Asian Character Recognition

14

Srirangaraj Setlur and Zhixin Shi

Contents

S. Setlur • Z. Shi
Department of Computer Science and Engineering, University at Buffalo, The State University of New York, Buffalo, NY, USA
e-mail: setlur@buffalo.edu; zshi@buffalo.edu

D. Doermann, K. Tombre (eds.), *Handbook of Document Image Processing and Recognition*, DOI 10.1007/978-0-85729-859-1_14,

Abstract

This chapter deals with the automated recognition of Asian scripts and focuses primarily on text belonging to two script families, viz., oriental scripts such as Chinese, Japanese, and Korean (CJK) and scripts from the Indian subcontinent (Indic) such as Devanagari, Bangla, and the scripts of South India. Since these scripts have attracted the greatest interest from the document analysis community and cover most of the issues potentially encountered in the recognition of Asian scripts, application to other Asian scripts should primarily be a matter of implementation. Specific challenges encountered in the OCR of CJK and Indic scripts are due to the large number of character classes and the resultant high probability of confusions between similar character shapes for a machine reading system. This has led to a greater role being played by language models and other post-processing techniques in the development of successful OCR systems for Asian scripts.

Keywords

Character recognition • Chinese • CJK • Classifiers • Devanagari • Features • Hindi • HMM • Indic • Japanese • Korean • Language models • Neural networks • OCR • Post-processing

Introduction

History and Importance

Asian scripts are among the oldest scripts of the world and hence have played an important role in shaping early civilizations and influenced the development of other modern scripts. Scripts in general have been subject to religious and cultural influences over time, and many scripts in use today have their origins rooted in early Asian scripts (▸Chap. 9 (Language, Script and Font Recognition)).

While Asian scripts such as CJK and Indic scripts support a large number of languages and dialects, the number of distinct scripts used to represent these languages is not as numerous. Asian scripts belong to three main classes [1]:

1. **East Asian scripts** are based on the East Asian or Han ideographs that were developed in China in the second millennium BCE. These are also known as the CJK scripts (for Chinese, Japanese, and Korean – the three main languages that use these scripts). The primary scripts in this category are Han (Chinese), Hiragana and Katakana (Japanese), and Hangul (Korean). Chinese ideographs are also used in modern Japanese (Kanji) and in modern Korean (Hanja).
2. **South and Southeast Asian Scripts** are based on the ancient Brahmi script, oldest instances of which have been found from third century BCE. South Asian scripts are used by at least 22 major languages and hundreds of minor languages or dialects spoken by populations in and around the Indian subcontinent. The prominent scripts and the major languages for which they are used

(in parentheses) are Devanagari (Sanskrit, Hindi, Marathi, Nepali to name a few), Bangla (Bengali, Asomiya, Manipuri), Gurmukhi (Punjabi), Gujarati (Gujarati), Oriya (Oriya, Santhali), Kannada (Kannada), Tamil (Tamil), Telugu (Telugu), and Malayalam (Malayalam).

Southeast Asian scripts are also derived from Brahmi and exhibit some idiosyncrasies that distinguish them from South Asian scripts. The scripts belonging to this category are Thai, Philippine scripts, and Indonesian scripts (Balinese and Buginese). This chapter covers OCR research on only the South Asian or Indic scripts.

3. **West Asian Scripts** are influenced by the Phoenician script – these have a strong Arabic influence and hence will be covered by the chapter on Middle Eastern Character Recognition (►Chap. 13 (Middle Eastern Character Recognition)).

Evolution of the Problem

The successful advent of the first generation OCR systems for machine-printed Latin characters in the early 1960s spurred an immediate interest in the machine reading of non-Latin scripts. Among the CJK scripts, the earliest reported attempt at printed Chinese character recognition was in 1966 [2]. Research into recognition of Indic scripts such as Devanagari began in the early 1970s [3]. Despite these early endeavors, research into recognition of Indic scripts did not pick up steam until the 1990s whereas there has been a more widespread and sustained interest in recognition of CJK scripts over the years. This is perhaps a result of the fragmentation of the market since many of the Indic scripts and languages are localized to small geographic regions.

Applications

While traditional OCR application domains such as postal routing and bank check recognition have also been targeted in the context of Asian character recognition, digital library applications have been seen to be particularly appealing due to the rich heritage of the ancient cultures that gave birth to these languages and scripts. Hence, interesting preprocessing applications for enhancement of historical documents on media such as papyrus, palm-leaf, and stone inscriptions prior to word spotting, transcript mapping, OCR, and other downstream applications have also seen considerable research effort.

Main Difficulties

Asian scripts, in general, share the common trait that a single written form is used to represent multiple languages that are distinct and, many a time, mutually

unintelligible. This often results in regional variations in writing of the same character. Asian scripts also have a large number of character classes. This section describes the difficulties that Asian scripts pose for machine reading.

CJK Scripts

The CJK scripts owe their origin to the Han ideographs developed in China in the second millennium BCE. Several standard romanizations are commonly used to refer to East Asian ideographic characters. They include hànzì (Chinese), kanzi (Japanese), kanji (colloquial Japanese), hanja (Korean), and chuhán (Vietnamese). The following English terms used to describe these character sets are interchangeable: Han character, Han ideographic character, East Asian ideographic character, or CJK ideographic character.

The Unicode Standard has a Han character repertoire of 75,215 characters. However, less than 4,000 of these make up over 99 % of the characters seen in actual use today. So, for OCR applications, the effective target class space is slightly less than 4,000 characters which still is a daunting number.

Indic Scripts

Indic scripts are all *abugidas* or writing systems where the effective unit in the script is the orthographic syllable where the core represents a vowel or a consonant with an inherent vowel. The consonant can be preceded by one or more optional consonants. The canonical structure of this orthographic syllable unit is (((C)C)C)V where C is a consonant and V is a vowel. This usually also corresponds to a phonological syllable.

Large Character Set: All nine Indic scripts more or less share the same relatively small basic set of vowels and consonants. However, syntactically the number of character shapes tends to be in the hundreds since consonants and vowels combine according to the canonical form (((C)C)C)V to form distinct characters. In addition to independent vowels and consonants, Indic scripts use alphabetic pieces known as vowel modifiers or dependent vowel signs. The position of the vowel modifiers or *maatraas* and the resulting variation in the shape of the consonants is very different in each of the nine scripts.

Figure 14.1 shows the vowels in all nine scripts and a single consonant with each of the vowel modifiers applied. Some vowels are not present in all scripts and the corresponding entries are shown blank. The basic consonants in each of the nine scripts are shown in Fig. 14.2. These scripts are also characterized by a large number of consonant conjunct forms where the characters change shape depending on their context. The appearance of the consonants is sometimes also affected by its ordering with respect to other characters. This results in a large number of character glyphs and hence poses a challenge for reading systems.

The nature of the variations in shapes between characters is physically very minute and hence classifiers find it difficult to overcome this confusion.

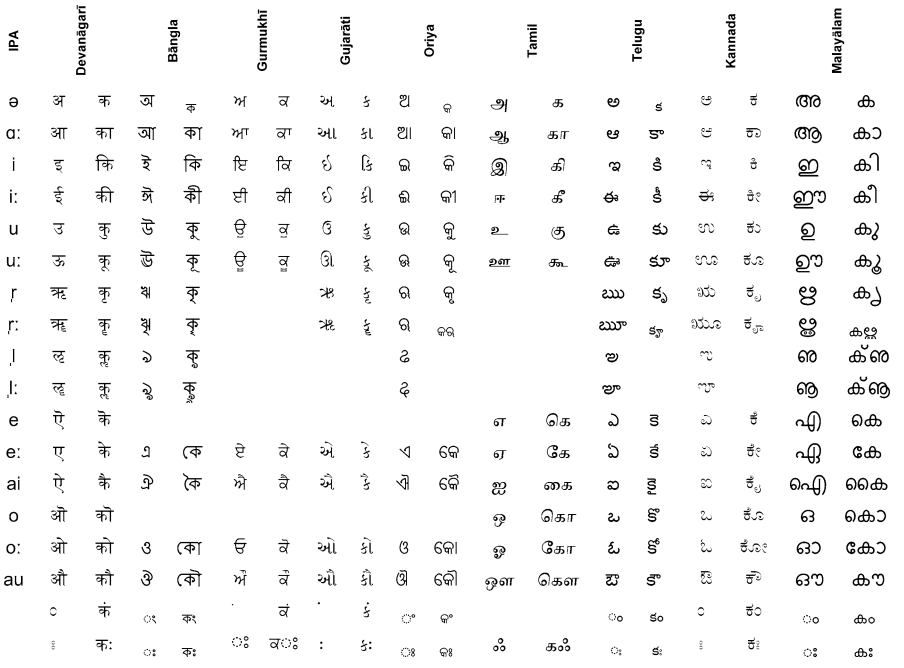


Fig. 14.1 Vowels and a consonant with corresponding vowel modifiers

Additionally, handwritten characters are further compounded by writer idiosyncrasies in writing these complex shapes. Post processing using language models attains greater importance in the classification of such large character sets with minor structural differences.

Summary of the State of the Art

While the state of the art in CJK character recognition has seen significant advances over the past few decades, evaluation of early research results was on relatively smaller private data sets, which made the comparison of the efficacy of different methods very difficult. Recent years have seen the release of large openly accessible data sets such as CASIAHWDB/OLHWDB [4] and international contests such as the ICDAR 2011 Chinese Handwriting Recognition Competition [5] that have made the objective assessment of the state of the art in recognition of CJK scripts feasible.

Recognition systems for machine-printed CJK documents are mature and readily available. The recognition performance for machine-printed documents has reached

IPA	Devanagari	Bangla	Gurmukhi	Gujarati	Oriya	Tamil	Telugu	Kannada	Malayalam
k	क	ক	ਕ	ક	କ	க	క	ಕ	ക
kʰ	ख	খ	ਖ	ખ	ଖ		ఖ	ಖ	ഖ
g	ग	গ	ਗ	ગ	ଗ		గ	ಗ	ഗ
gʱ	घ	ঘ	ਘ	ઘ	ଘ		ఘ	ಘ	ഘ
ŋ	ङ	ঙ	ਙ	ઙ	ଙ	ங	ఙ	ಙ	ങ
c	च	চ	ਚ	ચ	ଚ	ச	చ	ಚ	ച
cʰ	छ	ছ	ਛ	છ	ଛ		ఛ	ಛ	ഛ
ɟ	ज	জ	ਜ	જ	ଜ	ஜ	జ	ಜ	ജ
ɟʱ	झ	ঝ	ਝ	ઝ	ଝ		ఝ	ಝ	ഝ
ɲ	ञ	ঞ	ਞ	ઞ	ଞ	ஞ	ఞ	ಞ	ഞ
ʈ	ट	ট	ਟ	ટ	ଟ	ட	ట	ಟ	ട
ʈʰ	ठ	ঠ	ਠ	ઠ	ଠ		ఠ	ಠ	ഠ
ɖ	ड	ড	ਡ	ડ	ଡ		డ	ಡ	ഡ
ɖʱ	ढ	ঢ	ਢ	ઢ	ଢ		ఢ	ಢ	ഢ
ɳ	ण	ণ	ਣ	ણ	ଣ	ண	ణ	ಣ	ണ
t̪	त	ত	ਤ	ત	ତ	த	త	ತ	ത
t̪ʰ	थ	থ	ਥ	થ	ଥ		థ	ಥ	ഥ
d̪	द	দ	ਦ	દ	ଦ		ద	ದ	ദ
d̪ʱ	ध	ধ	ਧ	ધ	ଧ		ధ	ಧ	ധ
n	न	ন	ਨ	ન	ନ	ந	న	ನ	ന
n	ऩ					ன			
p	प	প	ਪ	પ	ପ	ப	ప	ಪ	പ
pʰ	फ	ফ	ਫ	ફ	ଫ		ఫ	ಫ	ഫ
b	ब	ব	ਬ	બ	ବ		బ	ಬ	ബ
bʱ	भ	ভ	ਭ	ભ	ଭ		భ	ಭ	ഭ
m	म	ম	ਮ	મ	ମ	ம	మ	ಮ	മ
j	य	য	ਯ	ય	ଯ	ய	య	ಯ	യ
r	र	র/ৰ	ਰ	ર	ର	ர	ర	ರ	ര
r	ऱ					ற	ఱ	ಱ	റ
l	ल	ল	ਲ	લ	ଲ	ல	ల	ಲ	ല
ɭ	ळ		ਲ਼	ળ	ଳ	ள	ళ	ಳ	ള
ɻ	ऴ					ழ		ೞ	ഴ
ʋ	व	ৱ	ਵ	વ		வ	వ	ವ	വ
ɕ	श	শ	ਸ਼	શ	ଶ		శ	ಶ	ശ
ʂ	ष	ষ		ષ	ଷ	ஷ	ష	ಷ	ഷ
s	स	স	ਸ	સ	ସ	ஸ	స	ಸ	സ
h	ह	হ	ਹ	હ	ହ	ஹ	హ	ಹ	ഹ

Fig. 14.2 Consonants in Indic scripts

high levels of accuracy. For example, the THOCR system developed by the Department of Electronic Engineering, Tsinghua University, reports the highest correct character recognition rate of 98.6 % [6].

Handwritten document analysis and recognition in CJK scripts has been a very active research topic over the past 40 years. Despite these research efforts handwritten text recognition can still be considered an unsolved problem as evidenced by low accuracies on freely written samples [7]. To stimulate research in this field, the National Laboratory of Pattern Recognition (NLPR), Institute of Automation of Chinese Academy of Sciences (CASIA), has released large data sets of free handwriting such as CASIA-HWDB/OLHWDB [4].

A sense for the accuracy of Chinese handwriting recognition can be obtained by reviewing the most recent results of the ICDAR 2011 Chinese Handwriting Recognition Competition [5]. The results from multiple participants represent the state-of-the-art performance in handwritten Chinese character recognition for both online as well as offline documents. The databases CASIA-HWDB and CASIA-OLHWDB contain offline and online handwritten characters and continuous text written by 1,020 writers. The samples include both isolated characters as well as handwritten text. The data sets of isolated characters contain about 3.9 million samples of 7,356 classes (7,185 Chinese characters and 171 symbols), and the datasets of handwritten text contain about 5,090 pages and 1.35 million character samples. All the data have been segmented and annotated at character level, and each data set is partitioned into standard training and test subsets.

In the case of OCR of Indic scripts, much of the published research is on Devanagari and Bangla scripts and there is relatively little work on other scripts. In the absence of standard open data sets and competitions, results have been reported on closed sets of various sizes and in some cases on data sets of unspecified size. Also, much of the work appears to be on digit and character recognition (both machine print and handwritten) and there are only a handful of papers that specify performance results at the word recognition level. Even though character recognition results are reported in the mid-to-upper 90 % range, word recognition performance tops out at about 87 % even with the use of various post-processing methods and language models. While references to in-house end-to-end systems have been reported in the literature, they do not appear to be widely available commercially even for the processing of machine-printed documents. Initiatives such as the Technology Development for Indian Languages (TDIL) program of the Government of India have served to coordinate the OCR research and development efforts for Indic scripts in all of the premier academic research labs in the country. In a consortium-mode project, this effort has led to significant progress in the development of OCRs for a number of Indic scripts. The BBN Byblos OCR System also has Indic language support. The state of the art clearly shows significant performance gains in the past few years in basic recognition but also indicates that much still needs to be done before handwritten and multilingual, multi-script documents can be automatically processed for digital transcription.

Recognition of CJK Scripts

This section highlights the key challenges in the recognition of documents in Chinese, Japanese, and Korean scripts.

Preprocessing

Traditionally CJK texts are printed or handwritten using black ink on evenly colored background such as plain white paper or red paper in the case of special documents of cultural significance. Binarization of scanned CJK documents is not very different from documents in other scripts. Targeted preprocessing techniques required for CJK document images are for text line orientation detection, word segmentation, character segmentation, and script identification.

CJK documents may have either horizontal or vertical text layout. Both text layout directions may be used in the same document page. When horizontal layout is used, the text reads from left to right and the line order is from top to bottom. When vertical layout is used, the text reads from top to bottom and the line order is from right to left. Text line orientation detection algorithms presented in [8, 9] are based on the observation that inter-character spacing is generally smaller than interline spacing. A connected-component-based approach using a minimal spanning tree is applied in calculating the predominant direction of the text line.

Most text line separation algorithms designed for Latin-based language document images are also applicable to CJK documents. Due to the complexity of the character structure, the text sizes in CJK documents are generally bigger than Latin-based language documents. This is a minor source of concern for those algorithms which are size dependent/sensitive. Overall text line separation for CJK documents, both machine-printed as well as handwritten, is relatively easier than for Latin-based language documents.

Segmentation

In Latin-based language documents, words are generally separated by relatively wider white spaces within a text line. Words that are split across text line are hyphenated. But in Chinese documents words are not delimited by any mark or space. Line boundaries cannot be used for word segmentation either. Punctuations indicate only the boundary of sentences. In Japanese documents the word boundaries are often found at the transition from Hiragana characters to Kanji. In Korean documents, words are separated by spaces but may be broken across text lines without hyphenation. For above reasons, most CJK document recognition systems proceed directly to character segmentation after line separation.

Accurate character segmentation is a vital step in the processing of CJK documents. When a document includes only CJK characters, segmentation of printed

characters is generally an easy problem. Simple projection profile methods are often sufficient. When a document contains multiple scripts such as Chinese with English, or CJK character with Latin numerals, character segmentation may be nontrivial. For multiple script documents, a two-path approach has worked well where the first pass labels the text line segments which are either CJK or Latin characters, and then in the second pass, the characters in each language are segmented with a language-dependent algorithm.

Segmentation of handwritten CJK characters is a much more complicated problem. There are several methods for handwritten CJK character segmentation based on a combination of character recognition and statistical approaches. Final character segmentation results are determined from multiple segmentation candidates.

Character Recognition

Character recognition algorithms for CJK can be broadly categorized into three groups: radical-based algorithms, stroke-based algorithms, and holistic approaches.

Radical-Based Algorithms

Most CJK character such as Chinese characters and Korean characters are logographic, which are visual symbols representing a minimal meaningful unit of the language. Each character is a graph that is composed of simple basic graphs called radicals. Each radical appears in a specific position, such as left-hand radical and upper radical. A small number of such radical primitives can be put together to compose many different Chinese and Korean characters [10].

Radical-based approaches decompose the large set of characters into a much smaller set of radicals. Then the character recognition is reduced to matching of radicals to a character image. The best match determines the identity of the character (Fig. 14.2). Modeling radicals is the most important processing step in radical-based recognition. An active shape modeling method has been successfully used for modeling radicals [10]. Firstly a radical character image is converted to its skeleton representation. Then the critical landmark points on the skeleton are labeled. Finally using the landmark points as features, a principal component analysis is applied to calculate the main shape parameters that capture the radical variations. In [10] a subset of 200 most common and productive radicals were selected to cover 2,154 Chinese characters out of the 3,755 GB2313 set. Based on the possible location of radical locations in forming the character, the character image is decomposed into multiple partitions such as left-right, up-down, or quadric sections. To match radicals in a character, a chamfer distance is used to find the optimal match of radicals in the modeled set to that in the partitions of the character image.

Stroke-Based Algorithms

Stroke-based approach uses finer level structures for composing a character image. Rather than modeling radicals, stroke-based approach decomposes a character image into a set of strokes, which are usually defined as the writing segment from a pen-down to a pen-up in the ideal cases. The recognition of the character is reduced to recognition of the stroke shapes and positional information of the stroke set in forming the character [11].

Recognition of a stroke image is usually done by extracting the direction features such as the four directions or eight directions (Fig. 14.3).

To extract stroke segments, segment feature points are first identified on the skeleton of the character image. The segment feature points include end points and fork points. Skeleton sections between feature points are extracted as stroke segments. Statistical-structural scheme based on Markov random fields (MRFs) can be used in the recognition of the stroke sequence [12]. The stroke segments are represented by the probability distribution of their locations, directions, and lengths. The states of MRFs encode the information from training samples. Through encouraging and penalizing different stroke relationships by assigning proper clique potentials to spatial neighboring states, the structure information is described by neighborhood system and multi-state clique potentials and the statistical uncertainty is modeled by single state potentials for probability distributions of individual stroke segment. Then, the recognition problem can be formulated into MAP-MRF framework to find an MRF that produces the maximum a posteriori for unknown input data. More details can be found in [12].

Statistical Feature Approaches

Although radical-based approach and stroke-based approach reduce the number of target classes to a small set of radicals or strokes, they often encounter difficulties in matching radicals to the character image or finding the optimal stroke sequence. Statistical approach is used to directly extract features from the character image. The most popular features for Chinese character recognition include peripheral shape features, stroke density features, stroke directional features, and gradient features [13, 14].

Peripheral features [13]: To extract peripheral features, an input character image is first partitioned into 2×4 bins. The character image is separated into left and right parts each of which consists of four horizontal strips labeled by $S_h^1 S_h^2 \ldots S_h^8$ in the character image frame. Suppose strip S_h^i consists of k lines. Let $|\lambda_h^i|$ be the distance between the outermost stroke edge and the character image frame along i-th line and $H_h^i \times V_h^i$ be the subarea of the i-th horizontal strip S_h^i. The horizontal peripheral background area in the i-th horizontal strip S_h^i can be represented by the following formula:

$$A_h^i = \left\{ \sum_{i=1}^{k} |\lambda_h^i| \right\} / (H_h^i \times V_h^i)$$

Fig. 14.3 Example of the two-pass method for character segmentation. *Left*: Identification of multiple language script. *Right*: Character segmentation result

Similarly, horizontal peripheral line difference Ψ_h^i, vertical peripheral back ground A_v^i, and vertical peripheral line difference Ψ_v^i are defined as

$$A_h^i = \left\{ \sum_{i=1}^{k} |\lambda_h^i| \right\} / (H_h^i \times V_h^i)$$

$$\Phi_h^i = \Re_h^i / \sum_S \Re_h^i$$

$$\psi_h^i = \left\{ \sum_{i=1}^{k} ||\lambda_h^i - \lambda_h^{i+1}|| \right\} / (H_h^i \times V_h^i)$$

Stroke Density Features [13]: 16 stroke density features can be extracted in two different ways, depending on the directions of the elastic meshing used (see Fig. 14.4). Suppose there are m horizontal inspection lines in horizontal strip S_i, the crossing count on inspection line r_j is

$$\frac{1}{2} \sum\nolimits_x \overline{f(x,y)} f(x+1,y)$$

The horizontal regional stroke density (HRSD) is the quotient of the accumulation of the crossing counts in S_i divided by the number of horizontal inspection lines in horizontal strip S_i. It is denoted by $\Re_h^i$ and computed by the following formula:

$$\Re_h^i = \left\{ \frac{1}{2} \sum_{j=1}^{m} \sum\nolimits_x \overline{f(x,y)} f(x+1,y) \right\} / m$$

Similarly horizontal regional stroke density distribution (HRSDD) Φ_h^i, vertical regional stroke density (VRSD) $\Re_v^i$, and vertical regional stroke density distribution (VRSDD) Φ_v^i can be calculated by (Fig. 14.5)

$$\Phi_h^i = \Re_h^i / \sum_S \Re_h^i$$

$$\Re_v^i = \left\{ \frac{1}{2} \sum_{j=1}^{m} \sum\nolimits_x \overline{f(x,y)} f(x,y+1) \right\} / m, \qquad \Phi_v^i = \Re_v^i / \sum_S \Re_v^i$$

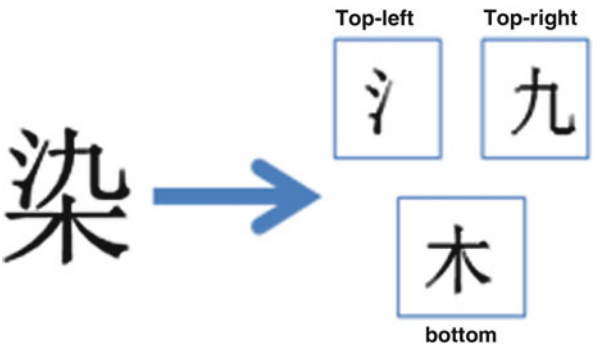


Fig. 14.4 Example of a Chinese character decomposed into idealized radicals

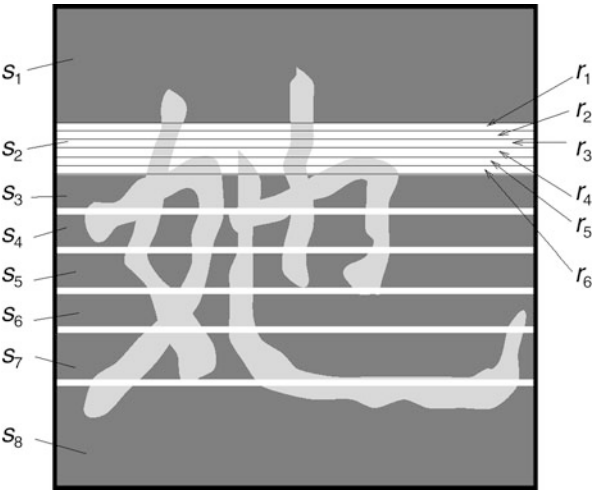


Fig. 14.5 Stroke density features (From Ref. [13])

Gradient Features [14]: To extract gradient features, 3×3 Sobel operators are used to get the horizontal and vertical grayscale gradient at each image pixel, respectively. L directions are defined with an equal interval $2\pi/L$, and the gradient vector is decomposed as illustrated in Fig. 14.6. An L-dimensional gradient code is calculated at each image pixel. To quantify the gradient code into a feature vector, the 65×65 normalized image is divided equally into 13×13 sub-blocks; within each sub-block the L-dimensional gradient codes are summed up and then the resolution of sub-block is down sampled to 7×7 by a Gaussian filter, resulting in a $d-$dimensional gradient feature, where $d = 7 \times 7 \times L$. Finally a variable transformation $\boldsymbol{y} = \boldsymbol{x}^{0.4}$ is applied on each element of the extracted feature vector to make its distribution more Gaussian-like (Fig. 14.7).

Classification

Classification is one of the key modules in any feature-based OCR system. Many classification methods have been designed and implemented in OCR systems for

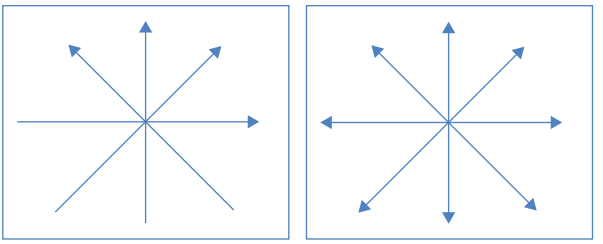


Fig. 14.6 Stroke features are extracted from four directions (*left*) or eight directions

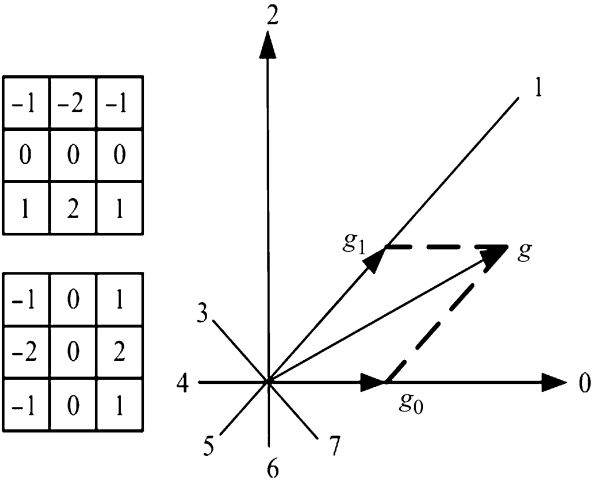


Fig. 14.7 Gradient feature extraction using Sobel operators and gradient vector decomposition (eight directions for illustration) (From Ref. [14])

recognition of Asian scripts. Among them the most popular and successful methods include nearest-neighbor (NN)-based methods, methods based on quadratic discriminant functions and their modifications (QDF and MQDF), and convolutional neural networks (CNN).

Nearest-Neighbor Classifiers

The nearest-neighbor classifiers are built upon preselected prototypes which are feature vectors with assigned class ids [6]. The nearest-neighbor (NN) rule assigns an unknown sample to the class of its nearest prototype. The performance of a nearest-neighbor classifier is affected by the selection scheme of the training prototypes, the number of the prototypes, and the metrics used for measuring the distances. The k-nearest-neighbor (k-NN) rule is a popular nonparametric technique. The performance of k-NN classifier asymptotically approximates the Bayesian classifier if the number of training samples approaches infinity. The nearest-neighbor (1-NN) rule is a special case of k-NN rule. It was proved that the asymptotic classification error of 1-NN rule is bounded by twice the Bayesian error rate [15]. However due to the requirement of large size prototypes needed especially in the case of Asian character recognition, which deals with large number of classes, the computation efficiency and storage space are among the challenges.

Many approaches have been used to improve the performance of NN classifier in various aspects. To improve the classification performance of NN rule under a finite number of samples, different distance metrics have been used. In addition to improving the efficiency, the selection of prototypes from the original training sample set by adding or pruning can preserve the classification performance by using some heuristics. Some algorithms have been proposed to select prototypes with the aim of optimizing the classification performance [16].

Modified Quadratic Discriminant Function Classifier (MQDF)

For regular handwritten Chinese character recognition the feature distributions are near Gaussian distributions, which can be classified by the modified quadratic discriminant function classifier (MQDF). For a d-dimensional feature vector x, the QDF distance can be represented as follows:

$$g_i(x) = (x - \mu_i)^T \phi_i^{-1} (x - \mu_i) + \log |\phi_i|, i = 1, 2, \ldots, C$$

where C is the number of character classes and μ_i and ϕ_i denote the mean vector and the covariance matrix of class ω_i. Applying decomposition on ϕ_i and replacing the minor eigenvalues with a constant σ^2 to compensate for the estimation error caused by small training set, the MQDF f or a d-dimensional feature vector x is given as formula:

$$g_i(x) = \frac{1}{\sigma^2} \left\{ ||x - \mu_i||^2 - \sum_{j=1}^{q} \left(1 - \frac{\sigma^2}{\lambda_{ij}}\right) [\phi_{ij}^T (x - \mu_i)]^2 \right\} + \sum_{j=1}^{q} \log \lambda_{ij} + (d - q) \log \sigma^2, i = 1, 2, \ldots, C$$

The formula defines the distance of sample x and class ω_i. λ_{ij} and ϕ_{ij} denote the jth eigenvalue in descending order and the corresponding eigenvector of the covariance matrix of class ω_i, respectively; q denotes the number of dominant principal axes, $q < d$. The parameters are usually estimated using maximum likelihood (ML) framework on the training set. During classification, MQDF calculates distances between the test sample and each class according to above formula and arranges classes according to their corresponding distances in the increasing order to obtain the multiple candidate recognition results.

Convolutional Neural Network (CNN)

Convolutional neural networks are a type of neural networks which combine feature extraction and classification. They have been shown to perform better than probability density function approaches such as Gaussian Bayesian methods and Gaussian mixture models or nonparametric models such as K-nearest-neighbor classifiers. Biologically inspired from the human visual system, the distinctive

characteristic feature of a convolutional neural network is its integral functionality in combining multiple modules of conventional recognition systems. Usually character recognition systems have three basic modules: preprocessing feature extraction, and classification. In convolution neural network approach, raw images are taken as input and appropriate features are implicitly extracted for classification using a trainable neural network.

Convolutional feed forward neural networks were first described by LeCun et al. [17]. Convolutional neural networks are based on the modeling of the human retina structure. They are designed to recognize two-dimensional shapes with a high degree of invariance to shift, scaling, and distortion [17]. The main advantage of a convolutional neural network architecture is its implicit capability for automatic feature extraction. The input is the raw image and the output layer represents the winning classes as a classifier. Different layers are responsible for feature extraction, mapping, and subsampling tasks in addition to a traditional multilayer perceptron structure in the last stage working as a classifier. To train a CNN, the back propagation algorithm is successfully generalized for CNN's topology, which is known as deep neural networks learning [18]. The general topology of a CNN uses three architectural structures: local receptive field, shared weights, and spatial sub sampling [17]. Specifically, CNN architecture is made of one input layer and three types of hidden layers and one output layer. The first kind of hidden layers is responsible for convolution; the second kind of hidden layers is responsible for local averaging, subsampling, and resolution reduction; and the third kind of hidden layers act as a traditional multilayer perceptron classifier. In the first and second types, the extracted local features are saved in the output planes which are called feature maps. Then the extracted spatial features are classified in the last kind of layers which is a multilayer perceptron with two hidden layers.

Convolutional neural networks have been implemented and used successfully for Asian language character recognition. A CNN with nine hidden layers was used by one of the entries in the ICDAR 2011 Offline Chinese Handwriting Competition [5].

Post-processing

Post-processing is a vital step for improving recognition performance especially for Asian character recognition due to the high likelihood of confusion between character shapes given the large set of classes. In order to improve the recognition rate, language models that can utilize context can be used to correct the raw recognition results. Post-processing approaches based on linguistic knowledge also include using a lexicon [19, 20] or syntactic and semantic rules [21] to correct the spelling of words and using statistical language models (LMs) [22, 23] to select the best sequence from the candidate characters given by the OCR engine.

Statistical language models (LMs) have been widely used for the contextual post-processing to improve the accuracy in recognizing the Chinese scripts [24–27].

In Asian languages, conventional n-gram LMs such as character bigram, character trigram, or word bigrams have been utilized in post-processing of recognition results. Class-based LMs have also been shown to improve the performance of recognition systems when combined with conventional word-based LMs [28]. Hybrid bigram LMs combining both word- and class-based bigrams have also been tried [29].

The most widely used LMs are the n-gram models The conventional n-gram Chinese LMs can be based on either characters or words. Based on characters, for $n = 2, 3$, the character-based bigram model and the character-based trigram model are expressed, respectively, as follows:

$$p(S) = p(s_1) \prod_{t=2}^{T} p(s_t | s_{t-1})$$

and

$$p(S) = p(s_1) p(s_2 | s_1) \prod_{t=3}^{T} p(s_t | s_{t-2} s_{t-1})$$

where $S = s_1 s_2 \ldots s_T$ is a sequence of characters given by an isolated character recognizer, in which each output s_t may include the top K candidates $c_1 c_2 \ldots c_k$ with the corresponding distance measurement values $D = d_1 d_2 \ldots d_K$. The output of the system $O = o_1 o_2 \ldots o_T$ is the final sentence.

Considering the top K candidates for each output $s_{t,}$ there are K^T possible sentences. The post-processing task is then to select the best sentence from all the K^T sentences. Applying the rule of maximal posterior probability, the output O of a recognition system is formulated as

$$O = \arg\max_{S} p(S) \times \prod_{t=1}^{T} p(s_t | x_t)$$

where is one of above statistical LMs; $p(s_t \quad | \quad x_t)$ is the posterior probability of s_t given x_t, which can be computed from candidate confidence.

Similarly the word-level bigram and trigram models are

$$p(S) = p(w_1) \prod_{t=2}^{T} p(w_t | w_{t-1})$$

and

$$p(S) = p(w_1) p(w_2 | w_1) \prod_{t=3}^{T} p(w_t | w_{t-2} w_{t-1})$$

where $S = w_1 w_2 \ldots w_T$ is a sequence of words given by a word recognizer.

Recognition of Indic Scripts

This section highlights the key challenges in the recognition of documents in Indic scripts.

Preprocessing

Preprocessing of document images for OCR applications usually involves (i) page segmentation to narrow in on the text regions, (ii) noise removal of various types, (iii) binarization to separate the foreground text from the background since most recognition techniques have been designed for binarized images, and (iv) line segmentation (▶Chaps. 4 (Imaging Techniques in Document Analysis Process) and ▶8 (Text Segmentation for Document Recognition)). In general, most of the methods developed for preprocessing are script-agnostic. However, occasionally there are features of certain scripts that need to be taken into account while applying general preprocessing algorithms and, for some script-specific traits, may even require targeted preprocessing. There are also some document types that are more frequently encountered in conjunction with certain scripts. The emphasis of this chapter is on only those elements of preprocessing that have particular relevance to a specific Asian script itself or to document types that are more frequently encountered in the context of Asian scripts.

A few of the preprocessing issues that require adaptations for Indic scripts are addressed here:

(i) The *shirorekha* or headline which is a horizontal bar that connects characters in a word in scripts such as Devanagari and Gurmukhi – location and removal of these *shirorekha*s in machine-printed documents is fairly trivial and can be achieved using simple projection profile-based methods [30]. Skew is also easy to detect due to the locally linear *shirorekha* lines [31, 32]. However, in handwritten documents, the *shirorekha*s may not be very uniform and effective detection and removal requires localized adaptations of the projection profile approach [33, 34]. Morphological operations such as erosion and dilation have also been used to identify *shirorekhas* in handwritten images [35].

(ii) Line segmentation Separating text lines correctly and grouping diacritics and other small components with the correct line are crucial to the success of downstream OCR processes. In handwritten documents in general, it is common to see variability in skew between different lines in the same document leading to varying distances between lines as well as overlapping and touching of strokes from multiple lines. This is further exacerbated in handwritten documents using scripts such as the Indic scripts due to the presence of vowel modifiers and conjuncts that are dispersed all around the base character region leading to touching between strokes from multiple lines. Localized adaptive projection profile [36] and morphology-based methods [37] have been used to detect the location of text lines, but splitting touching components intelligently using

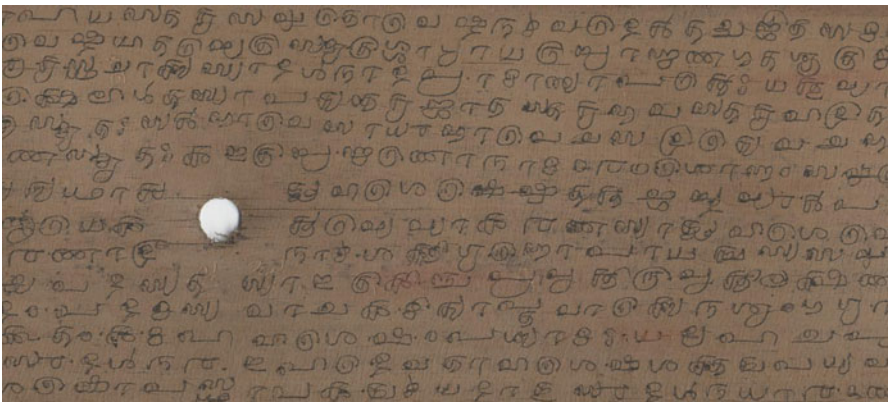


Fig. 14.8 Historical palm-leaf manuscript

stroke following and other approaches appears to have not been investigated in any great detail for handwritten Indic script documents.

(iii) Historical documents in Indic scripts can be found on a variety of non-paper surfaces such as palm leaves, parchment, and stone and wood inscriptions. Ancient palm-leaf manuscripts relating to religion, science, medicine, and astronomy are still available for reference today due to many ongoing efforts for preservation of ancient documents by libraries and universities around the world. These manuscripts typically last a few centuries but with time the leaves degrade and the writings become illegible. Image-processing techniques have been used to help enhance the images of these manuscripts so as to enable readability of the written text for human eyes as well as machine reading. Figure 14.8 shows an example of a palm-leaf manuscript.

Most document image enhancement algorithms have been designed primarily for binarization of modern documents. An overview of the traditional thresholding algorithms for text segmentation are given in [38] which compares various techniques such as Otsu's, entropy techniques, and the minimal error technique. Entropy-based methods specifically designed for historical document segmentation that deal with the noise inherent in the paper especially in documents written on both sides such as background noise and bleed-through text using direct image matching and directional wavelets as well as other methods designed for improving human readability while maintaining the original "look and feel" of the documents fail on palm-leaf manuscripts due to low contrast and varying color intensity of the background.

One method used for enhancing digital images of palm-leaf and other historical manuscripts approximates the background of a grayscale image using piecewise linear and nonlinear models. Normalization algorithms are used on the color channels of the palm-leaf image to obtain an enhanced gray scale image and an adaptive local connectivity map is used to try to segment lines of text from the enhanced images with the objective of facilitating techniques such as keyword

Fig. 14.9 Palm-leaf manuscript image after preprocessing (binarization and line segmentation)

spotting or partial OCR and thereby making it possible to index these documents for retrieval from a digital library. Figure 14.9 shows the results of line segmentation on the binarized image obtained using this method [39].

Segmentation

Segmentation of characters can be a particularly critical issue for Indic scripts depending on the system architecture. This stems from the use of vowel modifiers and conjunct consonants.

(i) Placement of vowel modifiers – in scripts such as Devanagari, the modifiers are typically connected to the characters and can usually be found on either side of or above and below the base character, e.g., the base consonant ka (क) takes the shapes (का,कि,की,कु, कू,कृ, के, कै, को, कौ, कं, कः) with different vowel modifier, whereas in scripts such as Kannada, the modifiers can involve multiple pieces positioned around the character (up, down, left, right) and may or may not be connected to the base character, e.g., the base consonant ka (ಕ) takes the shapes ಕಾ, ಕಿ, ಕೀ, ಕು, ಕೂ, ಕೃ, ಕೆ, ಕೇ, ಕೈ, ಕೊ, ಕೋ, ಕೌ, ಕಂ, ಕಃ.

(ii) Conjunct consonants – in Devanagari, conjuncts are represented by partial character shapes for the initial consonants followed by the full consonant shape for the last consonant (e.g., sya (स्य), ghne (घ्ने)) with the last consonant taking the vowel modifiers. In Kannada, the conjunct consonants have the first consonant in the sequence retain its complete shape whereas the second and subsequent consonants are smaller components sometimes with a new distinct shape under the first consonant and the first consonant takes the vowel modifiers (e.g., kree (ಕ್ರೀ), dve (ದ್ವೆ)). In Tamil, the initial consonants are represented in their full form with a dot and the last consonant in its full form (e.g., pra (ப்ர)) and the last consonant takes the vowel modifiers. So, depending on the script and the techniques being used, one would need to decide whether to segment

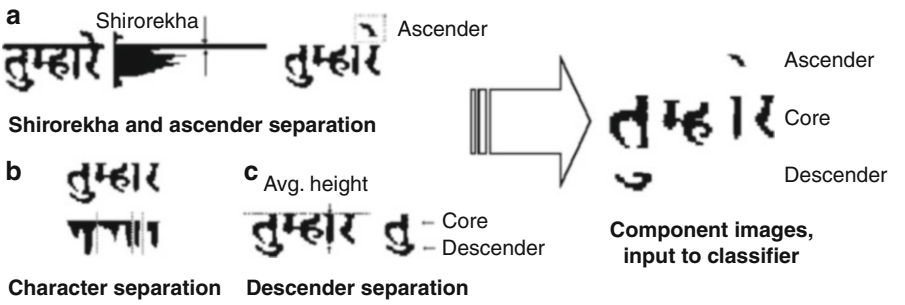


Fig. 14.10 Segmentation-driven OCR for Devanagari

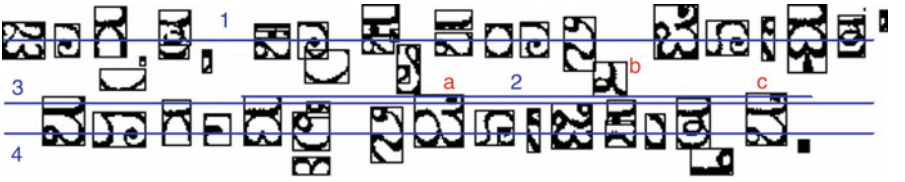


Fig. 14.11 Line segmentation in a Kannada document [43]

the character images into semantic sub-parts such as half-consonant, consonant, vowel, and vowel modifier prior to recognition or whether to treat the shape as a separate class.

Segmentation-driven OCR techniques split words into smaller units, with some using composite characters and others using finer semantic subparts as classification units. In the case of Devanagari, horizontal and vertical profiles are used to remove the shirorekha and separate the characters within the word. Word separation techniques developed for Latin characters work well for word segmentation in other Indic scripts that do not use the shirorekha (Fig. 14.10).

In the former design, the character images are processed directly using classifiers to obtain recognition results. The number of character classes is very large when compared to the number of unique semantic subparts. The ILT image data set [40] has 973 character classes, and the EMILLE text data set [41] has 5,573 character classes, and both have approximately 128 unique semantic subparts. Typically, classifier design increases in difficulty with an increase in class space. For example, 537 character classes from the ILT data set occur less than 25 times each. Due to lack of training samples, these classifiers often drop rare characters from the class space. Since a large number of classes have few samples, removing rare characters from the class space adversely affects overall OCR results. Using characters as the unit of classification for Devanagari is therefore less desirable [42] (Fig. 14.11).

Font variations and poor print quality add to the challenge. Techniques that assume uniform font attributes and rely on information extracted from a paragraph are not very robust especially when considering multi-font documents. A method

Fig. 14.12 Generating block adjacency graph representation of a Devanagari word image

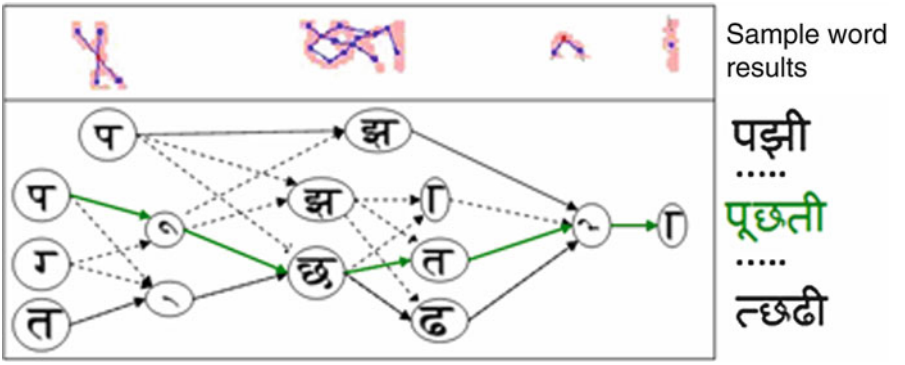


Fig. 14.13 An example of a scheme where recognition results are generated based on multiple segmentation hypotheses and post-processing based on language models is used to identify the correct hypothesis

that does not depend on font statistics and accommodates character distortions is critical to process typical Indic documents.

Techniques that use the semantic subparts as classification units benefit from a reduction in the number of classes. However, segmenting characters into the semantic subparts before classification is often nontrivial and depends on using a representation that is suited to carry out this segmentation. Simple segmentation using structural attributes like location and number of vertical strokes in a character or average character size in the paragraph often fails in segmenting conjuncts such as (स्य) ghne (घ्ने) since the characters are conjoined with each other along nonlinear boundaries.

Graph-based representations provide a better handle on segmenting these connections along nonlinear boundaries so that the split segments are more representative of the true character shapes and hence can provide better classification performance. Block adjacency graphs have been used as a successful representation scheme to deal with touching characters and conjuncts [42] (Fig. 14.12).

A recognition-driven segmentation approach uses multiple segmentation hypotheses to generate character recognition results followed by the use of post-processing techniques such as dictionary lookup for lexicon pruning and language models to identify the correct segmentation hypothesis (Fig. 14.13).

Discriminative Features

Selection of the right set of features that help discriminate between the desired classes is at the core of the recognition task. In approaches such as template matching, the entire image can be considered as the feature and in the case of machine text, the approach can provide good results. But the dimensionality of the feature is proportional to the size of the image and the high dimensionality ends up being a significant bottleneck for a real-time system. So, an added element to consider in designing effective features is to keep the dimensionality of the feature vector to a manageable size. It can be seen that many of the structural and statistical features that have been used for Latin-based scripts have also been successfully used for Indic scripts. A few of the most effective features that have been used for Indic script recognition will be reviewed in this section.

(i) Structural or geometric features: Structural features tend to explicitly capture structural aspects of text such as ascenders, descenders, loops, branch points, and end points. So, in the case of Indic scripts, features such as presence or absence of a vertical bar as in क vs ड would be a structural feature. They are useful features in the recognition of machine-printed text but are not very reliable features in the recognition of handwritten text due to the wide variations seen in handwriting in representing the same structure.

(ii) Statistical features: Statistical features are numerical measures computed over images or regions of images. They include, but are not limited to, measures such as pixel densities, histograms of slope along strokes, moments, curvature, and Fourier descriptors. Statistical features are among the most widely used features for handwritten character and word recognition. Extraction of reliable features and meaningful quantization of the measurements are the most challenging aspects of using statistical features.

Since high dimensionality is the major bottleneck for template-matching approaches, using principal component analysis for dimensionality reduction and choosing a sufficient number of features, researchers have been able to attain high character recognition rates on scanned clean machine-printed documents partially subject to degradation models. While this method worked well on cleaner document images, a decrease in performance can be expected for newspaper and other real-life documents.

Tree-based classifiers that use simple features to roughly partition the class space into more manageable number of classes have also been successfully used in machine-printed Devanagari OCR. This method can be extended to other Indic scripts also by selecting the appropriate sets of features for pruning (Fig. 14.14).

Gradient, structural, and concavity (GSC) features are a combination of features that have been effective across many scripts including Latin and Arabic and have been successfully used in Indic script recognition as well. The gradient features represent the local orientation of handwritten strokes, the structural features extend the gradient features to longer distances and provide information about stroke

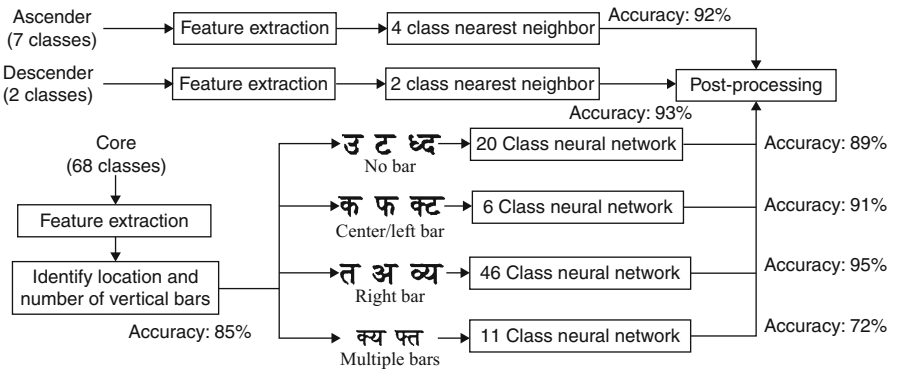


Fig. 14.14 Tree-based (multistage) classifier scheme for Devanagari OCR

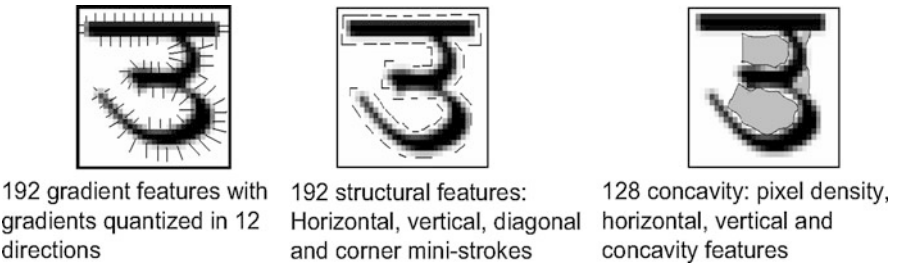


Fig. 14.15 GSC features from Devanagari character

trajectories, and the concavity features serve to detect certain stroke relationships at long distances. They also include information such as pixel density, existence of loops, and arcs and their direction. The extraction of the features is done by using mathematical filters such as Sobel operators and scanning and tracing the character image. For each type of feature, statistical sampling is done using fixed size grids. The character image is size normalized before extracting features. The nature of the GSC feature emphasizes the shape of the characters at three different levels of granularity, fine, medium, and coarse, and hence has been effective in obtaining high recognition rates on the ILT data set [40] which contains scanned images of documents spanning a wide spectrum of variations in the quality of paper and print [40]. Even with these features, it is hard to discern the fine details such as the minute differences in vowel modifiers that are used extensively in some Indic scripts. Post-processing can help overcome some of these problems (Fig. 14.15).

Variable grid sizes have been used with GSC to account for the varying widths of patterns.

Most machine-printed systems in the literature report performance at the character level on data sets that have perfectly segmented characters.

Classification

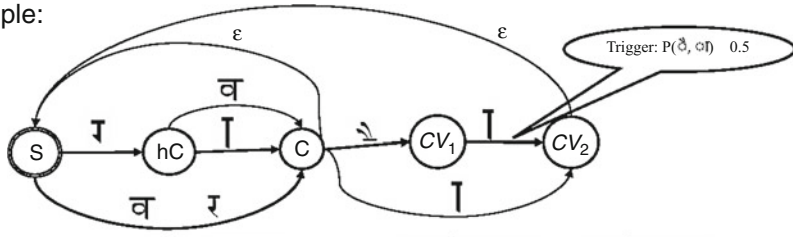
Many types of features and classification approaches have been reported in the OCR literature for Latin script recognition. Various combinations of these features and classification schemes have been tried with varying levels of success for Indic script recognition. Two broad approaches to classifier design for Indic scripts have been using complex classifier models for directly classifying the sample into one of the n-classes possible or using a hierarchical divide-and-conquer strategy where an ensemble of binary or k-ary ($k < n$) classifiers are integrated to form the final n-class classifier. Several flavors of neural-network based classifiers (MLP, PNN), k-nearest-neighbor (KNN), HMM-based approaches, support vector machines (SVM), modified quadratic discriminant function (MQDF), and generalized Hausdorff image comparison have all been used for Indic script recognition. Given the wide range of feature-classifier combinations that have been used for character recognition in Indic scripts and the lack of sufficient information about the various data sets over which the numbers have been reported, it is difficult to generalize any particular combination of features-classifiers as being superior to others for Indic script recognition. However, an analysis of the word recognition approaches that have been reported indicates that effective post-processing is very crucial.

Post-processing

In order to get satisfactory recognition results at the word and sentence levels, post-processing appears to be a necessity, especially for Indic scripts where the differences between some character shapes is very minute leading to confusions that cannot be overcome during classification. Initial approaches to achieve improved word-level performance consisted of techniques such as a simple dictionary lookup for error correction. Use of probabilistic language models to enhance recognition performance prior to using dictionary lookup that have been used successfully for Latin OCR was first used for Indic scripts in [42].

In [42], a stochastic finite state automaton (SFSA) is used to combine rule-based script composition validity checking and a probabilistic n-gram language model into a single framework. Different linguistic modeling units were evaluated in deciding the granularity that would be best suited for Devanagari word recognition. The general idea in designing an SFSA was to create a finite-state machine (FSM) that accepts strings from a particular language and assigning data-driven probabilities to transitions between states. If a candidate word is accepted, a score or probability is obtained and if rejected, corrections can be made using the minimum set of transitions that could not be traversed [44]. Here, the SFSA is modeled using script composition rules, and the transition probabilities are modeled on character or composite character n-grams. Transition probabilities for the SFSA are obtained from text corpus using the formula:

- Stochastic FSA can represent rules and statistical measures.
- Example:



A simplified FSA to reject {र ै ा र व} and accept {र ै ा र व}and {र ै ा र व}

S: Start/Accept state
hC: State after accepting half-consonant
C: State after accepting full-consonant
CV_1,CV_2,: States after accepting vowel modifiers

Fig. 14.16 Stochastic FSA used for Devanagari text recognition

$$a_{ij}(o) = \frac{\text{count of observing } o \text{ going from state } i \text{ to } j}{\text{Number of transitions from state } i \text{ to } j}$$

An n-gram model specifies the conditional probability of each unit, or token of text c_n with respect to its history $c_n, \ldots c_{n-1}$:

$$P(c_n|c_{1\ldots n-1}) = \frac{P(c_1 \ldots c_n)}{P(c_1 \ldots c_{n-1})} = \frac{\text{Count}(c_1 \ldots c_n)}{\text{Count}(c_1 \ldots c_{n-1})}$$

and perplexity values are reported over the test set (Pp):

$Pp(D,q) = 2^{-\frac{1}{N}\sum_x \log q(x)}$ where q represents the probability assigned by an n-gram model. A perplexity of k indicates that while predicting a token, the language model is as surprised as a uniform, independent selection from k tokens (Fig. 14.16).

Conclusion

As indicated in the preceding sections, there has been considerable progress in the development of techniques for Asian character recognition. Many different features and classification approaches have been successfully used to achieve high levels of accuracy in character recognition. As with other scripts and languages, handwritten text recognition is still a big challenge for Asian scripts. Unlike Arabic and cursive Latin scripts where a significant challenge is due to the difficulty of segmenting words into characters, the challenge for Asian scripts is in the ability to deal with the large number of character classes that differ only slightly in shape

leading to confusion between classes during classification. Language models for post-processing are likely to continue to play a big role in the quest to develop practical end-to-end systems that can process free-form handwritten text. Increasing availability of large data sets and a heightened interest in competitions at ICDAR and other conference venues offer the promise of rapid advances in the near future.

Cross-References

▸Analysis of the Logical Layout of Documents
▸Language, Script, and Font Recognition
▸Page Segmentation Techniques in Document Analysis
▸Text Segmentation for Document Recognition

References

1. The Unicode Standard, 6.0.0 (2011) The Unicode Consortium, Mountain View
2. Nagy G (1988) Chinese character recognition: a twenty-five year retrospective. In: Proceedings of the 12th international conference on pattern recognition, Rome, pp 163–167
3. Pal U, Chaudhuri BB (2004) Indian script character recognition: a survey. Pattern Recognit 37:1887–1899
4. Liu CL et al (2011) CASIA online and offline Chinese handwriting databases. In: Proceedings of 11th ICDAR, Beijing
5. Liu CL et al (2011) ICDAR 2011 Chinese handwriting recognition competition. In: ICDAR, Beijing
6. Ding X (2009) Advanced topics in character recognition and document analysis: research works in intelligent image and document research lab, Tsinghua University. In: Proceedings of the SPIE, San Jose, CA
7. Liu CL et al (2010) Chinese handwriting recognition contest 2010. In: Proceedings of the 2010 Chinese conference on pattern recognition, Chongqing, China
8. Kimura F (2007) OCR technologies for machine printed and hand printed Japanese text. In: Digital document processing. Major directions and recent advances. Springer, London
9. Baird HS, Ittner DJ (1993) Language-free layout analysis. In: Proceedings of the second international conference on document analysis and recognition, Tsukuba
10. Suen Y, Huang EM (1984) Computational analysis of the structural compositions of frequently used Chinese characters. In: Computer processing of Chinese and Oriental languages, World Scientific Publishing, Singapore
11. Tang KT, Leung H (2007) Reconstructing strokes and writing sequences from Chinese character images. In: Proceedings of the international conference on machine learning and cybernetics, Hong Kong
12. Zeng J, Liu ZQ (2005) Markov random fields for handwritten Chinese character recognition. In: ICDAR 2005: Proceedings of the 9th international conference on document analysis and recognition, Seoul
13. Tang Y et al (1998) Offline recognition of Chinese handwriting by multifeature and multilevel classification. IEEE Trans PAMI 20(5):556–561
14. Liu HL, Ding X (2005) Handwritten character recognition using gradientfeature and quadratic classifier with multiple discrimination schemes. In: Proceedings of the ninth international conference on document analysis and recognition, Seoul

15. Liu CL, Nakagawa M (2001) Evaluation of prototype learning algorithms for nearest neighbor classifier in application to handwritten character recognition. Pattern Recognit 34(3): 601–615
16. Srihari SN, Hong T, Shi Z (1997) Cherry Blossom: a system for reading unconstrained handwritten page images. In: Symposium on document image understanding technology (SDIUT), Washington, DC
17. LeCun Y, Bengio Y (1995) Convolutional neural network for images, speech, and time series. In: Arbib MA (ed) The handbook of brain theory and neural networks. MIT, Cambridge
18. LeCun Y, Bottou L, Bengio Y, Haffner P (1998) Gradient based learning applied to document recognition. Proc. IEEE, 86(11): 2278–2324
19. Wimmer Z, Dorizzi B (1999) Dictionary preselection in a neuro-Markovian word recognition system. In: Proceedings of the 5th international conference on document analysis and recognition, Bangalore
20. Procter S, Illingworth J, Mokhtarian F (2000) Cursive handwriting recognition using hidden Markov models and a lexicon-driven level building algorithm. Proc IEE Vis Image Signal Process, pp 332–339
21. Marti U, Bunke H (1999) A full English sentence database for off-line handwriting recognition. In: Proceedings of the 5th international conference on document analysis and recognition, Bangalore
22. Brakensiek A, Willett D, Rigoll G (2000) Unlimited vocabulary script recognition using character n-grams. In: Proceedings of the 22nd DAGM symposium, Kiel. Tagungsband Springer
23. Zhuang L, Bao T, Zhu XY (2004) A Chinese OCR spelling check approach based on statistical language models. In: Proceedings of the IEEE international conference on system, man and cybernetics, Hague
24. Tung CH, Lee HJ (1994) Increasing character recognition accuracy by detection and correction of erroneously identified characters. Pattern Recognit 27(9):1259–1266
25. Chang CH (1996) Simulated annealing clustering of Chinese words for contextual text recognition. Pattern Recognit Lett 17(1):57–66
26. Lee HJ, Tung CH (1997) A Language model based on semantically clustered words in a Chinese character recognition system. Pattern Recognit 30(8):1339–1346
27. Wong PK, Chan C (1999) Post-processing statistical language models for a handwritten Chinese character recognizer. IEEE Trans Syst Man Cybern Part B Cybern 29(2):286–291
28. Martin S, Liermann J, Ney H (1998) Algorithms for bigram and trigram word clustering. Speech Commun 24:9–37
29. Li YX, Tan CL, Ding X (2005) A hybrid post-processing system for offline handwritten Chinese script recognition. Pattern Anal Appl 8:272–286
30. Chaudhuri BB, Pal U (1997) An OCR system to read two Indian language scripts: Bangla and Devanagari. In: Proceedings of the 4th international conference on document analysis and recognition, Ulm
31. Chaudhuri BB, Pal U (1997) Skew angle detection of digitized Indian script documents. IEEE Trans PAMI 19(2):182–186
32. Pal U, Mitra M, Chaudhuri BB (2001) Multi-skew detection of Indian script documents. In: Proceedings of the 6th conference on document analysis recognition, Seattle
33. Agrawal P, Hanmandlu M, Lall B (2009) Coarse classification of handwritten Hindi characters. Int J Adv Sci Technol 10:43–54
34. Hanmandlu M, Agrawal P, Lall B (2009) Segmentation of handwritten Hindi text: a structural approach. Int J Comput Process Lang 22(1):1–20
35. Shaw B, Parui SK, Shridhar M (2008) A segmentation based approach to offline handwritten Devanagari word recognition. In: Proceedings of the IEEE international conference information technology, Tampa, FL
36. Chaudhuri BB, Bera S (2009) Handwritten text line identification in Indian scripts. In: Proceedings of the 10th international conference on document analysis and recognition, Barcelona

37. Roy P, Pal U, Lladós J (2008) Morphology based handwritten line segmentation using foreground and background information. In: Proceedings of the international conference on frontiers in handwriting recognition, Montréal, pp 241–246
38. Leedham G et al (2002) Separating text and background in degraded document images – a comparison of global threshholding techniques for multi-stage threshholding. In: Proceedings of the eighth international workshop on frontiers in handwriting recognition (IWFHR'02), Niagara-on-the-Lake. IEEE Computer Society, p 244
39. Shi Z, Setlur S, Govindaraju V (2004) Digital enhancement of palm leaf manuscript images using normalization techniques. In: 5th international conference on knowledge based computer systems, Hyderabad
40. Setlur S et al (2003) Creation of data resources and design of an evaluation test bed for Devanagari script recognition. In: Proceedings of the 13th international workshop research issues data engineering: multi-lingual information management (RIDE-MLIM), Hyderabad
41. Baker P et al (2004) Corpus linguistics and South Asian languages: corpus creation and tool development. Lit Linguist Comput 19(4):509–524
42. Kompalli S, Setlur S, Govindaraju V (2009) Devanagari OCR using a recognition driven segmentation framework and stochastic language models. IJDAR 12:123–138
43. Umesh RS, Pati PB, Ramakrishnan AG (2009) Design of a bilingual Kannada-English OCR. In: Govindaraju V, Setlur S (eds) Guide to OCR for Indic scripts. Springer, London, pp 97–124
44. Amengual JC, Vidal E (1998) Efficient error-correcting Viterbi Parsing. IEEE Trans PAMI 20(10):1109–1116
45. Liu CL, Jaeger S, Nakagawa M (2004) Online recognition of Chinese characters: the state-of-the-art. IEEE Trans PAMI 26(2):198–213
46. Liu C-L et al (2013) Online and offline handwritten Chinese character recognition: benchmarking on new databases. Pattern Recognit 46(1):155–162
47. Govindaraju V, Setlur S (ed) (2009) Guide to OCR for Indic scripts. Springer, London, 325p
48. Jayadevan R et al (2011) Offline recognition of Devangari script: a survey. IEEE Trans Syst Man Cybern 41(6):782–796

Further Reading

This chapter provides an overview of the key issues pertaining to the recognition of Asian scripts. An introduction to the history of recognition of Chinese characters can be found in [2]. A benchmarking of effective techniques for offline and online Chinese OCR can be found in [45] and [46]. An overview of the research issues in Indic OCR (offline as well as online) and information retrieval can be found in [47]. All major research groups working in this area are represented in this book, which is divided into sections on recognition of Indic scripts and retrieval of Indic documents. A recent overview of Devanagari script recognition can be found in [48].