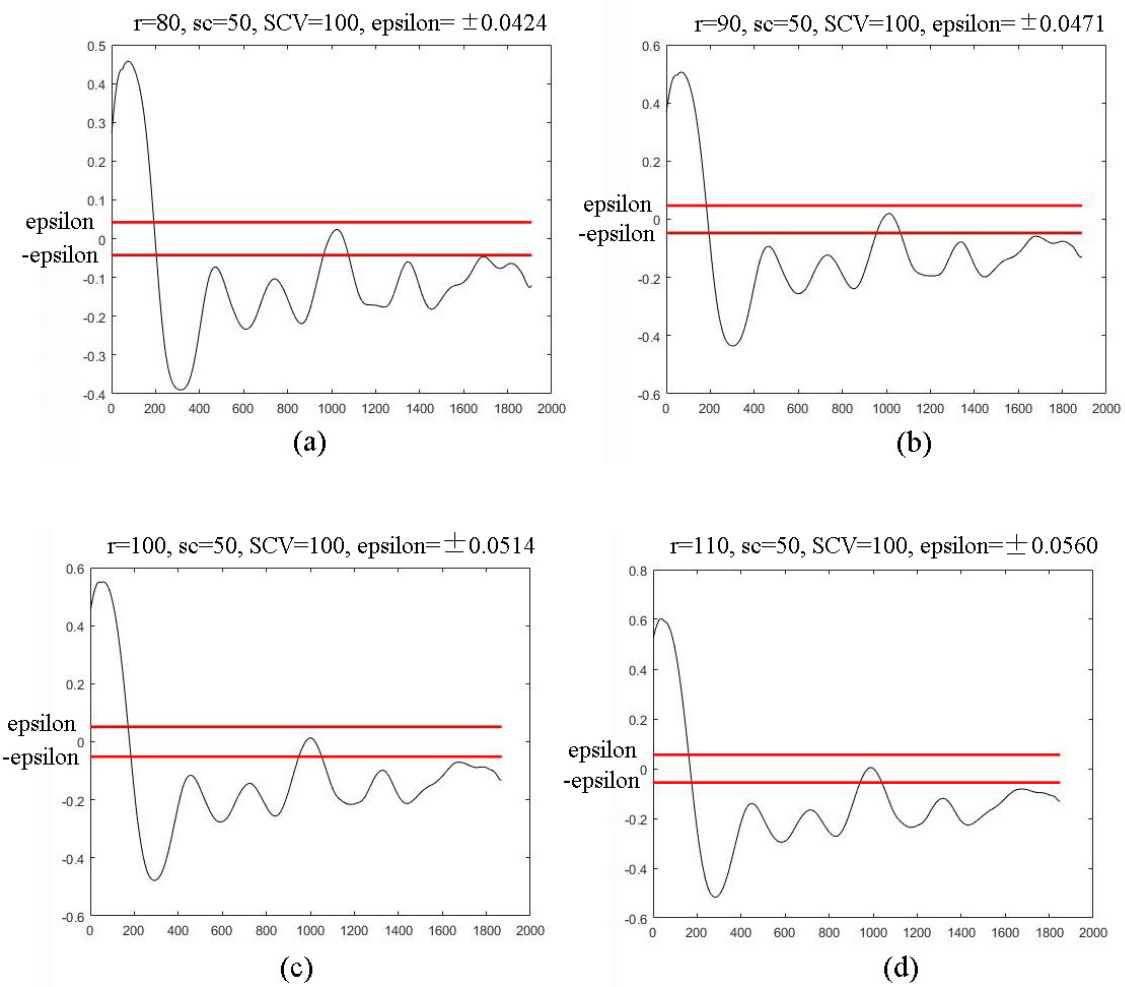


Figure 4.16: The graph of the curvature computations with respect to r, sc, scv, ϵ.

Paramters	Patterns	Feature tuple
r=40, sc=50, scv=100, ε= ± 0.0223	(0, 1, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1)	(1, -1)
r=50, sc=50, scv=100, ε= ± 0.0274	(0, 1, 0, -1, 0, -1, 0, -1, 0, -1)	
r=60, sc=50, scv=100, ε= ± 0.0324	(1, 0, -1, 0, -1, 0, -1)	
r=70, sc=50, scv=100, ε= ± 0.0375	(1, 0, -1, 0, -1, 0, -1)	
r=80, sc=50, scv=100, ε= ± 0.0424	(1, 0, -1, 0, -1)	
r=90, sc=50, scv=100, ε= ± 0.0471	(1, 0, -1, 0, -1)	
r=100, sc=50, scv=100, ε= ± 0.0514	(1, 0, -1, 0, -1)	
r=110, sc=50, scv=100, ε= ± 0.0560	(1, 0, -1, 0, -1)	

Figure 4.17: The results of computations of the sequences of consonant *"la"* with respect to the parameters– r, sc, scv, ϵ.

To compute the geometric features tuple, it requires four parameters as follows:

(a) r : a radius of the disk B_r, to compute the curvatures of the edge segment. The radius has to be large enough with respect to the arc length of the edge segment.

(b) sc : for smoothing the edge segment $\gamma(s_p)$.

(c) scv : for smoothing the values of the function $f(p, r)$ in equation 4.6. The values of the function f representing the curvature distribution of the curved edge.

(d) ϵ : for determining the corresponding decompositions of the function f to obtain the sequences stated in equation 4.7.

In the following, we formulate an algorithm to compute the sequence.

Algorithm 4.1.2 (Computation of the sequence)

1. *For the edge segment of the character, two endpoints p_s and p_e of the curve segment are determined.*

2. *Define r, sc, scv and ϵ, where sc and scv are odd numbers $\in \mathbb{N}^*$*

3. *Smoothen the edge segment $\gamma(s_p) = (x(s_p), y(s_p))$ by*

$$\hat{x}(s_p) = \frac{1}{sc} \sum_{i=-\frac{(sc-1)}{2}}^{\frac{(sc-1)}{2}} x(s_p - i), \qquad \hat{y}(s_p) = \frac{1}{sc} \sum_{i=-\frac{(sc-1)}{2}}^{\frac{(sc-1)}{2}} y(s_p - i) \tag{4.10}$$

4. *For every point $p(\hat{x}(s_p), \hat{y}(s_p)) \in \gamma$*

- *Define disk $B_r(p)$ and consider the counter-clockwise orientation of the boundary of $B_r(p)$*
- *Determine a_1, where a_1 is the first intersecting point of the disk $B_r(p)$ and the edge.*
- *Determine a_2, where a_2 is the second intersecting point of the disk $B_r(p)$ and the edge.*
- *The area $|M_{p,r}|$ can be approximated by the area of the sector of the disk $B_r(p)$. It is defined by*

$$|M_{p,r}| \cong \frac{r.l}{2} \quad or \quad \frac{r^2\theta}{2} \tag{4.11}$$

  where l is the minor arc of the disk between intersecting points a_1 and a_2. θ is the central angle subtended by minor arc l.
- *Compute $f(s) = f(p,r) = \dfrac{|M_{p,r}|}{\frac{1}{2}|B_r(p)|} - 1$, where $B_r(p) = \pi r^2$.*

5. *Smoothen $f(s)$ by*

$$f^*(s) = \frac{1}{scv} \sum_{i=-\frac{(scv-1)}{2}}^{\frac{(scv-1)}{2}} f(s-i) \tag{4.12}$$

6. *$\varphi_\epsilon(s)$ is determined by*

$$\varphi_\epsilon(s) = \begin{cases} +1 & if \quad f^*(s) > \epsilon \\ 0 & if \quad |f^*(s)| \leq \epsilon \\ -1 & if \quad f^*(s) < -\epsilon \end{cases} \tag{4.13}$$

In figures 4.18 and 4.19 below, we illustrate the application of the algorithm to compute curvature distributions of consonants *"nā"* and *"la"*.

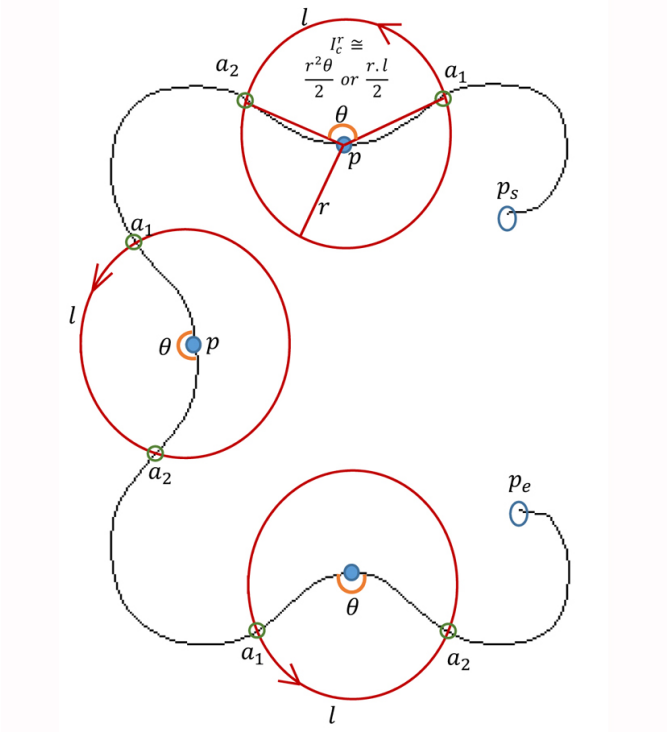


Figure 4.18: The curvature of the curve segment of consonant *"nā"* is approximated by the sector of the disk.

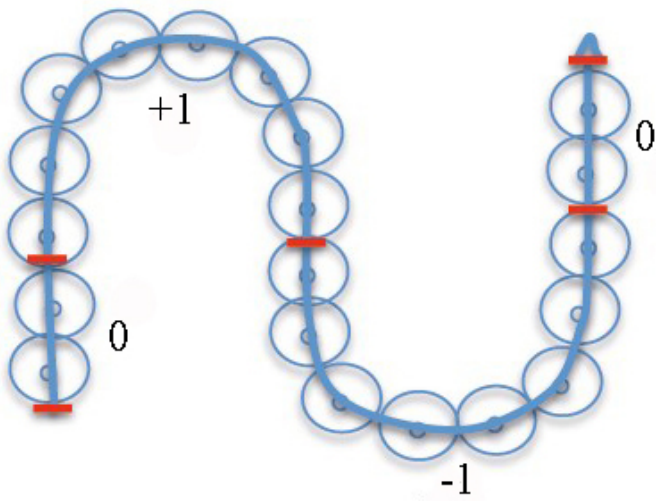


Figure 4.19: The curvature computation of curve segment of consonant *"la"*.

We illustrate the differentiation of the consonant *"la"* and the consonant *"nā"* based on the their geometric features in figure 4.20.

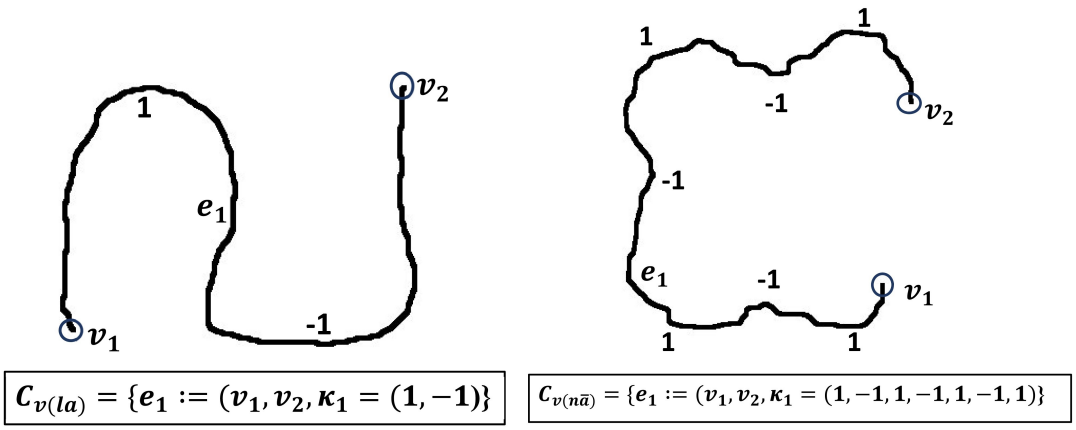
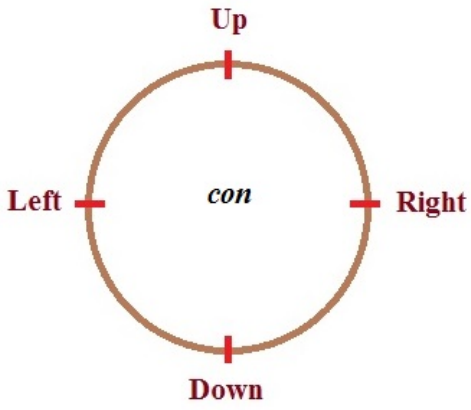


Figure 4.20: The differentiation between consonant*"la"* and consonant *"nā"* by their geometric features.

4.2 Characterization of the Combinations

Define:

- CON as the set of consonants.
- SUB as the set of sub-consonants.
- DV as the set of dependent vowels.



The elements of SUB and DV can be attached to a consonant $con \in CON$ at the locations *"Down"*, *"Right "*, *"Up "*, and *"Left "*. Following a fixed rule, the list of possible appendices attached to $con \in CON$ is shown below. The value 1 means *"attachable "* and value 0 *"unattachable "*.

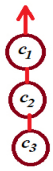
	1	2	3	4
Point				
Down	1	1	1	1
Right	1	0	0	0
Up	1	0	0	0
Left	1	0	0	0

Figure 4.21: The rule to attach sub-consonants, dependent vowels to a consonant.

Definition of the symbols:
symbol 1 means a point c can be attached.
symbol 2 two points c_1, c_2 not in the same line can be attached.
symbol 3 two points c_1, c_2 in the same line can be attached.
symbol 4 three points c_1, c_2 and c_3 in the same line can be attached.
Assume:
In case *"1", "2"and "3"* : $c, c_j \in SUB$ respectively DV.
In case *"4"* : c_1 and $c_2 \in SUB$, $c_3 \in DV$ or SUB.
For all the combinations, up to 3 elements of SUB and less than 3 elements of DV can be attached.

To identify the combinations, we develop an algorithm to decompose the combinations. The algorithm is based on the characteristics of the ancient Khmer writing system. After we separate the combinations into consonants, sub-consonants and dependent vowels, the methods to identify the ancient Khmer characters as described in section 4.1 are applied.

The algorithm can be implemented after the character segmentation described in chapter 3 section 3.5 is done. In figure 4.22, we show an example of the character segmentation of inscription K49.

After the character segmentation, the text areas of script data are marked by a set of horizontal blank space lines $\mathcal{L}_h = \{h_1, h_2, ..., h_n\}$. Between horizontal blank space lines, the text areas of the script data are also marked by a set of vertical blank space lines $\mathcal{L}_v = \{v_1, v_2, ..., v_n\}$. From two sets $\mathcal{L}_h$ and $\mathcal{L}_v$, the script data is separated into isolated characters including consonants, independent vowels and the combinations. The position of every character is defined by $P_{ch} = \{h_i, h_j, v_i, v_j\}$, for example the position of character *"ra "* is defined by $P_{ra} = \{h_1, h_2, v_1, v_2\}$.

Considering text area between two horizontal blank space lines h_1 and h_2, we draw two other

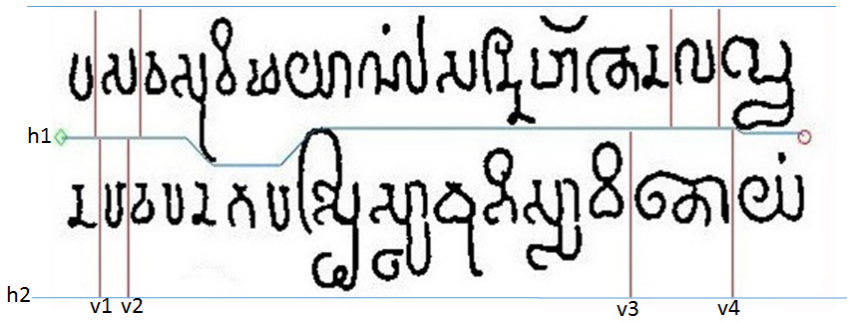


Figure 4.22: The text domain marked by horizontal and vertical blank space lines after character segmentation of K49.

horizontal line segments called upper bound line segment L_u and lower bound line segment L_l as shown in figure 4.22. The domain between h_1 and h_2 is divided into three parts– $domain_{h_1}^{L_u}$, $domain_{L_u}^{L_l}$ and $domain_{L_l}^{h2}$. We call:

- $domain_{h_1}^{L_u}$ as "super script".
- $domain_{L_u}^{L_l}$ as "main body".
- $domain_{L_l}^{h2}$ as "sub script".

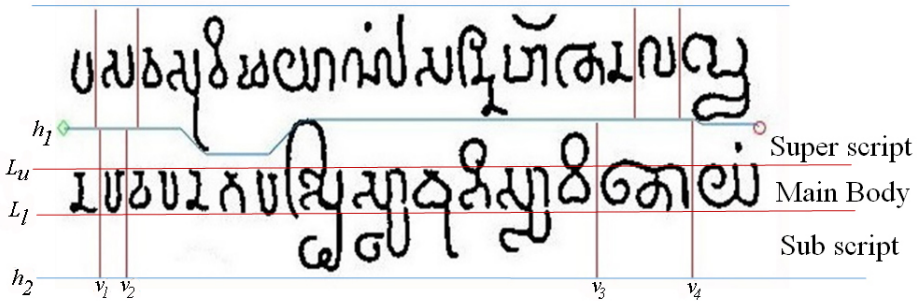


Figure 4.23: The text domain marked by horizontal and vertical blank space lines, upper bound line and lower bound line.

The combination are the characters which lie on the domain $domain_{com} =: domain_{L_u}^{L_l} \cup domain_{h_1}^{L_u}$, $domain_{L_u}^{L_l} \cup domain_{L_l}^{h_2}$ or $domain_{L_u}^{L_l} \cup domain_{h_1}^{L_u} \cup domain_{L_l}^{h_2}$.

In order to compute $domain_{com}$, the connected pixels of the characters are tracked. we define an array in figure 4.23 to store the information of tracking connected pixels of the characters. Each element of the array represents $domain_{h_1}^{L_u}$, $domain_{L_u}^{L_l}$ and $domain_{L_l}^{h2}$. The value 1 or 0 is assigned to each element of the array if pixels of the characters exist or don't exist in each domain.

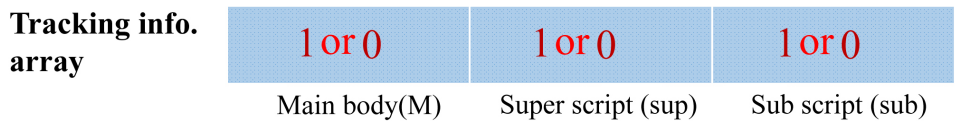


Figure 4.24: Tracking information array

In the tracking information array, the combination are the characters which the element of super script or sub script is not 0. The combinations are then separated into consonants, sub-consonants and dependent vowels based on: (1) tracking information array and (2) horizontal, vertical, upper bound and lower bound line segments. The example of the algorithm to separate the combinations is illustrated in figure 4.24 and 4.25.

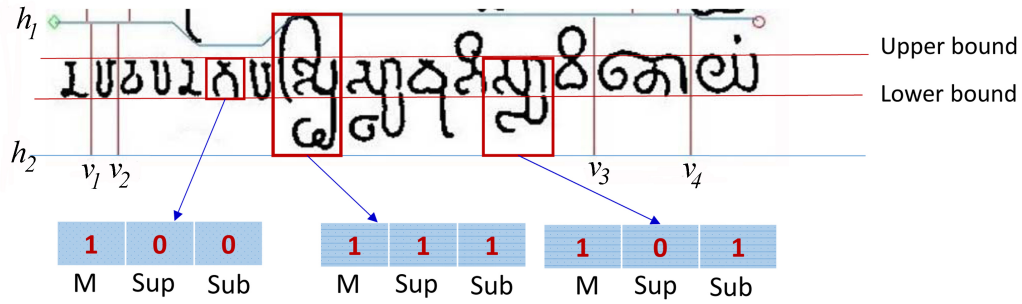


Figure 4.25: The information of connected pixels in tracking information array

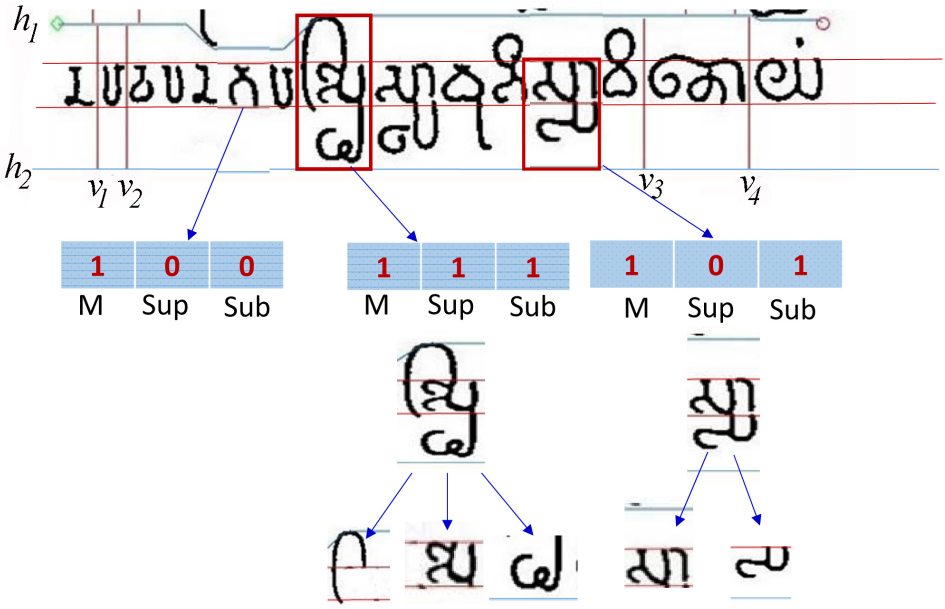


Figure 4.26: The characterization of the combinations from algorithm and the decomposition of the combinations into consonants, sub-consonants and dependent vowels.

The developed algorithm can decompose the combinations in which the sub-consonants and dependent vowels are attached to the point *"Down "* and *"Up "* of the consonants. The algorithm need to be further developed to decompose the combinations in which dependent vowels are attached to the point *"Left "* and *"Right "* of the consonants.

4.3 Summary

In summary, chapter 4 gave new approaches to identify the ancient Khmer characters in the pre-Angkorian period. First, we applied concept of graph theory to represent the structures of the characters. We determined two types of features– topological and geometric– to identify the characters. We described concepts of topology to extract topological features based on the endpoints and junction points of the characters and integral geometry to extract geometric features of the characters having the same topological structures. We gave the description of the algorithm to characterize the combinations of consonants, sub-consonants and dependent vowels.

In chapter 5, we give examples of numerical solutions of implementing analytic solutions in chapter 3 and this chapter.

Chapter 5

Examples

In chapter 5, we give some examples of numerical solutions of implementing analytic solutions which are described in chapter 3 and 4. In the first part of this chapter, we present the results of transfer of 3D Inscription data to 2D representations. We also show examples of reductions of information on 2D representations of the inscriptions to essential script data and to topological and geometrical structures of the script. In the second part of the chapter, we provide examples of the identification of the ancient Khmer script by topological and geometric features. We also show numerical solutions of computing geometric features by curvature approximation.

5.1 Transfer of 3D Inscription Data to 2D Representations

We have scanned ten inscriptions using *Breuckmann smart SCAN-3D-HE scanner* at Angkor Conservation Center (ACC), Siem Reap, Cambodia. Investigating *field of view (FOV)* of 3D scanner to capture the small script in the inscriptions during scanning, we found that *FOV*475mm optimally fits to scan the Khmer inscriptions. It takes 30 minutes to 2 hours to scan an inscription. For an inscription, ten to forty scans were required. The time of scanning relies on the size of an inscription.

3D data models of the inscriptions obtained from the scanning have been processed by using Optocat and Meshlab tools to remove unnecessary mesh, align the mesh and optimize the number of vertices to represent the geometric shapes of the inscriptions. The results of K127 and K852 of removing unnecessary mesh and aligning mesh are shown in figure 5.1 and 5.2.

We then computed the feature vectors of the inscriptions as described in chapter 3 section 3.2 and visualized the script data from the inscriptions. Finally, we received the 2D representations of original inscriptions. The 2D representations are raster images.

The results of 2D representations of inscriptions K127 and K852 are displayed in figure 5.3

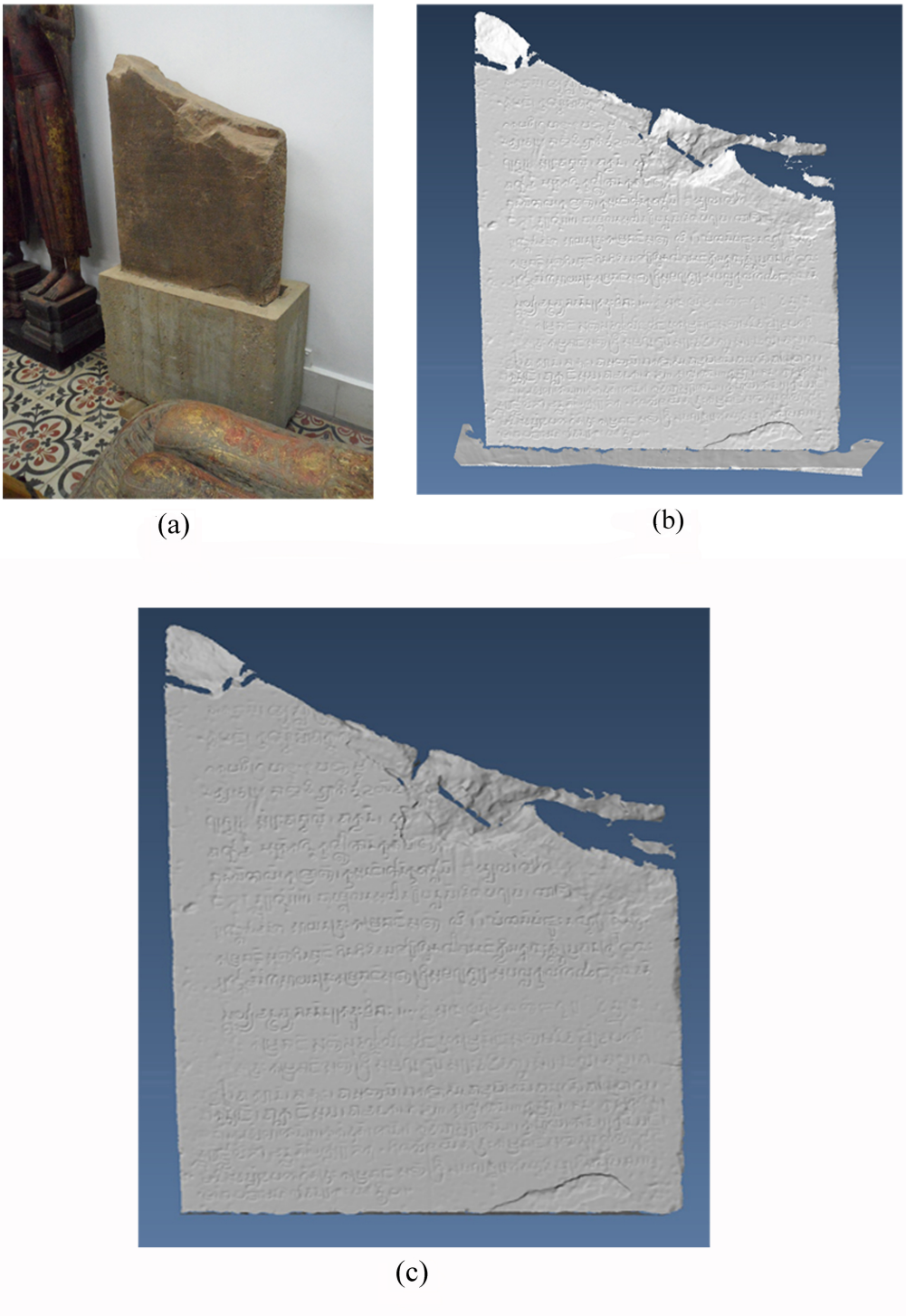
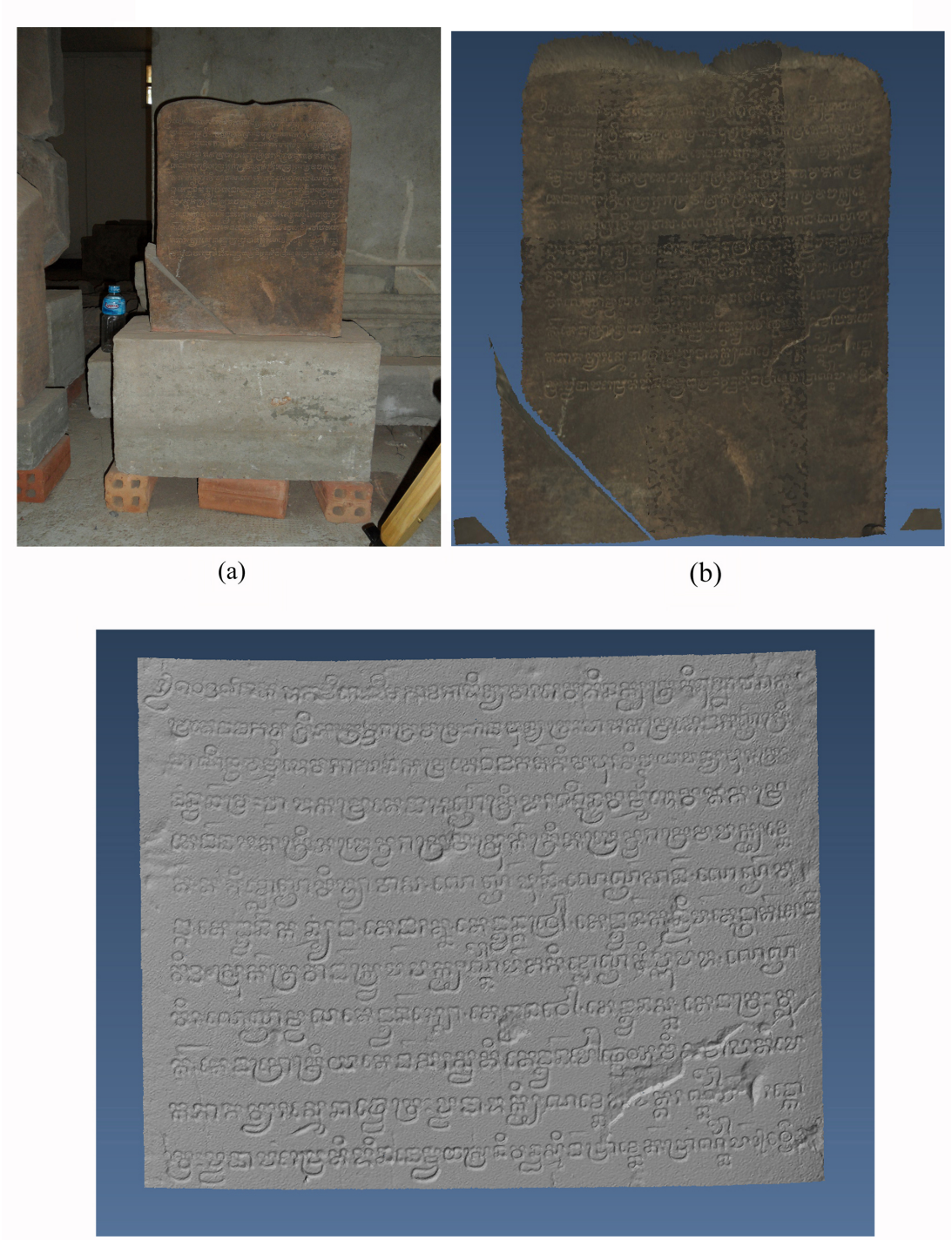


Figure 5.1: (a) The original inscription K127. (b) 3D model of inscription K127. (c) 3D model of inscription K127 after mesh clean-up.

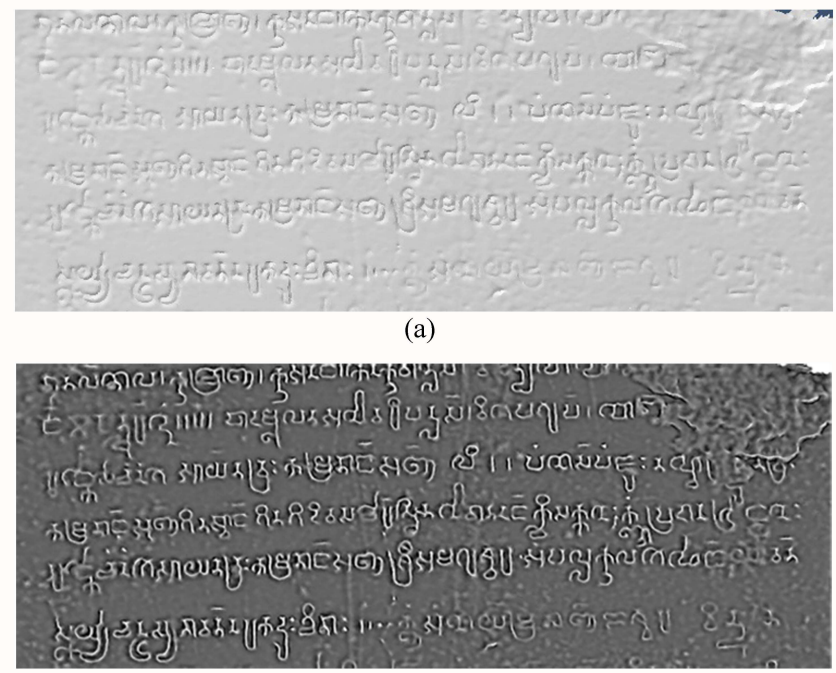


(a) (b)

(c)

Figure 5.2: (a) The original inscription K852. (b) 3D model of inscription K852. (c) 3D model of inscription K852 after mesh clean-up.

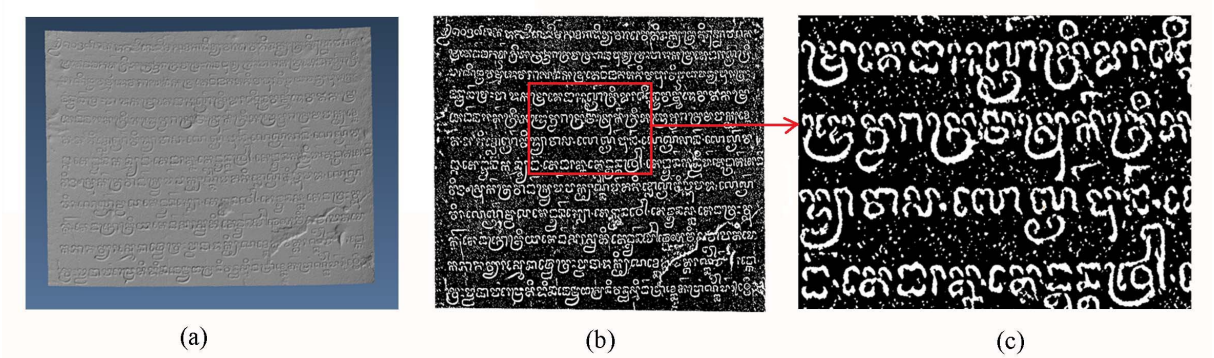
and 5.4 below. We use $\hat{V}_{r=7.50mm}$ to compute feature vectors of inscription K127 and $\hat{V}_{r=4.00mm}$ to compute feature vectors of inscription K852.



(a)

(b)

Figure 5.3: (a) A part of 3D model of inscription K127. (b) The result of script visualization from a curvature-based method using $\hat{V}_{r=7.50mm}$.



(a) (b) (c)

Figure 5.4: (a) 3D model of inscription K852. (b) the result of script visualization from a curvature-based method using $\hat{V}_{r=4.00mm}$. (c) A part of script data of inscription K852.

The curvature-based method to compute the feature vectors of the inscriptions is estimated by volume integral invariant the volume $V_r(\partial D := \partial\Phi \cap B_r(p_i))$. The volume integral invariant is computed with respect to radius of the sphere $B_r(p_i)$. In the figure 5.5, we show the results of script visualization of inscription K852 with respect to radii $r = \{0.30, 4.00\}$ in mm.

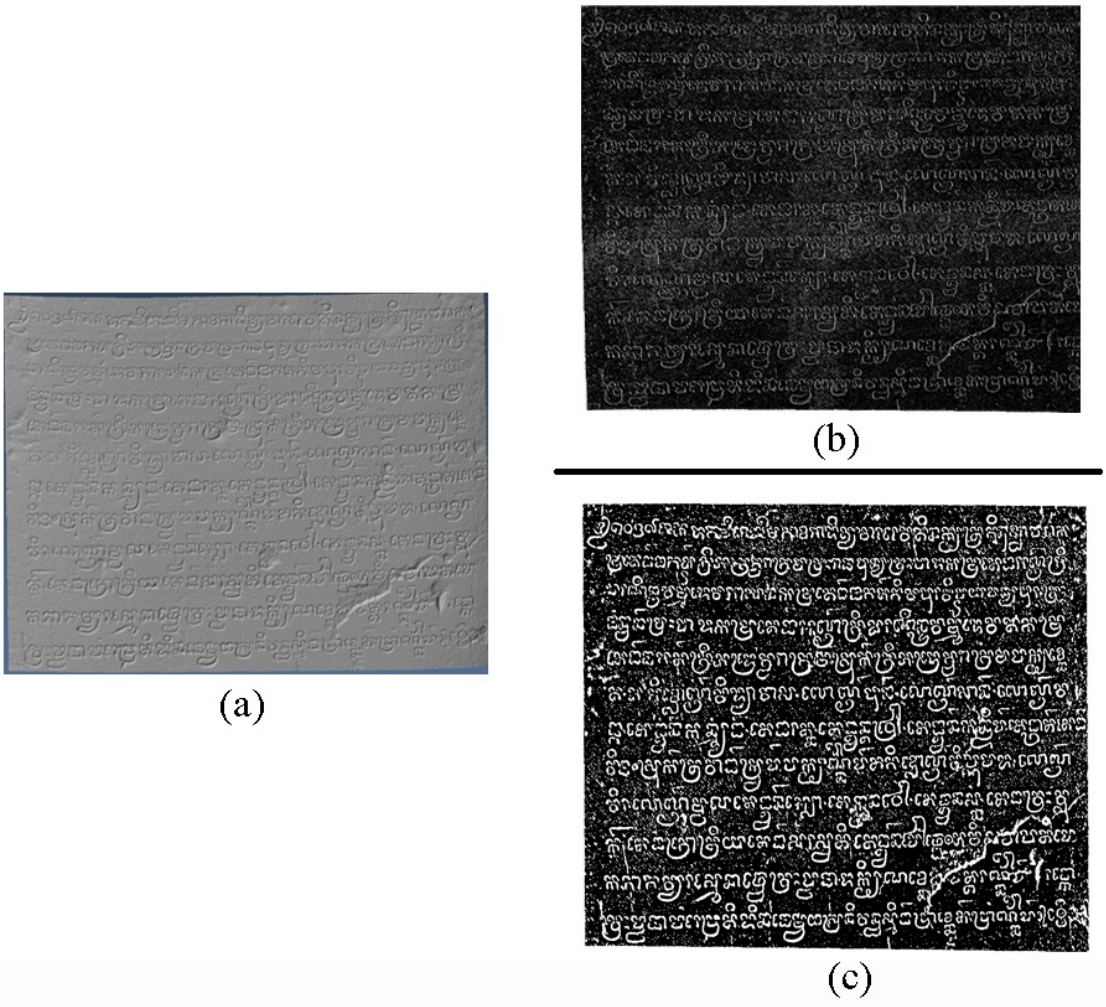


Figure 5.5: (a) The 3D model of inscription K852. The results of a curvature-based method using (b) $\hat{V}_{r=0.30mm}$ and (c) $\hat{V}_{r=4.00mm}$.

Based on our experiences in computations of feature vectors of Khmer inscriptions, we have to select radius of a sphere $r \geq h$, the depth of script engraved on the stones.

5.2 Reduction in Information on 2D Representations to Essential Script Data

In this section, we show the examples of numerical results of thresholding methods– global and local thresholding– which have been described in chapter 3 section 3.3. First, we show the results of applying global thresholding to two consonants– *la* and *ha*. The two consonants were extracted from inscription K127. The raster images of these two characters have ho-

mogeneous backgrounds. We manipulate threshold value $t = \{0.30, 0.40, 0.50, 0.60, 0.70\}$ and optimal threshold value t^* obtained from Otsu's method. The results are depicted in figure 5.6 and 5.7 .

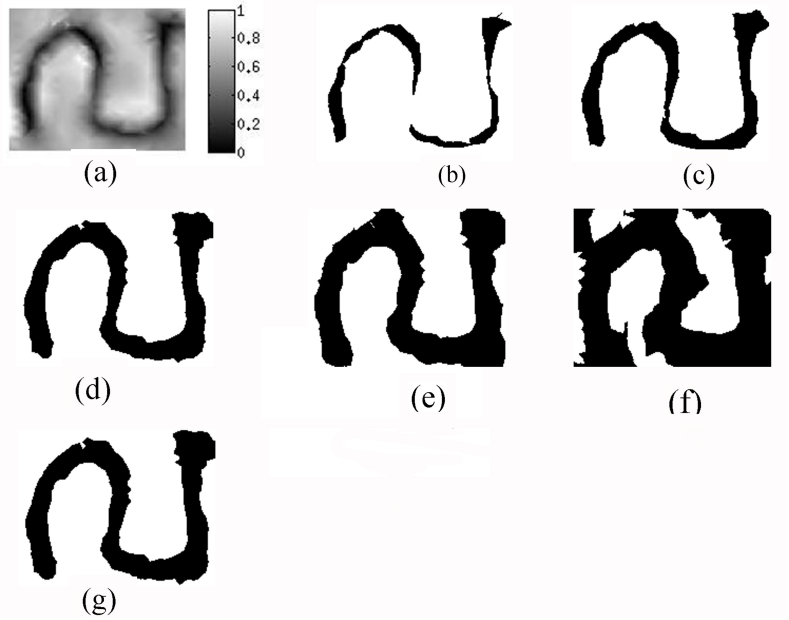


Figure 5.6: (a) Gray-scale image of consonant "*la*". The extraction of essential script data using global thresholding: (b) $t = 0.30$, (c) $t = 0.40$, (d) $t = 0.50$, (e) $t = 0.60$, (f) $t = 0.70$ and (g) $t^* = 0.5255$.

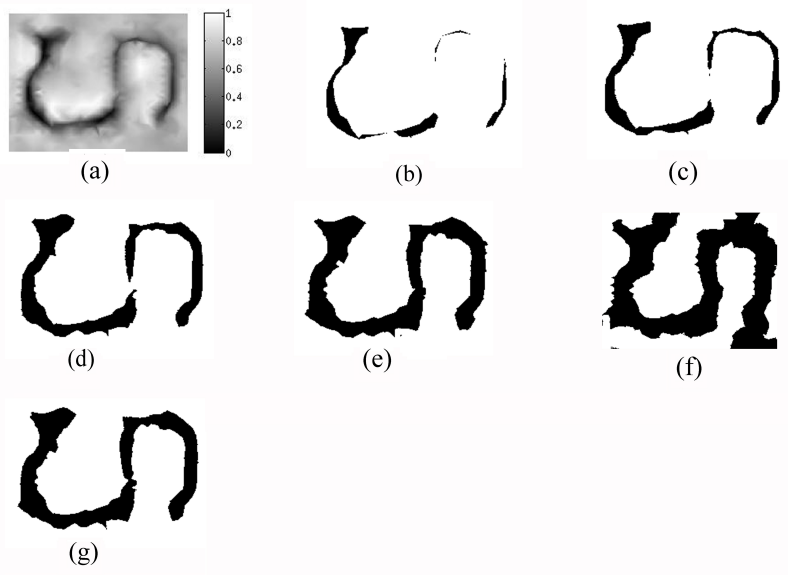


Figure 5.7: (a) Gray-scale image of consonant "*ha*". The extraction of essential script data using global thresholding: (b) $t = 0.30$, (c) $t = 0.40$, (d) $t = 0.50$, (e) $t = 0.60$, (f) $t = 0.70$ and (g) $t^* = 0.5686$.

Applying global thresholding technique using several threshold values, we obtained different results. We selected the threshold value which can preserve the connectivity of the extracted script structures. In figure 5.6 and 5.7, the threshold value $t \in [0.5, 0.6]$, i.e. $t^* = \{0.5255, 0.5686\}$, could be selected because the connectivity of the script structure was preserved. Therefore, t^* from Otsu's method is the best choice.

In the following example, we also show more results of applying global threshold to a part of inscription K127 which was displayed in figure 5.3. In the raster image of inscription K127, we used black pixels to denote the essential script structures against the white background. The background of inscription K127 is considered as homogeneous background.

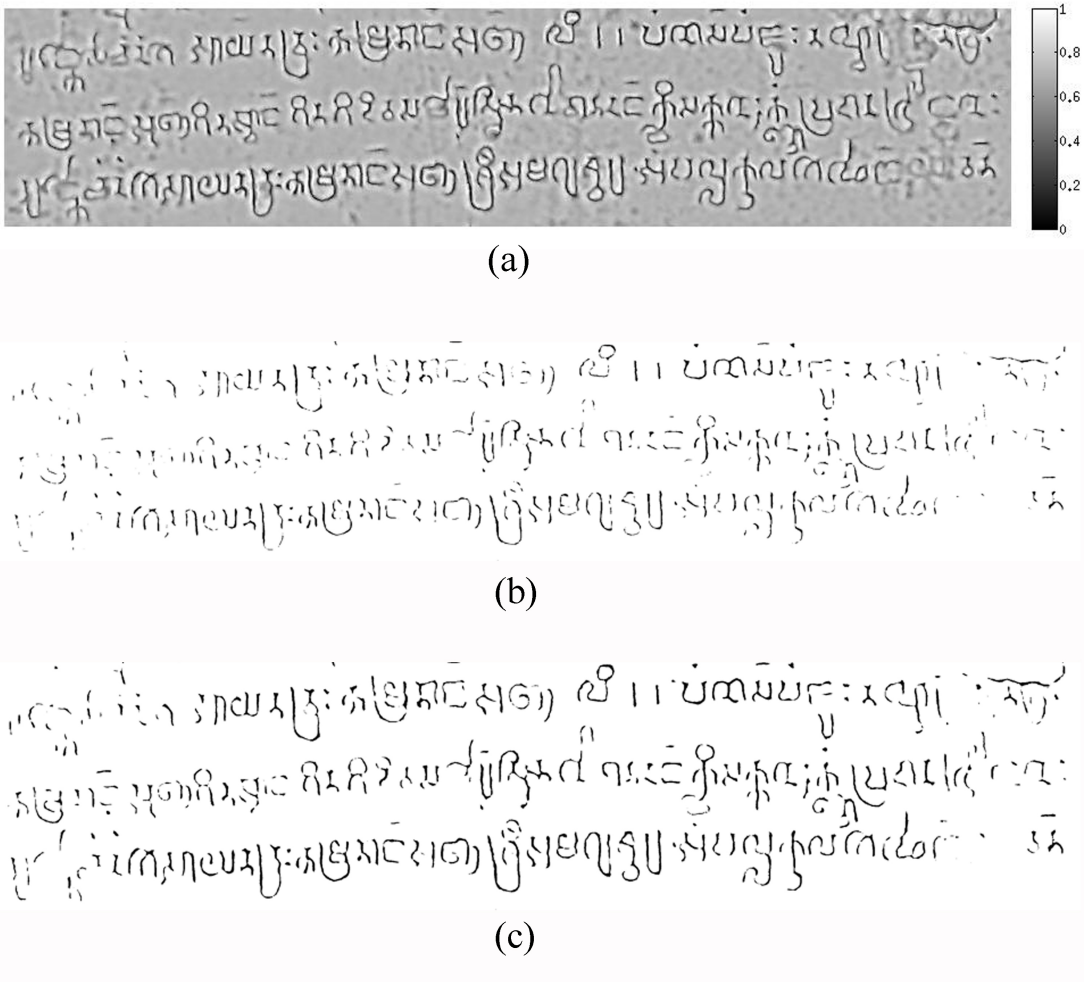


Figure 5.8: (a) Gray-scale image of inscription K127. The extraction of essential script data from gray-scale image using global thresholding: (b) $t = 0.30$ and (c) $t = 0.40$.

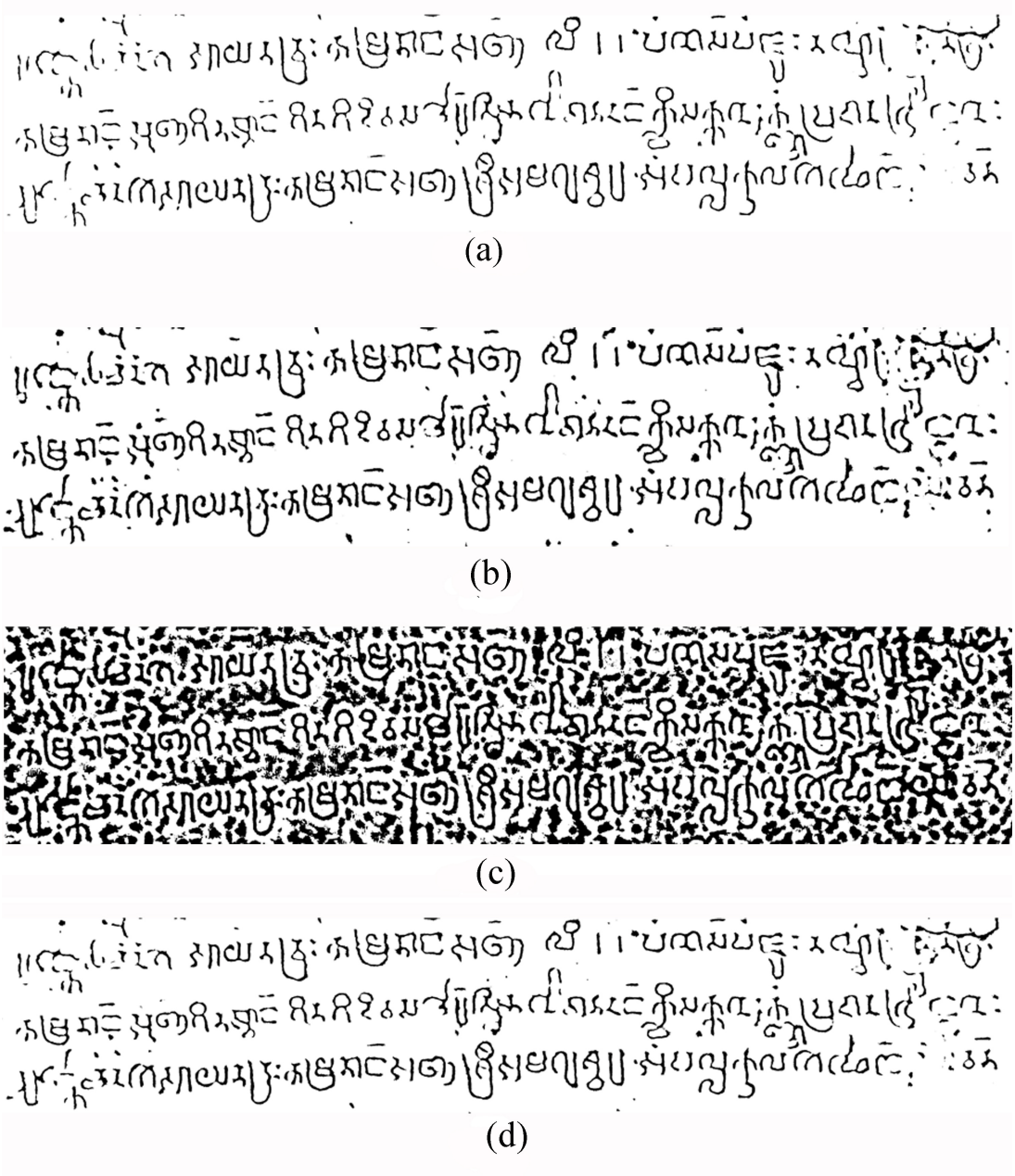


Figure 5.9: The extraction of essential script data from gray-scale inscription 127 using global thresholding: (a) $t = 0.50$, (b) $t = 0.60$, (c) $t = 0.70$ and (d) $t^* = 0.5451$ of Otsu's method.

From the above examples, we showed the application of global thresholding technique on homogeneous background of raster images of the Khmer inscriptions.

Next, we give more results of sauvola's local thresholding method. We applied local thresholding method to the images of the inscriptions which have inhomogeneous background. We already provided an example of implementing sauvola's method to extract the essential script data on inscription K193 in chapter 3 section 3.3 (p. 31-32). In this section, we give another ex-

periment results of extracting the essential script data on inscription K1250 using several small windows $n \times n$ in figure 5.10 and 5.11. We also give the result of applying Otsu's method to extract the essential script data on inscription K1250.

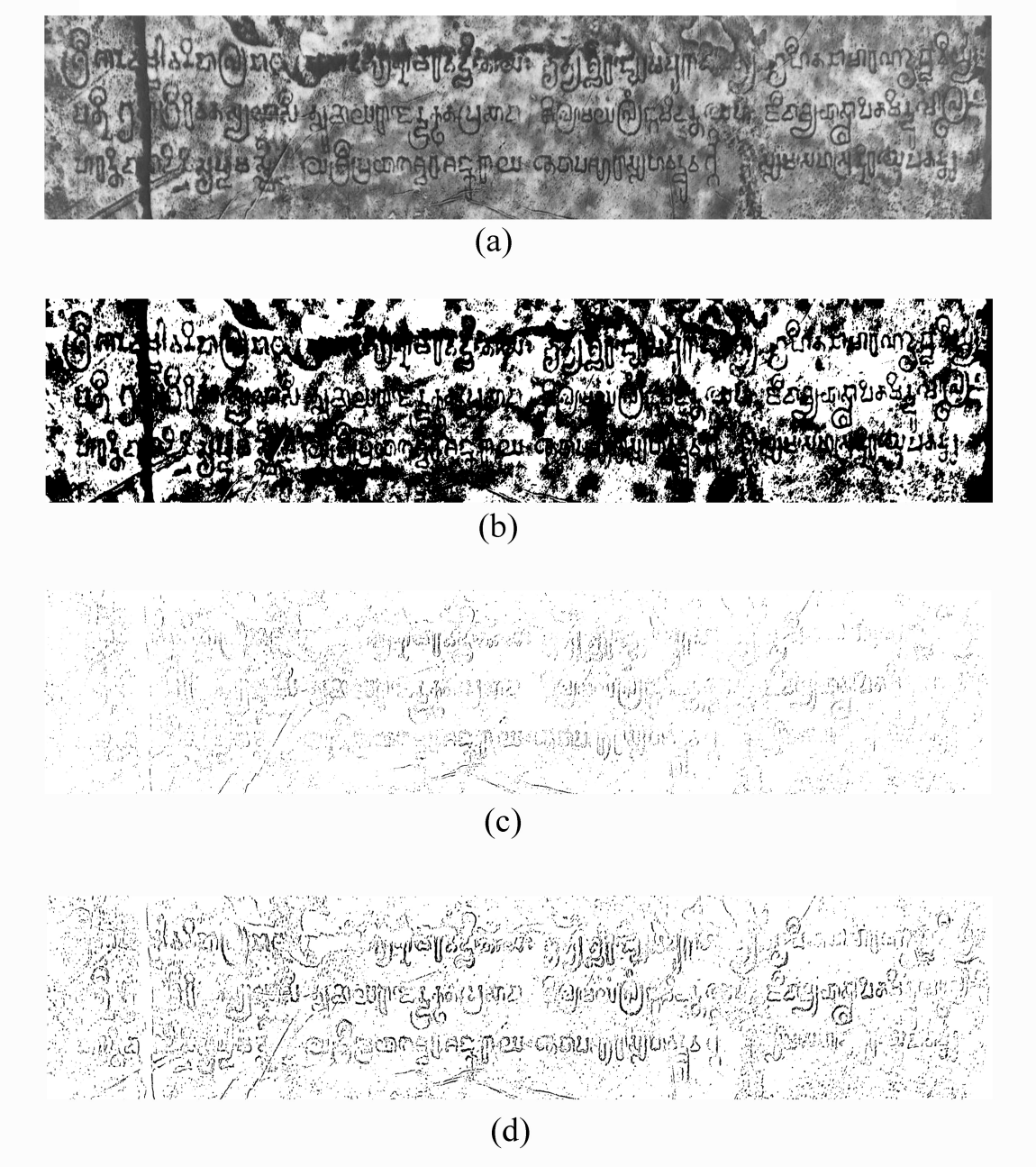


Figure 5.10: (a) Gray-scale image of inscription K1250. The extraction of the essential script data from gray-scale image of inscription K1250 using (b) Otsu's method with optimal threshold $t^* = 0.5294$ and using sauvola's method with small window: (c) 15×15 and (d) 25×25.

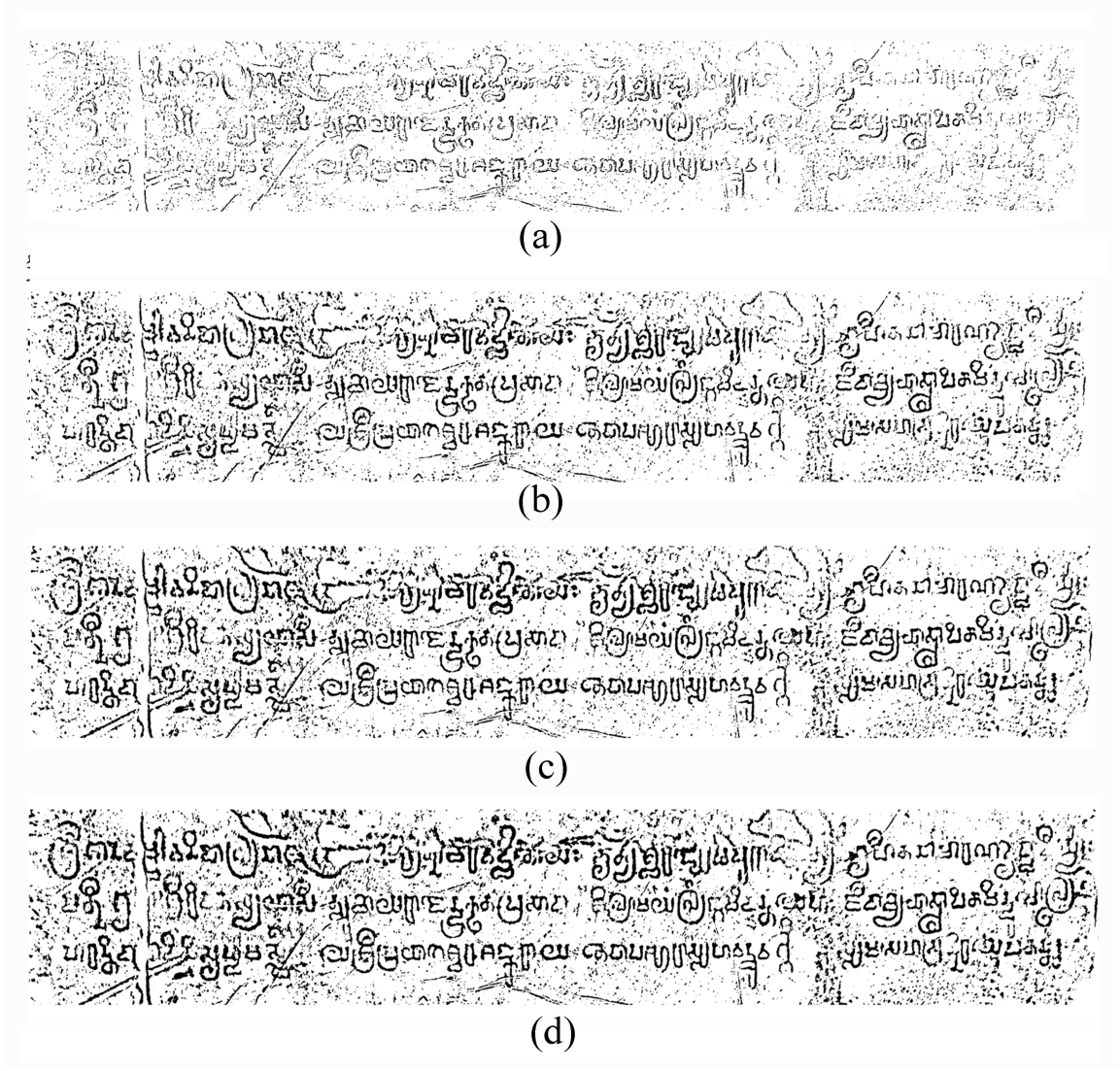


Figure 5.11: The extraction of essential script data from gray-scale image of inscription K1250 using sauvola's method with small window: (a) 35×35, (b) 45×45, (c) 55×55 and (d) 65×65.

As we explained in chapter 3 section 3.3 (p.30), obtaining the satisfactory result from sauvola's method is the process of trial and error. However, according to our defined criteria in chapter 3 section 3.3 (p. 30), we can claim that the result from small window 65×65 in figure 5.11d is the satisfactory result.

5.3 Characterization of Ancient Khmer Characters

Characterization of Ancient Khmer Based on Topological Features

In this section, we give the results of characterization of ancient Khmer characters based on the topological features. Topological features of the ancient Khmer characters are defined by the total number of nodes of degree $1, 3, 4, 5$ except degree 2. So, the topological feature tuples of the ancient Khmer characters are denoted by

$$V_{TFC(kc)} = (n_{d_1}, n_{d_3}, n_{d_4}, n_{d_5}) \tag{5.1}$$

where n_{d_1} is endpoints and $n_{d_3}, n_{d_4}, n_{d_5}$ are junction points of degree 3, 4 and 5, respectively.

The following are the examples of the characterization of some characters based on their topological features. The examples are shown in figure 5.12 and 5.13.

- $V_{TFC(kc)} = (2, 0, 0, 0)$

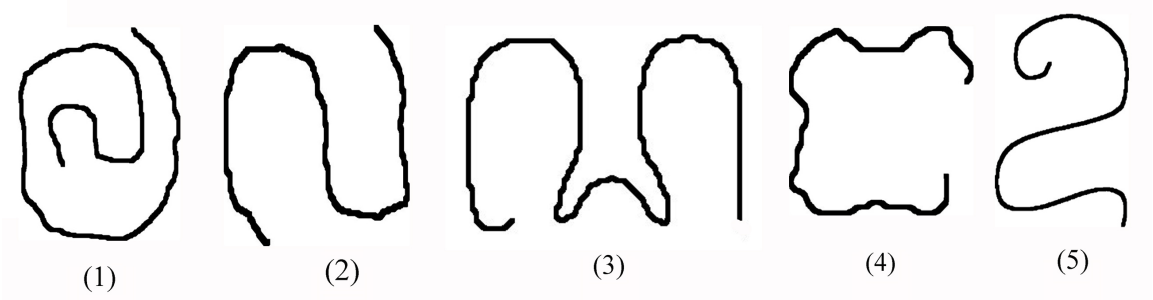


Figure 5.12: (1) Consonant *"la1"*. (2) Consonant *"la2 "*. (3) Consonant *"na"*. (4) Consonant "*nā*". (5) Independent vowel *"O "*.

- $V_{TFC(kc)} = (3, 1, 0, 0)$

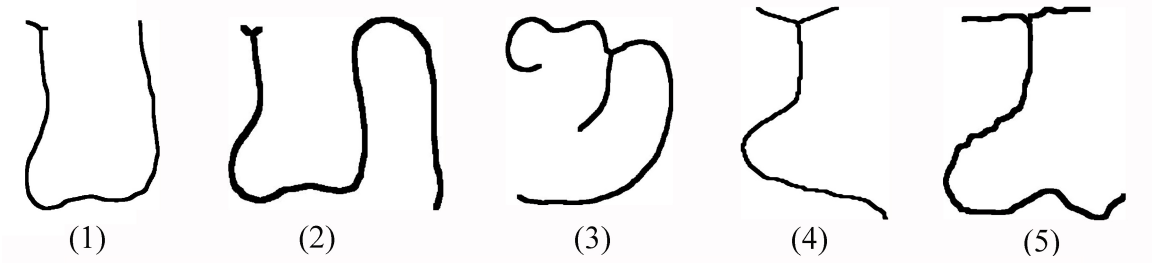


Figure 5.13: (1) consonant *"pa"*. (2) consonant*"ha"*. (3) consonant *"ňa"*. (4) consonant*"da"*.(5) consonant*"ta"*.

We give the topological feature tuples to characterize of ancient Khmer characters in table 5.1. Based on the feature tuples, we can classify the ancient Khmer Characters into 15 classes. Some classes are unique, that is, only character exists in the class.

n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}	Number of characters in the class
0	0	0	0	5
0	2	0	0	2
1	1	0	0	8
1	1	1	0	1
2	0	0	0	25
2	0	1	0	3
2	2	0	0	9
3	1	0	0	15
3	1	1	0	1
3	3	0	0	1
4	0	1	0	2
4	2	0	0	5
4	4	0	0	1
5	1	1	0	2
5	3	0	0	8

Table 5.1: The topological feature tuples of ancient Khmer characters.

Characterization of Ancient Khmer Characters Based on Geometric Features

When the characters have the same topological features, e.g. the consonants in figure 5.12 and 5.13, we use the geometric features to further identify the characters. The geometric features are defined by the curvature distribution of the edges of the characters. In this section, we give some examples of computing curvature with respect to r, sc, scv, ϵ. We used the characters in figure 5.12 for curvature computation. The results of the curves representing the curvature distributions of each character are in the appendix D. In the below tables, we display the sequences, and geometric feature tuples of the characters.

Parameters	Sequences	Feature tuple
$r = 40$, $sc = 50$, $scv = 100$, $\epsilon = 0.0223$	(0, 1, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1, 0, -1)	(1, -1)
$r = 50$, $sc = 50$, $scv = 100$, $\epsilon = 0.0274$	(0, 1, 0, -1, 0, -1, 0, -1, 0, -1)	
$r = 60$, $sc = 50$, $scv = 100$, $\epsilon = 0.0324$	(1, 0, -1, 0, -1, 0, -1)	
$r = 70$, $sc = 50$, $scv = 100$, $\epsilon = 0.0375$	(1, 0, -1, 0, -1, 0, -1)	
$r = 80$, $sc = 50$, $scv = 100$, $\epsilon = 0.0424$	(1, 0, -1, 0, -1)	
$r = 90$, $sc = 50$, $scv = 100$, $= 0.0471$	(1, 0, -1, 0, -1)	

Table 5.2: The results of computations of the sequences of the consonant *"la1"* with respect to the parameters– r, sc, scv, ϵ.

Parameters	Sequences	Feature tuple
$r = 100$, $sc = 50$, $scv = 100$, $\epsilon = 0.0347$	(1, 0, -1, 0, -1)	(1, -1)
$r = 110$, $sc = 50$, $scv = 100$, $\epsilon = 0.0371$	(1, 0, -1, 0, -1)	
$r = 120$, $sc = 50$, $scv = 100$, $\epsilon = 0.0396$	(1, 0, -1, 0, -1)	
$r = 130$, $sc = 50$, $scv = 100$, $\epsilon = 0.0423$	(1, 0, -1)	
$r = 140$, $sc = 50$, $scv = 100$, $\epsilon = 0.0450$	(1, 0, -1)	
$r = 150$, $sc = 50$, $scv = 100$, $\epsilon = 0.0478$	(1, 0, -1)	

Table 5.3: The results of computations of the sequences of the consonant *"la2"* with respect to the parameters– r, sc, scv, ϵ.

Parameters	Sequences	Feature tuple
$r = 40$, $sc = 50$, $scv = 100$, $\epsilon = 0.0367$	(1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0)	(1, -1, 1, -1, 1)
$r = 50$, $sc = 50$, $scv = 100$, $\epsilon = 0.0414$	(1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0)	
$r = 60$, $sc = 50$, $scv = 100$, $\epsilon = 0.0445$	(1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1, 0)	
$r = 70$, $sc = 50$, $scv = 100$, $\epsilon = 0.0459$	(1, 0, -1, 0, 1, 0, -1, 0, 1, 0)	
$r = 80$, $sc = 50$, $scv = 100$, $\epsilon = 0.0483$	(1, 0, -1, 0, 1, 0, -1, 0, 1, 0)	
$r = 90$, $sc = 50$, $scv = 100$, $\epsilon = 0.0520$	(1, 0, -1, 0, 1, 0, -1, 0, 1, 0)	

Table 5.4: The results of computations of the sequences of the consonant *"na"* with respect to the parameters– r, sc, scv, ϵ.

Parameters	Sequences	Feature tuple
$r = 40, sc = 50, scv = 100, \epsilon = 0.0186$	(1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1)	
$r = 50, sc = 50, scv = 100, \epsilon = 0.0229$	(1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1)	
$r = 60, sc = 50, scv = 100, \epsilon = 0.0265$	(1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1)	(1, -1, 1, -1, 1,
$r = 70, sc = 50, scv = 100, \epsilon = 0.0291$	(1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1)	-1, 1)
$r = 80, sc = 50, scv = 100, \epsilon = 0.0299$	(1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1)	
$r = 90, sc = 50, scv = 100, \epsilon = 0.0296$	(1, 0, -1, 0, 1, 0, -1, 0, 1, 0, -1, 0, 1)	

Table 5.5: The results of computations of the sequences of the consonant *"nā"* with respect to the parameters– r, sc, scv, ϵ.

Parameters	Sequences	Feature tuple
$r = 40, sc = 50, scv = 100, \epsilon = 0.0216$	(1, 0, -1, 0, 1)	
$r = 50, sc = 50, scv = 100, \epsilon = 0.0269$	(1, 0, -1, 0, 1)	
$r = 60, sc = 50, scv = 100, \epsilon = 0.0318$	(1, 0, -1, 0, 1)	(1, -1, 1)
$r = 70, sc = 50, scv = 100, \epsilon = 0.0365$	(1, 0, -1, 0, 1)	
$r = 80, sc = 50, scv = 100, \epsilon = 0.0406$	(1, 0, -1, 0, 1)	
$r = 90, sc = 50, scv = 100, \epsilon = 0.0442$	(1, 0, -1, 0, 1)	

Table 5.6: The results of computations of the sequences of the independent vowel *"o"* with respect to the parameters– r, sc, scv, ϵ.

We get the geometric feature tuples for the characters– consonant *"la"* (C_{la}), consonant *"na"* (C_{na}), consonant "*nā*"($C_{n\bar{a}}$) and independent vowel *"O"* (INV_O) as follows.

- From table 5.2, the geometric feature tuple of $C_{la1} = (1, -1)$.
- From table 5.3, the geometric feature tuple of $C_{la2} = (1, -1)$.
- From table 5.4 the geometric feature tuple of$C_{na} = (1, -1, 1, -1, 1)$.
- From table 5.5, the geometric feature tuple of $C_{n\bar{a}} = (1, -1, 1, -1, 1 - 1, 1)$.
- From table 5.6, the geometric feature tuple of $INV_O = (1, -1, 1)$.

5.4 Summary

In conclusion, chapter 5 gave the examples of implementing the methods which we found in chapter 3 and 4. We showed results of transfer of 3D inscriptions to 2D representations using curvature-based method. We also provided the numerical solutions to reduce 2D representations to essential script data. Finally, we showed the numerical results to characterize ancient Khmer characters based on topological and differential geometric features.

In chapter 6, we present the software tool to implement the methods in chapter 3 and 4 for analyzing characters and inscriptions in pre-Angkorian period.

Chapter 6

Description of the Developed Software

In chapter 6, we provide the description of developed software to implement the mathematical methods which were described in chapter 3 and 4. This software framework is called *Khmer Inscriptions Analysis Software Tool* (KIAST). The software framework is built in numerical analysis software– Matlab. The interface of the software framework is also constructed to give user-friendly interaction between users and software tool. The framework consists of four important parts– image processing, image analysis, visualization and existing techniques of image enhancements.

The chapter starts with the architecture of KIAST. It is followed by image processing part. We then present image analysis part. At last, visualization and existing techniques of image enhancements are provided.

6.1 Architecture of KIAST

The KIAST is programmed in numerical analysis software– Matlab. Matlab is one of the practical numerical analysis software which is widely used in computational science. Matlab is the matrix-oriented software which is suitable to implement image processing and analysis of images. Moreover, Matlab is the independent platform, i.e. it supports several important operating system such as Window, Linux and Mac.OS. It also allows programmers to design interface for user-friendly interaction with users and write other languages including C, C++, Java and so on.

The KIAST consists of four important parts– image processing, image analysis, visualization and some general tools which are used for image manipulation. The components of KIAST is shown in figure 6.1. The interface of KIAST is also displayed in figure 6.2.

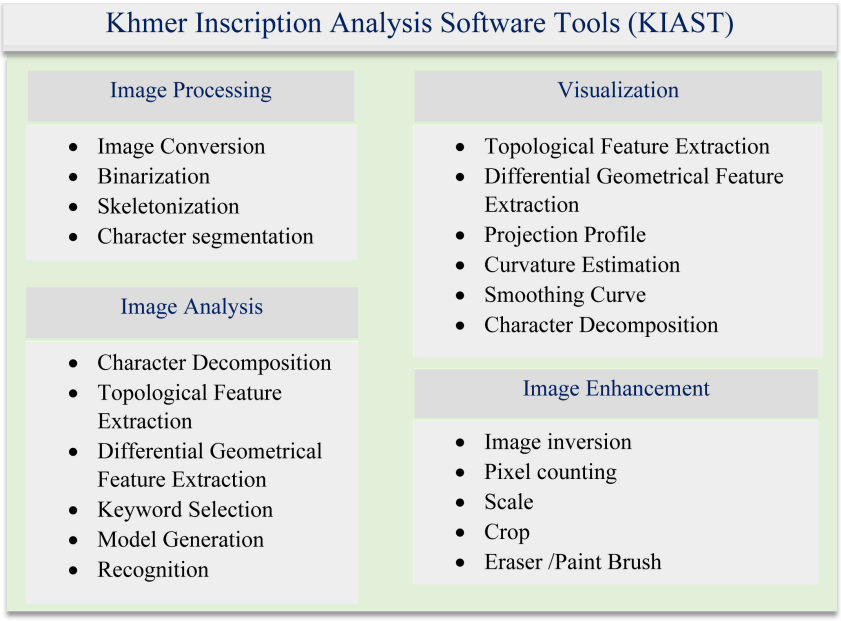


Figure 6.1: The architecture of Khmer Inscriptions Analysis Software Tool (KIAST)

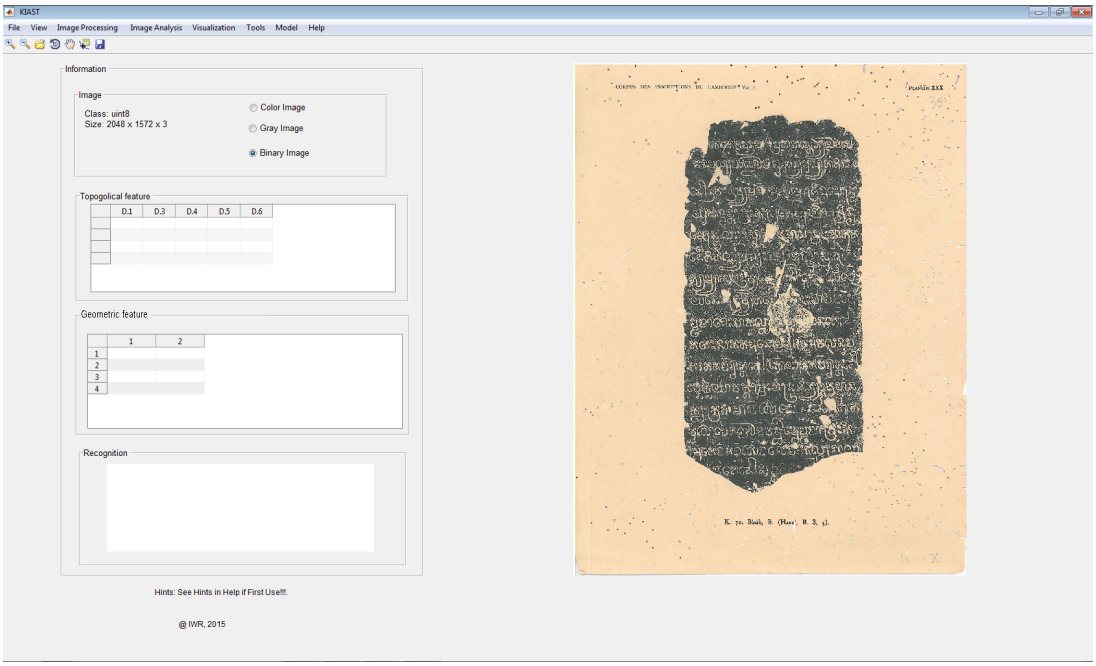


Figure 6.2: The interface of KIAST

We show the flowchart to analyze and recognize the ancient Khmer characters and inscriptions in figure 6.3 and we describe functionality of KIAST in the following sections.

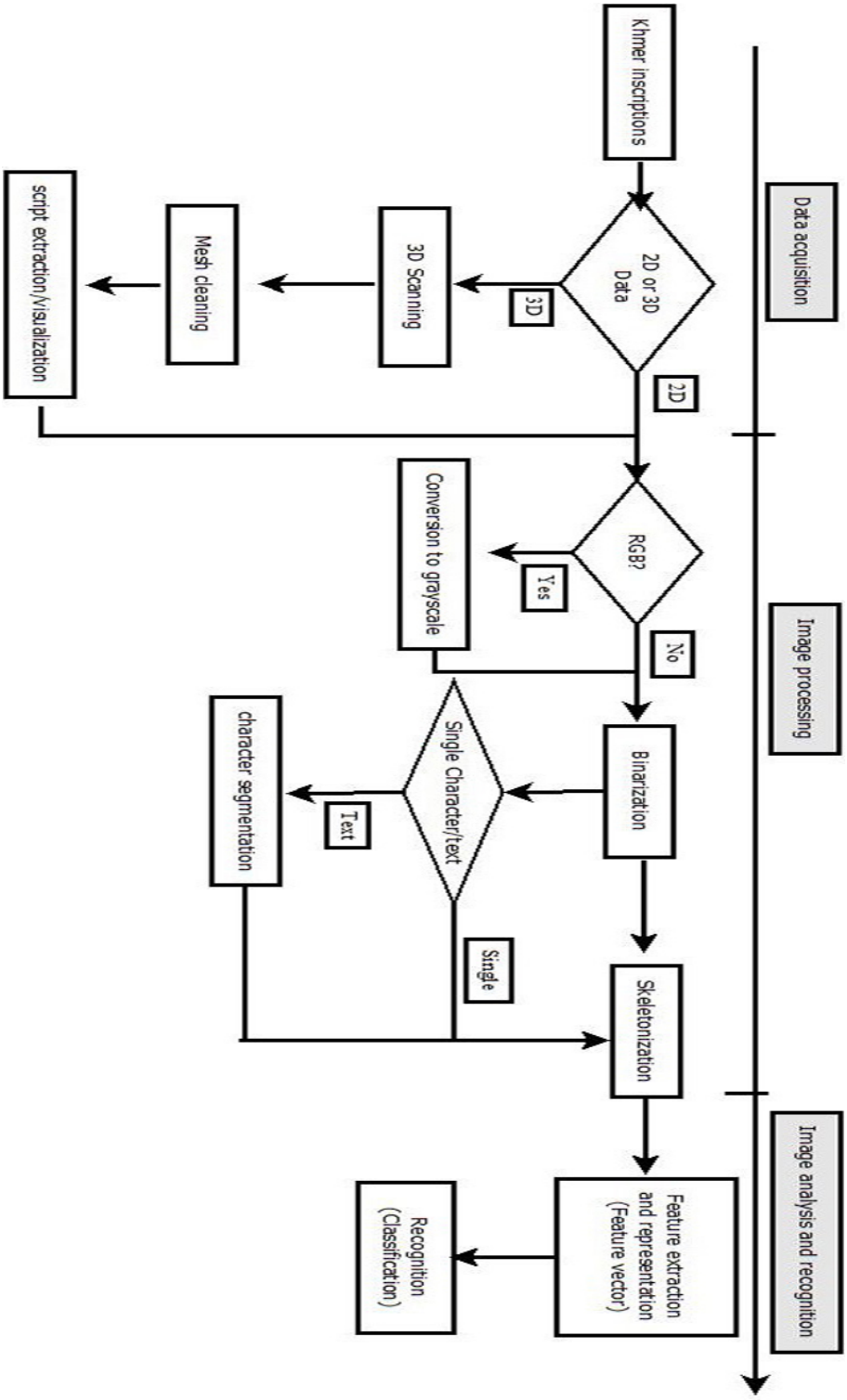


Figure 6.3: The flowchart to analyze and recognize the ancient Khmer characters and inscriptions

6.2 Image Processing Software Framwork

Image processing framework deals with the implementation of the mathematical methods which were described in chapter 3. It includes binarization (global and local thresholding), thinning operation (skeletonization) and character segmentation.

6.2.1 Binarization

To do binarization, the input image has to be in gray scale. If input image is RGB image, it needs to be converted. The equation to convert from RGB image to gray-scale image is defined below.

$$I_{grayscale}(x, y) = 0.299 * R(x, y) + 0.587 * G(x, y) + 0.114 * B(x, y) \tag{6.1}$$

where $I_{grayscale}(x, y)$ is the intensity of pixel at the position (x, y) and $R(x, y), G(x, y), B(x, y)$ are red, green, and blue channel of RGB image, respectively. The intensity of pixel $I_{grayscale}(x, y)$ is the integer value ranging in [0-255]. In KIAST, we provide tool bar *"Grayscale Image"* to convert color image to gray-scale image. The operation is shown in figure 6.4.

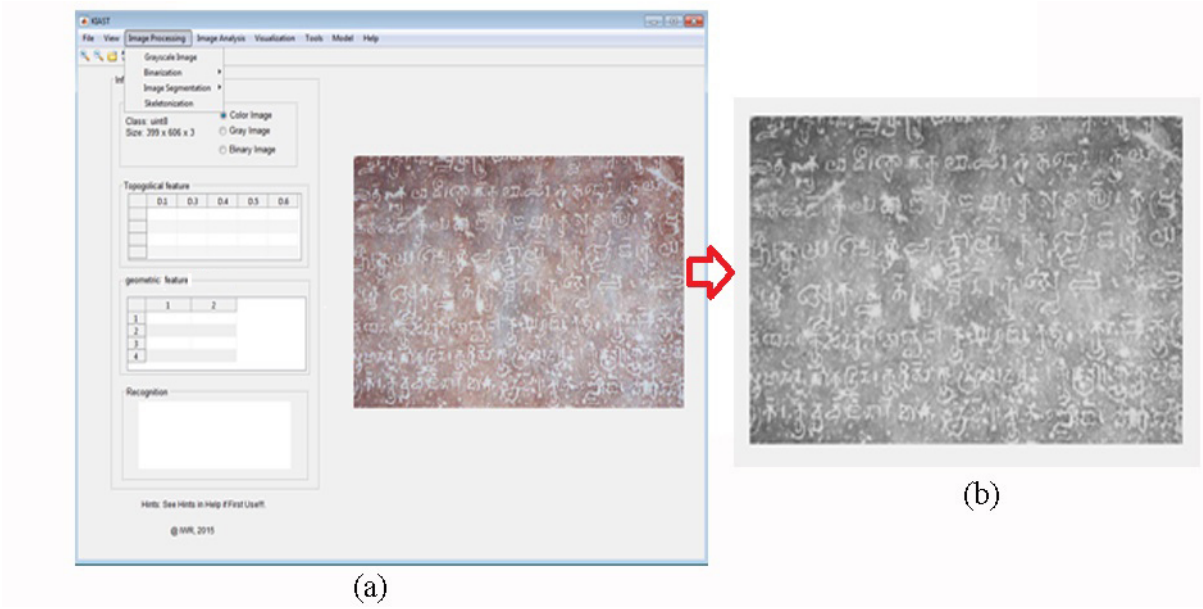


Figure 6.4: (a) The operation of converting color image to gray-scale image. (b) The gray image after conversion.

The image information is also displayed in a panel of the interface as seen in figure 6.5.

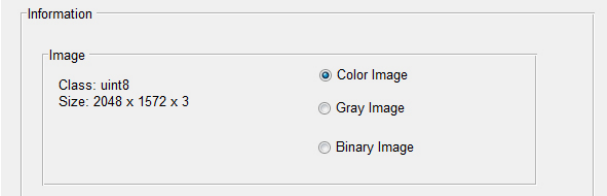


Figure 6.5: The panel to show the information about image.

Binarization consists of the global threshold and local threshold. Global threshold applies one threshold into gray-scale image to classify all the pixels into two classes 1s and 0s while local threshold considers a threshold to apply for each pixel. We have explained global and local thresholding methods in chapter 3 section 3.3. In KIAST interface, the tool bar *"Binarization"* is offered for global and local thresholding. The tool bar is shown in figure 6.6.

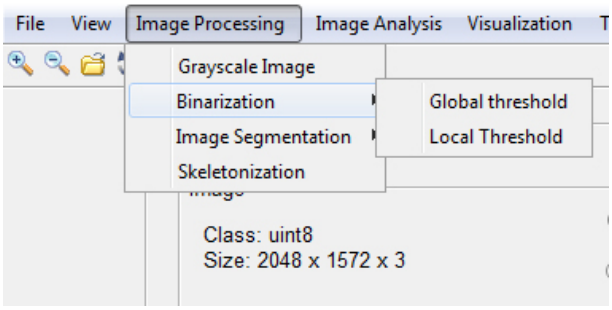


Figure 6.6: The tool bar of the global and local threshold methods in KIAST interface.

We also provide the window as displayed in figure 6.7a to manipulate the global threshold, if the default threshold of otsu's method does not satisfy the wanted results. For a local threshold, a parameter $n \times n$ of small window is required as in figure 6.7b. The local threshold is the best choice, if the inscription images contain noise and intensities of image pixels are not homogeneous between the script data and background.

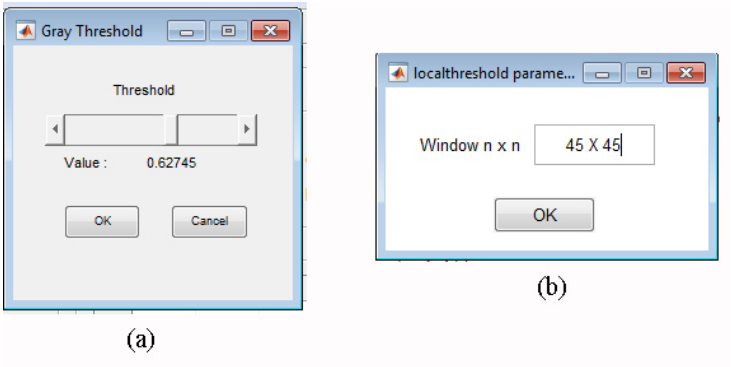


Figure 6.7: (a) The windows to adjust the global threshold. (b) The window to adjust local threshold.

6.2.2 Skeletonization

As explained in chapter 3 section 3.4, we used thinning operation– the so-called skeletonization– to reduce the script data into essential topological and geometrical structures of the script. It was done by Zhang and Suen's method. In KIAST, the skeletonization does not require any parameters. The algorithm automatically optimize the structure of the script by preserving the connectivity of the character structures. The tool bar *"Skeletonization"* and the thinning operation are shown in figure 6.8 and 6.9.

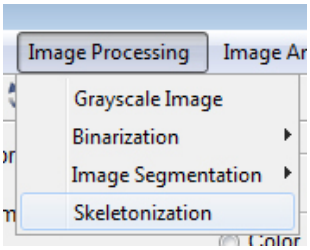


Figure 6.8: The tool bar of the skeletonization.

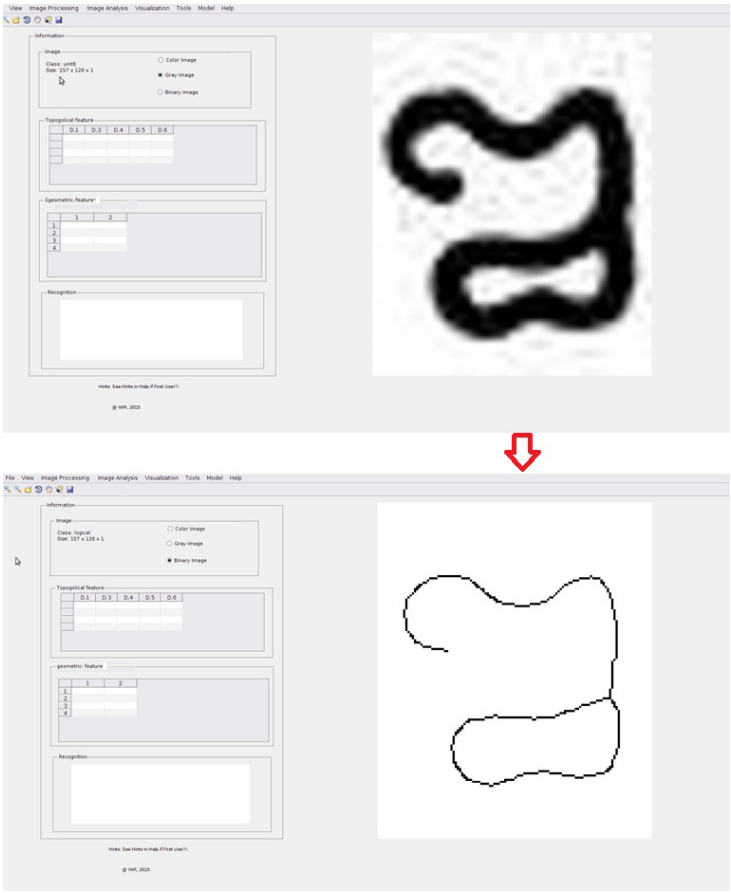


Figure 6.9: The example of the thinning operation in KIAST.

6.2.3 Character Segmentation

The character segmentation needs three parameters: initial state $S_{initial}(x, y)$, goal states $S_{goal}(x, y)$ and sub plane $M(x, y)$. The algorithm will work in the predefined sub-plane to reduce computation time. The three parameters can be directly assigned in the axis object which is used to display inscription images in the KIAST interface. The character segmentation is shown in figure 6.10.

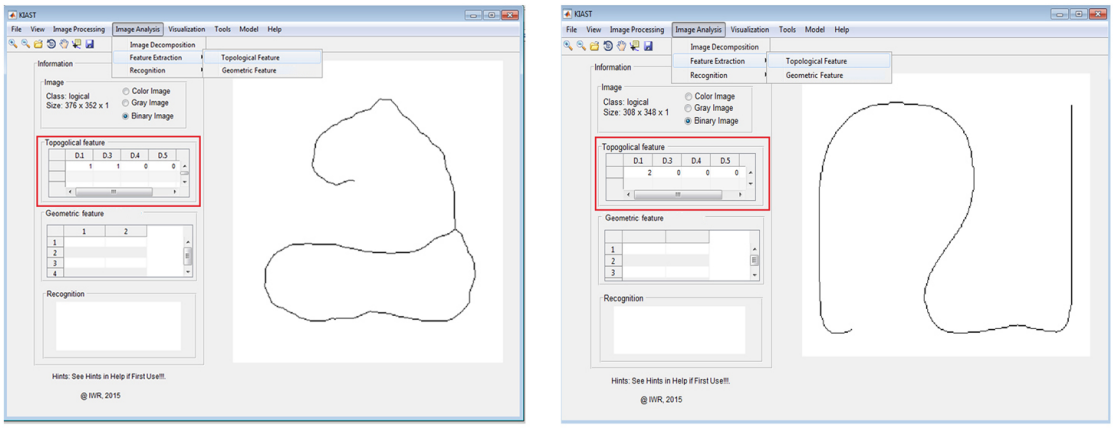
Figure 6.10: The character segmentation in KIAST

6.3 Image Analysis Software Framework

We design the image analysis framework to implement methods described in chapter 4 to analyze the structures of ancient Khmer characters and recognize the characters.

6.3.1 Topological Features Extraction

In KIAST, a function to compute topological features of characters is provided. We show the tool bar linked to the function in figure 6.11. To compute the topological features of a character, we need to provide the binary skeleton image of the characters. There is no parameter for the computation. A function of KIAST automatically detects the endpoints and junction points of various degrees from the skeleton of a character structure and then sum of endpoints and junction points of each degree.



(a) (b)

Figure 6.11: The computation of topological features of consonant– (a) *"kha"* and (b) "la". The results of the computations are shown in tables marked in red rectangles.

6.3.2 Geometric Feature Extraction

To compute geometric features of a character, it requires parameters as explained in chapter 4 section 4.1.

In KIAST, a window is given to manipulate the parameters. The window for the parameter adjustment to compute the features is shown in figure 6.12.

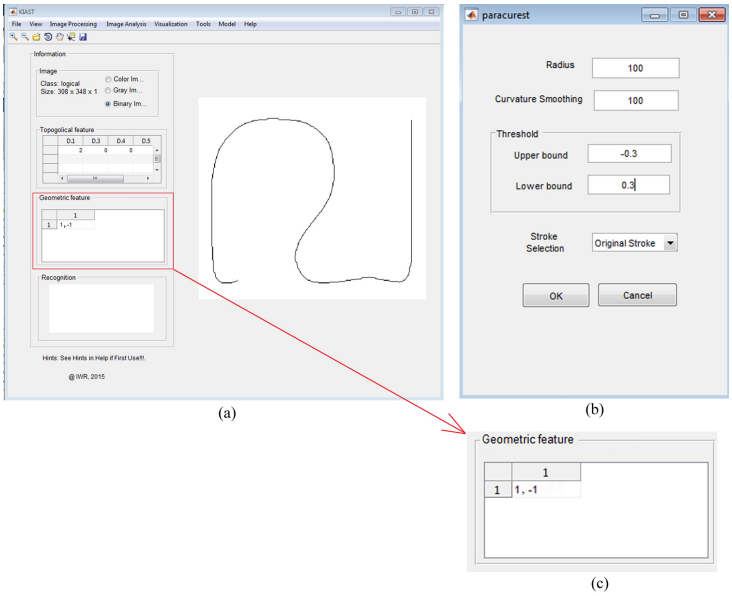


Figure 6.12: (a) The main window to display the result of computing the geometric features. The result is shown in red rectangle. (b) The window to manipulate the parameters. (c) The table to show the result.

In figure 6.12a, we computed the geometric features of consonant *"la"*. The result is shown in a table marked in red rectangle. The table of the geometric features is shown in figure 6.12c.

As shown in figure 6.12b, to compute the geometric features of consonant *"la"*, we used parameters: radius of disk $r = 100$, curvature smoothing $scv = 100$ and threshold $t = \pm 0.3$, to compute the original curve segment without smoothing. We can select curve segments with or without smoothing. If stroke selection → original stroke, it means we are computing curve segment without smoothing.

The parameters to smooth the curve segments are provided in other tool bar called *"Stroke Smoothing"* as depicted in figure 6.13.

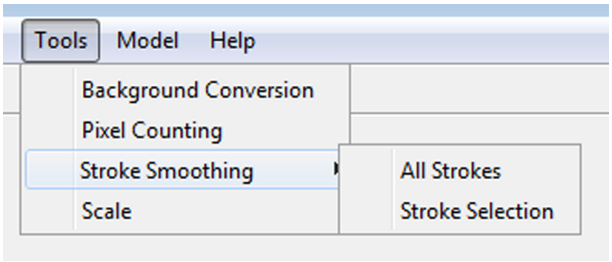


Figure 6.13: The tool bar to smooth the curve segments.

Selecting *Stroke Smoothing* → *All Strokes*, it means we intend to smooth multi-curve segments of the character. Then we have to provide the parameters of smoothing as described in figure 6.14. Semicolons are used to separate the parameter values of each curve segments.

Figure 6.14: The window to adjust the parameters to smooth the curve segments.

Figure 6.14 illustrates that we smooth 3 curve segments c_1, c_2 and c_3 (*Stroke No*:1; 2; 3). Each curve segment is computed with different smoothing parameters given as follows.

- Smoothing curve c_1 by using span $s_1 = 20$ and smoothing method "moving average"
- Smoothing curve c_2 by using span $s_2 = 30$ and smoothing method "loess"
- Smoothing curve c_3 by using span $s_3 = 40$ and smoothing method "lowess"

We can also smooth a single curve by selecting *Stroke Smoothing* → *Stroke Selection* as in figure 6.13. Then a window of the parameter adjustment is displayed as in figure 6.15.

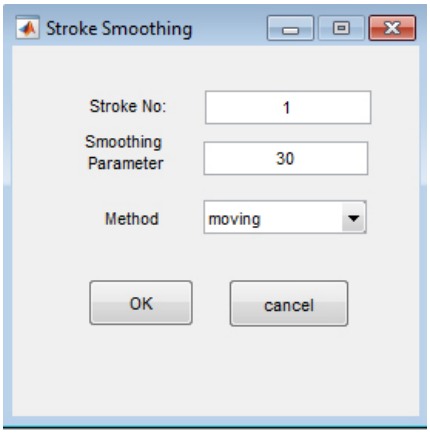


Figure 6.15: The window to adjust the parameters to smooth a single curve segment.

We can find all the information about the curve segments of the characters in workspace of KIAST as displayed in figure 6.16 (p. 93).

6.3.3 Recognition

In KIAST, the results of recognition are displayed in *Recognition* panel as in figure 6.18. To recognize the character in KIAST, first we have to have training model– topological feature training model and geometric feature training model of ancient Khmer characters and predefined class of character names of both training models.

We designed two steps to characterize the characters. First, an unknown character was characterized based on its topological features. Finally, we used geometric features to further classify the character.

As shown in Figure 6.18b, Based on the topological features of the character $TFC_{kc} = (2, 0, 0, 0)$, the topological class of character is displayed. We then computed the geometric features of the character; we further characterized the character. The final result is shown, i.e. the character is identified as consonant *"la"* as illustrated in figure 6.18c.

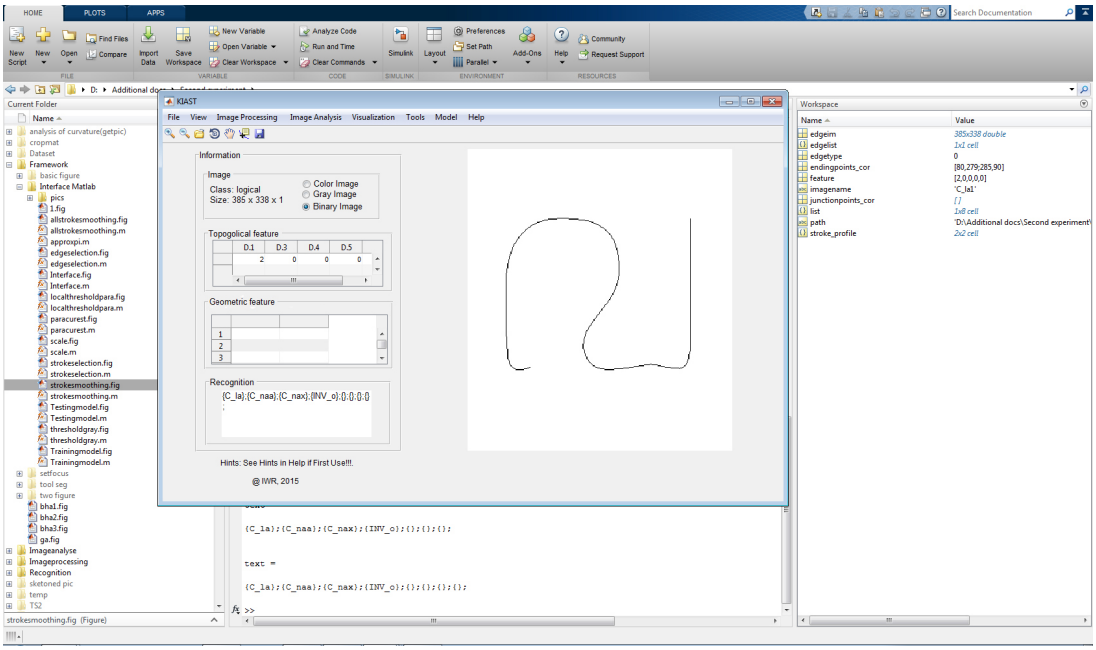


Figure 6.16: KIAST environment to show the information about the processes and variables.

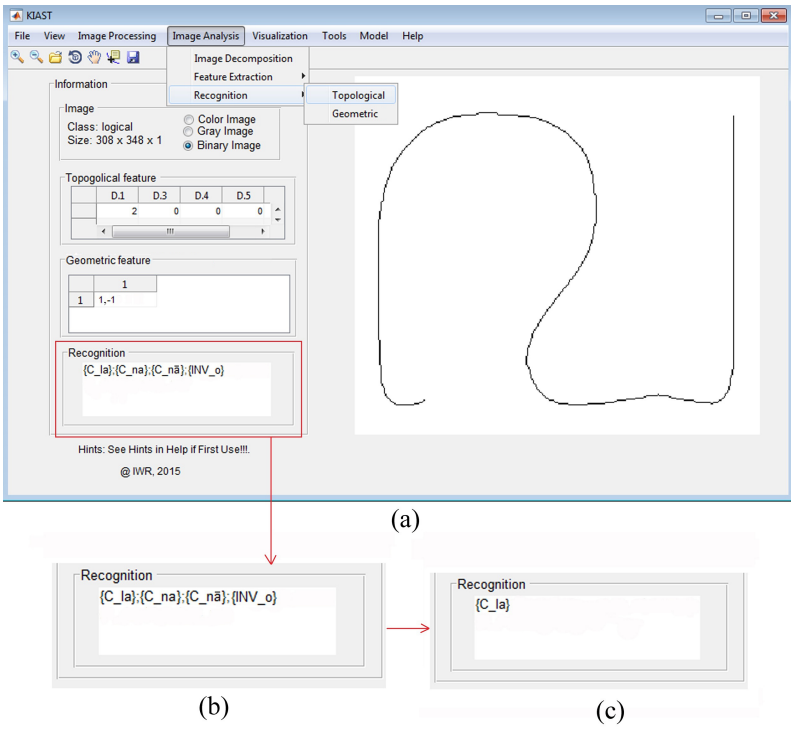


Figure 6.17: The character recognition processes in KIAST interface.

6.4 Visualization

In KIAST, we are able to visualize the analysis of the characters structures such as endpoint and junction, curved edge segments and so on. The tool bar of *Visualization* in KIAST is depicted in figure 6.19 and the visualization of structure analysis of consonant *"la"* such as the endpoints and curvature distribution of the curve segment are displayed in figure 6.20.

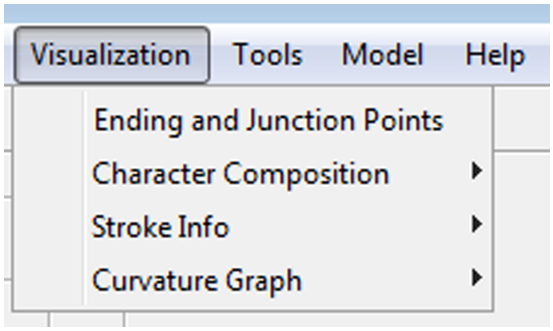
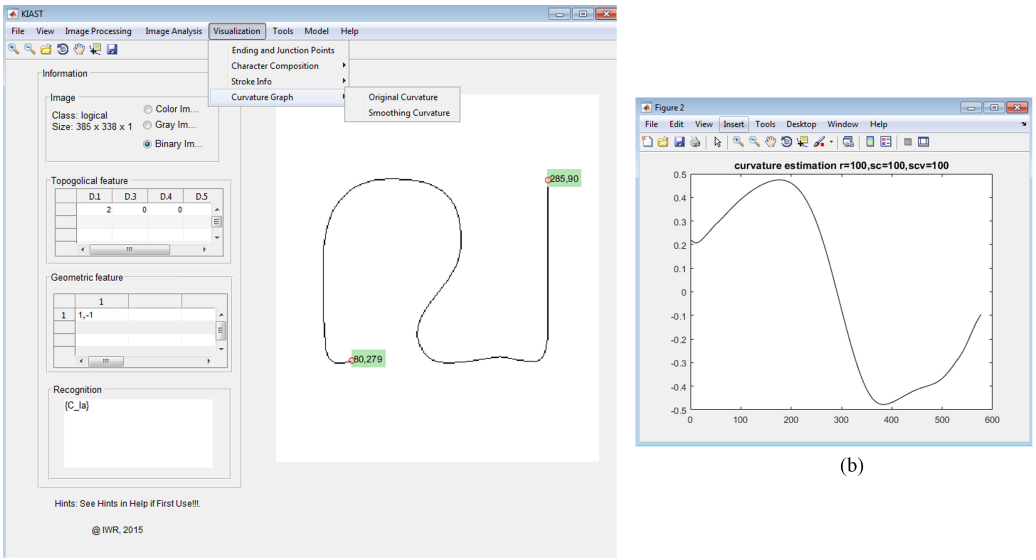


Figure 6.18: The tool bar of the visualization in KIAST interface.



(a) (b)

Figure 6.19: The visualization of structure analysis of consonant *"la"* such as (a) the endpoints and (b) the curvature of the character edge.

We also provide image enhancement tools such as image crop, image scale, eraser/paint brush, etc. in KIAST to improve geometrical structure of characters.

6.5 Summary

In chapter 6, we described the developed software– KIAST– to analyze structure of ancient Khmer characters and identify the characters. KIAST is developed based on the mathematical methods described in chapter 3 and 4. We also included some image processing tools in KIAST. However, KIAST which is attached in this thesis is in beta version. We developed it for experiment purposes of this research. The software tool will be expanded in a further project for general use. The future work of KIAST will be described in chapter 7.

Chapter 7

Conclusion and Outlook

Thesis Summary and Conclusion

In our PhD research project, we developed mathematical and computational methods to process and analyze Khmer inscriptions in pre-Angkorian period. Based on mathematical methods and algorithms presented in the thesis, we have developed the software framework– the so-called KIAST– to automatically process and analyze characters of pre-Angkorian inscriptions.

Instead of acquiring a digital version of inscriptions by using stone-rubbing or 2D camera, we presented the technique of 3D scanner to acquire original 3D inscriptions and we applied a curvature-based method to transfer 3D inscription data to 2D representations. The 3D scanning approach can also be used for digital preservation of the original Khmer inscriptions because this approach can preserve geometric properties of the inscriptions much more than stone-rubbing and taking a photograph.

Furthermore, we suggested mathematical methods based on topology and integral geometry to identify ancient Khmer characters of pre-Angkorian inscriptions. Analyzing the structures of the characters, we proved that the ancient Khmer characters can be discriminated by their topological features which are defined by total number of endpoints and junction points of various degrees and by geometric features which are defined by the curvature distribution of the character edges. We also presented an algorithm to decompose the complex structures of the combinations of consonants, sub-consonants and dependent vowels which can be treated with the methods developed for consonants and vowels. Finally, we developed the software– KIAST– to implement the mathematical methods presented in this thesis.

As a result, based on topological features, the characters can be classified into 15 classes. Among 15 classes, 4 classes are unique, that is, there exists only one character in the class. Based on geometric features, we can differentiate the characters which have the same topologi-

cal structures.

In summary, we can present in this thesis an efficient solution to the problem of processing and analyzing Khmer inscriptions, dating back in particular to the pre-Angkorian period.

Outlook

Although a successful approach and tools to digitally process and analyze Khmer inscriptions were developed in this thesis, much work still remains to be done in the future. The following topics are suggested for further research.

- Eroded surfaces of the inscriptions lead to damaged and incomplete geometric shape of script data. They also produce noisy data. Therefore, we suggest to develop mathematical and computational methods to distinguish the script and the eroded parts (noisy data) to restore the geometric shapes of script data to be readable again, at least for experts.

- Our approach and methods can be extended to analyze and characterize the script of Angkorian and post-Angkorian inscriptions.

- So far the software KIAST was mainly directed to the ”experimental” investigations of the PhD thesis. It has to be further developed into a professional tool. The software is expected to have the functionality of converting the results of character identification into an editable file format. In this thesis, we have developed a vector graphics font of ancient Khmer characters in pre-Angkorian period. This font can be used to support the functionality of converting the results of character identification into editable file format. In a further step, KIAST should be transfered to other platforms such as Qt and Eclipse using C++ and Java.

- Text processing of Khmer inscriptions is a crucial demand to supplement vanished characters caused by erosions or destruction. Educated guesses of the missing parts can be based on their neighboring characters. The guesses also need the reference to a dictionary explaining the meanings of ancient Khmer words.

Appendix A

Standard Ancient Khmer Characters

Ancient Khmer Consonants

Translit-eration	Pre-Ang-korian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern 1	Modern 2
ka	ᩀ ᩁ	ᨠ	ᨠ	ᨡ	ᨢ	ᨣ	ក	ក
kha	ᨤ	ᨥ	ᨤ	ᨦ	ᨥ	ᨥ	ខ	ខ
ga	ᨧ	ᨨ	ᨧ	ᨩ	ᨨ	ᨧ	គ	គ
gha	ᨫ	ᨫ	ᨫ	ᨫ	ᨫ	ᨫ	ឃ	ឃ
na	ᨬ	ᨭ	ᨬ	ᨭ	ᨬ	ᨬ	ង	ង
ca	ᨯ	ᨮ	ᨰ	ᨱ	ᨯ	ᨮ	ច	ច
cha	ᨰ	ᨰ	ᨰ	ᨱ	ᨯ	ᨰ	ឆ	ឆ
ja	ᨳ	ᨲ	ᨳ	ᨴ	ᨳ	ᨲ	ជ	ជ
jha	ᨵ	ᨶ	ᨶ	ᨷ	ᨶ	ᨶ	ឈ	ឈ
na	ᨹ	ᨺ ᨻ	ᨹ	ᨽ	ᨻ	ᨹ	ញ	ញ
ta	ᨿ	ᨾ	ᨾ	ᨸ	ᨾ	ᨿ	ដ	ដ

ANCIENT KHMER CONSONANTS

Translit-eration	Pre-Angkorian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern 1	Modern 2
tha	[illegible]						ឋ	ឋ
da		[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ឌ	ឌ
dha	[illegible]	[illegible]	[illegible]	[illegible]			ឍ	ឍ
na	[illegible] [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ណ	ណ
ta	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ត	ត
tha	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ថ	ថ
da	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ទ	ទ
dha	[illegible] [illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ធ	ធ
na	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ន	ន
pa	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ប	ប
pha	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ផ	ផ
ba	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ព	ព
bha	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	ភ	ភ

Ancient Khmer consonants

Translit-eration	Pre-Angkorian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern 1	Modern 2
ma	ᨾ	ᨿ	ᩁ	ᩃ	ᩉ	ᩋ	ម	ម
ya	ᨿ	ᩀ	ᩂ	ᩄ	ᩆ	ᩈ	យ	យ
ra	ᩁ	ᩅ ᩁ	ᩃ ᩅ	ᩃ	ᩁ ᩅ	ᩁ	រ	រ
la	ᩃ ᩃ	ᩃ	ᩃ	ᩃ	ᩃ	ᩃ	ល	ល
va	ᩅ	ᩅ	ᩅ	ᩅ	ᩅ	ᩅ	វ	វ
ßa/©a	ᩑ	ᩑ	ᩑ	ᩑ	ᩑ	ᩑ	ឝ	ឝ
sa	ᩔ	ᩔ	ᩔ	ᩔ	ᩔ	ᩔ	ឞ	ឞ
sa	ᩈ	ᩈ	ᩈ	ᩈ	ᩈ	ᩈ	ស	ស
ha	ᩉ	ᩉ	ᩉ	ᩉ	ᩉ	ᩉ	ហ	ហ
la						ᩊ	ឡ	ឡ
a	ᩋᩋᩋᩋ	ᩋ	ᩋ	ᩋ	ᩋ	ᩋ	អ	អ

Ancient Khmer Sub-Consonants

Translit-eration	Pre-Angkorian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern 1	Modern 2
ka								
kha								
ga								
gha								
na								
ca								
cha								
ja								
jha								
ña								
ta								

Ancient Khmer Sub-Consonants

Traslit-eration	Pre-Angkorian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern 1	Modern 2
tha								
dha								
dha								
na								
ta								
tha								
da								
dha								
na								
pa								
pha								
ba								
bha								
ma								

Ancient Khmer Sub-Consonants

Translit-eration	Pre-Angkorian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern 1	Modern 2
ya	្យ	្យ	្យ	្យ	្យ	្យ	◌្យ	◌ៈ
ra	◌្រ	◌្រ	◌្រ	◌្រ	◌្រ	◌្រ	◌្រ	◌ៗ
la	◌្ល	◌្ល	◌្ល	◌្ល	◌្ល	◌្ល	◌្ល	◌ឬ
va	◌្ឝ	◌្ឝ	◌្ឝ	◌្ឝ	◌្ឝ	◌្ឝ	◌្ឝ	◌ឭ
ßa/ ©a	◌្ឞ	◌្ឞ	◌្ឞ	◌្ឞ	◌្ឞ	◌្ឞ	◌្ឞ	◌ឮ
śa/ sha	◌្ឝ	◌្ឝ	◌្ឝ	◌្ឝ	◌្ឝ	◌្ឝ	◌្ឝ	◌ឯ
sa	◌្ស◌្ស	◌្ស	◌្ស	◌្ស	◌្ស	◌្ស	◌្ស	◌ឰ
ha	◌្ហ◌្ម	◌្ហ◌្ម	◌្ហ◌្ម	◌្ហ	◌្ហ	◌្ហ	◌្ហ	◌ឱ
a	◌្អ	◌្អ	◌្អ	◌្អ	◌្អ	◌្អ	◌្អ	◌ឪ

Ancient Khmer Dependent Vowels

Translit-eration	Pre-Angkorian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern
a							◌
i							◌ិ
í							◌ី
u							◌ុ
◌a--u							◌ូ
◌a-uo							◌ួ
ie							◌ឿ
e							◌េ
ae							◌ែ
ai							◌ៃ
o							◌ោ
au							◌ៅ
(a)m							◌ំ
(a)h							◌ះ
Viram							

Ancient Khmer Dependent Vowels

Translit-eration	Pre-Angkorian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern
a							◌
i							◌ិ
í							◌ី
u							◌ុ
va--u							◌ូ
va-uo							◌ួ
ie							◌ើ
e							◌េ
ae							◌ែ
ai							◌ៃ
o							◌ោ
au							◌ៅ
(a)m							◌ំ
(a)h							◌ះ
Viram							

Ancient Khmer Numbers

Translit-eration	Pre-Angkorian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern
0	ꞏ	ꞏ	ꞏ	ꞏ	ꞏ	๐	0
1	ᛁ	ᛁ ๑	ᛁ ๑	ᛁ ๓	๑ ๑	๑	๑
2	ᛁᛁ	ᛁᛁ ๒	ᛁᛁ ๒	ᛁᛁ ๒	๑๑ ๓	๗	๒
3	ᛁᛁᛁ	ᛁᛁᛁ ๒	ᛁᛁᛁ ๒	ᛁᛁᛁ ๒	๑๑๑ ๓	๗	๓
4	ᛁᛁᛁᛁ ๔	ᛁᛁᛁᛁ ๒	ᛁᛁᛁᛁ ๒	ᛁᛁᛁᛁ ๒	๑๑๑๑ ๔	๔	๔
5	ᛁᛁᛁᛁᛁ ๕	ᛁᛁᛁᛁᛁ ๖	ᛁᛁᛁᛁᛁ ๖	ᛁᛁᛁᛁᛁ ๖	๑๑๑๑๑ ๕	๕	๕
6	ᛁᛁᛁᛁᛁᛁ ๖ ๑	๗	๖	๖	๖	๖	๖
7	ᛁᛁᛁᛁᛁᛁᛁ ๗ ๗	๗	๗	๗	๗	๗	๗
8	ᛁᛁᛁᛁᛁᛁᛁᛁ ๘	๘	๘	๘	๘	๘	๘
9	ᛁᛁᛁᛁᛁᛁᛁᛁᛁ ๙	๙	๙	๙	๙	๙	๙
10	ᛁᛁᛁᛁᛁᛁᛁᛁᛁᛁ ๙	๑ ๐.	๑ ๐.	๑ ๐.	๑๐ ๐.	๑๐	๑๐
11	๙ᛁ	๑ᛁ ๑๑	๑ᛁ ๐๐	๑ᛁ ๐๐	๑๐ᛁ ๑๑	๑๑	๑๑
12	๙ᛁᛁ	๑ᛁᛁ ๑๒	๑ᛁᛁ ๐๒	๑ᛁᛁ ๐๒	๑๐ᛁᛁ ๑๓	๑๗	๑๒
13	๙ᛁᛁᛁ	๑ᛁᛁᛁ๐๒	๑ᛁᛁᛁ ๐๒	๑ᛁᛁᛁ ๐๒	๑๐ᛁᛁᛁ ๑๓	๑๗	๑๓
14	๙ᛁᛁᛁᛁ	๑ᛁᛁᛁᛁ๐๒	๑ᛁᛁᛁᛁ ๐๔	๑ᛁᛁᛁᛁ ๐๒	๑๐ᛁᛁᛁᛁ ๑๔	๑๖	๑๔

Ancient Khmer Numbers

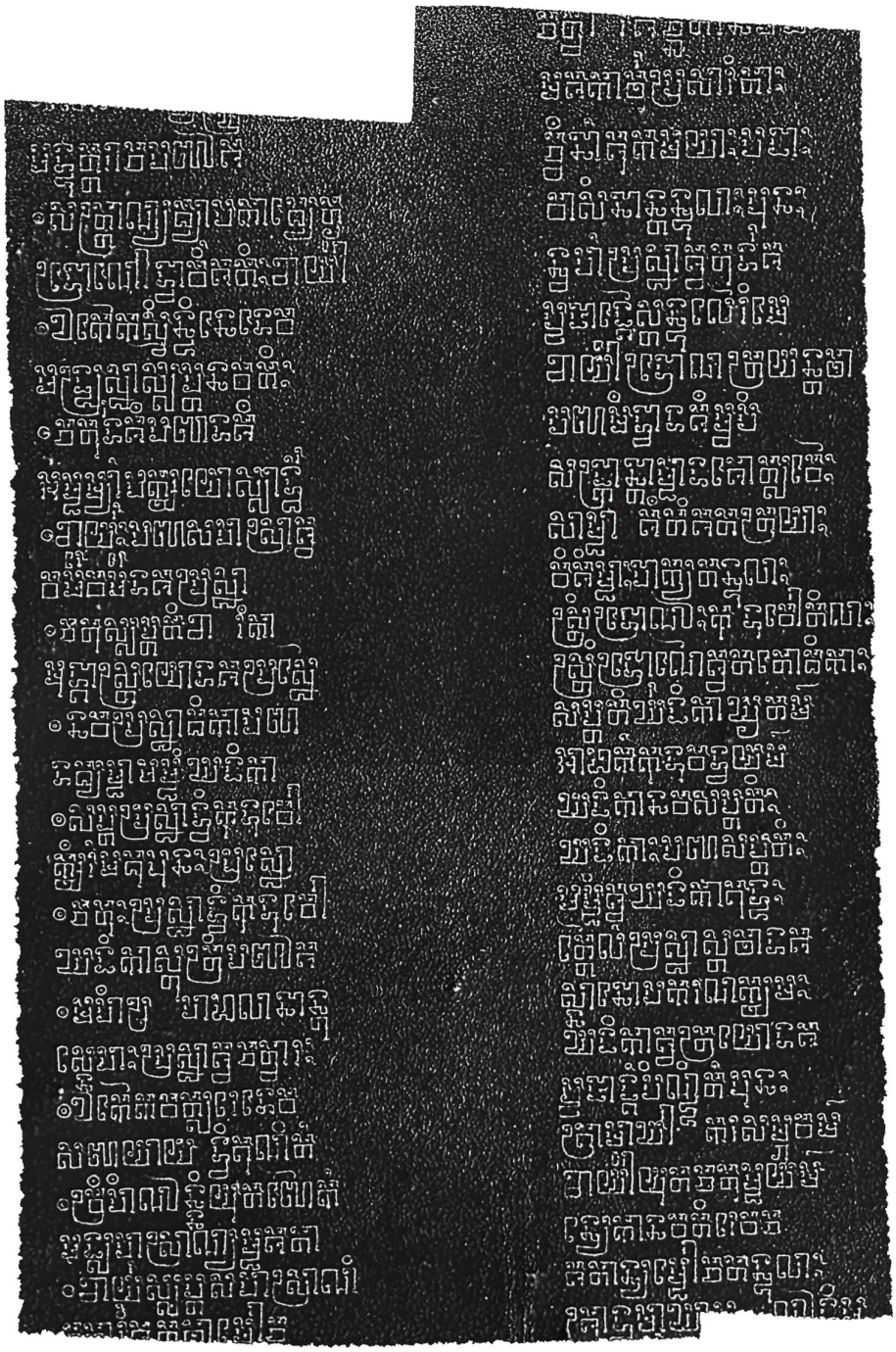
Translit-eration	Pre-Angkorian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern
15	ᩔᩍ	ᩋᩖ ᩑᩖ	ᩋᩖ ᩒᩖ	ᩊᩖ ᩓᩖ	ᩑᩖ ᩑᩖ	ᩑᩖ	๑๕
16	ᩔᩋ	ᩋᩗ ᩑᩗ	ᩋᩗ ᩒᩗ	ᩊᩗ ᩓᩗ	ᩑᩗ ᩑᩗ	ᩑᩗ	๑๖
17	ᩔᩀ ᩔᩐ	ᩋᩘ ᩑᩘ	ᩋᩘ ᩒᩘ	ᩊᩘ ᩓᩘ	ᩑᩘ ᩑᩘ	ᩑᩗ	๑๗
18	ᩔᩁ	ᩋᩙ ᩑᩙ	ᩋᩙ ᩒᩙ	ᩊᩙ ᩓᩙ	ᩑᩙ ᩑᩙ	ᩑᩘ	๑๘
19	ᩔᩃ	ᩋᩃ ᩑᩃ	ᩋᩛ ᩒᩛ	ᩊᩛ ᩓᩛ	ᩑᩃ ᩑᩃ	ᩑᩙ	๑๙
20	ᩑ	ᩋ ᩍ.	ᩋ ᩍ.	ᩊ ᩎ.	ᩋ ᩏ.	ᩒ๐	๒๐
21	ᩑ᩠	ᩋ᩠ ᩍᩑ	ᩋ᩠ ᩍᩒ	ᩊ᩠ ᩎᩒ	ᩋ᩠ ᩏᩑ	ᩒ᩠ᩑ	๒๑
28	ᩑᩁ	ᩋᩙ ᩍ.	ᩋᩙ ᩍ.	ᩊᩙ ᩎ.	ᩋᩙ ᩏ.	ᩒᩘ	๒๘
30	ᩑᩔ	ᩋᩔ ᩎ.	ᩋᩔ ᩎ.	ᩊᩔ ᩎ.	ᩋᩉ ᩐ.	ᩒ๐	๓๐
33	ᩑᩔᩍ	ᩋᩔᩍ ᩎᩎ	ᩋᩔᩍ ᩎᩎ	ᩊᩔᩍ ᩎᩎ	ᩋᩉᩍᩐᩐ	ᩒᩒ	๓๓
40	ᩑ ᩒ.	ᩋ᩵ ᩎ.	ᩋ᩵ ᩎ.	ᩊ᩵ ᩎ.	ᩋ᩵ ᩐ.	ᩓ๐	๔๐
42	ᩑᩍ	ᩋ᩵ᩍ ᩎᩍ	ᩋ᩵ᩍ ᩎᩍ	ᩊ᩵ᩍ ᩎᩎ	ᩋ᩵ᩍ ᩐᩏ	ᩓᩒ	๔๒
50	ᩑᩔ ᩘ.	ᩋ᩵ᩔ ᩑ.	ᩋ᩵ᩔ ᩒ.	ᩊ᩵ᩉ ᩓ.	ᩋ᩵ᩉ ᩔ.	ᩕ๐	๕๐
55	ᩑᩔᩘ	ᩋ᩵ᩔᩑ ᩑᩑ	ᩋ᩵ᩔᩒ ᩒᩒ	ᩊ᩵ᩉᩓ ᩓᩓ	ᩋ᩵ᩉᩔ ᩔᩔ	ᩕᩕ	๕๕
60	ᩑᩋ ᩊ.	ᩋ᩵ᩋ ᩏ.	ᩋ᩵ᩋ ᩐ.	ᩊ᩵ᩋ ᩑ.	ᩋ᩵ᩋ ᩑ.	ᩖ๐	๖๐
70	ᩑᩋᩔ ᩁ.	ᩋ᩵ᩋᩔ ᩘ.	ᩋ᩵ᩋᩔ ᩔ.	ᩊ᩵ᩋᩉ ᩘ.	ᩋ᩵ᩋᩉ ᩕ.	ᩗ๐	๗๐
80	ᩑᩋ ᩍ.	ᩋ᩵ᩋ ᩑ.	ᩋ᩵ᩋ ᩒ.	ᩊ᩵ᩋ ᩓ.	ᩋ᩵ᩋ ᩓ.	ᩘ๐	๘๐

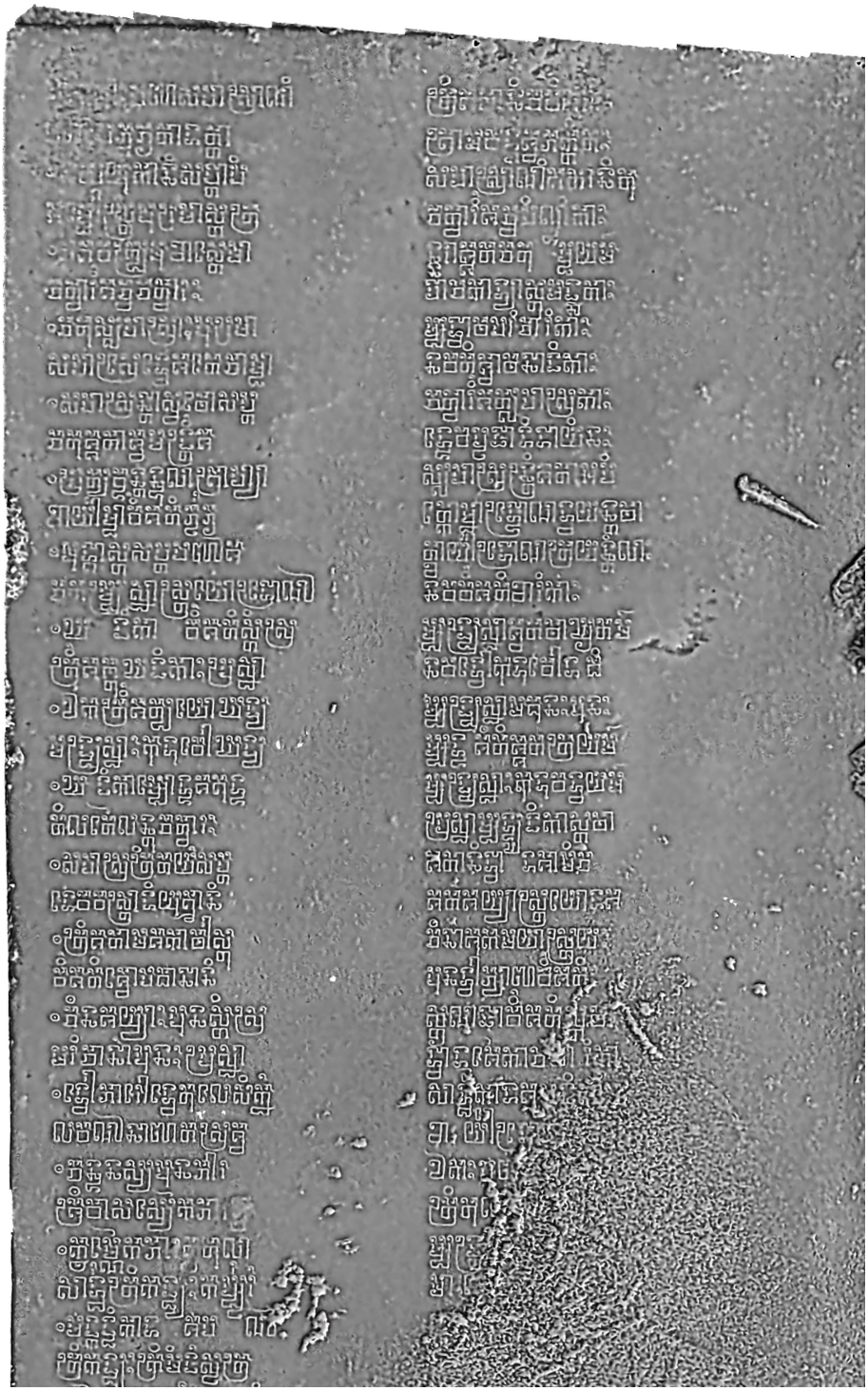
ANCIENT KHMER NUMBERS

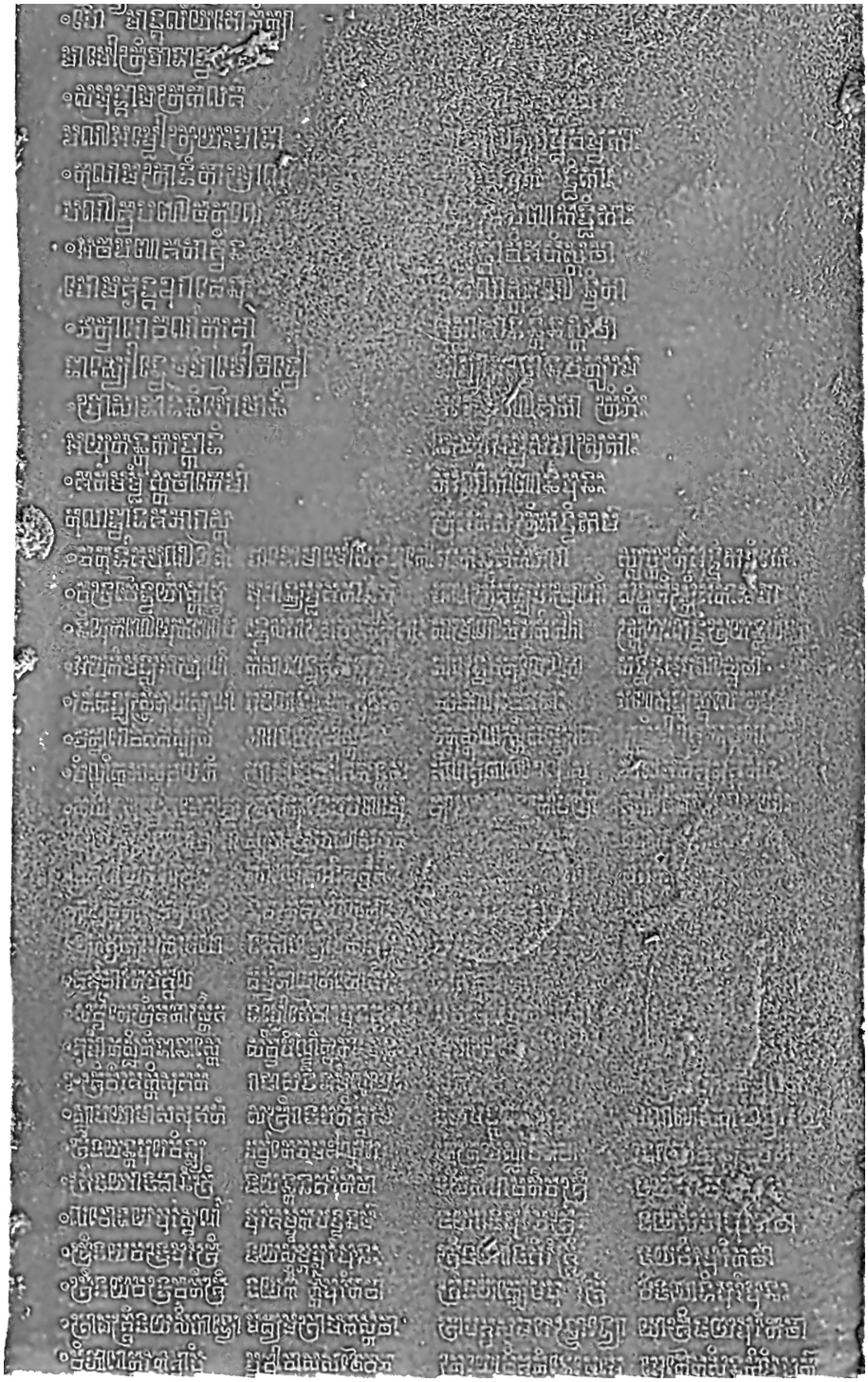
Translit-eration	Pre-Angkorian	Angkorian 1	Angkorian 2	Angkorian 3	Angkorian 4	Angkorian 5	Modern
90	ធ្នូខ	ធ្នូខ	ធ្នូខ	ធ្នូ+	ធ្នូx	៩០	៩០
100	ល	ល ១..	ល ១..	ល ១..	ល ១..	១០០	១០០
101	ល।	ល। ១.១	ល। ១.១	ល। ១.១	ល ១.១	១០១	១០១
110	លខ	លខ ១១.	លខ ១១.	ល+ ១១.	លx ១១.	១១០	១១០
124	លត្តខ	លត្តខ	លត្តខ	លត្តខ	លត្តខ	១២៤	១២៤
158	លធ្នូខT	លធ្នូខ។	លធ្នូខ។	លធ្នូ+។	លធ្នូx។	១៥៨	១៥៨
200	ល	ល	ល	ល	ល	២០០	២០០
300	ល	ល	ល	ល	ល	៣០០	៣០០
400	ស្លឹក। ៣៩	ស្លឹក।	ស្លឹក।	ស្លឹក।	ស្លឹក।	ស្លឹក១	១ស្លឹក =៤០០
452	ស្លឹក।ធ្នូខ॥	ស្លឹក।ធ្នូខ॥	ស្លឹក।ធ្នូខ॥	ស្លឹក।ធ្នូ+॥	ស្លឹក।ធ្នូx॥	៤៥២	១ស្លឹក១ផ្លូន ៣ដំបរ =៤៥២
800	ស្លឹក॥	ស្លឹក॥	ស្លឹក॥	ស្លឹក॥	ស្លឹក॥	ស្លឹក២	២ស្លឹក =៨០០
1200	ស្លឹក॥।	ស្លឹក॥।	ស្លឹក॥।	ស្លឹក॥।	ស្លឹក॥।	ស្លឹក៣	៣ស្លឹក =១២០០
1000	សហស្រ	សហស្រ	សហស្រ	សហស្រ	សហស្រ	១ពាន	១០០០
4000	ក្លុញ	ខ្លួញ	ខ្លាញ	ខ្លាញ	ខ្លាញ	៤០០០	៤០០០

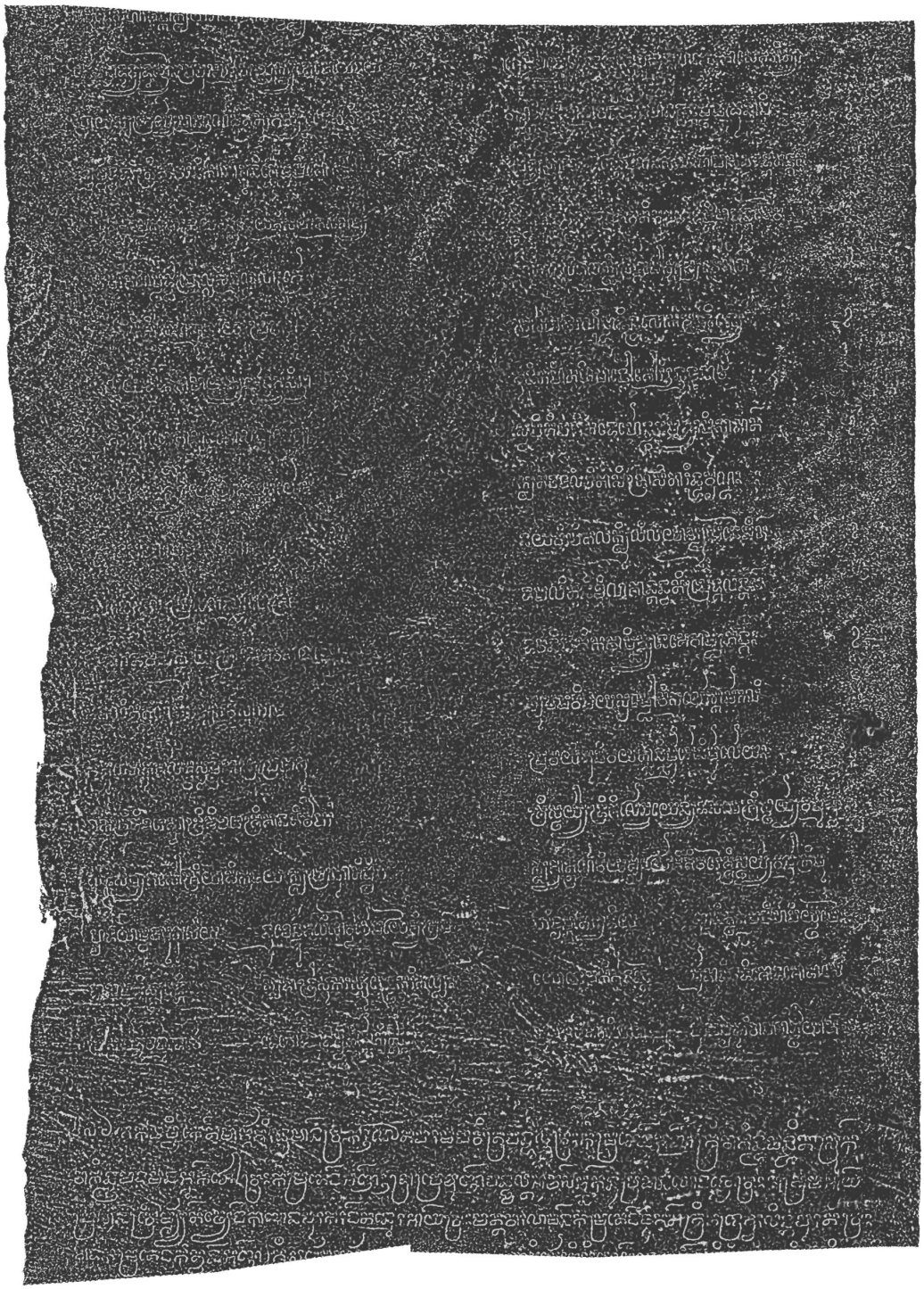
Appendix B

Further Results of Script Extraction





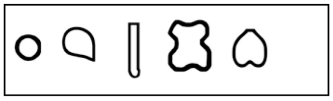




Appendix C

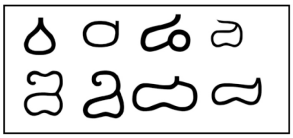
Characters in Topological Classes

n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
0	0	0	0



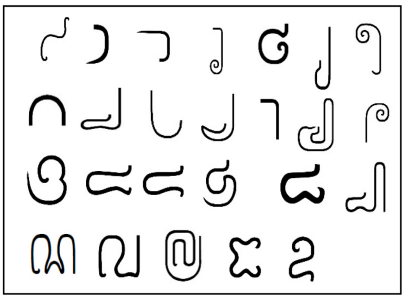
n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
0	2	0	0

n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
1	1	0	0



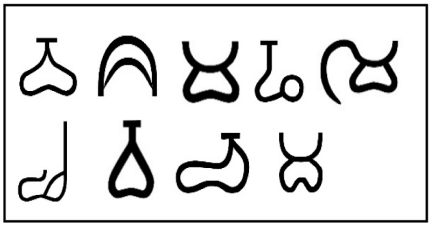
n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
1	1	1	0

n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
2	0	0	0

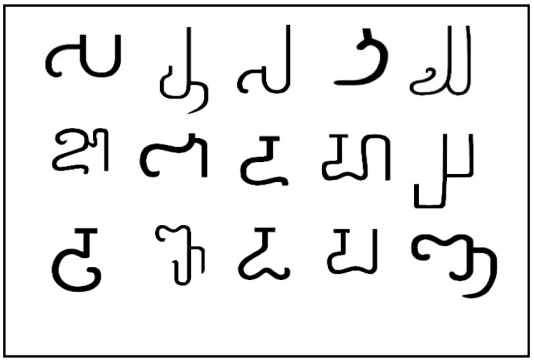


n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
2	0	1	0

n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
2	2	0	0



n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
3	1	0	0

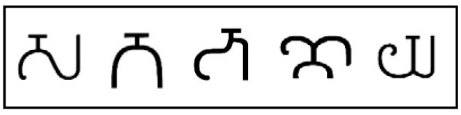


n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
3	1	1	0

n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
3	3	0	0

n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
4	0	1	0

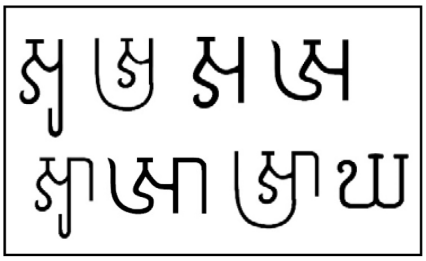
n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
4	2	0	0



n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
4	4	0	0

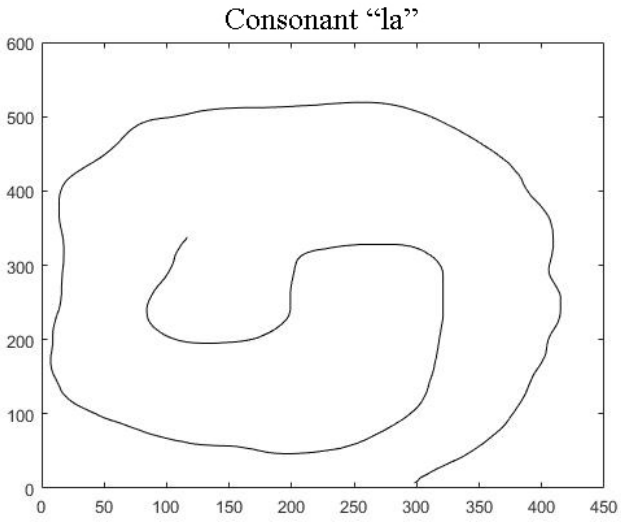
n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
5	1	1	0

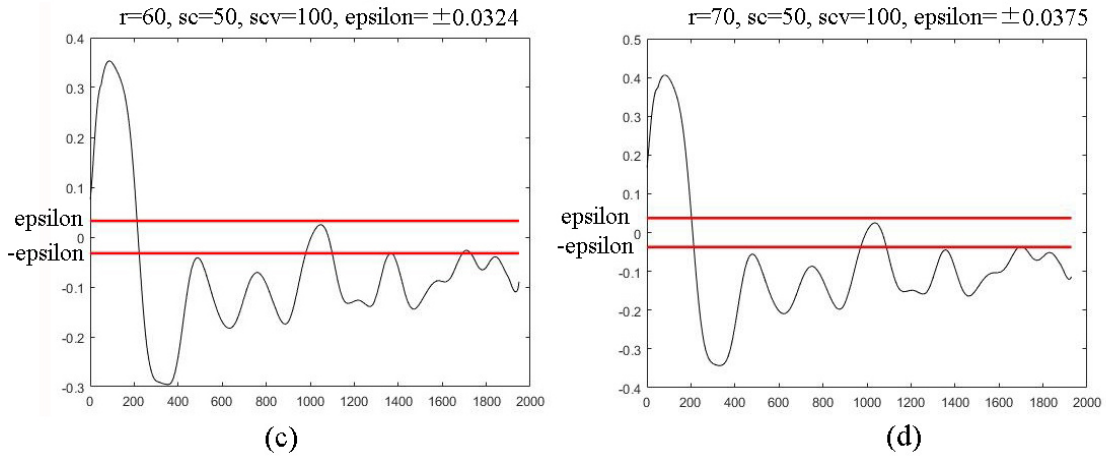
n_{d_1}	n_{d_3}	n_{d_4}	n_{d_5}
5	3	0	0



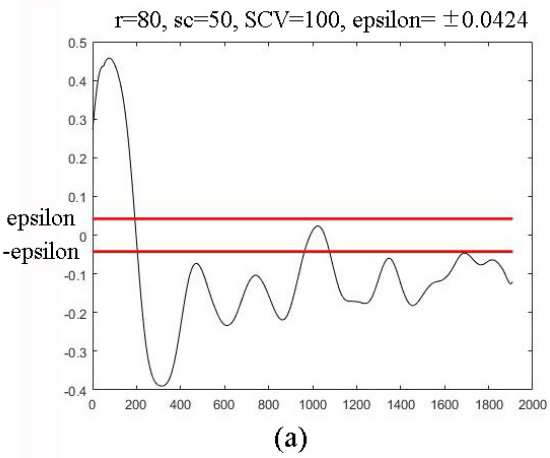
Appendix D

Curvature Computation

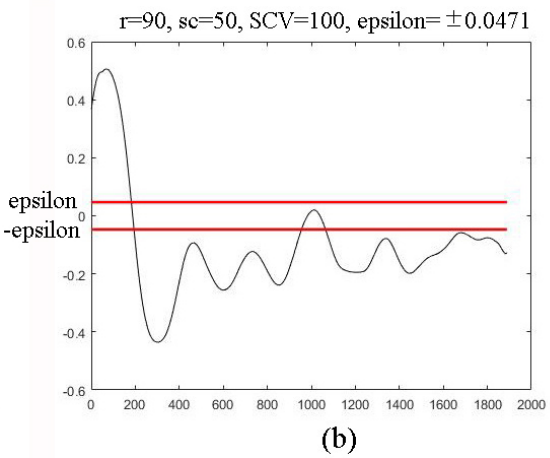




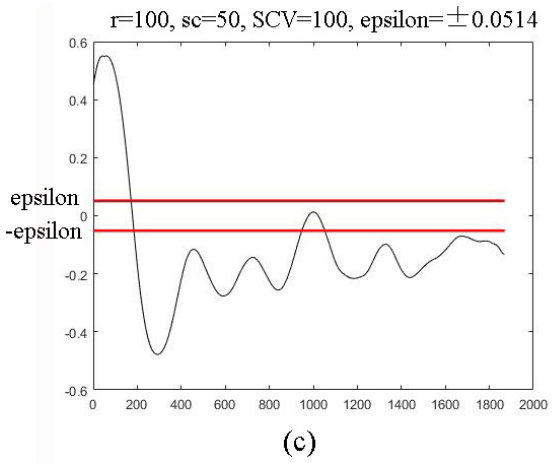
(c) (d)


r=80, sc=50, SCV=100, epsilon= ±0.0424
epsilon
-epsilon

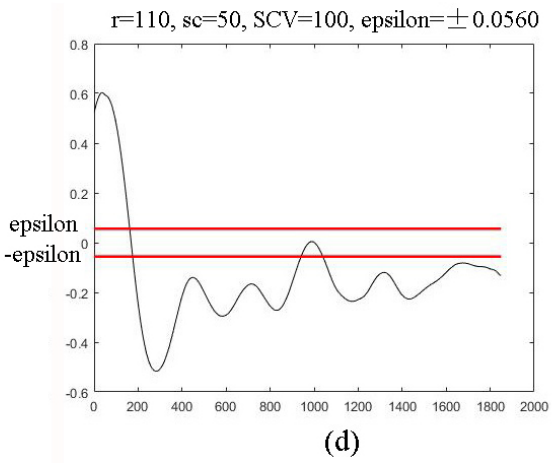
(a)


r=90, sc=50, SCV=100, epsilon= ±0.0471
epsilon
-epsilon

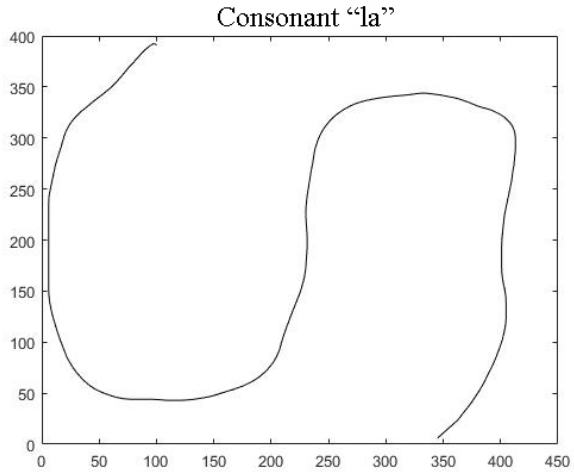
(b)

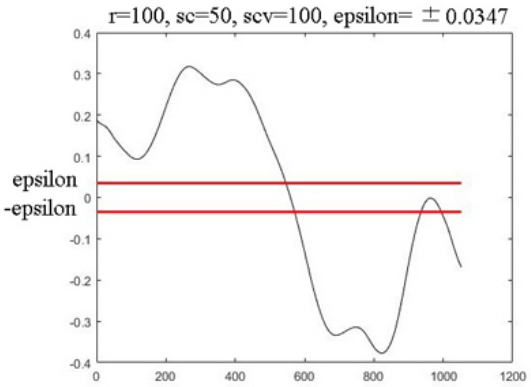

r=100, sc=50, SCV=100, epsilon=±0.0514
epsilon
-epsilon

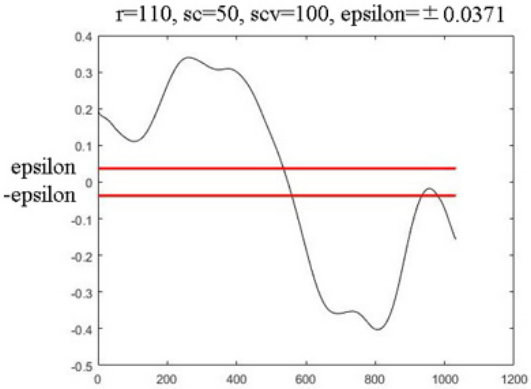
(c)

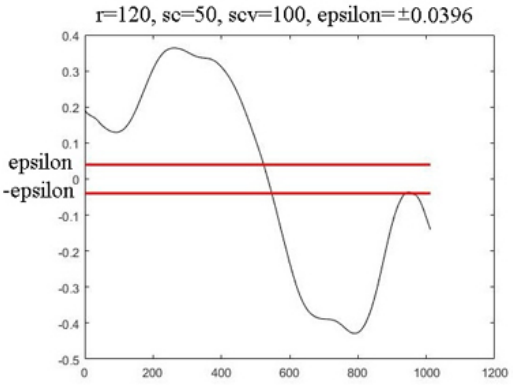

r=110, sc=50, SCV=100, epsilon=±0.0560
epsilon
-epsilon

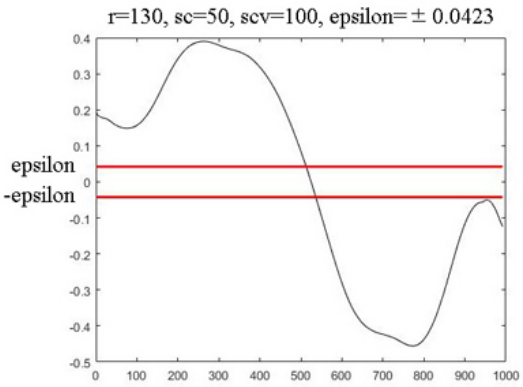
(d)

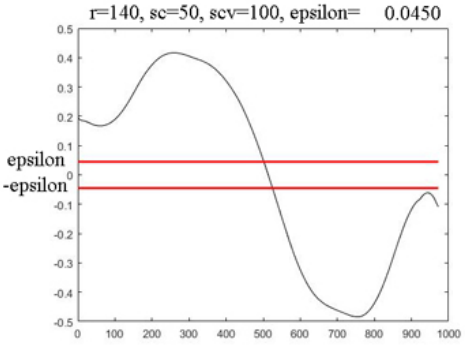

Consonant “la”

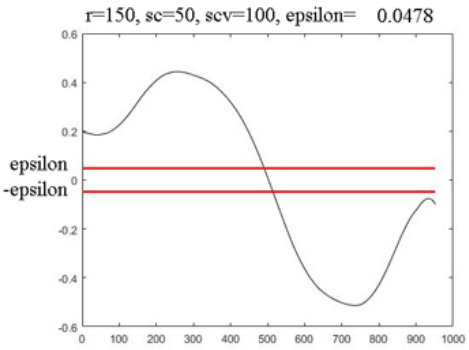

r=100, sc=50, scv=100, epsilon= ± 0.0347
epsilon
-epsilon

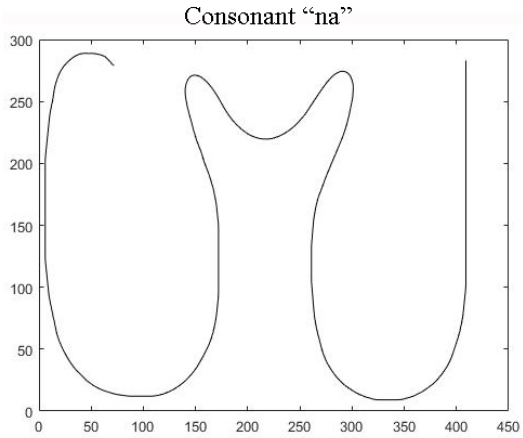

r=110, sc=50, scv=100, epsilon=± 0.0371
epsilon
-epsilon

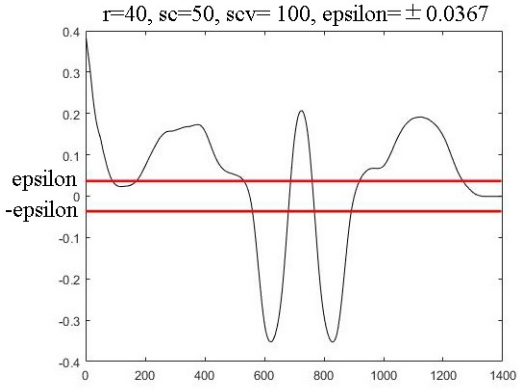

r=120, sc=50, scv=100, epsilon=±0.0396
epsilon
-epsilon

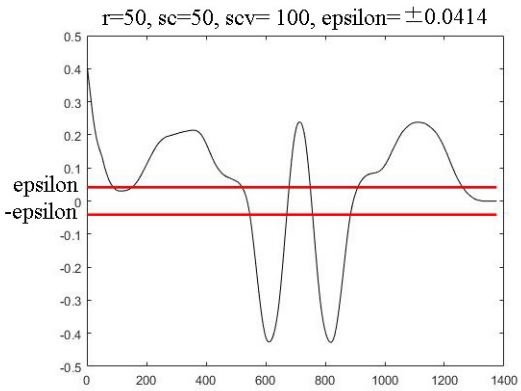

r=130, sc=50, scv=100, epsilon= ± 0.0423
epsilon
-epsilon

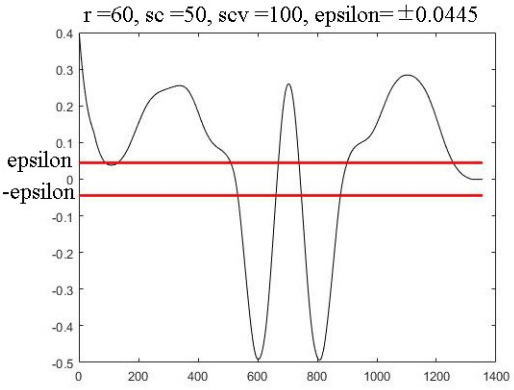

r=140, sc=50, scv=100, epsilon= 0.0450
epsilon
-epsilon

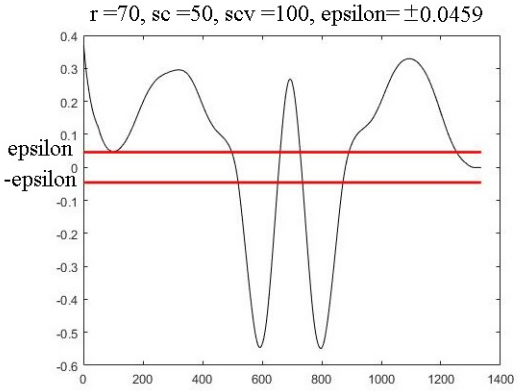

r=150, sc=50, scv=100, epsilon= 0.0478
epsilon
-epsilon

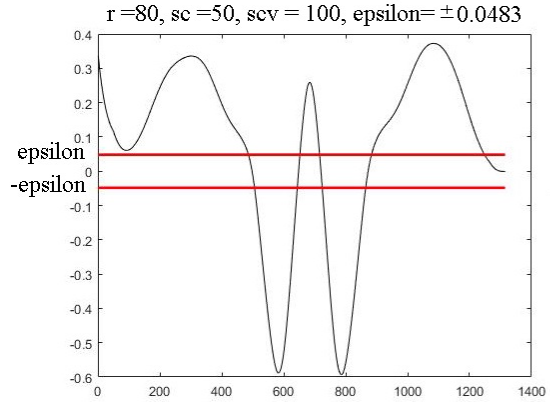

Consonant “na”

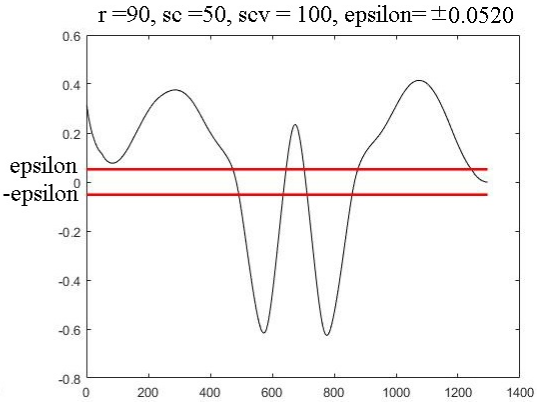

r=40, sc=50, scv= 100, epsilon=±0.0367
epsilon
-epsilon

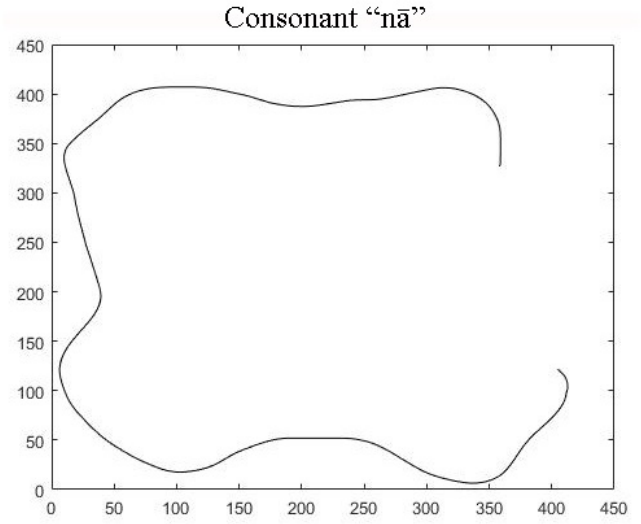

r=50, sc=50, scv= 100, epsilon=±0.0414
epsilon
-epsilon

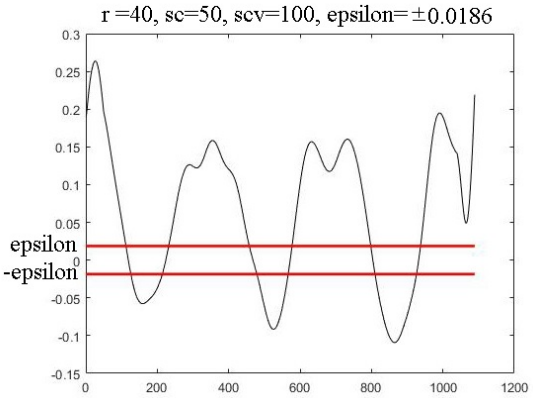

r =60, sc =50, scv =100, epsilon=±0.0445
epsilon
-epsilon

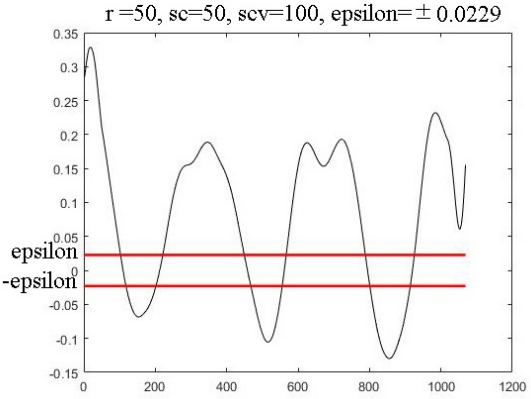

r =70, sc =50, scv =100, epsilon=±0.0459
epsilon
-epsilon

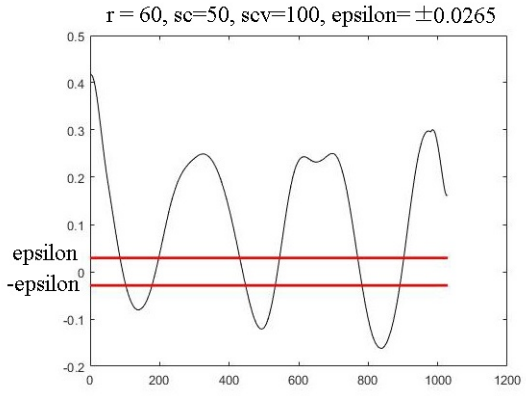

r =80, sc =50, scv = 100, epsilon=±0.0483
epsilon
-epsilon

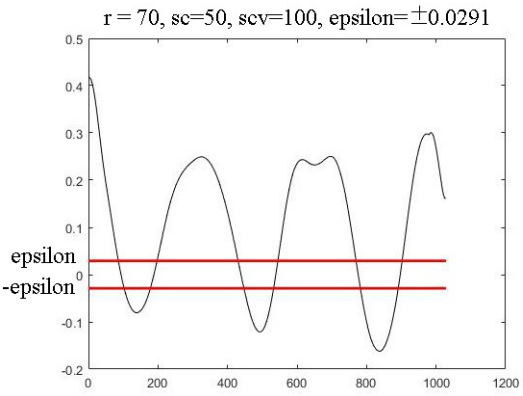

r =90, sc =50, scv = 100, epsilon=±0.0520
epsilon
-epsilon

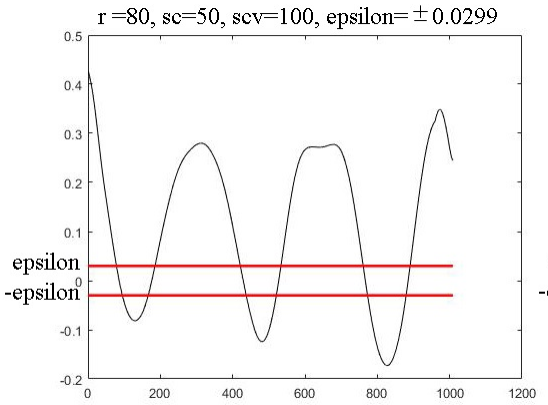

Consonant “nā”

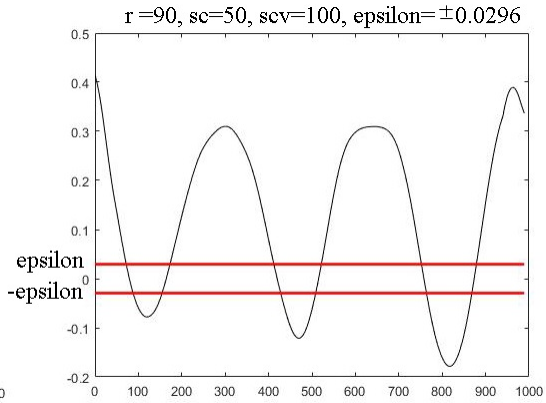

r =40, sc=50, scv=100, epsilon=±0.0186
epsilon
-epsilon

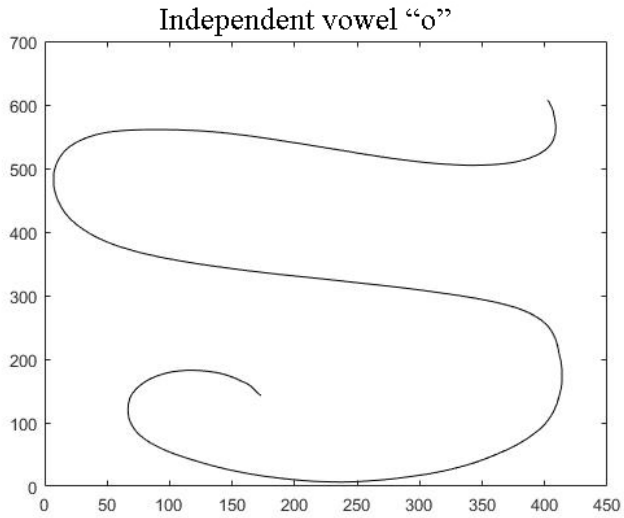

r =50, sc=50, scv=100, epsilon=± 0.0229
epsilon
-epsilon

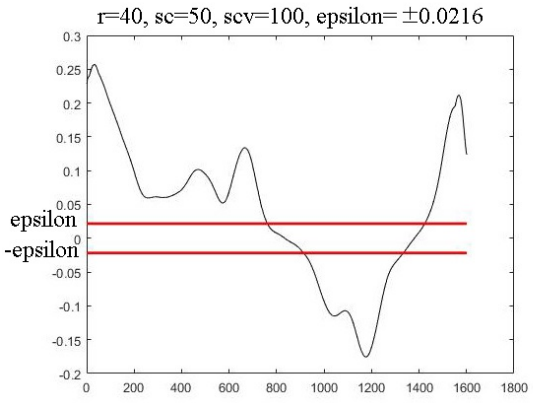

r = 60, sc=50, scv=100, epsilon=±0.0265
epsilon
-epsilon

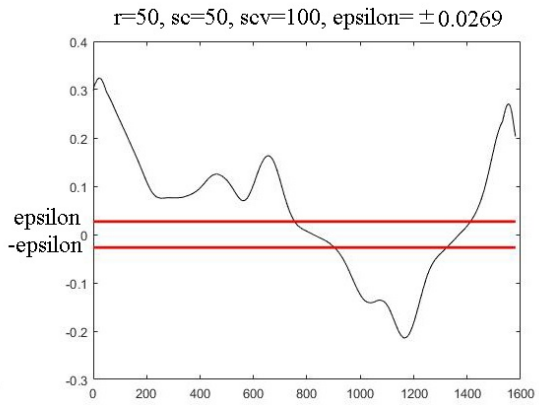

r = 70, sc=50, scv=100, epsilon=±0.0291
epsilon
-epsilon

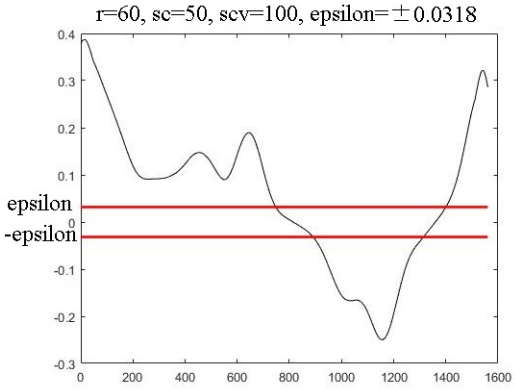

r =80, sc=50, scv=100, epsilon=±0.0299
epsilon
-epsilon

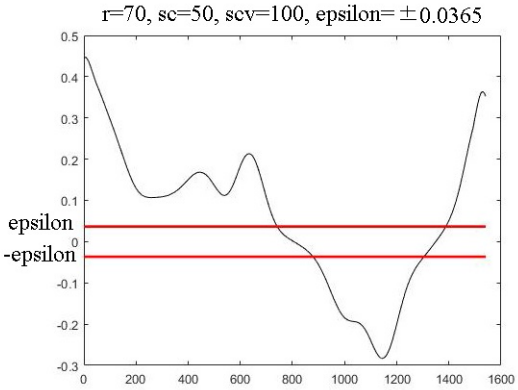

r =90, sc=50, scv=100, epsilon=±0.0296
epsilon
-epsilon

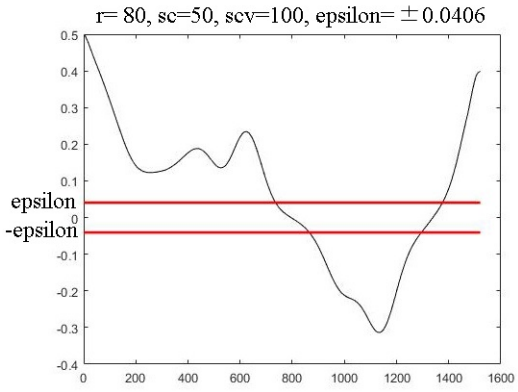

Independent vowel “o”

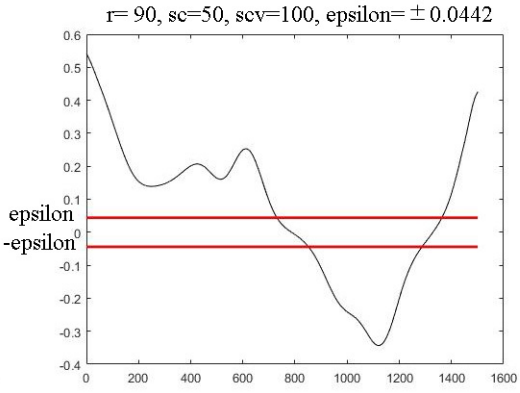

r=40, sc=50, scv=100, epsilon=±0.0216
epsilon
-epsilon


r=50, sc=50, scv=100, epsilon=±0.0269
epsilon
-epsilon


r=60, sc=50, scv=100, epsilon=±0.0318
epsilon
-epsilon


r=70, sc=50, scv=100, epsilon=±0.0365
epsilon
-epsilon


r= 80, sc=50, scv=100, epsilon=±0.0406
epsilon
-epsilon


r= 90, sc=50, scv=100, epsilon=±0.0442
epsilon
-epsilon

Bibliography

[1] 3d-scannen von inschriften mittelalterlicher grabsteine jüdischer friedhof "heiliger sand", worms. URL http://www.iwr.uni-heidelberg.de/groups/ngg/Worms/.

[2] 3d-reconstruction of the dancing shiva of the prasat thom in koh ker. URL http://scotty.iwr.uni-heidelberg.de/3Dpuzzle.shtml.

[3] L. W. Beineke, R. J. Wilson, and J. L. Gross. *Topics in Topological Graph Theory*. Cambridge University Press, 2009.

[4] S. W. Chen, S. T. Tung, and C. Y. Fang. Extended attributed string matching for shape recognition. *Computer Vision and Image Understanding*, 70(1), April 1998.

[5] M. Cheriet, N. Kharma, C. L. Liu, and C. Y. Suen. *Character Recognition Systems. A Guide for students and practitioners*. John Wiley & Sons, Inc, 2007.

[6] C. Chey, P. Kumhom, and K. Chamnongthai. Khmer pritned character recognition using wavelet descriptors. *International Journal of Uncertainty*, 14:337–350, 2006.

[7] H. Chhaily. *Inscriptions du Cambodge. Anthologie d'epigraphies ancienne du Cambodge*, volume 1, 2. 2011.

[8] G. Coedès. *Inscriptions du Cambodge*, volume 1-8. EFEO, 1937, 1942, 1951, 1952, 1953, 1954, 1964, 1966.

[9] R. M. O. Cruz, G. D. C. Cavalcanti, and T. I. Ren. An ensemble classifier for offline cursive character recognition using multiple feature extraction techniques. *IEEE*, 2010.

[10] R. L. Das, B. K. Prasad, and G. Sanyal. Hmm based offline handwritten writer independent english character recognition using global and local feature extraction. *International Journal of Computer Applications*, 46(10):45–50, May 2012.

[11] M. Dehmer and F. E. Streib. *Quantitative Graph Theory*. Taylor & Francis Group, LLC, 2015.

[12] D. Ferguson, M. Likhachev, and A. Stentz. A guide to heuristic-based path planning. *American Association for Artificial intelligence*, 2005. URL www.aaai.org.

[13] L. Finot. *Inscriptions du Cambodge*, volume 1, 2, 3, 4, 5, 6. EFEO, 1926, 1927,1928,1931,1937.

[14] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Acamedic Press, 1990.

[15] V. Ganapathy and K. L. Liew. Handwritten character recognition using multiscale neural network training technique. *World Academy of Science, Engineering and Technology*, 2008.

[16] B. Gatos, K. Ntzios, I. Pratikakis, S. Petridis, T. Konidaris, and S. J. Perantonis. An efficient segmentation-free approach to assist old greek handwritten manuscript ocr. *Pattern Analysis Application*, 2006.

[17] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Prentice Hall, second eidtion edition, 2002.

[18] R. C. Gonzalez, R. E. Woods, and S. L. Eddins. *Digital Image Processing Using Matlab*. Gatesmark Publishing, second edition edition, 2009.

[19] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber. A novel connectionist system for unconstrained handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31(5), May 2009.

[20] J. Gross, J. Yellen, and P. Zhang. *HandBook of Graph Theory*. Chapman and Hall/CRC, December 2013.

[21] J. Han, M. Kamber, and J. Pei. *Data Mining Concepts and Techniques*. Elsevier, third edition edition, 2012.

[22] P. Harrington. *Machine Learning in Action*. Manning Publication Co., 2012.

[23] P. E. Hart, N. J. Nilsson, and B. Raphael. A formal basis for the heuristic determination of minimum cost paths. *IEEE Transactions of Systems Science and Cybernetics*, ssc-4(2), July 1968.

[24] C. J. Hilditch. Linear skeleton from square cupboards. *Machine Intelligence*, 4:403–420, 1969.

[25] J. M. JACOB. Pre-angkor cambodia: evidence from the inscriptions in khmer concerning the common people and their environment. In D. A. Smyth, editor, *Cambodian Linguistics, Literature and History*, pages 299–318. School of Oriental and African Studies, University of London, London, 1993. ISBN 0-7286-0218-0. URL `http://purl.org/sealang/jacob1993pre-angkor.pdf`.

[26] J. M. JACOB. The ecology of angkor: evidence from the khmer inscriptions. In D. A. Smyth, editor, *Cambodian Linguistics, Literature and History*, pages 280–298. School of Oriental and African Studies, University of London, London, 1993. ISBN 0-7286-0218-0. URL `http://purl.org/sealang/jacob1993ecology.pdf`.

[27] C. Jacques. Khmer epigraphy. *Museum International, Angkor, a living museum, UNESCO*, 54(1-2):37–43, May 2002.

[28] B. Jähne. *Digital Image Processing*. Springer, Interdisciplinary Center for Scientific Computing, University of Heidelberg, sixth revised and extened edition edition, 2005.

[29] T. Jipeng, G. Kumar, and H. Chethan. Skew correction for chinese character using hough transform. *International Journal of Computer Applications*, 22(2), May 2011.

[30] D. Kiem. *Evolution de L'ecriture Khmére*. Apsara Library, 2000.

[31] M. S. Kim, K. T. Cho, H. K. Kwag, and J. H. Kim. Segmentation of handwritten characters for digitdigital korean historical documents. *Springer- Verlag Berlin, Heidelberg*, 2004.

[32] A. Konno and Y. Hongo. Postprocessing algorithm based on the probabilistic and semantic method for japanese ocr. *IEEE*, 1993.

[33] P. Kovesi. URL `http://www.peterkovesi.com/matlabfns/#edgelink`.

[34] S. Krömker. *Gigamesh Manual*. Interdisciplinary Center for Scientific Computing, Visualization and Numerical Geometry Group, Heidelberg University, Germany, 2011 .

[35] C. Lee and B. Wu. A chinese character stroke extraction algorithm based on contour information. *Pattern Recognition*, 31(6):651–663, 1998.

[36] R. S. . T. Lee and J. N. K. Liu. An oscillatory elastic graph matching model for recognition of offline handwritten chinese characters. In *Third International Conference on Knowledge-Based Intelligent Information Engineeing Systems*, Australia, 1999.

[37] S. Lewitz(Pou). Textes en khmer moyen. inscriptions modernes d'angkor vat. *BEFEO*, LVII, 1970.

[38] S. Lewitz(Pou). *Nouvelles Inscriptions du Cambodge*, volume 1, 2, 3. EFEO, Paris, 1989,2001.

[39] S. Lewitz(Pou). *Dictionnaire vieux khmer-franais-anglais*. Center de Documentation et de Recherche sur la civilisation Khmere, Paris, 1992.

[40] S. Lewitz(Pou). *Nouvelles Inscriptions du Cambodge*, volume 4. L'Harmattan, 2011.

[41] C.-L. Liu, S. Jaeger, and M. Nakagawa. Online recognition of chinese characters: The state-of-the-art. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(2), February 2004.

[42] S. Long. *Toponymie Khmére*. Royal University of Phnom Penh, 1997.

[43] S. Long. *Dictionnaire du Khmer Ancien (D'apaés les inscriptions du Cambodge du VIe-VIIIe. siécles)*. Cambodge, 2000.

[44] R. Lowrance and R. A. Wagner. An extension of the string-to-string correction problem. *Journal of the Association for Computing Machinery*, 22(2):177– 183, April 1975.

[45] H. Mara. *Multi-Scale Integral Invariants for Robust Character Extraction from Irregular Polygon Mesh Data*. PhD thesis, Interdisciplinary Center for Scientific Computing, Heidelberg University, September 2012.

[46] H. Mara, H. Jan, and S. Kromker. Gpu based optical character transcription for ancient inscription recognition. In *Proceedings of the 15th International Conference on Virtual Systems and Multimedia (VSMM2009)*, Vienna, Austria, September 2009.

[47] A. Marzal and E. Vidal. Computation of normalized edit distance and application. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(9), 1993.

[48] I. Methasate, S. Marukatat, S. Saetang, and T. Theeramunkong. The feature combination technique for offline thai character recognition system. *IEEE, Eighth ICDAR*, 2005.

[49] S. Mori and H. Nishida. *Optical Character Recognition*. Wiley Interscience, New Jersey, 1999.

[50] S. Mori and C. Y. Suen. Historical review of ocr research and development. *Proc. IEEE*, 80(7):1029–1058, July 1992.

[51] M. Nagendraprasad, P. S. P. Wang, and A. Gupta. Aalgorithm for thinning and rethickening binary digital pattern. *Digital Signal Processing*, 3(2):97–102, April 1993.

[52] G. Nagy. At the frontiers of ocr. *Processing of the IEEE*, 80(7):1093–1100, July 1992.

[53] P. Nguonphan. 3d scanning of khmer inscriptions. 2011.

[54] N. J. Nilsson. *Principles of Artificial Intelligence*. Springer- Verlag, 1982.

[55] K. Ntzios, B. Gatos, I. Pratikakis, T. Konidaris, and S. J. Perantonis. An old greek handwritten ocr systems. In *Proceedings of the 2005 Eight International Conference on Document Analysis and Recognition (ICDAR05)*, 2005.

[56] N. Otsu. A threshold selection method from gray-level histograms. *IEEE Transactions on Systems, Man and Cybernetics*, 9(1):62–66, January 1979.

[57] J. N. Philip. A manual of pre-angkorian khmer, the dated inscriptions. . URL `http://sealang.net/classic/khmer/manual/`.

[58] J. N. Philip. Seaclassics old khmer images. . URL `http://sealang.net/oldkhmer/image.htm`.

[59] J. N. Philip. *A Chronoligical Inventory of the Inscriptions of Cambodia*. Center for Southeast Asian Studies, University of Hawaii at Manoa, 1982.

[60] H. Pottmann, J. Wallner, Y. L. Yang, Y. K. Lai, and S. M. Hu. Principal curvatures from integral invariant viewpoint. *Computer Aided Geometric Design*, 24(8-9):428–442, 2007.

[61] H. Pottmann, J. Wallner, Q. Huang, and Y. Yang. Integral invariant for robust geometry processing. *Computer Aided Geometric Design*, 1(16):37–60, 2009.

[62] U. Quatember, B. Thuswaldner, R. Kalasek, C. Bathow, and B. Breuckmann. The virtual and physical reconstruction of the octagon and hadrians temple in ephesus. *Proc. 2nd Workshop on Scientific Computing for Cultural Heritage (SCCH)*, pages 20–23, 2009.

[63] M. Revow, C. K. l. Williams, and G. E. Hinton. Using generative models for handwritten digit recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18 (6):592–606, June 1996.

[64] S. V. Rice, G. Nagy, and T. A. Nartker. *Optical Character Recognition: An Illustrated Guide to the Frontier*. Kluwer Academic Publisher, 1999.

[65] A. Rosenfeld. A characterization of parallel thinning algorithms. *Information and Control*, 29:286–291, 1975.

[66] E. M. Rostie, P. Natarajan, M. Decerbo, and R. Prasad. The bbn byblos japanese ocr system. *IEEE, Computer Society*, 2004.

[67] J. Sauvola and M. Pietikäinen. Adaptive document image binarization. *Pattern Recognition*, 33:255–236, 2000.

[68] A. Schäfer. *An Optimizational Approach for an Algorithmic Reassembly of Fragmented Objects*. PhD thesis, Interdisciplinary Research Center for Scientific Computing (IWR), University of Heidelberg., January 2015.

[69] H. F. Schantz. *The History of OCR*. Technologies Users Association, Boston, 1982.

[70] F. Shafait, D. Keysers, and T. M. Breuel. Efficient implementation of local adaptive thresholding techniques using integral images. *Journal of Document Recogntion and Retrieval*, 2008.

[71] T. Som and S. Saha. Handwritten character recognition using fuzzy membership function. *International Journal of Emerging Technologies in Sciences and Engineering*, 5(2):11–15, Dec 2011.

[72] S. N. Srihari, T. Hong, and G. Srikantan. Machine-printed japanese document recognition. *Pattern Recognition*, 30(8), 1997.

[73] R. Stefanelli and A. Rosenfeld. Some parallel thinning aalgorithm for digital pictures. *Juournal of the Association for Computing Machinery*, 18(2):255–264, April 1971.

[74] A. Stentz. Optimal and efficient path planning for partially- known environments. In *IEEE international Conference on Robotics and Automation*, May 1994.

[75] C. Tanprasert and T. Koanantakool. Thai ocr: A neural network application. *IEEE TENCON- Digital Signal Processing Application*, 1996.

[76] C. C. Tappert, C. Y. Suen, and T. Wakahara. The state of the art in on-line handwriting recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(8), August 1990.

[77] W. H. Tsai and S. S. Yu. Attributed string matching with merging for shape recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Pami-7(4):453– 462, July 1985.

[78] Y. T. Tsay and W. H. Tsai. Attributed string matching by split and merge for online chinese character recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 15(2), February 1993.

[79] UNESCO. *Ancient Inscriptions of Cambodia*, volume 1,2,3,4. UNESCO, January 2002.

[80] K. Vanna. Khmer optical character recognition. Master's thesis, Kameyama Laboratory, Graduate School of Global Information and Telecommunication Studies, Waseda University, January 2011.

[81] S. Vong. *Angkor Inscriptions of Cambodia I*. Royal University of Phnom Penh, 2008.

[82] S. Vong. *Pre-Angkor Inscriptions of Cambodia I*. Royal University of Phnom Penh, 2009.

[83] S. Vong. *Post- Angkorian Inscriptions of Cambodia*. Royal University of Phnom Penh, 2011.

[84] R. A. Wagner. The string-to string correction problem. *Journal of the Association for Computing Machinery*, 21(1):168–173, January 1974.

[85] I. H. Witten, E. Frank, and M. A. Hall. *Data Mining, Practical Machine Learning Tools and Techniques*. Elsevier, third edition edition, 2011.

[86] S. I. Woolley, T. Davis, N. Flowers, J. P. Dutoi, A. Livingstone, and T. Arvanitis. Communicating cuneiform: The evolution of a multimedia cuneiform database. *The Journal of Visible Language*, 3(36):308–324, 2002.

[87] Y. Xu and G. Nagy. Prototype extraction and adaptive ocr. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(12), December 1999.

[88] K. Yu, J. Wu, and Y. Zhuang. Skeleton-based recognition of chinese calligraphic character image. *Springer- Verlag Berlin, Heidelberg*, 2008.

[89] T. Y. Zhang and C. Y. Suen. A fast parallel algorithm for thinning digital patterns. *Communication of the ACM*, 27(3):236–239, March 1984.

[90] X. F. Zhang, Y. Zhuang, J. Wu, and F. Wu. Hierarchical approximate matching for retrieval of chinese historical calligraphy character. *Journal of Computer Science and Technology*, 22(4), 2007.

[91] Y. Zhuang, X. Zhang, J. Wu, and X. Lu. Retrieval of chinese calligraphic character image. *Springer- Verlag Berlin, Heidelberg*, 2004.