FARSI HANDWRITTEN WORD RECOGNITION USING CONTINUOUS HIDDEN MARKOV MODELS AND STRUCTURAL FEATURES

By:

MOHAMMAD MEHDI HAJI

Advisors:

H. J. EGHBALI PH. D
S. D. KATEBI PH. D

JANUARY 2005

IN THE NAME OF GOD

FARSI HANDWRITTEN WORD RECOGNITION USING CONTINUOUS HIDDEN MARKOV MODELS AND STRUCTURAL FEATURES

BY:

MOHAMMAD MEHDI HAJI

THESIS

SUBMITTED TO THE SCHOOL OF GRADUATE STUDIES IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE (MSC.)

IN

COMPUTER ENGINEERING
SHIRAZ UNIVERSITY
SHIRAZ, IRAN

EVALUATED AND APPROVED BY THE THESIS COMMITTEE AS: **Excellent**

[illegible]............ H. J. EGHBALI, Ph. D., PROF. OF ELECTRONICS AND COMPUTER ENGINEERING (CHAIRMAN)

[illegible]............ S. D. KATEBI, Ph. D., PROF. OF CONTROL AND COMPUTER ENGINEERING (CHAIRMAN)

[illegible]............ A. TOWHIDI, Ph. D., PROF. OF CONTROL ENGINEERING

JANUARY 2005

Acknowledgements

It is not possible to forget the many people who contributed to this thesis, directly or indirectly. I would like to thank S. D. Katebi, H. J. Eghbali and A. Towhidi for their invaluable supervision, guidance and encouragements. I must also express my sincere thanks to all of the faculty members of Computer Science and Engineering Department of Shiraz University for their endeavors in my six years of academic education.

Special thanks must go to M. Khorsheed at University of Cambridge and M. Dehghan at Amirkabir University of Technology as they replied my each mail with requested information.

I would like to thank my best friends A. Yousefan, M. Adab, M. Eftekhari, A. Nayyeri, A. Sharifi, B. Haghhooei, R. Pariz, V. Daneshmand, M. Ahmadi, B. Daei, M. Hosseini, etc. for all their encouragements and contributions during this work. I would also like to thank all employees in the engineering school and particularly the Kharazmi library for their true effort and friendship.

I am also indebted to my family for their endless support, patience and providing a relaxing environment throughout my life.

ABSTRACT

FARSI HANDWRITTEN WORD RECOGNITION USING CONTINUOUS HIDDEN MARKOV MODELS AND STRUCTURAL FEATURES

By:

Mohammad Mehdi Haji

Recognizing handwritten words has been and still is one of the most challenging problems in Artificial Intelligence (AI). Words are rather complex patterns, having many variations in handwriting style. Despite the considerable progress achieved in recent years, performance of handwriting recognition systems is still far from human's both in terms of accuracy and speed.

A complete offline recognition system for Farsi handwritten words is presented. To the best of our knowledge, this work is the first to use continuous hidden Markov models with structural features to recognize Farsi handwritten words. Most parts of a complete recognition system are addressed. A new machine learning approach based on the naive Bayes classifier is developed for text segmentation. Four different algorithms for document image binarization are compared and contrasted. Different skew and slant correction algorithms are surveyed for handwritten documents, and the problem of multiple skews is dealt with in a two-stage process. The first stage corrects the global skew, and after extracting text lines, in the second stage, the skew of each line is corrected locally. Five different skeletonization algorithms are compared and contrasted with the main focus on preserving text characteristics. A simple and effective skeleton post-processing technique is also described. Most of the normalization methods are adaptive, meaning that they do not use any parameters to be set experimentally. Each word image is represented by a sequence of structural features. The features are independent of the baseline location, so the difficult and crucial problem of baseline detection is avoided. The recognition is performed by continuous hidden Markov models.

There is no publicly available dataset for Farsi handwritten word images, and it is not wise to compare different systems evaluated on different datasets. The executable version of training, recognition and evaluation modules of the system is provided in the thesis webpage, http://pasargad.cse.shirazu.ac.ir/~mhaji/handrec, so it can be trained and evaluated on different datasets. The proposed method achieves a maximum recognition rate of about 82% on a 100-word lexicon. The striking aspect of the recognition system is its excellent generalization performance, as in our experiments, the system trained with multi-font machine-printed word images could recognize handwriting.

Contents Page

Contents Page

Contents Page

List of Figures Page

List of Figures Page

List of Figures Page

CHAPTER 1

INTRODUCTION

1.1 General Characteristics of Farsi Script

Farsi and Arabic scripts have minor differences. From recognition point of view, Arabic script is a little more complicated, but similarities outweigh differences. Both scripts are written from right to left, and most letters are connected to the base line. As opposed to English, there is no lower or upper case in Arabic/Farsi. But the more distinguishing feature is that Arabic/Farsi texts whether machine-printed or handwritten are cursive, i.e. letters belonging to the same word are connected whenever possible. More precisely, all but six Farsi letters can be connected from both sides; the six letters 'د', 'ذ', 'ر', 'ز', 'ژ' and 'و' can be joined to the succeeding letter from the right side only. Thus, these letters cause discontinuity within the same word. This feature is in sharp contrast with Latin scripts in which texts can be written whether cursively or discretely. So, it is not surprising that Arabic/Farsi text recognition is more difficult than English, and even the problem of machine-printed text recognition is not yet completely solved for these two languages.

Table of Figure 1.1 depicts that every Arabic/Farsi letter can have up to four different shapes depending on the location of the letter within word. Farsi alphabet has 32 basic letters, with four letters 'پ', 'چ', 'ژ' and 'گ' more than Arabic. In Arabic/Farsi there are certain diacritics to indicate a difference in pronunciation and meaning from the same word when unmarked. For example, the Farsi word 'مرد' has two different pronunciation/meanings: 'مُرد' that means died, and 'مَرد' that means man. But the diacritics are not usually used in writing because in most cases the exact pronunciation/meaning can be inferred from the context. From recognition point of view, the diacritics are better not to be presented, because they may raise the recognition error. By looking at table of Figure 1.1, you can see that 18 out of the 32 Farsi letters have dots, appearing on above or below the baseline. More precisely, 10 letters have one dot, 3 have two dots and 5 have three dots. The number of dots of a certain letter does not change with its different forms, except for the letter 'ی' that has two dots in initial and middle forms and no dot in isolated and final forms. The Arabic/Farsi alphabet has sets of letters that the letters in each set are almost identical in the absence of dots; these sets are {ب, پ, ت, ث, ن}, {ج, چ, ح, خ}, {د, ذ}, {ر, ز, ژ}, {س, ش}, {ص, ض}, {ط, ظ}, {ع, غ} and {ف, ق}. Therefore, any erosion or deletion of the dots result in a misinterpretation, and so any denoising or skeletonization algorithm must take care of dots.

Character	Isolated	Initial	Middle	Final	Transliteration
Alef	آ	آ	ـا	ـا	a
Beh	ب	بـ	ـبـ	ـب	b
Peh	پ	پـ	ـپـ	ـپ	p
The	ت	تـ	ـتـ	ـت	t
Theh	ث	ثـ	ـثـ	ـث	Th
Jeem	ج	جـ	ـجـ	ـج	j
Cheh	چ	چـ	ـچـ	ـچ	ch
Heh	ح	حـ	ـحـ	ـح	h
Kheh	خ	خـ	ـخـ	ـخ	kh
Dal	د	د	ـد	ـد	d
Thal	ذ	ذ	ـذ	ـذ	th
Reh	ر	ر	ـر	ـر	r
Zeh	ز	ز	ـز	ـز	z
Zheh	ژ	ژ	ـژ	ـژ	zh
Seen	س	سـ	ـسـ	ـس	s
Sheen	ش	شـ	ـشـ	ـش	sh
Sad	ص	صـ	ـصـ	ـص	s
Zad	ض	ضـ	ـضـ	ـض	th
Tah	ط	طـ	ـطـ	ـط	t
Zah	ظ	ظـ	ـظـ	ـظ	z
Ain	ع	عـ	ـعـ	ـع	a
Ghain	غ	غـ	ـغـ	ـغ	gh
Feh	ف	فـ	ـفـ	ـف	f
Ghaf	ق	قـ	ـقـ	ـق	gh
Kaf	ک	کـ	ـکـ	ـک	k
Gaf	گ	گـ	ـگـ	ـگ	g
Lam	ل	لـ	ـلـ	ـل	l
Meem	م	مـ	ـمـ	ـم	m
Noon	ن	نـ	ـنـ	ـن	n
Waw	و	و	ـو	ـو	v
Heh	ه	هـ	ـهـ	ـه	h
Yeh	ی	یـ	ـیـ	ـی	y

Figure 1.1. The Farsi character set.
Each character can have up to four different shapes.

The above characteristics imply that each Arabic/Farsi word image consists of some connected components where each one represents a dot, connected dots, one letter or some connected letters. For example, the Farsi word 'باران' has 5 letters and 6 connected component, but the word 'کمد' has three letters and only one connected component.

Another aspect of Arabic and Farsi scripts which complicates the segmentation-based recognition techniques is that usually, and particularly for handwritten texts, succeeding letters overlap, and hence no vertical partitioning can exactly separate the letters from each other.

1.2 Methodology

The methods of handwritten and machine-printed text recognition can be divided into two categories (Amin, 1998): 1) Holistic strategies in which the recognition is globally performed on the whole representation of a word, so there is no need to segment a word to its individual characters. But it is necessary that we can segment the text lines into words, which is not always possible, because the intra-word space is sometimes greater than inter-word space; 2) Analytical strategies in which words are segmented either explicitly or implicitly. In the explicit segmentation, an attempt is made to isolate single letters which are then separately recognized (Vinciarelli, 2002); but in the implicit segmentation, the text (line or word) image is converted into a sequence of small size units (a sequence of observations) and the recognition is performed at this intermediate level rather than the word or character level.

Every holistic method uses a lexicon, a list of the allowed interpretations of the input word image. Usually the error rate of a holistic method increases with the lexicon size because the higher number of classes increases the probability of misclassification. Having a lexicon, the extracted feature vector of the input word is compared with feature vectors of all lexicon entries, and the entry having the maximum score (e.g. minimum distance or maximum probability) is assumed as the interpretation of the input. Holistic approaches have been successful when the lexicon is small, with a maximum size of a few hundred words. When the lexicon size increases, the number of compatible words for an input image rise and choosing the correct classification becomes difficult. Thus, the holistic approaches have been applied to such problems as postal address recognition or bank check reading where lexicon is limited and small. For general text recognition where the lexicon size is usually between 5000 and 10000, lexicon pruning techniques are effective because they have the double benefit of improving the recognition results and speeding up the system. For example when the input word image has no dot, all lexicon entries with no dot can be excluded from the list of alternatives. In (Zimmermann and Mao, 1999) an effective lexicon reduction technique based on key characters and word-length estimation is presented. Language models can also be used to reduce the number of alternatives.

Analytical methods perform implicit or explicit segmentation on the input image. In implicit segmentation, words are segmented into small units which are then transformed into a sequence of observation vectors. Each unit is usually a part of a letter, so that a number of successive units can belong to a single letter. On the other hand, in explicit segmentation, words are segmented directly into the single letters; which is usually fulfilled by a dynamic programming technique to find the optimal word hypotheses, thus explicit segmentation is more expensive than implicit segmentation. It is worth noting that Arabic/Farsi letters of the same font have different lengths, so word segmentation based on a fixed size width can't be applied, even for machine-printed words. Also according to Sayre's paradox (Vinciarelli, 2002): "a letter can not be segmented before having been recognized, and can not be recognized before having been segmented". It is clear that more segmentation error results in less recognition rate, and until now there is no method that is able to segment handwritten words exactly into letters. Therefore, explicit segmentation is more error prone and almost all successful handwritten recognition systems have used implicit segmentation.

After explicit segmentation, each letter can be individually recognized by a classifier which is usually an Artificial Neural Network (ANN) or Hidden Markov Model (HMM). Character recognition has been one of the most successful application areas of ANNs. The main advantages of an ANN over other classifiers is that it require less knowledge about the problem and being capable of implementing more complex partitioning of feature space. However, an ANN usually needs much training, and if a new class wants to be added later, the whole training process must be repeated.

Implicit segmentation results in a sequence of observations which is usually recognized by the HMM approach. HMMs are the prevalent technique in Automatic Speech Recognition (ASR), and in recent years they have proved to be very effective in handwritten recognition. A HMM can represent probability distribution over the sequence of data which has been used for its training, and it is not only being able to work on a sequence of small fragments, not necessarily corresponding to letters, but can also deal with noise in the sequence itself. A HMM assigns a probability to a sequence of observations which describes how probable the sequence is. Another advantage of HMMs is that their powerful theoretic framework limits the amount of heuristics to improve the system performance.

1.3 Feature Extraction

The extracted features can be either local or global. Global features, either structural (topological) or statistical, are usually used in holistic approaches. For example number of connected components, holes, ascenders and descenders are global structural features; coefficients of Fourier transform and invariant moments are global statistical features. Structural features do not depend on the writing style and can tolerate a high degree of variability, but they are not robust to noise and their extraction may be difficult. Statistical features, on the other hand, are easy to extract and robust, but they are not as stable as structural features with respect to cursive variability.

Analytical approaches usually use local features being extracted from small sliding windows. For example, percentage of foreground pixels within a window, foreground-background transition statistics, percentage of the foreground pixels in core, ascenders and descenders regions have been successfully used as local features.

1.4 Literature Survey

In (Khorsheed, 2000) two holistic approaches for recognition of handwritten Arabic words are presented. The first one, which uses global statistical features, transforms the word image into a normalized polar map, and then applies a 2D Fourier transform to extract features which are invariant to scaling, rotation and translation (displacement), and the recognition is simply based on Euclidean distance, i.e. the lexicon entry with minimum Euclidean distance to the input is returned as the answer. The second method utilizes structural feature vectors obtained from small strokes of the word skeleton. These vectors are then transformed into a sequence of observations that is fed to a HMM classifier. Khorsheed has surveyed the two possible models: a HMM for each word and a single HMM for all words.

Trenkle et al. (Trenkle et al., 1995) have presented a holistic method based on sliding neural networks for Arabic word recognition which scans continuously over a word image to search characters. The sliding neural network system uses networks in a two-stage approach. The first stage is to detect plausible locations of character centers, and at the second stage, the characters are recognized at those detected locations. During both stages, the neural network slides across the word image, producing a recognition signal at each location. Thus, there is no need to explicitly segment a word into its characters. By completing the second stage, the character hypotheses are combined by a dynamic programming algorithm which uses a pruned list of words to find the most probable word hypothesis. The recognition rate of the system is about 70% for a large lexicon of size 50,000 words.

In (Erlandson et al., 1996) a holistic approach for recognition of multi-font Arabic text is presented. The system computes a vector of structural features for each input word image which is then matched against a database of feature vectors of a lexicon by a vector matching algorithm. Like other systems of this kind, in the database there are several feature vectors corresponding to multiple fonts for each lexicon entry. In the training stage (building the database of feature vectors) noise models are also applied to word images, before feature extraction, to simulate low quality data, making the system robust to noise. The extracted features are dots, holes, junctions, endpoints, directional segments, directional cavities, descenders and intra-word gaps. By equipping the system with a lexicon pruner, a word recognition rate of 65% for a 48000 word lexicon was achieved, and the authors conclude that achieving a higher performance is very difficult.

In (Al-Yousefi and Udpa, 1992) statistical features and a quadratic Bayesian classifier are used to recognize Arabic characters. The striking aspect of the system is that a character is segmented into primary and secondary parts (dots and zigzags), thus reducing the number of classes from 28 (the number of basic letters of Arabic alphabet) to 18. The system uses a simple histogram-based method to explicitly segment a word into characters. A 9D feature vector extracted from normalized moments of the horizontal and vertical projections is used to classify primary characters, and a simple procedure classifies secondaries into one of the four classes (1 dot, 2 dots, 3 dots and zigzag). As expected, the segmentation method can not deal with handwritten text; the authors also didn't report the performance for machine-printed text. However, a recognition rate of 99.5% was reported for isolated forms of machine-printed characters of three different fonts and five different sizes.

In (Lu et al., 1999) a language-independent OCR system is introduced. Since segmentation at character or word level is problematic, the text lines are used as the basic unit for training and recognition, so there is no need to perform any segmentation. A 14-state left-to-right continuous HMM is employed to model each character, and a word model is then obtained by concatenating character models. Sets of statistical features are extracted from narrow vertical overlapping windows, and a Linear Discriminant Analysis (LDA) is applied to reduce the dimension of feature vector from 80 to 15. The system uses the same feature extraction, training and recognition modules for different scripts, but obviously, a separate language model is utilized for each script. The language models extracted from training corpora can improve recognition results of speech and text recognition systems by incorporating high level knowledge. The recognition process is a search for the most probable sequence of characters, given the input feature vectors, lexicon and language model. Since the classical Viterbi algorithm is slow when the language models are large, a multi-pass search algorithm is used instead. The system is trained for the Arabic, English and Chinese languages. By using character trigram and simple word unigram models for Arabic, a character error rate of between 0.8% and 4.7% is obtained for different test conditions.

In (Ahmed and Ward, 2000) an expert system for analysis and recognition of general symbols is introduced. The system uses structural pattern recognition techniques for modeling symbols by a set of straight lines, achieving high accuracy for explicitly segmented characters. A structural pattern recognition technique, by using a set of rules and grammars, describes relations between sub-patterns being able to build complex patterns. Their system stores some models for each symbol (an average of 97 models/symbol) and it is shown to be able to map similar styles of a symbol to the same representation. The main advantage of the system over a statistical pattern recognition technique is the ability to learn new symbols by simply adding their models to the system knowledge base. Generally, each statistical approach needs retraining for adding (recognizing) each new symbol (class).

In (Arica and Yarman-Vural, 2000) a technique is described to convert 2D information into a 1D observation sequence. Although there are 2D-HMMs for image analysis, but in practice, they are not as successful as 1D-HMMs, because of the large number of parameters which requires a large amount of training data. Therefore, 1D-HMMs are usually preferred even for analyzing 2D image data. The Arica's method extracts a set of directional skeletons of the binarized character by scanning the image matrix in various directions. These directional skeletons are then appended one after another to form a 1D observation sequence. By using a left-to-right discrete HMM as the recognizer, and about 200 samples per class for training, the system has achieved a recognition rate of 87-96% for English handwritten digits and letters.

In some systems (Amin and Mari, 1989) the vertical histogram of word image is used for explicit segmentation. The basic idea is that the histogram at connectivity points of a word has its least values. Thus, the word image separates into a number of segments which are then connected together to form the basic characters. Some systems extract key feature segments by tracing contour of the sub-words, and then identify the cut points in each segment. The baseline is important characteristic of Arabic/Farsi scripts; the connection point is where the normal thickness of the baseline changes. Based on this fact, Parhami and Taraghi (Parhami and Taraghi, 1981) have identified connection points of Farsi machine-printed words. In one recent study (Motawa et al., 1999), morphological operators have been applied to a word image to find singularities and regularities. Singularities represent the start, end or transition to another character; while regularities contain the required information for connecting a character to its successor, which means that regularities are promising candidates for segmentation. The boundary pixels or the contour, which provide important information of an object, can also be used for word segmentation. For example, a transition from a column having all its black pixels within the baseline boundaries to an unlike column corresponds to a connection point. Segmentation can also be performed by tracing the outer contour of a word and calculating the distance between the extreme points of intersection of the contour with a vertical line (El-Sheikh and Guindi, 1988).

In (Almuallim and Yamaguchi, 1987) a structural recognition method for Arabic handwritten words is introduced. The system is composed of four phases. In the first step words are thinned and the baseline is detected. Since segmenting a word into individual letters is difficult and error-prone, the words are segmented into strokes. The extraction of a stroke is done by finding its start point and then following the curve until an end point is reached. The first unvisited start point is found by a search for a black point from right to left around the baseline. The algorithm attempts to extract the strokes in the same order that have been written. Moreover, adjacent strokes which form a loop are also detected, because loops are unique features of some letters, simplifying the task of recognition. The strokes are then further classified using their topological and geometrical properties. Finally, based on their relative positions, strokes are combined in several steps into equivalent character string of the input word image. The stroke extraction algorithm can not handle all situations, resulting in incorrect segmentation of words, and the system failures. Also, like all structural analysis methods, noise can highly degrade the performance; for example, spurious branches in the image skeleton, caused by noise, confuse extraction and consequently combination of strokes. Another drawback is that their system does not utilize any learning method for parameter adjustment; so they have to be manually adjusted on a set of test images. The system has been tested on a set of 400 word images and a maximum recognition rate of about 91% has been reported. It must be Almuallim and Yamaguchi did not use a lexicon, but they evaluated the average word rather than character recognition rate. If we have the average character recognition rate R_c, a rough estimate of the average word recognition rate R_w is $(R_c)^5$, based on the fact that the average length of a word is 5. For example, when R_c is 90%, R_w is only about 60%. A better estimate for R_w is made by the following formula:

$$R_w = \sum_{l=1} \frac{N_l}{N}(R_c)^l \tag{1.1}$$

where N is the size of language lexicon, and N_l is the number of words of length *l*.

Dehghan et al. (Dehghan et al., 2001) present a holistic system for recognition of Arabic/Farsi handwritten words using discrete HMMs and Kohonen self-organizing vector quantization. After the preprocessing step which includes binarization, noise removal, slant correction, baseline and stroke width estimation, a stroke width compensation step is applied to have the stroke width of at least three pixels wide to ensure proper contour generation. Then, a word image is represented by the chain-code, and the histogram of chain-code directions of the image strips, scanned from right to left by a sliding window, is used as feature vector. In order to limit the number of observation symbols for discrete HMM training, the feature space must be quantized into a set of codebook vector; the weights of Kohonen self-organizing feature map (SOFM) are used as the codebook vectors. Since without much training data, HMM parameters, and specially the observation symbol probabilities, are poorly estimated, they have to be smoothed after training. In this system, a separate HMM is trained for each word, and the neighborhood information preserved in the SOFM is used for smoothing the observation probabilities of the HMMs, proved to be very effective. With a 198-word lexicon, a recognition rate of 65% is achieved by the system.

Both discrete and continuous HMMs have been successfully used for handwritten recognition, but due to their lower computation costs, discrete HMMs are more appealing. However, discrete HMMs inherently suffer from some problems (Rabiner, 1989) such as quantization error caused by quantizing of input vectors into a limited set leading to loss of information and recognition deterioration. To obviate these problems, Dehghan et al. (Dehghan et al., 2001) used fuzzy vector quantization instead of self-organizing vector quantization of the previous system. The fuzzy c-means clustering (FCM) algorithm is used to generate a fuzzy codebook, so a sequence of feature vectors extracted from the input image frames is now mapped to an observation sequence of membership vectors instead of a sequence of single values in the case of conventional discrete HMMs. Thus, a modified version of Baum-Welch re-estimation algorithm is used for training. The system performance is slightly improved by using FCM, with a recognition rate of 67.2% on the same dataset (198-word lexicon).

CHAPTER 2

TEXT SEGMENTATION

2.1 Introduction

A text segmentation algorithm aims at detecting text regions in an image. Identifying text areas in images have wide applications in document image analysis and understanding, image compression and content-based image retrieval. In document image binarization (Liu and Srihari, 1997) and skew correction (Avanindra and Subhasis Chaudhuri, 1997) algorithms, it is often necessary to remove non-text items from the input image because they usually require predominant text area to have an accurate estimate of text characteristics. Paper text is still one of the main sources of information and it is clear that huge amount of such valuable data in the paper form, makes their updating and retrieval much difficult. Thus, there is a need to convert the text from paper to electronic format. This task is usually done by an OCR engine and text extraction is an essential component in the page segmentation module of the engine (Wu et al., 1999).

Text segmentation also has applications in training-based image compression algorithms such as Vector Quantization (VQ), which need to classify the data into statistically consistent parts, and thereafter use an appropriate codebook for each part (Gersho and Gray, 1992). The text in natural images and video frames such as street signs, vehicle license plates, billboards, writing on shirts, sport scores, time and location stamps, is a powerful source of knowledge in building image and video indexing and retrieval systems (Chen et al., 2001). This kind of text also provides useful content information for video understanding and automatic navigation systems.

Due to the wide range of applications, numerous methods for text segmentation also referred to as text detection have been proposed. Some of them require binary input images; which restricts their application when the text is embedded in an image with a complex background, because binarization techniques usually produce poor results for complicated images (Wu et al., 1997). On the other hand, some methods also use the color information to detect text areas; color information can be helpful, but it is not available in all situations. Moreover, for a human observer, intensity information is enough to segment the text areas. Therefore, most methods perform text segmentation on gray-scale images; even if a color input image is available, it is first converted to gray-scale (Wu et al., 1999; Chen et al., 2001).

The main text segmentation methods in the literature can be classified into connected component-based (Fletcher and Kasturi, 1988), edge-based (Pietikäinen and Okun, 2001; Jie Xi et al., 2001) and texture-based methods (Li and Gray, 1998). Connected component-based ones are bottom-up approaches that work by grouping small components satisfying several heuristic constraints into successively larger components to form text lines and columns. They are relatively independent of changes in text size and orientation, but having difficulties with complex images with non-uniform backgrounds, because in such cases thresholding techniques can not produce the expected binary image, for example, if a text string touches a graphical object in the original image, they may form one connected component in the resultant binary image.

The basic idea of the edge-based algorithms is that the edges of text symbols are typically stronger than those of noise, textured-background and other graphical items (Yuan and Tan, 2000; Chen et al., 2001; Jie Xi et al., 2001). In these top-down techniques, a binary edge image is first generated using an edge detector, and then adjacent edges are connected by applying morphological operations or other algorithms such as run-length smoothing (Jie Xi et al., 2001). Connected components of the resultant image are the candidate text regions, as each one represents either several merged lines or a graphical item. Then, these regions are decomposed into smaller regions by analyzing their vertical and horizontal projection profiles, and finally each of these small regions satisfying certain heuristic constraints is labeled as text. Edge-based methods are fast and can detect text in complex backgrounds but are restrictive to detect only horizontally or vertically aligned text strings.

Text segmentation can also be taught of as a special case of texture segmentation in which characters correspond to texels. By treating text as a distinct texture, a texture segmentation algorithm can be applied to separate them. In texture-based methods the input image is usually considered as a composite of two (text and non-text) or three (text, picture and background) texture classes. Many segmentation algorithms employ a classification window (block) of a certain size in the hope that all or majority of pixels in the window belong to the same class (Choi and Baraniuk, 2001). Thereafter, a classification algorithm can be used to label each window in the feature space. For example, in (Deng and Latifi, 2000) the number of classes is two, and a 2-means classification is used to classify each block of the image as text or non-text according to its local energy in the wavelet transform domain. By using a 3-means clustering in (Wu et al., 1999) each image pixel is labeled as text, picture or background according to a 9-D feature vector based on Gaussian filtering. A large number of statistical and geometrical features have been proposed for texture segmentation such as features of co-occurrence matrix, spatial gray-level dependency matrix (Ohya et al., 1994), the Fourier power spectrum, moments of wavelet coefficients (Unser, 1995), Gaussian filters (Wu et al., 1999), Gabor filters (Jain and Farrokhnia, 1991), Voronoi tessellation (Tuceryan and Jain, 1990). Among these, wavelet based features are of most interest. The wavelet transform has become a very effective tool in texture segmentation and classification due to its multi-resolution properties. It provides a powerful transform domain for modeling images that are well characterized by their edges.

In texture-based methods, irrespective of the employed features, the size of classification window is crucial. A large window results in robust segmentation in homogeneous regions but poor segmentation along the boundaries between regions. On the other hand, classification using small windows is not reliable because small amount of data (pixels) do not provide sufficient statistical information.

All of the methods have difficulties with multi-size text strings and text-like texture areas. The former causes false negatives, while the latter results in false positives. The problem of detecting text strings of different sizes can be addressed by pyramid approaches (Wu et al., 1997) to some extent, while reducing false positives needs more sophisticated approaches; for example in (Chen et al., 2001) a Support Vector Machine (SVM) is utilized for this task. Despite the many efforts spent on the text segmentation problem, there is no general method to detect arbitrary text strings; because in the most general form, detection must be insensitive to noise, background model and lighting conditions. Also, it must be invariant to text language, color, size, font and orientation even in a same image.

The literature on text segmentation is extensive but there appears to be very little appropriate literature on using machine learning techniques on this subject. A text segmentation algorithm should have adaptation and learning capability, but a learner usually needs much time and training data to achieve satisfactory results, which restricts its practicality. To overcome these problems, a simple procedure for generating training data from manually segmented images is presented, and then a Naive Bayes Classifier (NBC), which is fast both in training and application phase, is applied. It will be shown that surprisingly excellent results can be obtained by this simple classifier.

2.2 Naive Bayes Classifier

The Naive Bayes Classifier (NBC) is applicable to learning tasks where each instance is described by a conjunction of attribute values and a target function which takes a value from a finite set *V*. A set of training examples for the target function is provided, a new instance described by the attribute values (a_1, a_2, …, a_n) is then presented, and the learner is asked to predict the target value or classification. The Bayesian approach to classify the new instance is to assign the most probable that is the Maximum A Posteriori (MAP) hypothesis, given the attribute values that describe the instance (Mitchell, 1997).

$$v_{\mathrm{MAP}} = \arg\max_{v_j \in V} P(v_j \mid a_1, a_2, ..., a_n) \tag{2.1}$$

where v_{MAP} is the most probable target value. Using Bayes' theorem Equation (2.1) can be written as follows:

$$\begin{aligned} v_{\mathrm{MAP}} &= \arg\max_{v_j \in V} \frac{P(a_1, a_2, ..., a_n \mid v_j) P(v_j)}{P(a_1, a_2, ..., a_n)} \\ &= \arg\max_{v_j \in V} P(a_1, a_2, ..., a_n \mid v_j) P(v_j) \end{aligned} \tag{2.2}$$

Using training data the two terms in Equation (2.2) must be calculated. It is very easy to estimate each $P(v_j)$ by counting the frequency of occurrence of each target value in the training data. However, estimating the different $P(a_1, a_2, …, a_n)$ terms in this way is not possible unless a huge set of training data is available. In order to make the classifier much more practical and computationally efficient, the simplifying assumption that the attribute values are conditionally independent given the target value is used. This independence assumption implies that:

$$P(a_1, a_2, ..., a_n \mid v_j) = \prod_i P(a_i \mid v_j) \tag{2.3}$$

Substituting Equation (2.3) into Equation (2.2) results in the approach used by NBC, given by Equation (2.4):

$$v_{\mathrm{NB}} = \arg\max_{v_j \in V} P(v_j) \prod_i P(a_i \mid v_j) \tag{2.4}$$

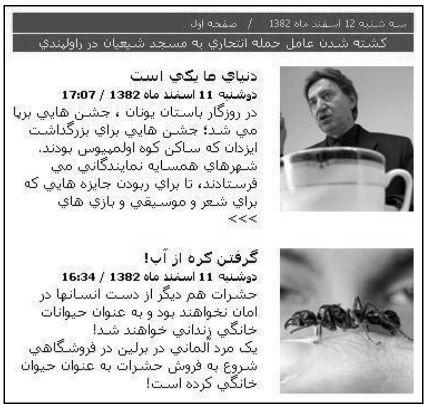
where v_{NB} denotes the target value output given by the NBC.

Despite the fact that the independence assumption is often violated in practice, NBC has shown itself a serious competitor with more sophisticated classifiers. This classifier is shown to be very effective in many practical domains such as text categorization and medical diagnosis (Mitchell, 1997). NBC has several distinctive features which make it suitable for the text segmentation task. First, it is a probabilistic classifier, i.e. it outputs posterior probability distribution over classes. In this work, text segmentation is treated as a two-class classification task, and a probabilistic classifier is appropriate here since it assigns a score to each instance expressing the degree to which that instance is thought to be positive. The second advantage of NBC is that the learning task is not sensitive to the relative number of training instances in the positive (text) and negative (non-text) classes. It is only important to have non-zero probability estimates in Equation (2.4). Lastly, in naive Bayes methods, learning time is short and actually linear in the number of training examples making it suitable for real-time learning. From Equation (2.4) it is clear that learning is simply done through counting the frequency of various data combinations within the training examples.

2.3 Training Data Generation

A large training set facilitate the task of learning, tuning and comparing various classifiers. A simple procedure was implemented to generate a large set of training data from a small set of hand-segmented images. A set of eight images, selected from a wide category, was used for extracting the training data. The images contained both English handwritten and machine-printed texts with different fonts, sizes and intensity values. Furthermore, since the method is intended to be language-independent, two Farsi document images were also included. For each training image a binary mask is created manually. The mask contains white rectangles correspond to the text strings (Figure 2.1).

The proposed algorithm is a block-based segmentation which use features in Discrete Cosine Transform (DCT) domain. It is observed that DCT-18 features are different for text and non-text textures (Chaddha et al., 1995), so the same features are used in this work. These are 18 elements of an 8x8 transformed image block with indices: 4, 5, 6, 12, 13, 14, 20, 21, 22, 44, 45, 46, 52, 53, 54, 60, 61 and 62 when counting coefficients at 1 and going line after line, denoted by A_1 to A_{18}. The procedure used to generate training data file is outlined in Algorithm 2.1, where I(i1:i2, j1:j2) notation is used to refer to the sub-image specified by the rectangle with (j1,i1) top-left corner and (j2,i2) bottom-right corner. The vertical sampling period, denoted by vp, was chosen to be 4 and horizontal sampling period, denoted by hp, was set to 8 in this work.



سه شنبه 12 اسفند ماه 1382 / صفحه اول

کشته شدن عامل حمله انتحاري به مسجد شيعيان در راولپندي

دنياي ما يکي است

دوشنبه 11 اسفند ماه 1382 / 17:07

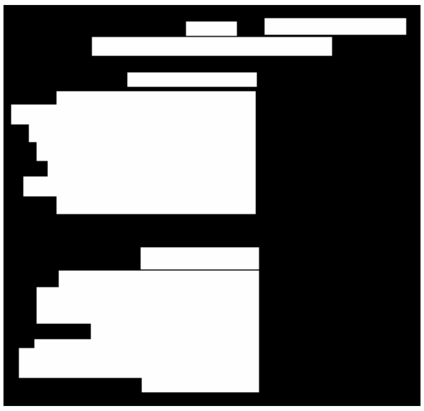
در روزگار باستان يونان ، جشن هايي برپا
مي شد؛ جشن هايي براي بزرگداشت
ايزدان که ساکن کوه اولمپيوس بودند.
شهرهاي همسايه نمايندگاني مي
فرستادند، تا براي ربودن جايزه هايي که
براي شعر و موسيقي و بازي هاي
<<<

گرفتن کره از آب!

دوشنبه 11 اسفند ماه 1382 / 16:34

حشرات هم ديگر از دست انسانها در
امان نخواهند بود و به عنوان حيوانات
خانگي زنداني خواهند شد!
يک مرد آلماني در برلين در فروشگاهي
شروع به فروش حشرات به عنوان حيوان
خانگي کرده است!

(a) A part of a Farsi document



(b) The text mask of (a)

Figure 2.1. A document image and its corresponding text mask.

```
for each training image I and its corresponding mask M
{
  for i = 0 : vp : $8*\lfloor \mathrm{rows(I)}/8 \rfloor - 1$
  {
    for j = 0 : hp : $8*\lfloor \mathrm{columns(I)}/8 \rfloor - 1$
    {
      [$A_1$ $A_2$ $A_3$ … $A_{18}$] = dct18( I(i:i+7, j:j+7) )
      if M(i:i+7, j:j+7) has more white than black pixels
      {
        /* it is a positive training instance */
        write [$A_1$ $A_2$ $A_3$ … $A_{18}$ 1] to the output file.
      }
      else
      {
        /* it is a negative training instance */
        write [$A_1$ $A_2$ $A_3$ … $A_{18}$ 0] to the output file.
      }
    }
  }
}
```

Algorithm 2.1 The procedure for generating training data file for the 'IsText' concept.

For small squares, such as 8x8, the DCT is more efficiently computed by the DCT transform matrix T given by Equation (2.5) for an NxN block.

$$T_{pq} = \begin{cases} \dfrac{1}{\sqrt{N}} & p = 0, 0 \le q \le N-1 \\ \sqrt{\dfrac{2}{N}} \cos \dfrac{\pi(2q+1)p}{2N} & 1 \le p \le N-1, 0 \le q \le N-1 \end{cases} \tag{2.5}$$

Then, the 2D-DCT of the square matrix A can be computed by $T \times A \times T'$.

	continuous	discrete
A_1	(-inf,-15.8]	S2
	(-15.8,-0.7]	S1
	(-0.7,0.8]	CE
	(0.8,16.1]	B1
	(16.1,inf)	B2

	continuous	discrete
A_2	(-inf,-13.1]	S2
	(-13.1,-0.4]	S1
	(-0.4,0.3]	CE
	(0.3,11.3]	B1
	(11.3,inf)	B2

	continuous	discrete
A_3	(-inf,-9.5]	S2
	(-9.5,-0.3]	S1
	(-0.3,0.4]	CE
	(0.4,11.4]	B1
	(11.4,inf)	B2

	continuous	discrete
A_4	(-inf,-11.5]	S2
	(-11.5,-0.5]	S1
	(-0.5,0.4]	CE
	(0.4,11.3]	B1
	(11.3,inf)	B2

	continuous	discrete
A_5	(-inf,-10]	S2
	(-10,-0.3]	S1
	(-0.3,0.2]	CE
	(0.2,9.4]	B1
	(9.4,inf)	B2

	continuous	discrete
A_6	(-inf,-6.3]	S2
	(-6.3,-0.3]	S1
	(-0.3,0.2]	CE
	(0.2,6.6]	B1
	(6.6,inf)	B2

	continuous	discrete
A_7	(-inf,-10.6]	S2
	(-10.6,-0.4]	S1
	(-0.4,0.3]	CE
	(0.3,8.5]	B1
	(8.5,inf)	B2

	continuous	discrete
A_8	(-inf,-7.3]	S2
	(-7.3,-0.2]	S1
	(-0.2,0.2]	CE
	(0.2,6.2]	B1
	(6.2,inf)	B2

	continuous	discrete
A_9	(-inf,-5.2]	S2
	(-5.2,-0.2]	S1
	(-0.2,0.2]	CE
	(0.2,4.8]	B1
	(4.8,inf)	B2

	continuous	discrete
A_{10}	(-inf,-4.6]	S2
	(-4.6,-0.2]	S1
	(-0.2,0.2]	CE
	(0.2,4.3]	B1
	(4.3,inf)	B2

	continuous	discrete
A_{11}	(-inf,-3.3]	S2
	(-3.3,-0.1]	S1
	(-0.1,0.2]	CE
	(0.2,3.7]	B1
	(3.7,inf)	B2

	continuous	discrete
A_{12}	(-inf,-3.4]	S2
	(-3.4,-0.2]	S1
	(-0.2,0.2]	CE
	(0.2,2.9]	B1
	(2.9,inf)	B2

	Continuous	discrete
A_{13}	(-inf,-3.4]	S2
	(-3.4,-0.1]	S1
	(-0.1,0.1]	CE
	(0.1,3.3]	B1
	(3.3,inf)	B2

	continuous	discrete
A_{14}	(-inf,-2]	S2
	(-2,-0.1]	S1
	(-0.1,0.1]	CE
	(0.1,2]	B1
	(2,inf)	B2

	continuous	discrete
A_{15}	(-inf,-2]	S2
	(-2,-0.1]	S1
	(-0.1,0.1]	CE
	(0.1,2]	B1
	(2,inf)	B2

	Continuous	discrete
A_{16}	(-inf,-2]	S2
	(-2,-0.2]	S1
	(-0.2,0.2]	CE
	(0.2,3]	B1
	(3,inf)	B2

	continuous	discrete
A_{17}	(-inf,-2.3]	S2
	(-2.3,-0.1]	S1
	(-0.1,0.1]	CE
	(0.1,2.4]	B1
	(2.4,inf)	B2

	continuous	discrete
A_{18}	(-inf,-2.2]	S2
	(-2.2,-0.1]	S1
	(-0.1,0.2]	CE
	(0.2,2.3]	B1
	(2.3,inf)	B2

Figure 2.2. The discretization rules for the DCT-18 features.

Using the above procedure, about 100,000 training instances were generated from the eight images, but there was no need for such a large amount of data because it was observed that only a small fraction of these data provides reasonable estimates for the terms of Equation (2.4). So in order to reduce the computational cost, 10,000 instances were selected randomly and used for the purpose of learning.

The NCB is a discrete classifier, and hence each attribute value must be converted to discrete form. For this purpose, each continuous attribute value was converted to only five discrete values: ‘S2’, ‘S1’, ‘ZE’, ‘B1’ or ‘B2’ (respectively for ‘very small’, ‘small’, ‘around zero’, ‘big’ and ‘very big’). The discretization rules were set in such a way to have approximately 2000 instances in each of the 5 bins for each attribute value. Therefore a different set of rules is used for each of the 18 attributes as given in Table of Figure 2.2.

2.4 Training

For the 'IsText' concept, let v_1 = 'Yes' and v_2 = 'No'. The evaluation of conditional probabilities is carried out on the discretized training data and the results are given in Table of Figure 2.3. Since all estimated probabilities are non-zero, no attempt is made to smooth them. When NBC is used, no conditional probability is allowed to be zero because only a zero value causes the estimate of zero in Equation (2.3) which is a biased underestimate of the probability. The *m*-estimate of probability (Mitchell, 1997) is simple and effective technique to avoid zero probability estimates.

It must be mentioned that the probability estimates of a NBC can also be acceptable if some of the underlying independence assumptions are violated. It is well-known that NBC is the optimal classifier when the independence assumptions are satisfied, but Rish (Rish, 2001) has shown that NBC also works well for functionally dependent features. The optimality of NBC has proved for some problems that have a high degree of feature dependencies such as disjunctive and conjunctive concepts (Domingos and Pazzani, 1997). By analyzing the impact of distribution entropy on the classification error, Rish has demonstrated that NBC is a good performer for low-entropy (almost deterministic) feature distributions.

2.5 Classification

No prior information about the source image is assumed, and so $P(v_1) = P(v_2) = 0.5$. Therefore, according to Bayes' rule:

$$P(\text{Text}) = \frac{P(a_1 \mid v_1)P(a_2 \mid v_1)...P(a_{18} \mid v_1)}{P(a_1 \mid v_1)P(a_2 \mid v_1)...P(a_{18} \mid v_1) + P(a_1 \mid v_2)P(a_2 \mid v_2)...P(a_{18} \mid v_2)} \tag{2.6}$$

The usual decision criterion (Equation (2.4)) suggests selecting the class with the highest posterior probability, or if *P*(Text) exceeds 0.5 the input block should be labeled as text.

P(A_1|V)		V	
		Yes	No
A_1	S2	0.3199	0.0938
	S1	0.1496	0.2496
	CE	0.0687	0.3212
	B1	0.1369	0.2462
	B2	0.3250	0.0893

P(A_2|V)		V	
		Yes	No
A_2	S2	0.3116	0.0821
	S1	0.1656	0.2458
	CE	0.0490	0.3475
	B1	0.1428	0.2274
	B2	0.3309	0.0972

P(A_3|V)		V	
		Yes	No
A_3	S2	0.3228	0.1018
	S1	0.1513	0.2661
	CE	0.0566	0.3112
	B1	0.1654	0.2329
	B2	0.3038	0.0881

P(A_4|V)		V	
		Yes	No
A_4	S2	0.3454	0.0635
	S1	0.1384	0.2568
	CE	0.0442	0.3333
	B1	0.1291	0.2869
	B2	0.3429	0.0595

P(A_5|V)		V	
		Yes	No
A_5	S2	0.3459	0.0684
	S1	0.1344	0.2390
	CE	0.0416	0.3576
	B1	0.1304	0.2674
	B2	0.3478	0.0677

P(A_6|V)		V	
		Yes	No
A_6	S2	0.3323	0.0771
	S1	0.1386	0.2522
	CE	0.0448	0.3191
	B1	0.1403	0.2822
	B2	0.3440	0.0694

P(A_7|V)		V	
		Yes	No
A_7	S2	0.3342	0.0584
	S1	0.1359	0.2464
	CE	0.0473	0.3578
	B1	0.1166	0.2666
	B2	0.3659	0.0709

P(A_8|V)		V	
		Yes	No
A_8	S2	0.3387	0.0572
	S1	0.1367	0.2585
	CE	0.0395	0.3655
	B1	0.1209	0.2515
	B2	0.3643	0.0673

P(A_9|V)		V	
		Yes	No
A_9	S2	0.3412	0.0578
	S1	0.1318	0.2795
	CE	0.0452	0.3174
	B1	0.1314	0.2733
	B2	0.3503	0.0720

P(A_{10}|V)		V	
		Yes	No
A_{10}	S2	0.3537	0.0610
	S1	0.1359	0.2941
	CE	0.0450	0.3212
	B1	0.1221	0.2608
	B2	0.3433	0.0629

P(A_{11}|V)		V	
		Yes	No
A_{11}	S2	0.3697	0.0553
	S1	0.1287	0.3015
	CE	0.0404	0.3358
	B1	0.1192	0.2505
	B2	0.3421	0.0569

P(A_{12}|V)		V	
		Yes	No
A_{12}	S2	0.3473	0.0623
	S1	0.1285	0.2623
	CE	0.0570	0.3667
	B1	0.1206	0.2445
	B2	0.3465	0.0642

P(A_{13}|V)		V	
		Yes	No
A_{13}	S2	0.3543	0.0618
	S1	0.1295	0.3151
	CE	0.0334	0.2672
	B1	0.1380	0.2952
	B2	0.3448	0.0606

P(A_{14}|V)		V	
		Yes	No
A_{14}	S2	0.3609	0.0457
	S1	0.1177	0.2782
	CE	0.0385	0.3523
	B1	0.1143	0.2738
	B2	0.3687	0.0500

P(A_{15}|V)		V	
		Yes	No
A_{15}	S2	0.3571	0.0584
	S1	0.1105	0.2994
	CE	0.0408	0.2738
	B1	0.1280	0.3047
	B2	0.3636	0.0637

P(A_{16}|V)		V	
		Yes	No
A_{16}	S2	0.3719	0.0853
	S1	0.0968	0.2168
	CE	0.0570	0.3597
	B1	0.1344	0.2714
	B2	0.3400	0.0669

P(A_{17}|V)		V	
		Yes	No
A_{17}	S2	0.3495	0.0661
	S1	0.1280	0.2894
	CE	0.0427	0.3057
	B1	0.1346	0.2818
	B2	0.3452	0.0570

P(A_{18}|V)		V	
		Yes	No
A_{18}	S2	0.3355	0.0669
	S1	0.1361	0.3142
	CE	0.0488	0.3324
	B1	0.1354	0.2265
	B2	0.3442	0.0601

Figure 2.3. The conditional probabilities for the 'IsText' concept.

However, there is no justification for such a decision criterion, and especially when the probability estimates are inaccurate. In (Lachiche and Flach, 2003) it is shown that if the NBC decision criterion is treated as an additional model parameter, which has to be learned from the training data, rather than a fixed threshold, significant improvements will result.

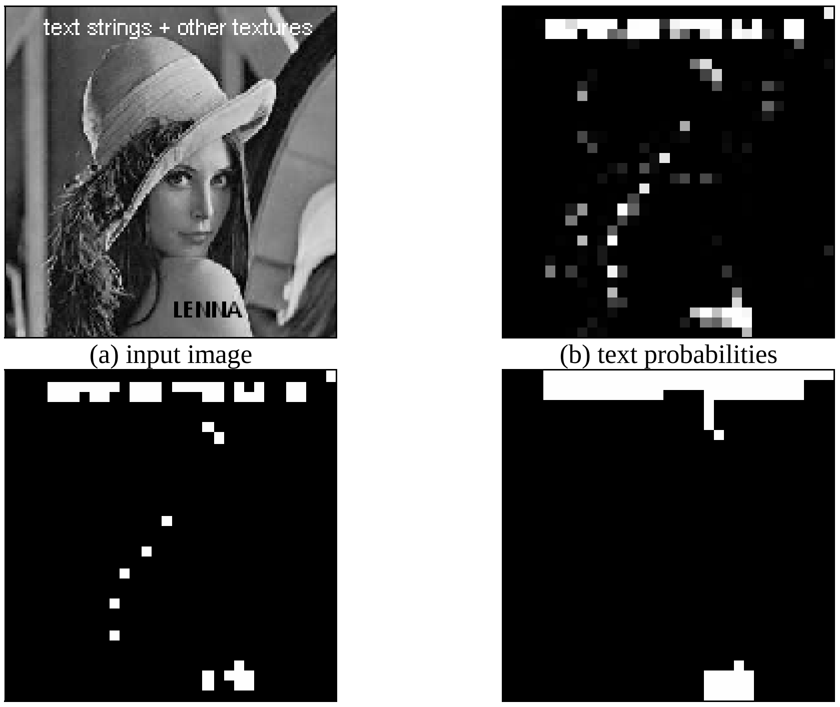
After classification, the image is post-processed by morphological operations to fill small black (non-text) holes within white (text) areas in order to reduce false negatives. In the classification phase, if a high decision threshold is selected (rather than 0.5) for the text class, the number of false positives (the block mistakenly marked as text) is obviously reduced, because only almost confident text blocks are classified as text. The threshold of 0.8 has worked well in our experiments, so to classify an 8x8 block of image, the DCT-18 features by Equation (2.5) is evaluated, and then their nominal equivalents are computed according to the rules of Table of Figure 2.2. Lastly, P(Text) is evaluated using Equation (2.6) and the conditional probabilities of Figure 2.3; if P(Text) exceeds 0.8, the input block is classified as text. This way a binary image is formed, with white pixels for text and black pixels for non-text areas. In order to improve the segmentation accuracy, this image should be post-processed.

2.6 Postprocessing

The post-processing step is based on the following assumptions: 1) the input image has more false negatives than false positives and 2) text areas are usually large and do not contain non-text areas (holes). In the first step, all isolated white pixels (without any white 8-neighbor) are removed, and then the morphological closing (dilation followed by erosion) with a 3x3 rectangular structuring element is applied.

In order to show its capabilities, the proposed text segmentation and post-processing algorithm is applied to the gray-scale image of Figure 2.4(a) which contains two texts of different colors and other textures. The output of naive Bayes classification is given in the image of Figure 2.4(b); each small square shows the probability that the corresponding square in the input image is thought to be text. Thresholding this image at 0.8 results in the binary image of Figure 2.4(c) having less false positives. The final text mask obtained by the post-processing step is given in the image of Figure 2.4(d).

The above experiment shows that the proposed method is not very sensitive to non-uniform background and works well if the text is darker or lighter than the background. In contrast, many existing approaches assume that background is uniform, showing poor performance when this assumption is not satisfied.



(a) input image (b) text probabilities

(c) image (b) thresholded at 0.8 (d) image (c) after post-processing

Figure 2.4. Applying the proposed text segmentation and postprocessing algorithm to an image with complex background.

CHAPTER 3

BINARIZATION

3.1 Introduction

Since many vision algorithms and operators only handle two-level (binary) images, binarization (thresholding) is a major step in such algorithms to convert gray-scale images into binary images. In binarization algorithms, a threshold or a threshold surface is usually computed first and then if a pixel has a higher intensity than that threshold or value of the threshold surface in that point, it is labeled as foreground (object); otherwise it is labeled as background. Due to the fact that binarization is usually applied in primary steps of a vision algorithm, say a recognition problem, and its result greatly influences the performance of the whole system, from the early days of automatic image processing much attention is devoted to this task. Binarization is challenging for gray-level images with poor contrast, strong noise and variable modalities in histograms, and it is still a difficult problem in vision.

Thresholding methods are divided into two classes: global and local. In global methods a single threshold is computed and applied to the whole image. Among many proposed global thresholding algorithms, Otsu's statistical method (Otsu, 1979), Tsai's moment-preserving method (Tsai, 1985) and Kapur et al.'s entropy method (Kapur et al., 1985) are satisfactory, and since Otsu's method is fast and easy to implement, perhaps it is the most widely used. Obviously global techniques can not produce satisfactory result when the gray-scale input image has non-uniform shading or its histogram is multi modal. Local (adaptive) thresholding algorithms, in contrast, use a separate threshold for each pixel or a small group of neighboring pixels based on the information contained in a neighborhood. In comparison with global methods, local algorithms usually involve more computation and so they are slower when running on a single-processor computer. A local algorithm is better suited for parallel processing and dealing with large and high resolution images which can not be completely kept in memory; they are usually superior in extracting characters with uneven gray-levels because of adaptation to local image properties, but do not necessarily yield better recognition results because they often do not preserve character stroke connectivity. Of the local algorithms, Niblack's method (Niblack, 1989) is simple and very effective and according to one experiment (Trier and Taxt, 1995) it is the best operator when the goal is character recognition.

General purpose thresholding methods such as Otsu's and Niblack's are not aware of the fact that the image being processed is a document image that has some special features, and they can not use this valuable information. Therefore researchers have developed algorithms specially designed for document image binarization. In (Liu and Srihari, 1997) a texture feature based thresholding algorithm is introduced to cope with images with complex patterns; In (Wu and Matmatha, 1998) a simple and effective method is proposed to separate text from textured, hatched or shaded background. A document binarization method for low-quality camera images is proposed in (Seeger and Dance, 2001).

3.2 The Otsu's Method

This is one the most widely used global thresholding techniques in machine vision. Although this method is not specially designed for document image binarization, for clean document image with simple backgrounds, it produces satisfactory results. As opposed to some algorithms which need a priori knowledge about the number of peaks in histogram, Otsu's method is completely automatic and it does not need any user defined parameter. It can also be used in more sophisticated binarization algorithms. This method selects the threshold based on the minimization of the within-group variance of the two groups of pixels separated as a result of a global threshold. In order to evaluate the threshold, the probabilistic histogram of the image must be computed first. In the probabilistic histogram P, the value of P(i) represents the probability of i'th gray level in that image. For ordinary 8-bit gray level images, obviously there are 256 levels, so the values of P(0), P(1), …, P(255) must be evaluated. This is performed using the following formula:

P(i) = (number of pixels with gray level value of i) / (total number of pixels). (3.1)

If the histogram is bimodal, the best threshold is the value that separates the two modes of P from each other. If so, each threshold t determines a variance for the group of values that are less that or equal to t and a variance for the group of values greater than t. Otsu suggested that best threshold is that one which minimizes the weighted sum of within-group variances. Variance is a measure of homogeneity. A group with high variance will have low variance and a group of low homogeneity will have high variance. Therefore the criterion suggested by Otsu emphasizes high group homogeneity. An equal criterion is a dividing that maximizes the resulting squared differences between the group means which is related to the between-group variances. Due to the fact that the sum of within-group variances and between-group variances is a constant, both criteria cause the same result.

Having evaluated P, the best threshold is obtained as follows. Let s_W^2 be the weighted sum of group variances, that is, the within-group variance. Let $s_1^2(t)$ be the variance for the group with values less than or equal to *t*, and $s_2^2(t)$ be the variance for the group with values greater than *t*. Let $q_1(t)$ be the probability for the group with values less than or equal to *t* and $q_2(t)$ be the probability for the group with values greater than *t*. Let $m_1(t)$ be the mean of first group and $m_2(t)$ be the mean of second group. The within-group variance is defined by:

$$s_W^2(t) = q_1(t).s_1^2(t) + q_2(t).s_2^2(t) \tag{3.2}$$

Where

$$q_1(t) = \sum_{i=0}^{t} \mathrm{P}(i) \tag{3.3}$$

$$q_2(t) = \sum_{i=t+1}^{255} \mathrm{P}(i) \tag{3.4}$$

$$m_1(t) = \sum_{i=0}^{t} i.\mathrm{P}(i) / q_1(t) \tag{3.5}$$

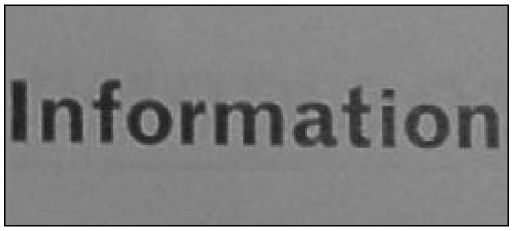
$$m_2(t) = \sum_{i=t+1}^{255} i.\mathrm{P}(i) / q_2(t) \tag{3.6}$$

$$s_1^2(t) = \sum_{i=0}^{t} (i - m_1(t))^2 \, \mathrm{P}(i) / q_1(t) \tag{3.7}$$

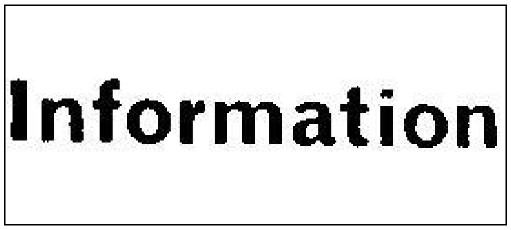
$$s_2^2(t) = \sum_{i=t+1}^{255} (i - m_2(t))^2 \, \mathrm{P}(i) / q_2(t) \tag{3.8}$$

Now the best threshold can be determined by a simple sequential search through all possible values of t to find the threshold that minimizes $s_W^2(t)$. From the statistical point of view, however, in many cases it is not necessary to try all possible 256 values, and computational time can be reduced. The interested reader can refer to the original paper (Otsu, 1979).

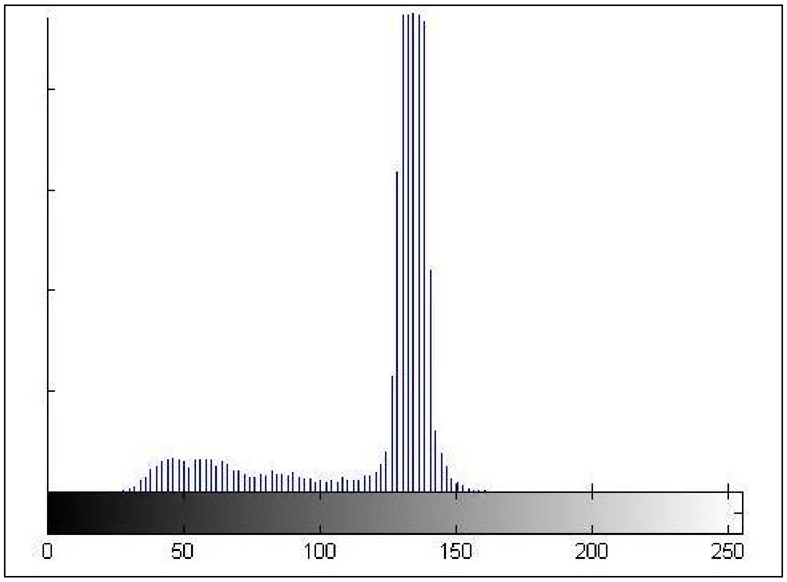
In Figure 3.1 the Otsu's algorithm is applied to a bimodal gray-level image. The selected threshold is 97 and the resultant binary image is useful since the original image has a bimodal distribution.



(a) Original image



(b) Binarized image



(c) Histogram of original image

Figure 3.1. Applying the Otsu's algorithm to a bimodal image.

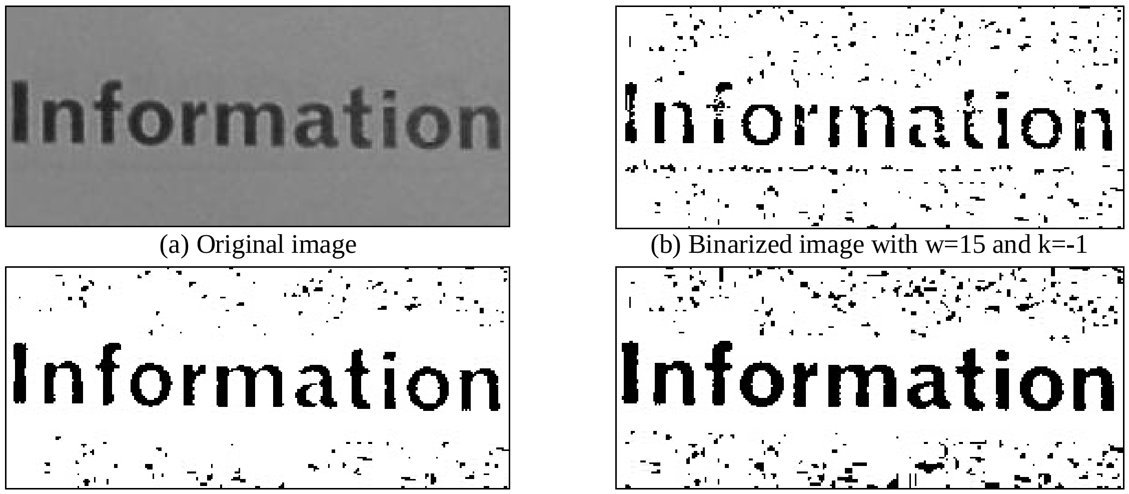
3.3 The Niblack's Method

As mentioned before, global thresholding techniques are not sufficient for binarizing document images with complex backgrounds; local methods in such cases are more useful and of them Niblack's local average method is selected because it is frequently cited to be promising. This method operates on the following threshold:

$$T(x,y) = M(x,y) + k.\sqrt{V(x,y)} \tag{3.9}$$

Where *M(x,y)* is the local mean and *V(x,y)* is the local variance computed in a moving $w \times w$ window. As seen, Niblack's method, like other local methods, has some user-defined parameters: *w* and *k*; which must be fine tuned by the user. The recommended value for *w* is 15 and a typical value for *k* is -1. These parameters are image-dependent; generally small values of *w* lead to noisy results and inconsistent stroke width and large values cause some characters to merge or split.

In Figure 3.2 the Niblack's algorithm is applied to a 120 by 275 gray-level image; as shown in Figure 3.2(b) the resultant binary image for default values of *w* and *k* is not useful because of split characters. The method becomes considerably slower as the window size becomes larger; in the experiments carried out in this work, for example, the computation time of Figure 3.2(c) was five times as much as that of Figure 3.2(b). Also, for large windows, it acts just like a global method.



(a) Original image (b) Binarized image with w=15 and k=-1

(c) Binarized image with w=30 and k=-1 (d) Binarized image with w=30 and k=-0.5

Figure 3.2. Applying the Niblack's algorithm to a bimodal image.

3.4 The Wu and Manmatha's Method

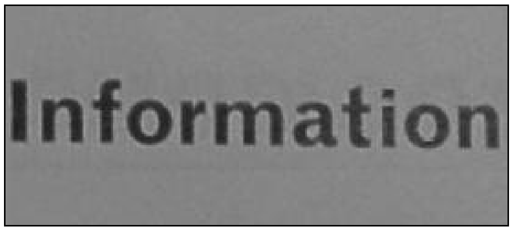
This is a simple and global method to binarize complex documents with text over textured/shaded backgrounds, poor contrast or considerable noise. The algorithm consists of two basic steps. First, the input image is smoothed using a Gaussian low-pass filter, causing text enhancement against background texture. Since many images do not have well-separated foreground and background, this step is necessary. Because the text has normally lower frequency than the shading, the smoothing operation affects the background more than foreground, and actually it tends to clean up the background. Second, the threshold is selected from the intensity histogram of smoothed image. Since the text is normally darker than other image objects, the threshold is set to the first valley counted from the left side of the histogram. To extract text against darker background, the last valley is selected instead. In both cases, the intensity histogram must be smoothed before threshold selection since it usually contains many local minima. This can be done by convolving the histogram again with a Gaussian kernel. This method is also effective when the bimodal histogram assumption is not valid.

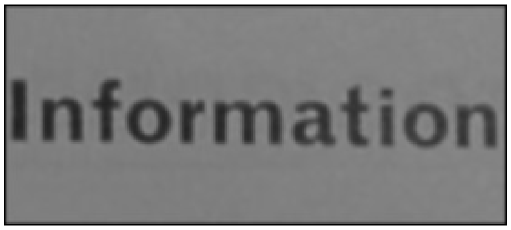
To compare this technique with the other implemented methods, it is applied to the same bimodal image as shown in Figure 3.3. The smoothed image obtained by a 5 by 5 Gaussian kernel with the variance of 1.44 is given in Figure 3.3(b). Figure 3.3(c) shows the histogram of smoothed image and Figure 3.3(d) depicts the smoothed histogram obtained by a Gaussian kernel of length 15. The selected threshold is 77 and the resultant binary image shown in Figure 3.3(e) is useful. This method is faster than Niblack's, but due to convolution which is a time consuming process, it is rather slower that Otsu's.

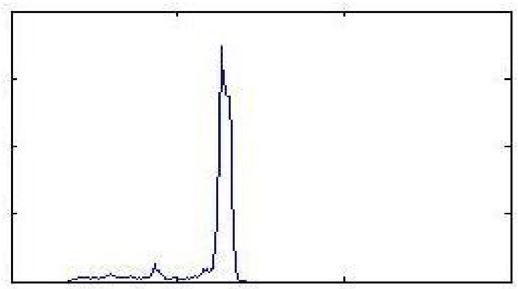
3.5 The Liu and Srihari's Method

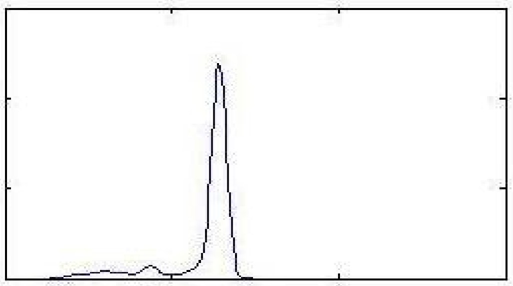
Like Wu and Manmatha's algorithm, this method belongs to the global category and specially designed for document image binarization. This method uses two fundamental features of document images to select a reasonable threshold. First, characters normally occupy a separable gray-level range in the histogram. Second, text images contain highly structured-stroke units. In this method the Otsu's algorithm is iteratively applied to the image histogram to find a limited number of (usually two) candidate thresholds. For each threshold value, the input image is binarized accordingly and some texture features are extracted from run-length histogram; the threshold for which these features have better values is considered best and applied as the global threshold.

To select the candidate thresholds, the entire histogram of the gray-level input image is first split using Otsu's algorithm, and in each subsequent iteration the part of the histogram with lower mean is further divided. This is because of the assumption that the


(a) Original image


(b) Smoothed image with a 5 by 5 Gaussian kernel


(c) Histogram of smoothed image


(d) Smoothed histogram of smoothed image

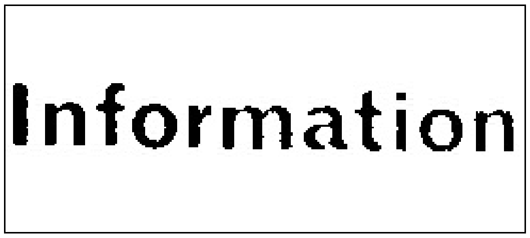

(e) Binarized image

Figure 3.3. Applying the Wu and Manmatha's algorithm to a bimodal image.

text is usually darker than other image items, and hence occupies the lower part of the histogram. This iterative strategy can handle an unknown number of histogram peaks. According to experiments, however, it is sufficient that Otsu's algorithm is only applied twice. Therefore the next step of the algorithm is to choose between just two candidate thresholds T_1 and T_2, $(T_1 > T_2)$.

Liu and Srihari concluded that the horizontal run-length histogram of a binarized image contains the essential information for evaluating quality of document image binarization.

The horizontal run-length histogram will then be denoted by R which is a one-dimensional array R(i), $1 \le i \le L$, where L is the longest horizontal run to be counted. The maximum possible length of a horizontal run-length is equal to or less than the number of columns of the binarized image C, so in implementation L is set to be C. In the histogram, R(i) is the count of the horizontal run-lengths of length i. Having the histogram, five texture features is extracted: Stroke Width (SW), Stroke-Like Pattern Noise (SPN), Unit-Run Noise (URN), Long-Run Noise (LRN) and Broken Character (BC). A detailed description about these features is found in the original paper, but for completeness, the definitions are given here (based on the assumption that in the input image the text is darker than other image items):

$$SW = \arg\max_{2 \le i \le L} R(i) \tag{3.10}$$

$$SPN = \frac{\max_{2 \le i \le L} R_1(i)}{\max_{2 \le i \le L} R_2(i)} = \frac{R_1(SW_1)}{R_2(SW_2)} \tag{3.11}$$

$$URN = \frac{R(1)}{\max_{2 \le i \le L} R(i)} = \frac{R(1)}{R(SW)} \tag{3.12}$$

$$LRN = \frac{\sum_{i>M} R(i)}{\max_{2 \le i \le L} R(i)} = \frac{\sum_{i>M} R(i)}{R(SW)} \tag{3.13}$$

$$BC = \frac{\min_{1 \le i \le SW} R(i)}{\max_{1 \le i \le L} R(i)} = \frac{\min_{1 \le i \le SW} R(i)}{\max(R(1), R(SW))} \tag{3.14}$$

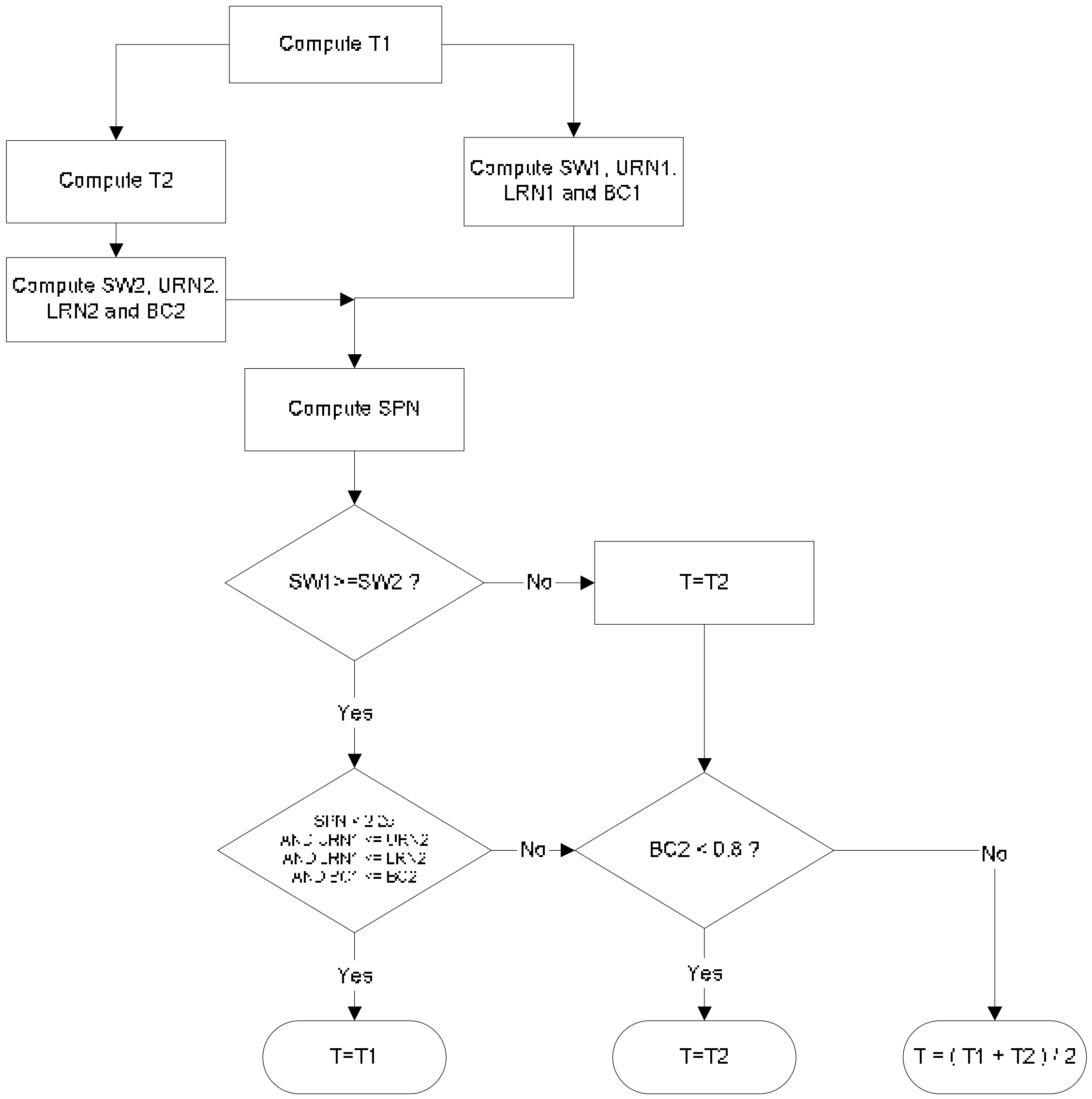
where R_1 is the histogram obtained with T_1 and R_2 is the histogram obtained with T_2; M is a constant that in this implementation was set to 3.

To select the best threshold from T_1 and T_2, the following scheme is applied. Let B_1 is the binarized image obtained with T_1, and B_2 is the binarized image obtained with T_2. First, the SW feature is checked, and the threshold with larger SW is selected. This is usually T_1, and the associated binarized image is the same as Otsu's. Hence, the next step is applied to verify and correct this selected threshold. In the second step the SPN, URN and LRN features are checked. If selected threshold (from the previous step) is T_1 and SPN is low (less than 2.25 in this implementation) and URN and LRN features of B_1 are less than URN and LRN features of B_2, the overall quality of B_1 is better than B_2 and the selected threshold is T_1. Otherwise, if the BC feature of B_2 is low (less than 0.8 in this implementation) the selected threshold will be T_2; otherwise, neither T_1 nor T_2 leads to an acceptable binary image, and the selected threshold is set to the average of T_1 and T_2.

The second step of this implementation is a little different from the original decision procedure (Liu and Srihari, 1997). By the following decision method, only three pre-specified values for M, low SPN and acceptable BC range are required; automatic tuning of these parameters seems difficult because of the lack of ground truth data, so the algorithm must be calibrated using a limited number of experiments and visual judgments. In this study, a set of ten images was used to determine them experimentally, and now they seem to be adequate for a broad range of images. The flowchart of the implemented method is presented in Figure 3.4.

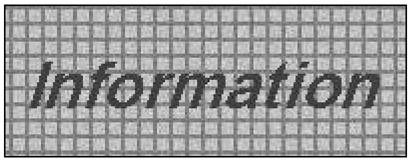
Although this method is able to deal with images with complex backgrounds, for comparison, it is applied to the same bimodal image of Figure 3.1. The first selected threshold is 97 (Otsu's threshold) and the second is 64. For 97 the SW feature is 7, and for 64 it is 5; and since other features associated with 97 have acceptable values, it is selected as the final threshold, and the output binarized image is the same as Otsu's result. Liu and Srihari's method is about two times slower that Otsu's but faster than Wu and Manmatha's. All of these global techniques, however, are fast enough so that can be used in any real-time application.

In order to show the superiority of Liu and Srihari's algorithm over others', it is applied to the rather complex image of Figure 3.5(a). The histogram of this image is presented in Figure 3.5(b), and as shown in Figure 3.5(c), Otsu's method fails to find the true threshold. But Liu and Srihrari's method successfully binarize the image; in this case, $T_1 = 153$ (which results in binary image of Figure 3.5(c)) and $T_2 = 97$. For T_1, $SW_1 = 2$, $URN_1 = 0.05$, $LRN_1 = 0.26$ and $BC_1 = 0.05$, and for T_2, $SW_2 = 4$, $URN_2 = 0.08$, $LRN_2 = 1.32$ and $BC_2 = 0.009$; since SW_2 is larger SW_1 and BC_2 is smaller than 0.8, the algorithm rejects T_1 in the favor of T_2 and the associated binary image is depicted in Figure 3.5(d) in which the small dots are due to the input image noise, and can be removed by a post-processing step.

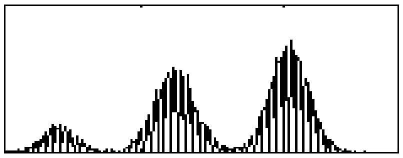

Compute T1
Compute T2
Compute SW1, URN1, LRN1 and BC1
Compute SW2, URN2, LRN2 and BC2
Compute SPN
SW1>=SW2 ?
No
T=T2
Yes
SPN < 2.0 AND URN1 <= URN2 AND LRN1 <= LRN2 AND BC1 <= BC2
No
BC2 < 0.8 ?
No
Yes
Yes
T=T1
T=T2
T = (T1 + T2) / 2
Figure 3.4. The flowchart of Liu and Srihari's method.

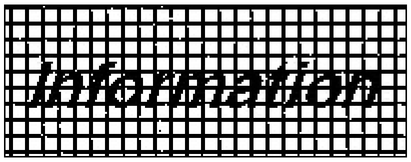
The Wu and Manmatha's method for the same image selects the threshold of 78 which yields the nicely binarized image of Figure 3.5(e). By inspection of Figure 3.5(f) in can be seen that the outcome of Niblack's adaptive method is useless because the background is left and there are broken characters; for both images of Figures (d) and (e), in contrast, stroke connectivity is maintained.

In further experiments, images with various size characters and images containing both machine-printed and handwritten text were processed; in each case, the overall quality of the Liu and Srihari's returned image was quite acceptable.

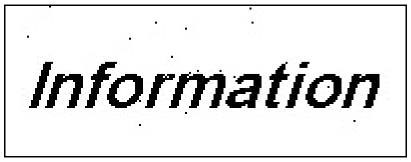


(a) The input image

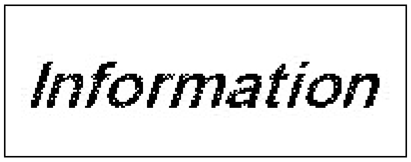

(b) Histogram of (a)



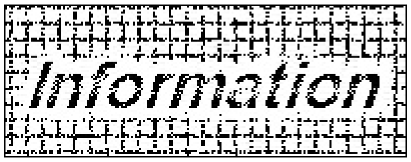
(c) Otsu's result



(d) Liu and Srihari's result



(e) Wu and Manmatha's result



(f) Niblack's result

Figure 3.5. Applying the four binarization methods to a non bimodal image.

3.6 Preprocessing

The outcome of binarization algorithms for low-resolution images sometimes can be enhanced by a preprocessing step termed "super-resolution" (Taylor and Dance, 1998); which is trading of gray-scale intensity resolution for spatial resolution. The block diagram of a binarization algorithm equipped with super-resolution is presented in Figure 3.6. In the first step, the input image is sharpened. There are several ways for this purpose (Jain, 1989), the simplest one, however, is to convolve the image with the negative of a Laplacian kernel. Next, this image upsampled (usually by a factor of 3), and then the binarization algorithm is applied as usual. Finally, the binarized image is downsampled to have the same size as input.

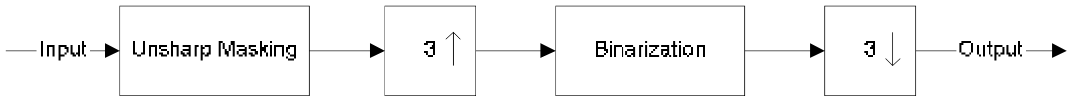
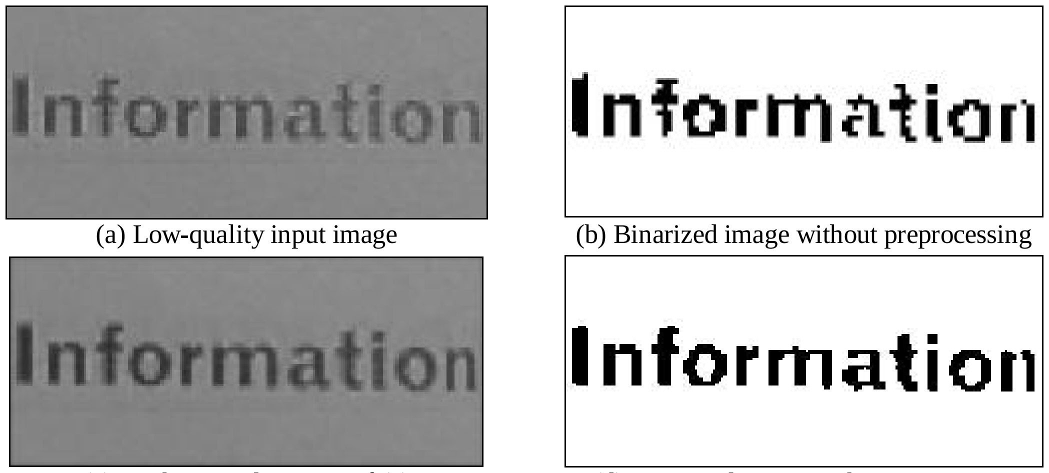


Figure 3.6 The block diagram of a binarization algorithm equipped with super-resolution.

Sometimes, this preprocessing step can be effective as shown in Figure 3.7.



(a) Low-quality input image (b) Binarized image without preprocessing

(c) Unsharpened version of (a) (d) Binarized image with preprocessing

Figure 3.7. Preprocessing can enhance the binarized output of low-quality images.

3.7 Postprocessing

The binarized output of a binarization algorithm often needs a postprocessing step particularly when the input image is noisy, and so causing noisy output. This was already shown in Figure 3.5(d). Usually binarized image contains some extra connected-components which are due to sudden intensity changes of in noisy regions of the input. A simple median filtering is not sufficient to remove this type of noise; since an extra component can be larger than one pixel in size, and so not removed by one pass. Moreover, the median filtering adversely affects other components by smoothing their corners (Figure 3.8) which may lead to higher recognition errors.

One of the effective postprocessing techniques, which can be incorporated into any thresholding algorithm, is surveyed in (Trier and Taxt, 1995) and its modified version is given below:

1. Smooth the original image by a 3 by 3 mean filter to reduce noise.
2. Calculate the gradient magnitude image G of the smoothed image using , e.g., Sobel's edge detector (Shapiro and Stockman, 2001).
3. Remove all isolated pixels (connected-components of size 1) of the binarized image.

4. For each remaining connected-component C_i of the binarized image, calculate the average gradient magnitude of its border pixels, using corresponding pixels in G, and call it M_i.
5. Compute the average of M_i 's and call it T, it is a criterion for the average strength of the connected-components.
6. Remove all connected-components of the binarized image having an average border pixels gradient below kT, where k is an image-dependent parameter; the value of 0.9 is a good choice for low amount of noise, but it must be increased for the input images containing more noise.

Figure 3.9 illustrates quality improvement gained by this postprocessing step. From Figure 3.9(c), it is clear that the median filter has caused broken characters, and thereby can not be used as a postprocessor. In contrast, Figure 3.9(d) shows that the mentioned postprocessing method has improved the quality of the noisy binarized image.

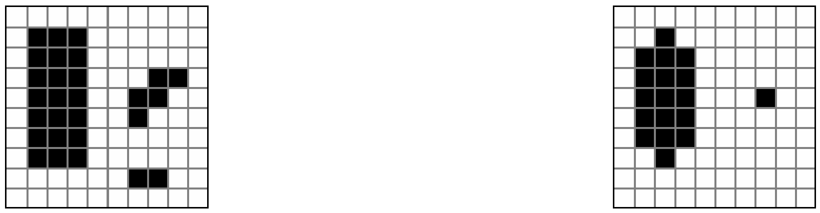
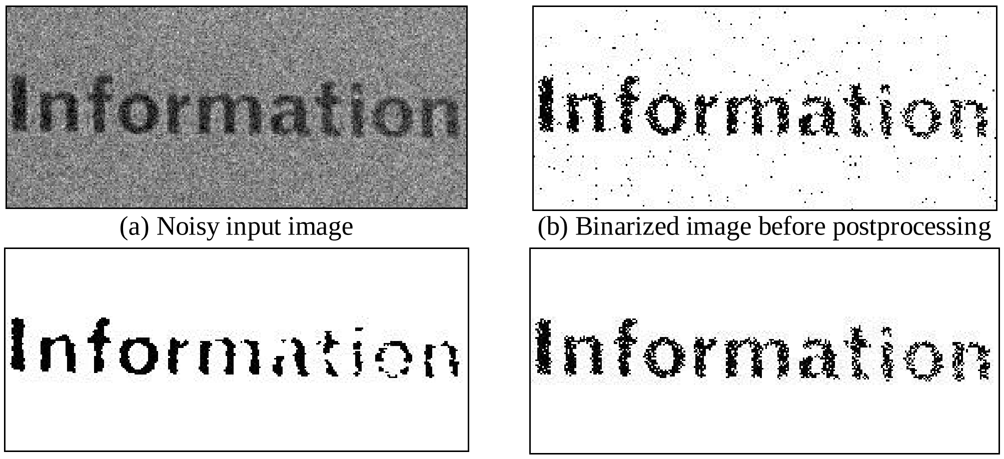

(a) A binary image containing noise and a shape (b) The image after applying a 3 by 3 median filter

Figure 3.8. A median filter can not remove all noise, and adversely affects corners of shapes.


(a) Noisy input image (b) Binarized image before postprocessing

(c) Image (b) after median filtering (d) Image (b) after postprocessing

Figure 3.9. Quality improvement of a binarized image by postprocessing.

CHAPTER 4

SKEW CORRECTION

4.1 Introduction

Document skew is a distortion that is often introduced during scanning or copying of a document and it is unavoidable. The skew angle is the angle that text lines deviate from the x-axis. Since page decomposition techniques require properly aligned images as input, document skew must be corrected in advance; otherwise, serious performance degradations will result.

In general there can be three types of skew within a page (Okun et al., 1999): 1) a global skew, when all text lines have the same orientation; 2) multiple skews, when some text lines have a different orientation than the others; and 3) non-uniform skew, when the orientation fluctuates within a text line. It must be noticed that a handwritten document image is usually expected to have multi-skew or even worse, non-uniform skew. A number of methods have been proposed for global skew detection. Nevertheless, it is assumed that even if there are multiple skews, they belong to a limited range, and hence we find the dominant global skew. Once the global orientation is detected, the document skew can be corrected by a rotation at this angle. In other words, "skew correction" is applied after "skew detection". Global skew detection algorithms can be divided into seven categories based on the underlying techniques: 1) projection profile (Shridhar and Kimura, 1995; Postl, 1986); 2) Hough transform (Jiang et al., 1997); 3) Fourier transform (Postl, 1986); 4) nearest-neighbor clustering (Yue Lu and Chew Lim Tan, 2003); 5) correlation (Avanindra and Subhasis Chaudhuri, 1997); 6) mathematical morphology (Najman, 2004); and 7) Artificial Neural Networks (Rondel and Burel, 1995). Some of these algorithms can detect a limited range of skew angles (usually varying from ± 5 to ± 45), while others are able to find and correct any skew angle (Okun et al., 1999). Some methods are designed for specific image formats, low-resolution or compressed images (Spitz, 1998). Some are designed to work with machine-printed documents (Changming Sun and Deyi Si, 1997), such methods can not deal with documents containing handwritten or non-Latin scripts. Most methods assume that text has already separated from graphics; otherwise it is often required that text is predominant in the image to have accurate estimates.

In projection profile based methods, histograms of foreground pixels or other features of connected-components (such as center of mass) are computed for a number of orientations close to the expected skew angle, and for each histogram a variation measure ,for example mean square deviation, is evaluated. The histogram that maximizes the variation corresponds to the global skew angle. The histogram at 0° is called horizontal projection profile; for a document without skew, the horizontal projection profile must have the maximum variation, and for skewed documents the histogram at skew angle has the maximum variation. The histogram with maximum variation has peaks whose widths are approximately equal to the average character height, and its valleys have minimum heights in comparison with other histograms. These methods are simple, robust and easy to implement; they can also work with gray-scale documents, tolerate noise and do not require predominant text area in the input image, but since the computation of histograms at different angles needs many image rotations which is a time-consuming operation, the range and resolution of detectable angles are restricted. Moreover, a projection profile based method may not find a good estimate in a multi-column document.

The Hough transform has been widely used for skew detection. This transformation maps each point in the original (x,y) plane to all points in the (θ, ρ) parameter plane that is the Hough space of lines through (x,y) with slope θ and distance ρ from the origin. A line in the original image forms a cluster in the parameter plane. Once the locations of the clusters are determined, the skew of each line and the average skew are easily evaluated by searching for a peak in the transformation space. The Hough transformation is useful not only in the detection of solid lines but also broken lines and even text lines. The high computational complexity of the Hough transform confines the detectable skew range. In order to reduce processing time, instead of applying the transform to entire the foreground pixels, it can be applied to other representative points such as edge points or center of mass of connected-components. The Hough transform is computed in O(n^2) time; also it needs a 2D accumulator. Therefore, methods based on this transform are usually slower than others. Another drawback is that when the text becomes sparse choosing a peak in the transform space is difficult, i.e., it can not be done by searching for the maximum value; because it is possible that the angle giving maximum value does not correspond to the skew angle. But as an advantage, it must be cited that the presence of graphics in the input image does not drastically degrade the accuracy. Just as the projection profile based methods can operate faster when the detectable skew range is limited, so the Hough transform based algorithms will benefit when the input document image is known to have a limited skew.

The basic idea of the methods based on nearest-neighbor clustering (NNC) is that the points belonging to the same line can not significantly deviate from that line. Generally, an algorithm based on this idea has the following steps. First, connected-components of the binarized input image are obtained. Then, the direction vector of all k-nearest-neighbors of connected-components are computed and accumulated in a histogram, and finally, the angle corresponding to the peak of histogram is returned as the document skew angle. It must be noticed that the presence of ascenders, descenders (i.e., upper and lower parts of characters) or dots cause connections that are not parallel to the text lines, thus reducing the accuracy. To remedy this problem, in some algorithms (Yue Lu and Chew Lim Tan, 2003; Okun, 1999) only connected-components satisfying certain size and/or positional conditions are taken into account, thereby, these algorithms must be tuned for their parameters. The main advantage of a method utilizing NNC is that it does not limit the detectable skew range; also it does not require predominant text area in the input image and can also deal with multi-column document images and even multiple skews. But, an algorithm of this type can only work with clean binarized images, and as mentioned before needs fine tuning. An accurate NNC based algorithm is presented in (Yue Lu and Chew Lim Tan, 2003); in this work connected-component chains with the largest possible number of nearest neighbor pairs are selected, and their slopes are computed to give the global skew angle.

The correlation function has also been used in skew estimation (Avanindra and Subhasis Chaudhuri, 1997). The basic idea is that the correlation between two columns (vertical lines) of the document image is maximized when one column is shifted relatively to the other such that character levels are aligned. The correlation based methods require predominant text area in the input image; otherwise, a prior text/graphics separation is necessary to have good estimates. But the major limitation is that such a method gives a true estimate only when the skew range is limited (usually from -10° to 10°), and fails to detect a high amount of skew angle, but it does not mean that correlation methods are not practical; because for ordinary scanned document the actual skew angle is quite small. These methods can deal with handwritten and non-Latin scripts as well, but text lines of different sizes degrade the accuracy, and as a further disadvantage, any correlation based algorithm use some parameters (usually two) which must be set for different types of documents beforehand. A fast correlation based method is presented in (Avanindra and Subhasis Chaudhuri, 1997), in which instead of finding the correlation for the entire image, it is calculated over randomly selected small windows to increase speed, and since these windows can be processed independently, as a further advantage, the algorithm can be implemented on a parallel hardware. It is a Monte Carlo probabilistic algorithm that needs at least half the input image area is occupied by text to ensure that the probability of a randomly selected window is higher than 0.5.

In the methods based on the Fourier transform (Postl, 1986) the direction having maximum density in transform space is regarded as the skew angle. These methods are not implemented in this study, but it is clear that a vertical line in the input image will have the maximum density direction. Thus, generally, finding the true skew angle in the transform space is not easy and straightforward. Also, it is often said that the Fourier transform is computationally expensive for large methods (Changming Sun and Deyi Si, 1997). The Fast Fourier Transform (FFT) was first used in (Postl, 1986); in this method The coefficients of the power spectrum are calculated and stored in a buffer. Then, directional criteria for a number of angles are calculated. Last, the angle that maximizes the directional criterion is taken as the document skew angle.

Artificial Neural Networks (ANN) have been widely used for document analysis and recognition, but not much work is dedicated to the problem of skew detection. In (Rondel and Burel, 1995) two neural networks are used to detect the global skew; the first one gives a rough estimate which is used to initialize the weights of the second network.

Then, the second network outputs the document skew angle. No ANN based method is implemented in this study, but due to numerous advantages of ANNs, they are worthy of study for the skew detection problem.

A typical method based on mathematical morphology iteratively applies special morphological operators (modified versions of opening and closing) to the input image to form one connected-component (blob) from each text line. Then, a line is fitted to each blob and its slope is accumulated in an angle histogram; finally, the angle corresponding to the histogram peak is returned as the skew angle. It must be mentioned that there are fast implementations for the two basic morphological operators (i.e., erosion and dilation), for example, the Fourier transform can be used for this purpose. Advanced operators can be derived by the combination of erosion and dilation, and used in skew estimation. All morphological operators are applicable to both binary and gray-scale images, but there exist faster implementations for binary images (Nadadur and Haralick, 2000).

In the rest of this chapter, the Hough transform is surveyed and its basic algorithm for skew detection is given, and due to the importance of this transform and lack of visual examples in the literature, a number of examples are presented to illustrate how this transform can be useful in skew detection. Then, the basic idea of the projection profile technique is clarified, and finally, a simple skew detection procedure, satisfying script-independency, is proposed in detail, and it will be shown that this projection profile based procedure is robust enough to be used in a real recognition system.

4.2 The Hough Transform for Skew Detection

Hough transform is a general method for detecting arbitrary curves (lines, ellipses, etc.) in gray-scale images. Hough Line Transform (HLT), as it is clear from the name, aims to detect straight lines and is a popular method for skew detection. In HLT, conceptually, all possible lines (at all orientations and positions) are placed into the image and the number of pixels on each line are counted and stored in the corresponding position of the Hough space.

The simple version of HLT for skew detection is given in Algorithm 4.1. It must be mentioned that since the ordinary line equation $y = mx + b$ does not work for vertical lines, $d = c.\cos(\theta) - r.\sin(\theta)$ is used as an alternative, where d is the perpendicular distance from the origin of the image (upper left corner) to the line, and θ is the angle this perpendicular makes with horizontal (column) axis. In order to compute gradient magnitude and direction of the input image, any edge detector can be used. The constant gradient_threshold in the algorithm is used to only take strong enough edge points, and it is reasonable to set its value to the average gradient magnitude of the input image. An alternative approach is to use the binary edge map B_M instead of M, with 1's representing strong edge points and 0's representing background, and now the if command must be changed accordingly:

```
  if B_M[r,c] > 0
  {
    d = round(absolute(c×cos(D[r,c]) - r×sin(D[r,c])));
    A(d,D[r,c]) = A(d,D[r,c]) + 1;
  }
```

Once the execution have been completed and the accumulator array has been filled, the angles corresponding to the local peaks of the Hough space (accumulator array) represent the dominant skew angles of the input document image. So theoretically, a Hough transform based method is also able to detect multiple skews. The accumulator array does tell us about where the line segments begin and end, and in the skew detection there is no need for this information.

```
Let I[r,c] be the input gray-scale image having R rows and C columns.
Let M[r,c] be the gradient magnitude of I[r,c].
Let D[r,c] be the gradient direction of I[r,c].
Let A[ρ,θ] be the accumulator array (the Hough space).

A = 0; // initialize the accumulator to zero.
for r = 0 to R-1
{
  for c = 0 to C-1
  {
    if M[r,c] > gradient_threshold
    {
      d = round(absolute(c×cos(D[r,c]) - r×sin(D[r,c])));
      A(d,D[r,c]) = A(d,D[r,c]) + M[r,c];
    }
  }
}
```

Algorithm 4.1 HLT for skew detection

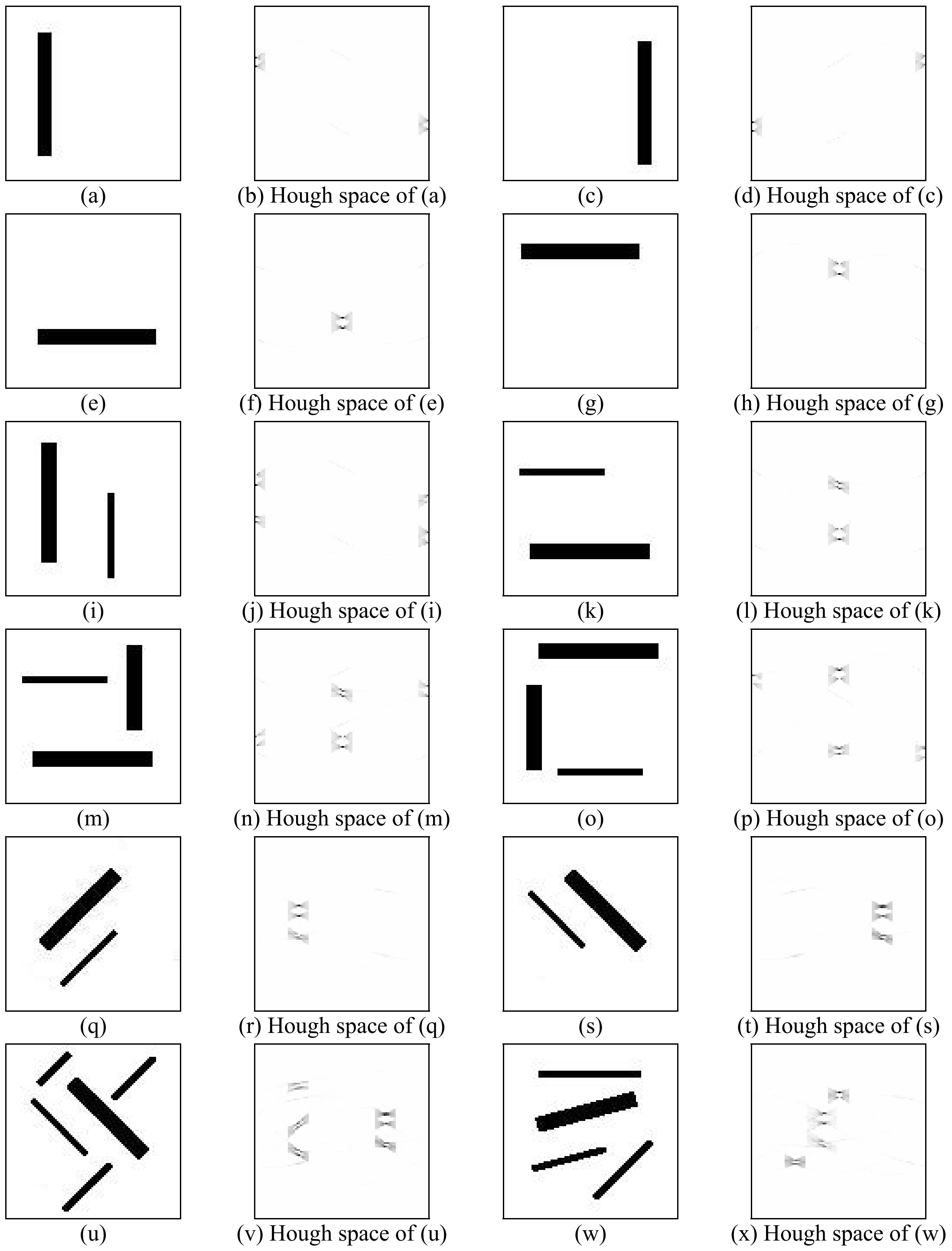
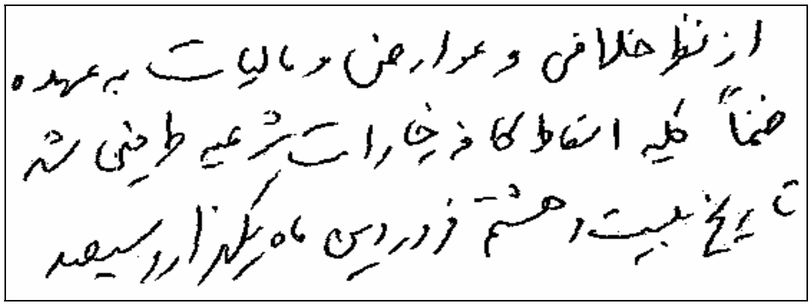


Figure 4.1. Applying HLT to simple binary images.

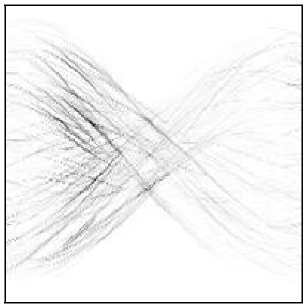
In Figure 4.1, HLT is applied to twelve images, containing one to four rectangles of different sizes and at various directions and positions; for each case the Hough space is depicted; it is clear that the relative position of the objects does not change the angles corresponding to the Hough space peaks, and as mentioned before, the objects at different directions will form different clusters in the Hough space, so HLT may be used to correct multiple skews.

Figure 4.2 shows the Hough space of two handwritten document images. Figure 4.2(b) has no peak corresponding to the document skew angle, and for this non-Latin document image the HLT technique fails. By further experiments, we found out that the method also fails for low resolution images.

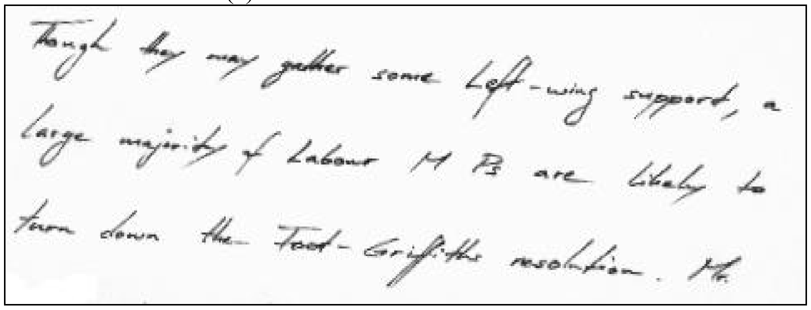


از نظر خلاف و عوارض روایت به عهده
ضمناً کلیه اشتغالات فرخارات شرعی طرفین شد
تاریخ بیست و هشتم فروردین ماه کار را رسید

(a) A handwritten Farsi document

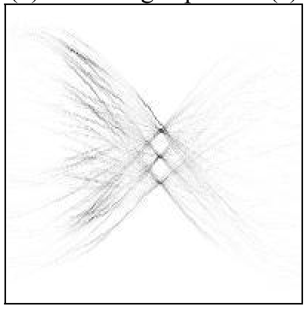


(b) The Hough space of (a)



Though they may gather some Left-wing support, a large majority of Labour MPs are likely to turn down the Foot-Griffiths resolution. Mr

(c) A handwritten English document



(d) The Hough space of (c)

Figure 4.2. Applying HLT to handwritten document images.

4.3 The Projection Profile Method for Skew Detection

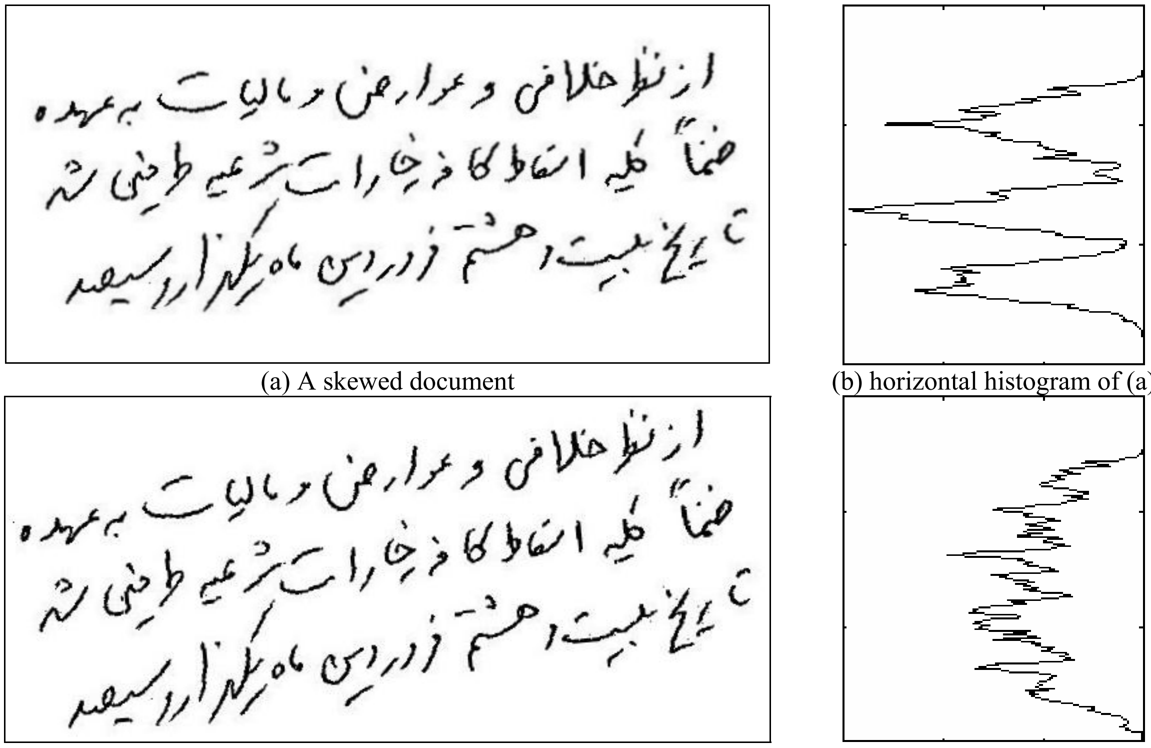
It is expected that the projection profile at the global skew angle of the document has narrow peaks and deep valleys, depending on weather the projection passes through a text line or between text lines. Figure 4.3, for example, shows a document image at two different directions and the associative horizontal projection profiles. Obviously, at this point we need a criterion to select the better projection profile. Let *f* be a function returning its maximum value for the horizontal projection profile at the global skew angles, then the global skew angle of the gray-scale image I is:

$$global\_skew\_angle = \arg\max_{\theta_{\min} \le \theta \le \theta_{\max}} f(horizontal\_projection\_profile(rotate(\mathrm{I}, \theta))) \qquad (4.1)$$

It may seem, at first sight, that variance or autocorrelation are good choices for *f*, but as noted by Bloomberg (Bloomberg et al., 1995) neither can be a good measure. The variance function usually results in a broad peak, being difficult to choose the global skew from; the autocorrelation function is more computationally demanding, and giving a large oscillating signal for the projection at the global skew angle. In that reference the goodness measure is taken as the sum of the squares of the successive differences of the projection profile (histogram). Formally, if the histogram is denoted by h, SD is given by:

$$SD = \sum_i (h(i) - h(i-1))^2 \qquad (4.2)$$

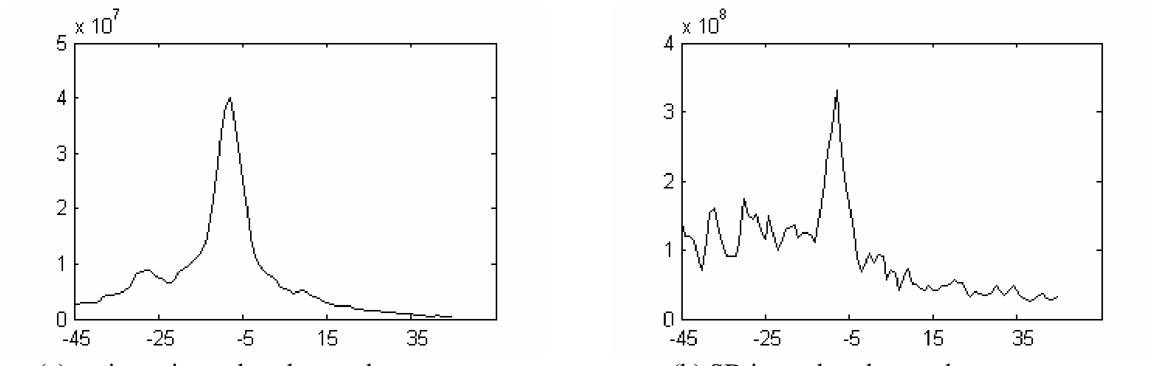
This function has a very sharp peak at the global skew angle, leading to very accurate results, but on the other hand, such a narrow peak restricts the use of the binary search to find the maximizing angle. There are three modifications which can speed up the basic method.


(a) A skewed document (b) horizontal histogram of (a)

(c) The same document with more skew (d) horizontal histogram of (c)

Figure 4.3. The projection profile technique for skew detection.

First, for computing the projection profile at a certain angle, it is not necessary to rotate the image by the angle, and then compute the horizontal projection profile. One possibility is to shear the image in vertical direction which is faster than rotation, and as proved in (Slavik and Govindaraju, 2001): "correcting first for skew by rotation and then for slant by a shear transformation in the horizontal direction is equivalent to first correcting for slant by a shear transformation in the horizontal direction and then for skew by a shear transformation in vertical direction". The other possibility is to compute the sum of pixels along parallel lines at an angle; Algorithm 4.2 is for this purpose.



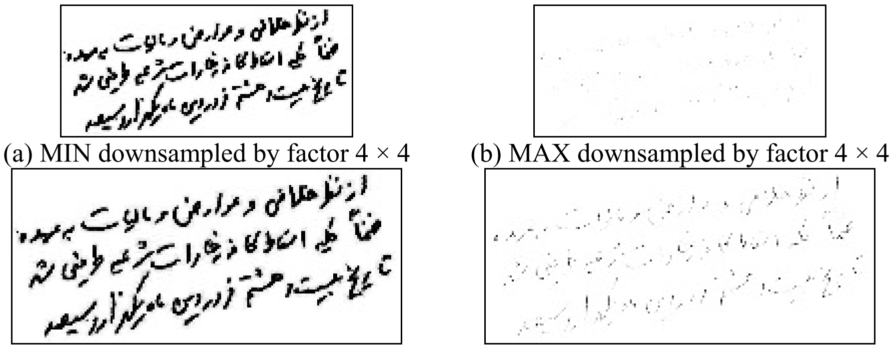
(a) variance is used as the goodness measure (b) SD is used as the goodness measure

Figure 4.4. Plotting goodness measure of projection profiles of Figure 4.3(c) against angles -45^o to 45^o.

Second, if in (4.1) f has only one maximum, it can be found by a binary (Algorithm 4.3) rather than the exhaustive search in the range [$\theta_{\min}, \theta_{\max}$], thus reducing the runtime. Bloomberg has suggested performing the binary search on the variance of the projection values, which has a sufficiently wide peak, but it may also fail because as you see in Figure 4.4(a) the function has local maxima. But, it seems that when the skew range is limited (e.g., -5^o to 5^o) the function has only one maximum, and so the binary search is possible.

Third, another advantage of projection profile based methods is that they actually don't need high resolution input images. Obviously, any image operation such as rotation or shear transformation is done faster for smaller images. Therefore, reducing the size of input image, as much as structure of text lines is preserved, leads to faster processing. This can be done by a MIN or MAX downsampling technique depending on whether the background is lighter than text or darker. These two techniques are faster than the ordinary downsampling methods, because the latter usually perform interpolation and smoothing to achieve better visual quality, which is not necessary for skew detection. In the MIN downsampling technique, the minimum of each M × N rectangle (when non-overlapping rectangles are considered) of the original gray-scale image is chosen as the value of output image in that location; as opposed to the MAX technique in which "maximum" performs the same job. For a binary image, MIN and MAX correspond to logical AND and OR, thus even a faster processing will result.

Figure 4.5 shows the downsampled versions of the 260 × 580 gray-scale image of Figure 4.3(c). It is clear that for ordinary document images with lighter background, the MIN method must be used; and as shown in Figure 4.5(a), a rough estimate of the skew angle can be made in the low-resolution image as well. Having a coarse estimate, the angle range $[\theta_{\min}, \theta_{\max}]$ can be restricted, because the actual skew angle is somewhere around it, and a more accurate result can be found in a higher resolution. This is a coarse-to-fine search strategy in which the approximate location of a solution is found quickly in a large and low-resolution space. Then, this estimate is refined successively in smaller spaces with higher resolutions.


(a) MIN downsampled by factor 4 × 4 (b) MAX downsampled by factor 4 × 4

(c) MIN downsampled by factor 3 × 3 (d) MAX downsampled by factor 3 × 3

Figure 4.5. Image downsampling using MIN and MAX techniques.

```
Let I[r,c] be the input gray-scale image having R rows and C columns.

projection_profile = 0; // initialize all elements to zero.
for r1 = 0 to R-1
{
  for c1 = 0 to C-1
  {
    r2 = r1.cos(θ) + c1.sin(θ); // new row after rotation
    projection_profile[r2] = projection_profile[r2] + I[r1,c1];
  }
}
```

Algorithm 4.2 Computing the projection profile at angle θ

As mentioned before, any projection profile based method tends to fail with unaligned text lines in multiple columns, however, according to experiments carried out in this work, for any other type of document image, whether machine-printed or handwritten, of any size and script, the method is able to correct the global skew angle. Figure 4.6 show that the method works well in the presence of considerable amount of noise. It seems that no other algorithm is so robust to noise. For such a noisy image with many broken characters, it is not surprising that any method, relying on structural information, fails.

```
Let f be the function, assumed to have only one maximum in the range [$x_{min}$, $x_{max}$].

$x_1$ = $x_{min}$;
$x_3$ = $x_{max}$;
$x_2$ = ($x_1$ + $x_3$) / 2;

while |$x_2$-$x_1$| > error
{
  $x_{12}$ = ($x_1$ + $x_2$) / 2;
  $x_{23}$ = ($x_2$ + $x_3$) / 2;
  maximizer = arg max (f($x_1$), f($x_{12}$), f($x_2$), f($x_{23}$), f($x_3$));

  if maximizer == $x_1$
  {
    $x_3$ = $x_{12}$;
    $x_2$ = ($x_1$ + $x_3$) / 2;
  }
  else if maximizer == $x_{12}$
  {
    $x_3$ = $x_2$;
    $x_2$ = $x_{12}$;
  }
  else if maximizer == $x_2$
  {
    $x_1$ = $x_{12}$;
    $x_3$ = $x_{23}$;
  }
  else if maximizer == $x_{23}$
  {
    $x_1$ = $x_2$;
    $x_2$ = $x_{23}$;
  }
  else
  {
    $x_1$ = $x_{23}$;
    $x_2$ = ($x_1$ + $x_3$) / 2;
  }
}

return arg max (f($x_1$), f($x_2$), f($x_3$));
```

Algorithm 4.3 Binary search for finding the maximizer of a function

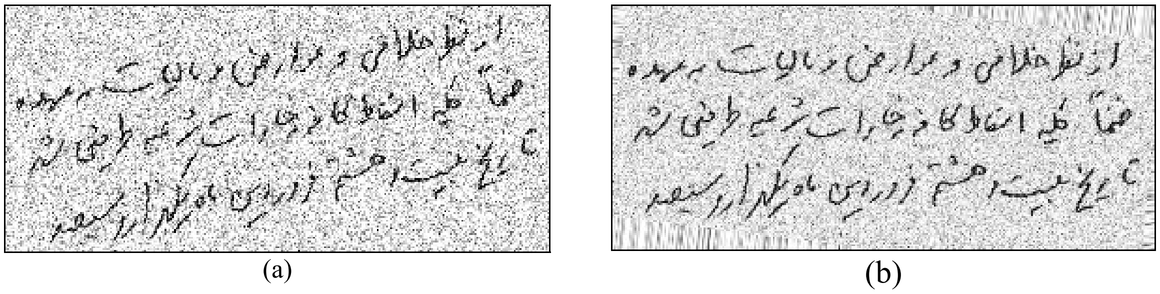
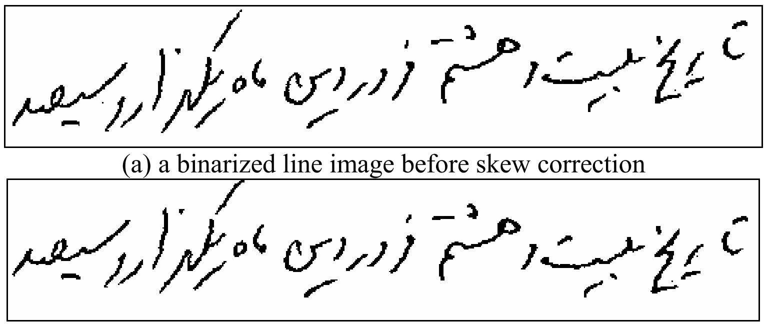

(a) (b)

Figure 4.6. A noisy image before (a) and after (b) skew correction using the projection profile based method.

4.4 Dealing with Multiple Skews

It is often expected that handwritten text lines slightly deviate from the global skew angle. In such cases, global skew correction followed by page segmentation result in a number of line (or word) images to be processed. Therefore, it is useful to perform a local skew correction in each line (or word) image.

A simple method for local skew detection is to fit a line to all text pixels in the line (or word) image. Due to its wide range of applications, line fitting a well-studied problem in statistics. The basic least square method for 2D space, which assumes y as the dependant variable, is not appropriate for vision tasks, partly because the mathematical definition of error as a difference along y-axis is not a true geometrical distance (Shapiro and Stockman, 2001); the disadvantage is more pronounced when the points are arranged in a near vertical direction. As mentioned in (Yuan and Tan, 2000), a better approximation is acquired by treating x and y not as statistical variables but as locations of points, and in this case, the error is defined as the sum of distances perpendicular to the orientation of the fitted line. Algorithm 4.4, based on evaluation of eigenvalues, is for this purpose, and skew correction by line fitting is illustrated by Figure 4.7.


(a) a binarized line image before skew correction

(b) and after skew correction

Figure 4.7. Skew correction by line fitting.

It is clear that, by fitting a line to text pixels (black pixels for ordinary document images) using Algorithm 4.4, $\tan^{-1}(m)$ gives the skew angle.

This method is very fast, but as mentioned before, it can not be applied to the whole document. For example, it fails when the image has more columns than rows; even if it does not fail, its estimate is not as accurate as other skew detection methods. Nevertheless, this method can be applied to the whole document image to find an estimate θ_R of the actual skew angle θ_A and reducing the search space from $[\theta_{\min}, \theta_{\max}]$ to $[\theta_R - E, \theta_R + E]$, where E must be selected so that θ_A falls within $[\theta_R - E, \theta_R + E]$. According to our experiments, $E = 3^{\circ}$ is a reasonable choice.

```
Let {(x_i, y_i)} be the set of points to be fitted by mx+y_0
$\bar{x} = \frac{1}{N}\sum x_i$ , $\bar{y} = \frac{1}{N}\sum y_i$ ;   // averages of x and y coordinates
$\hat{x}_i = x_i - \bar{x}$ ,   $\hat{y}_i = y_i - \bar{y}$ ;   // standardize data points
$a = \frac{1}{N}\sum \hat{x}_i^2$ , $b = \frac{1}{N}\sum \hat{x}_i\hat{y}_i$ , $c = \frac{1}{N}\sum \hat{y}_i^2$ ;
$\lambda_{1,2} = \frac{(a+c) \pm \sqrt{(a-c)^2 + 4b^2}}{2}$ ; // eigenvalues of the matrix [a b; b c]
$\lambda = \min(\lambda_1, \lambda_2)$ ;
$m = \frac{-b}{\lambda - a}$ ;
$y_0 = \bar{y} - m\bar{x}$ ;
```

Algorithm 4.4 Line fitting by evaluation of eigenvalues

CHAPTER 5

SLANT CORRECTION

5.1 Introduction

Slant is the deviation of average near-vertical strokes from the vertical direction. Slant correction is an attempt to reduce the range of variations of handwritten and machine-printed texts. In handwritten text, slant is due to the specific writing style, and in machine-printed text it is an innate feature of certain fonts. It is clear that slant is non-informative, but slanted words may considerably degrade the performance of the whole system (Kavallieratou et al., 2000), so another normalization step which must be performed before segmentation, feature extraction, training and recognition is to remove or reduce the slant influence as much as possible.

The literature includes a number of methods for uniform slant correction (Shridhar and Kimura, 1995; Changming Sun and Deyi Si, 1997; Kavallieratou et al., 2000) and some of them are robust, script-independent and applicable to both handwritten and machine-printed texts. The uniform slant correction techniques perform successfully when all near-vertical strokes have the same slant angle, which is usually the case for machine-printed words. So as far as the recognition of machine-printed text is concerned, there is no room for further study about slant removal methods. On the contrary, in handwritten text, the slant angle usually varies within each word (Figure 5.1), and hence a uniform slant correction is not optimum.

In all uniform slant correction techniques, the average slant angle is estimated first and then a shear transformation in horizontal direction is applied to the word (or line) image to correct its slant. The most effective methods are based on the analysis of vertical projection profiles (histograms) at various angles (Shridhar and Kimura, 1995; Kavallieratou et al., 2000); actually these techniques are identical to the projection profile based methods for skew correction, except that here the histograms are computed in vertical rather than horizontal direction and shear transformation is used instead of rotation.

Some method use statistics of chain-coded stroke contours; for example in (Shridhar and Kimura, 1995) the chain elements at 45^o, 90^o and 135^o are counted, then a simple formula is used to estimate the slant angle; according to our experiments, this method does not produce accurate result for handwritten words. In (Changming Sun and Deyi Si, 1997) two methods has proposed; the first one computes the histogram of gradient orientation of the input word image and returns the histogram peak as the slant angle; the second method fits a minimum bounding parallelogram to each connected-component of the binarized image, such that top and bottom sides of each parallelogram are parallel to x-axis, then the slant angle is chosen as the median value of all parallelogram angles. In the handwritten recognition system described in (Procter et al., 2000), two methods are used in combined, and the overall slant estimate is taken as the mean of the two estimates.

To the best of our knowledge, the only survey on non-uniform slant correction is presented in (Uchida et al., 2001), in which the problem is formulated as the optimal estimation of local slant angles at all horizontal positions. The optimal local slant angles which maximize a cost function, while satisfying several constraint for the global and local validity, are efficiently searched for a by a dynamic programming (DP) technique. Unfortunately, this method sometimes over-corrects slants of some alphabets such as the Latin 'X' or Farsi/Arabic letter 'ر' (Reh). So it can sometimes degrade the performance of recognition system, and this non-uniform technique can not be used.

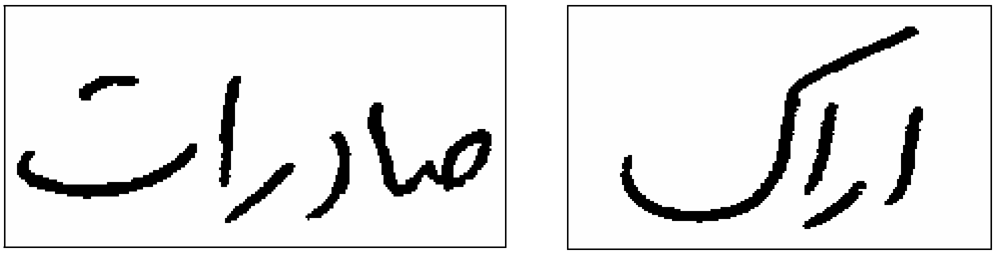

(a) Non-uniform slant (b) Uniform Slant

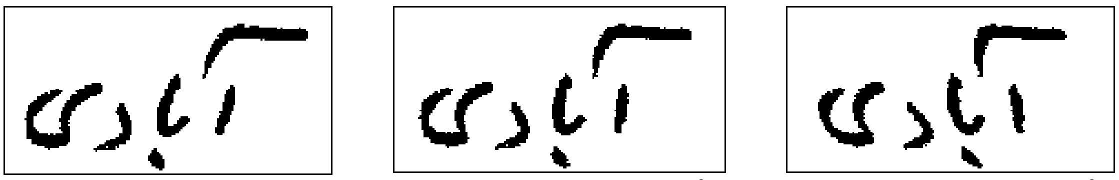
Figure 5.1. Examples of slanted handwritten words.

5.2 Horizontal Shear Transformation

In this linear transformation, each pixel (x,y) is transformed to new coordinate (x_s,y_s) by Equation (5.1), where y_c is the y-coordinate of the center and θ is the angle of transformation; for slant correction, y_c is set to half the number of image rows.

$$\begin{cases} x_s = x - (y_c - y).\tan(\theta) \\ y_s = y \end{cases} \tag{5.1}$$

By this transformation, the height of the image is not changed, while the width of the image will probably change. Figure 5.2 shows the results of shear transformation to a word image at two different angles.



(a) original slanted word (b) transformed by $\theta = -10^\circ$ (c) transformed by $\theta = -25^\circ$

Figure 5.2. Shear transforming a word image at different angles.

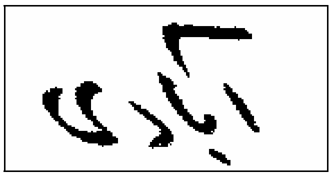
5.3 Projection Profile Technique for Slant Detection

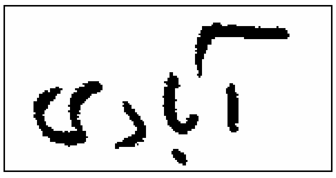
Like skew detection, here the basic idea is that the vertical histogram of a non-slanted word has higher peaks, deeper valleys and more variations than any other histogram. Figure 5.3 shows a handwritten word image at three different angles, the image of Figure 5.3(b) has less slant and its histogram has more and higher peaks. All we need is a criterion to judge between different histograms; in (Kavallieratou et al., 2000) the Winger-Ville distribution (WVD) is employed for this purpose. But it was found out that the same criterion utilized for skew detection can also work here. Therefore, in the proposed system, the slant angle of the line (or word) image I is estimated by the following formula:

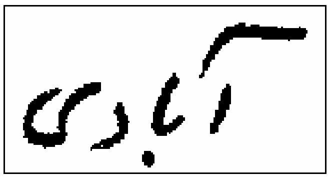
$$slant\_angle = \underset{\theta_{\min} \leq \theta \leq \theta_{\max}}{\arg\max}\, SD(vertical\_projection\_profile(horizontal\_shear(\mathrm{I},\theta))) \tag{5.2}$$

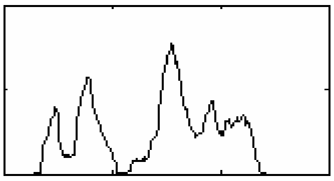
where *SD* is the sum of the squares of the successive differences of the projection profile, and search range is adequate be [-45°, 45°].

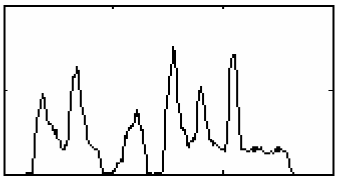
This method works well for both handwritten and machine-printed text. By being robust to noise and script independent, it is the optimal uniform slant estimator. Again, it is emphasized that this method requires a single line or word image as input. Obviously, vertical histograms of two or more text lines give no useful information about the slant. It contrasts with some other methods (Shridhar and Kimura, 1995; Changming Sun and Deyi Si, 1997) which employ structural information and can estimate the slant angle from the whole input document.


(a)


(b)


(c)


(d) Vertical histogram of (a)


(e) Vertical histogram of (b)

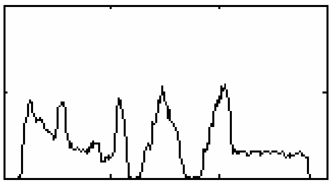
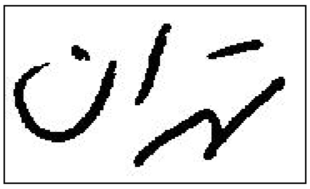
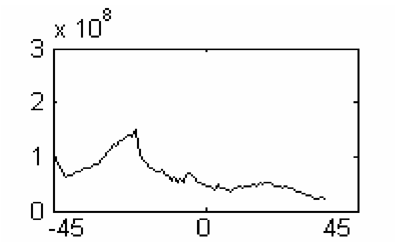

(f) Vertical histogram of (c)

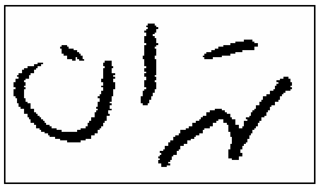
Figure 5.3. Vertical histograms of one image horizontally sheared at three different angles.

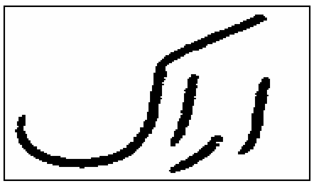
Figure 5.4 shows that the *SD* measure gives a maximum for the vertical histogram of horizontally sheared image at the slant angle, but the search space has local maxima which make it impossible to use the binary search.

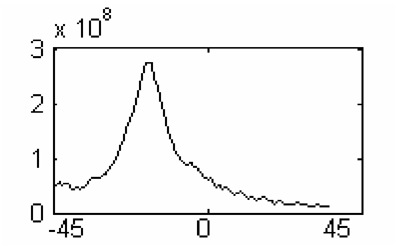

(a)



(b)


(c)


(d)



(e)

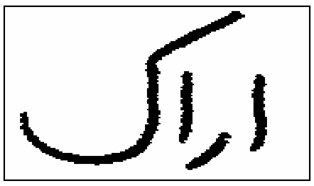
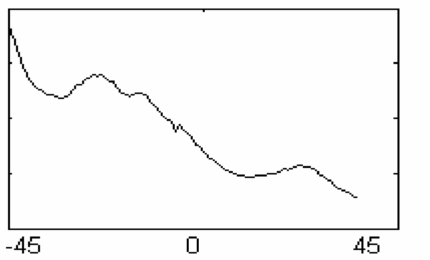
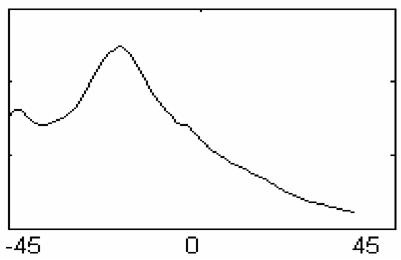

(f)

Figure 5.4. Plotting the *SD* measure of vertical histograms of sheared images from -45^o to 45^o. Each plot has a maximum corresponding to the slant angle.

Figure 5.5, shows that variance can not be used as the criterion, because it fails for image of Figure 5.4(a).



(a) for image of Figure 5.4(a), the maximum does not correspond to the slant angle.



(b) for image of Figure 5.4(e), variance gives the same result as *SD*.

Figure 5.5. Plotting variance of vertical histograms of sheared images from -45^o to 45^o.

Slant corrected words usually has jagged edges which may complicate the extraction of structural features. In order to remedy this problem, the image is smoothed by the rule set of Figure 5.6.

$$\begin{array}{|c|c|c|}\hline 1&1&1\\ \hline 1&0&0\\ \hline 1&1&1\\ \hline\end{array} \rightarrow \begin{array}{|c|c|c|}\hline 1&1&1\\ \hline 1&1&0\\ \hline 1&1&1\\ \hline\end{array}$$

(a) rule 1; rotating this rule at 90°, 180° and 270° gives rules 2, 3 and 4.

$$\begin{array}{|c|c|c|}\hline 1&0&0\\ \hline 1&1&0\\ \hline 1&0&0\\ \hline\end{array} \rightarrow \begin{array}{|c|c|c|}\hline 1&0&0\\ \hline 1&0&0\\ \hline 1&0&0\\ \hline\end{array}$$

(b) rule 5; rotating this rule at 90°, 180° and 270° gives rules 6, 7 and 8.

$$\begin{array}{|c|c|c|}\hline 1&1&1\\ \hline 1&0&1\\ \hline 1&1&1\\ \hline\end{array} \rightarrow \begin{array}{|c|c|c|}\hline 1&1&1\\ \hline 1&1&1\\ \hline 1&1&1\\ \hline\end{array}$$

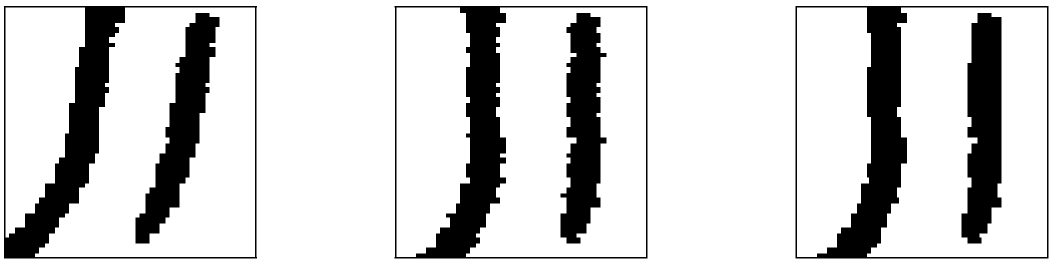
(c) rule 9; the complement of this rule, is rule 10.

$$\begin{array}{|c|c|c|}\hline 1&1&1\\ \hline 1&0&X\\ \hline 1&1&1\\ \hline\end{array} \rightarrow \begin{array}{|c|c|c|}\hline 1&1&1\\ \hline 1&1&X\\ \hline 1&1&1\\ \hline\end{array}$$

(d) rule 11; rotating this rule at 45°, 90°, 135°, 180°, 225°, 270° , and 315° gives rules 12 to 18.

Figure 5.6. The rule set for smoothing a slant corrected image, where 0 denotes background, 1 represents text and X means don't care.

Each rule is applied to all image pixels simultaneously, and the rules are applied one after another (i.e., rule n is applied to the image smoothed by rule n-1). These rules preserve the image connectivity, i.e. no rule breaks or merges connected components. Figure 5.7 demonstrates that the post-processing step can smooth jagged edges of a slant corrected word.



(a) A part of a slanted word (b) image (a) after slant correction (c) image (b) after smoothing

Figure 5.7. Applying rule-based smoothing after slant correction.

5.4 Comparison with a Structural Method

In order to show the effectiveness of the proposed method, it is compared with a structural slant correction technique (Shridhar and Kimura, 1995) which employs statistics of chain-coded image this way: the chain code of entire border pixels of the binarized image is extracted first, and then the slant is computed by:

$$\theta = \frac{\pi}{2} - \tan^{-1}(\frac{n_1 + n_2 + n_3}{n_1 - n_3}) \quad] \quad (5.3)$$

where n_1, n_2 and n_3 denote the number of chain elements at angles 45, 90 and 135 respectively.

Both methods are applicable to Farsi and English words. Figure 5.8 illustrates that for a handwritten word, chain-code based method is not as accurate as the proposed histogram based method, but the latter is significantly slower, because shear transformation at various angles is a time consuming operation. For slanted word of Figure 5.8(a), n_1 = 200, n_2 = 59 and n_3 = 30; so $\theta \approx 30^o$ and the sheared image is shown in Figure 5.8(b); while the histogram based method returns $\theta = 45^o$, leading to the less slanted word of Figure 5.8(c).

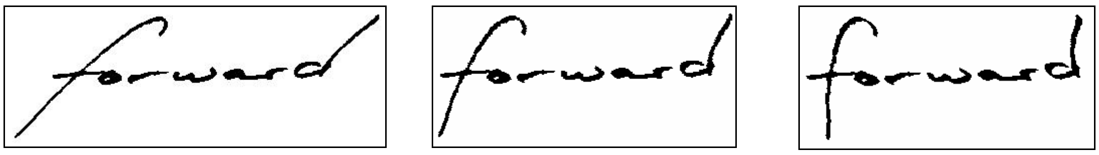

(a) A slanted word (b) Slant corrected word using chain-code based method (c) Slant corrected word using histogram based method

Figure 5.8. A comparison between two slant correction techniques for a handwritten English word.

Figure 5.9 shows that the chain-code based method fails in the presence of high noise; while the histogram based method still works properly.

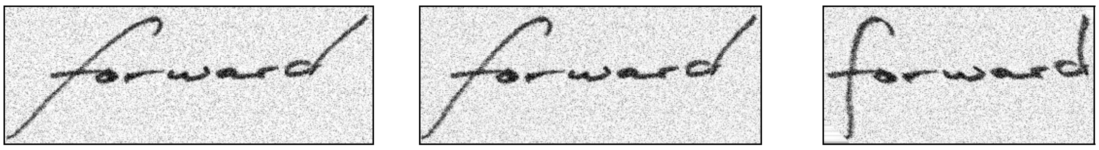

(a) A noisy slanted word (b) Slant corrected word using chain-code based method (c) Slant corrected using histogram based method

Figure 5.9. Structural slant correction methods tend to fail in the presence of high noise.

CHAPTER 6

SKELETONIZATION

6.1 Introduction

Skeletonization or medial axis transform (MAT) of a shape has been one the most surveyed problems in image processing and machine vision. A skeletonization (thinning) algorithm transforms a shape into arcs and curves of thickness one which is called skeleton. Ideally, the skeleton should retain basic structural properties of the original shape; it should be well-centered, well-connected (preserve connectivity information) and robust, and also allows a precise reconstruction (Ivanov et al., 2000). Over years, it has been found to be so difficult to get an algorithm that satisfies all of the requirements. There is no unique definition for skeleton, so different algorithms, with different definitions, produce different skeletons for the same shape.

By diminishing variability and distortion of instances of one class and reducing the amount of data to be handled, skeletonization simplifies classification. Skeletons have been proved to be effective in pattern recognition problems such as character recognition, fingerprint recognition, chromosome recognition and analyzing X-ray images. Skeletons provide compact representations that allow structural analysis of objects, and they have also applications in image compression.

The skeletonization techniques can be divided into two major categories (Ahmed, 1995): direct and indirect. The direct techniques produce skeletons by directly removing pixels from the pattern. The direct methods can be further classified into iterative and non-iterative. The iterative direct techniques compute skeletons by iteratively deleting removable boundary pixel either sequentially (Naccache and Shinghal, 1984) or parallel (Zhang and Suen, 1984) (Figure 6.1), until it causes no further changes to the image. A pixel is tested and marked to be removed if its neighbors (usually 8-neighbors) satisfy certain conditions. In sequential algorithms pixels are tested in a fixed order in all iterations and removing a pixel in an iteration depends on the resultant image of the previous iteration and the previous operations of this iteration. But, in parallel algorithms removing a pixel only depends on the result of the previous iteration, so all pixels can be tested independently in each iteration. The iterative methods yield thin and geometrically representative but not necessarily well-centered skeletons.

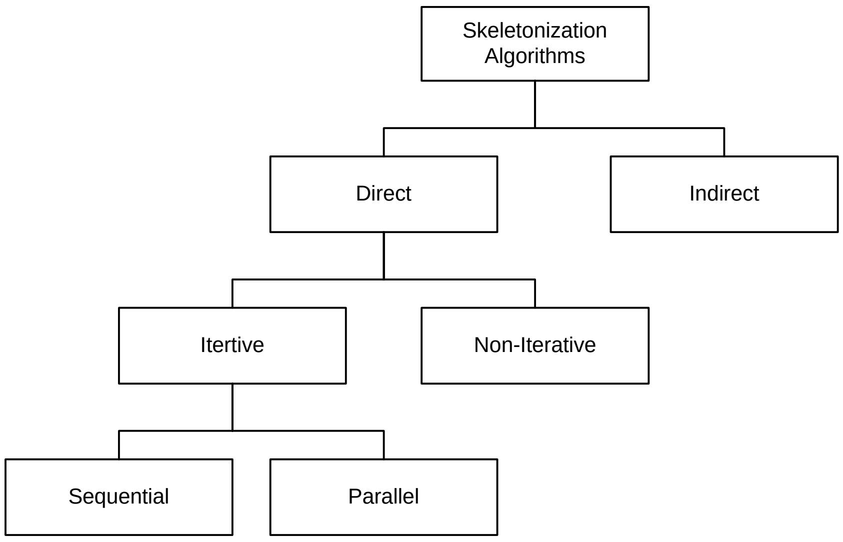


Figure 6.1. The classification of skeletonization algorithms.

The non-iterative techniques produce skeletons by connecting pixels having special properties. A pixel with special properties may be the middle pixel of a component of a scan line, the parts of polygonal regions where a pattern is divided into a set of regular or irregular polygons, etc. (Ahmed, 1995).

Indirect techniques are very similar to non-iterative techniques, and they are proved to perform better than some widely used direct techniques (Ahmed, 1995). Indirect techniques do not produce skeletons by removing or changing pixels, they rather construct skeletons by computing appropriate logical properties such as distributions of pattern pixels. In (Ahmed, 1995) an indirect technique is presented in which the skeleton is constructed by dividing shape pixels into a set of adjacent clusters and then connecting their centers. The cluster centers are computed by a modified version of the self-organizing feature map (SOM) algorithm.

Conventional skeletonization techniques implicitly assume connectivity of pixels inside image region, performing poorly on sparse (non-connective) shapes. The sparseness within image regions may be due to aging, uneven lighting or thresholding, and in document images, it may also occur because of poor ink quality. In (Singh et al., 2000) an indirect method utilizing SOM for the skeletonization of sparse shapes is introduced. The method requires neither well-separated shapes from background nor connectivity inside regions, so it can be used in developing robust vision systems. Given the pixel distribution of a shape, a piecewise-linear approximation of the shape skeleton is iteratively evolved by using a minimum spanning tree-based SOM. The adjacency relationships between the shape regions are detected and used in the evolution of the skeleton by constraining the SOM to lie on the edges of the Delaunay triangulation of the shape distribution. The final skeleton is obtained when the SOM converges. The method is invariant to Euclidean transformations and adaptive in terms of the topology of the shape distribution and in the number of map units.

Skeletonization algorithms are notorious for being slow on ordinary serial computers, and most of them suffer from irrational memory and CPU usage. These disadvantages are more pronounced for large images. For example, a drawing of standard A3 size, scanned at the typical resolution of 600 dpi, would be approximately 7000 × 10000 pixels and require 8.3 Mb of memory if treated as a binary image, which makes random access to different parts of the image very slow (Ivanov et al., 2000). However, for a typical word image of size 200 × 200, neither memory nor CPU inefficient usage is essential, and by using faster ubiquitous hardware, almost all algorithms are practical for text recognition. In (Ivanov et al., 2000) a fast and efficient skeletonization algorithm for large images is presented. Its main idea is to generate a special polyline for each raster line considering them in top to down direction, and then constructing the skeleton from points of these polylines. The obtained skeletons are precisely reconstructable, and the amount of required memory depends linearly on original image width, but not its area.

Ji and Piper (Ji and Piper, 1992) have developed a skeletonization algorithm by finding the points whose removal do not alter homotopy of the input image. They have proved that the Hilditch's condition is a sufficient condition for removing a single point from a binary image without altering its homotopy. The mathematical morphology operators erosion and dilation are used to construct skeletons. The computational complexity of the algorithm is $O(n^2)$ and the memory requirement is $O(n)$, where n is the linear scale of the image. The method is fast and can produce reconstructable and thin, but not necessarily of unitary thickness, skeletons.

There are hundreds of skeletonization algorithms in the literature. Of course it is not possible to implement and experiment with all of them. For the skeletonization of the Farsi script, five algorithms were implemented: two classical methods (SPTA

(Naccache and Shinghal, 1984) and Zhang-Suen's (Zhang and Suen, 1984)), DTSA (Sajjadi, 1996) designed for the Farsi scripts, one homotopy-preserving method (Ji and Piper, 1992) and the fully parallel Huang et al.'s method (Huang et al., 2003). In the rest of this chapter, we briefly describe each of these methods, showing that Huang's is better than others for the purpose of this work. Finally, a simple and effective skeleton post-processing procedure is described.

6.2 The SPTA

The Safe-Point Thinning Algorithm (SPTA) (Naccache and Shinghal, 1984) is a sequential method and like other iterative algorithms consists of iteratively deleting edge-points (points along the edges of a shape) while keeping end-points (points at the ends of a stroke), and also the shape connectedness should not be broken and excessive erosion (iteratively removing a stroke) should not be occurred.

Thinning is normally applied to binary images, and produces a binary image as output. Hereafter, it is assumed that shape pixels are represented by black pixels and background pixels are represented by white pixels. For a point p with the coordinate (x,y), the set of points with coordinates (x+1,y), (x-1,y), (x,y-1) and (x,y+1) are called its 4-neighbours, and its 8-neighbors are the set of points with coordinates (x+1,y), (x+1,y-1), (x,y-1), (x-1,y-1), (x-1,y), (x-1,y+1), (x,y+1) and (x+1,y+1) (Figure 6.2).

n_3	n_2	n_1
n_4	p	n_0
n_5	n_6	n_7

Figure 6.2. A point p and its 8-neighbors (n_0 to n_7).
The points n_0, n_2, n_4 and n_6 are also referred to as 4-neighbors of p.

In the SPTA, an edge-point is defined as a black pixel with at least one white 4-neighbor, an end-point is defined as a black point with at most one black 8-neighbor and a break-point is defined as a point whose deletion would break the connectedness of the pattern. The algorithm in each pass flags a point if it is an edge-point but not an end-point, nor a break-point, and nor must its possible deletion cause excessive erosion. All flagged points are removed at the end of a pass, and if there is no flagged point the procedure stops. An edge-point can be of one or more of the following types: 1) a left-edge point, having its left neighbor n_4 white; 2) a right-edge point, having its right neighbor n_0 white; 3) a top edge-point, having its top neighbor n_2 white; and 4) a bottom edge-point having its bottom neighbor n_6 white.

By examining different combinations of the 8-neighbors of a left-edge point p the authors have concluded that p can be safely removed (without breaking connectedness, end-point deletion and excessive erosion) if the boolean expression S_4 is true:

$$S_4 = n_0.(n_1 + n_2 + n_6 + n_7).(n_2 + \overline{n}_3).(n_6 + \overline{n}_5) \tag{6.1}$$

A boolean variable has the true value if its corresponding point is black and unflagged. Similarly, for a right-edge point, trueness of the expression S_0, for a top-edge point, trueness of the expression S_2 and for a bottom-edge point, trueness of the expression S_6 are sufficient conditions for safe deletion of the corresponding edge-points.

$$S_0 = n_4.(n_5 + n_6 + n_2 + n_3).(n_6 + \overline{n}_7).(n_2 + \overline{n}_1) \tag{6.2}$$

$$S_2 = n_6.(n_7 + n_0 + n_4 + n_5).(n_0 + \overline{n}_1).(n_4 + \overline{n}_3) \tag{6.3}$$

$$S_6 = n_2.(n_3 + n_4 + n_0 + n_1).(n_4 + \overline{n}_5).(n_0 + \overline{n}_7) \tag{6.4}$$

Each pass in the SPTA involves two scans, where all black points (the shape points) are examined in each scan. The scanning sequence can be either row-wise or column-wise. The first scan of a pass, flags safely removable left-edge points and safely removable right-edge points. In the second scan of the pass, safely removable top-edge points and safely removable bottom-edge points are flagged. At the end of the pass, all flagged points are removed (become white).

6.3 The Zhang-Suen's Algorithm

This algorithm has been used as basis of comparison for skeletonization algorithms for many years. It is a fast and simple parallel iterative algorithm, meaning that the new value for a pixel can be calculated using only the values from the previous iteration.

Each pass in the algorithm involves two sub-iterations, where in a sub-iteration, certain points are flagged, and at the end of the sub-iteration if there is no flagged point the algorithm stops; otherwise the flagged points are removed and the next sub-iteration starts. In the first sub-iteration, a pixel is flagged if it satisfies all of the following four conditions:

1. Its connectivity number is one. The connectivity number C_n of a pixel p can be defined as the number of transitions from black (foreground) to white (background) within the pixel 8-neighbors. It has a value in the range of zero to four.
2. It has at least two and at most six black neighbors.
3. At least one of n_0, n_4 and n_6 is white.
4. At least of n_0, n_2 and n_6 is white.

Now if there is no flagged point the algorithm stops, otherwise all flagged point are removed and the second sub-iteration starts where it is the same as the first sub-iteration except for conditions 3 and 4:

3. At least one of n_0, n_2 and n_4 is white.
4. At least one of n_2, n_4 and n_6 is white.

As it will be shown later, the Zhang-Suen's algorithm sometimes removes the letter dots, which carry the necessary information to distinguish certain Arabic/Farsi letters from each other. Therefore, this skeletonization must not be used in the context of Arabic/Farsi text recognition. Actually, it always removes 2×2 squares and sometime cause excessive erosion.

6.4 The DTSA

To overcome the problems of the Zhan-Suen's algorithm for the Arabic/Farsi scripts, Sajaddi proposed Decision Table Skeletonization Algorithm (DTSA). This parallel iterative algorithm involves four sub-iterations in each pass, and all shape (black) pixels are examined in each sub-iteration. Certain points are flagged within a sub-iteration; at the end of the sub-iteration if there is no flagged point the algorithm stops; otherwise the flagged points are removed and the next sub-iteration starts.

In the first sub-iteration, each left-edge point for which the boolean expression D_4 is true is flagged. In the second sub-iteration, each bottom-edge point for which the boolean expression D_6 is true is flagged. In the third sub-iteration, each right-edge point for which the boolean expression D_0 is true is flagged. In the forth (last) sub-iteration, each top-edge point for which the boolean expression D_2 is true is flagged. Where the definitions of left-edge point, right-edge point, top-edge point and bottom-edge point are the same as those of the SPTA and:

$$D_0 = S_0.(n_7 + n_1 + (n_2 + n_6 + n_3 \oplus n_5).(n_3 + n_5 + \overline{n}_2.\overline{n}_6)) \qquad (6.5)$$

$$D_2 = S_2.(n_1 + n_3 + (n_4 + n_0 + n_5 \oplus n_7).(n_5 + n_7 + \bar{n}_4.\bar{n}_0)) \qquad (6.6)$$

$$D_4 = S_4.(n_3 + n_5 + (n_6 + n_2 + n_7 \oplus n_1).(n_7 + n_1 + \bar{n}_6.\bar{n}_2)) \qquad (6.7)$$

$$D_6 = S_6.(n_5 + n_7 + (n_0 + n_4 + n_1 \oplus n_3).(n_1 + n_3 + \bar{n}_0.\bar{n}_4)) \qquad (6.8)$$

6.5 The Huang et al.'s Algorithm

Huang et al. (Huang et al., 2003) have proposed a fully parallel thinning algorithm which involves one iteration in each pass. It uses the information of 3×3 windows (i.e. the state of 8-neighbors) like the previous iterative algorithms, but in order to preserve connectivity, 3×4, 4×3 and 4×4 masks are also used. The algorithm is very efficient and robust to noise of contour.

All the following rules are applied simultaneously to each pixel *p* to determine whether it should be flagged or not:

- If *p* has zero, one or eight black neighbors, it is not flagged.
- If *p* has two black neighbors,

  It is flagged if the two neighbors are consecutive, i.e. *n*0 and *n*1 are black, or *n*1 and *n*2 are black, or *n*2 and *n*3 are black, …, or n7 and n0 are black.
- If *p* has three black neighbors,

  It is flagged if the three neighbors are consecutive, or if they match any of the following templates:

0	1	0
1	*p*	0
1	0	0

0	1	0
0	*p*	1
0	0	1

1	1	0
0	*p*	1
0	0	0

0	1	1
1	*p*	0
0	0	0

  Where 1 denotes a black and 0 denotes a white pixel.
- If *p* has four black neighbors,

  It is flagged if the four neighbors are consecutive, or if they match any of the following templates:

1	1	0
0	*p*	1
0	0	1

0	1	1
1	*p*	0
1	0	0

- If p has five black neighbors,

  It is flagged if the five neighbors are consecutive.
- If p has six black neighbors,

  It is flagged if the six neighbors are consecutive.

- If p has seven black neighbors,

  It is flagged if its white neighbor is a 4-neighbor.

These rules remove two-pixel-width rectangular patterns, resulting in loss of information or pattern connectivity. To obviate this problem, the pixel *p* is preserved (not flagged) if it matches any of the following templates:

x	0	x
1	*p*	1
1	1	1
x	0	x

x	0	0
1	1	0
0	*p*	0
0	0	x

x	0	0
0	*p*	0
0	1	1
0	0	x

0	0	0	0
0	*p*	1	0
0	1	1	0
0	0	0	0

x	0	0	0
0	*p*	1	0
0	0	1	x

x	1	1	x
0	*p*	1	0
x	1	1	x

0	0	0	x
0	1	*p*	0
x	1	0	0

At the end of a pass, if there is no flagged pixel the algorithm stops; otherwise the flagged pixels are removed and the next pass starts.

6.6 Experimental Results

For evaluating the quality of the implemented skeletonization algorithms, the following items are considered: the width and connectivity of skeleton, excessive erosion and robustness to border noise. Rather than go into a long detailed explanation, ineffectiveness of the homotopy-preserving and Zhang-Suen's algorithms is simply shown by actual examples. In the first experiment, the algorithms are applied to the image of Figure 6.3(a), which contains simple geometrical objects. Figure 6.3(b) shows that the homotopy-preserving algorithm removes small objects, and Figure 6.4(d) shows that the Zhang-Suen's algorithm removes the 2×2 square and excessively erodes the two-pixel-width slanted line. The SPTA, DTSA and Huang et al.'s algorithms provide acceptable outputs. The results of the SPTA and DTSA are similar, but the former is more computationally expensive.

The algorithms are applied to the Farsi (Figure 6.4(a)) and English (Figure 6.5(a)) character set. As shown in Figure 6.4(b) and Figure 6.5(b), the homotopy-preserving algorithm does not preserve connectivity. The Zhang-Suen's algorithm removes some of the dots (Figure 6.4(d)), so some letters have the same skeleton, for example 'ت' and 'ث', which leads to misidentification. Also notice the skeleton of 'K', in the image of Figure 6.5(d), which has been excessively eroded. Thus, the homotopy-preserving and Zhang-Suen's algorithms are applicable neither for Arabic/Farsi nor for English. The other three algorithms produce acceptable skeletons for both character sets.

To compare the five algorithms in the presence of border noise, the image of Figure 6.6(a) is presented to each of them. Figure 6.6(b) shows that the homotopy-preserving algorithm is not robust to the border noise, and the skeleton is not of unitary thickness.

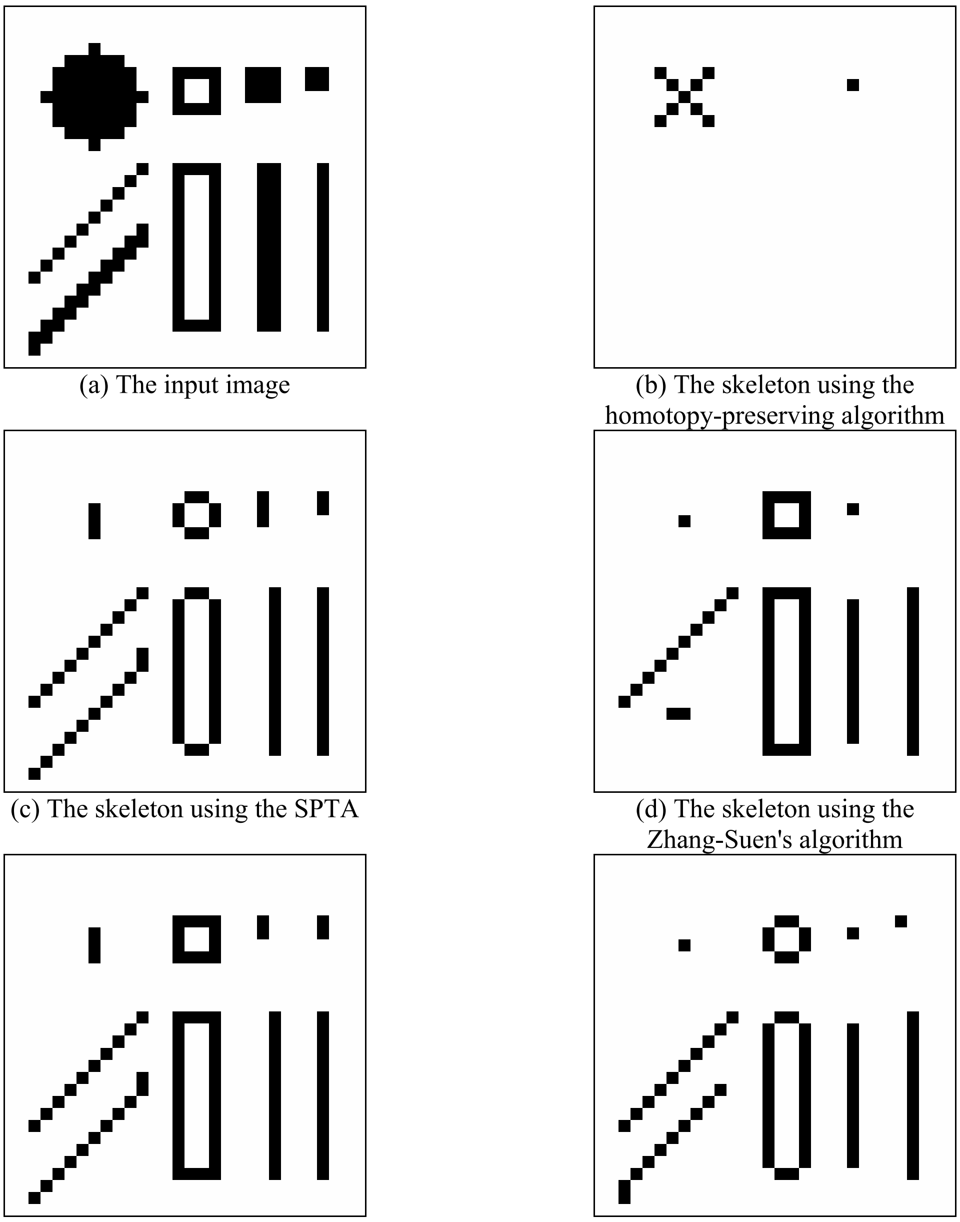
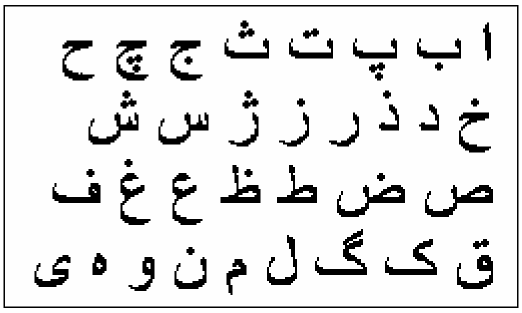
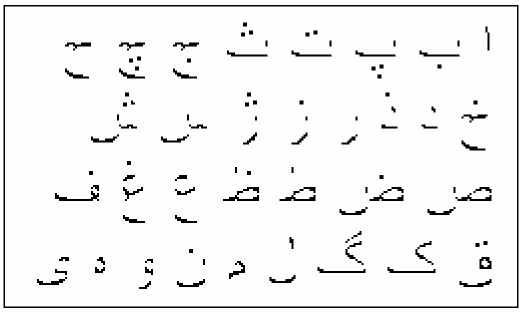
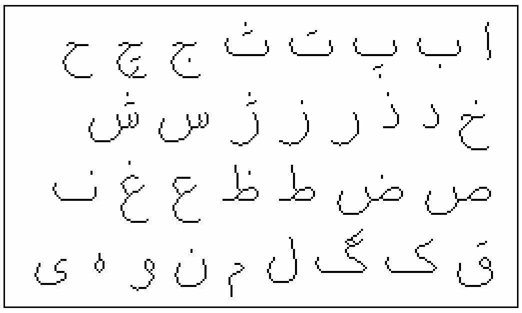

(a) The input image
(b) The skeleton using the homotopy-preserving algorithm
(c) The skeleton using the SPTA
(d) The skeleton using the Zhang-Suen's algorithm
(e) The skeleton using the DTSA
(f) The skeleton using the Huang et al.'s algorithm

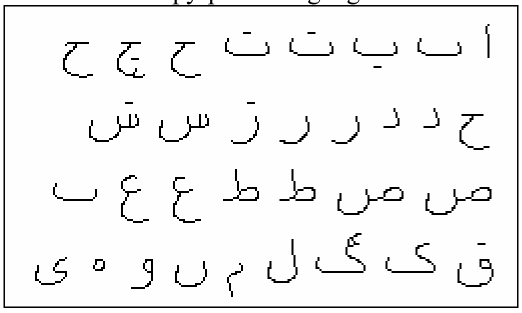
Figure 6.3. Applying the implemented skeletonization algorithm to an image containing simple geometrical patterns.

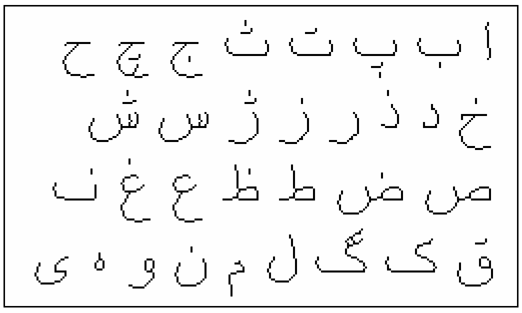
The Zhang-Suen's algorithm produce no spurious branch, meaning its robustness to the border noise, but as illustrated before, it has serious drawbacks that prevent its applicability. Among the other three algorithms, Huang et al.'s is more robust to noise; the resultant skeleton has only one spurious branch. Figure 6.6(e) shows that the DTSA is very sensitive to the border noise.


(a) The input image


(b) The skeleton using the homotopy-preserving algorithm


(c) The skeleton using the SPTA


(d) The skeleton using the Zhang-Suen's algorithm


(e) The skeleton using the DTSA

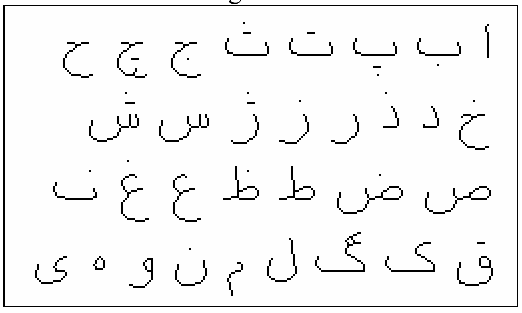
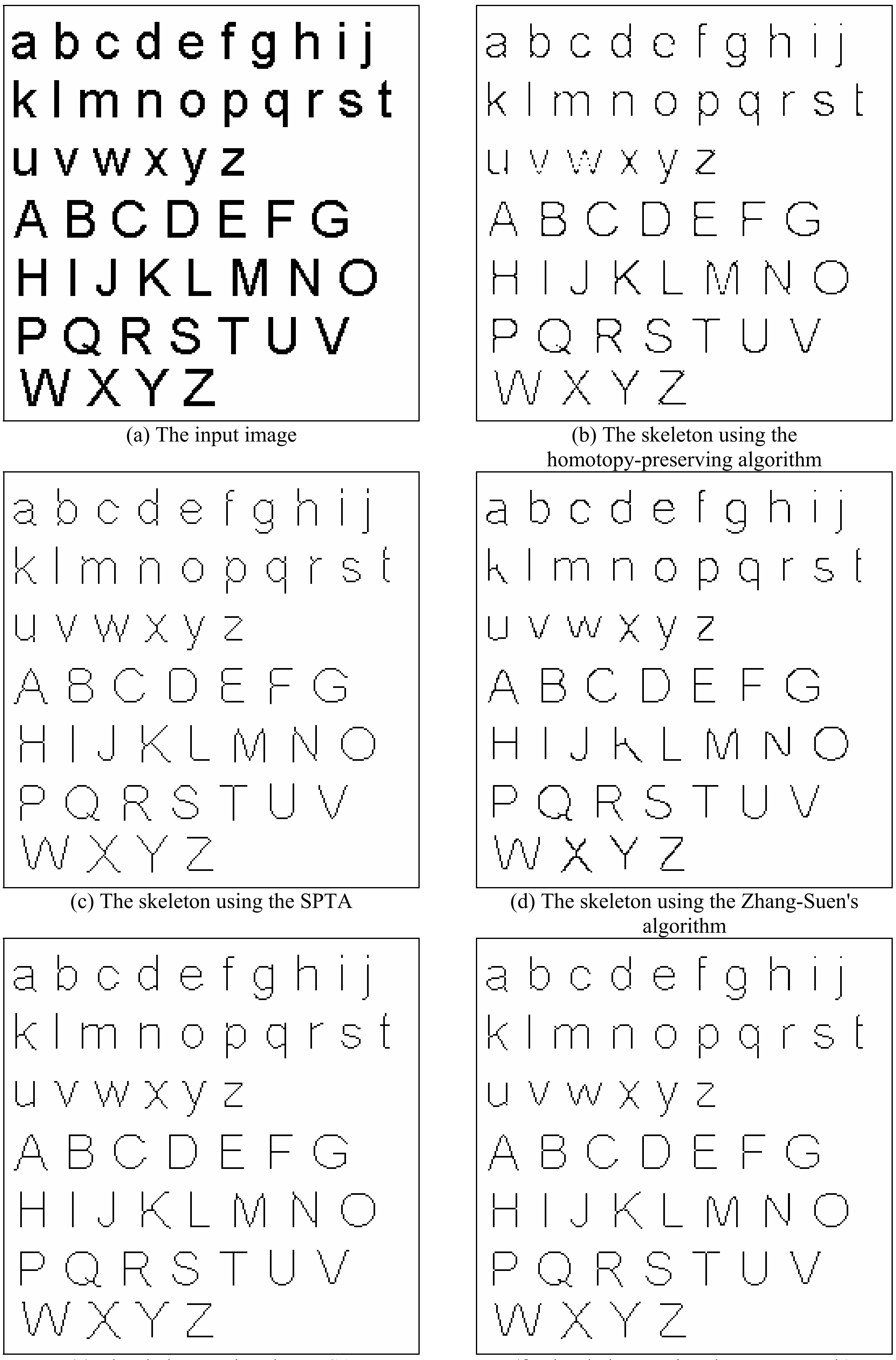

(f) The skeleton using the Huang et al.'s algorithm

Figure 6.4. Applying the implemented skeletonization algorithm to the Farsi character set.



(a) The input image

(b) The skeleton using the homotopy-preserving algorithm

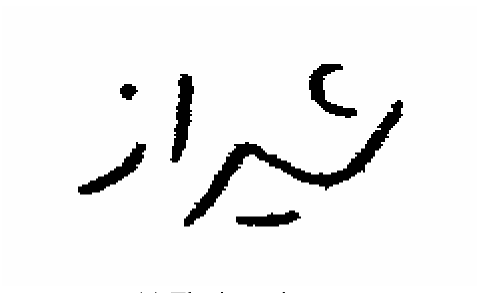
(c) The skeleton using the SPTA

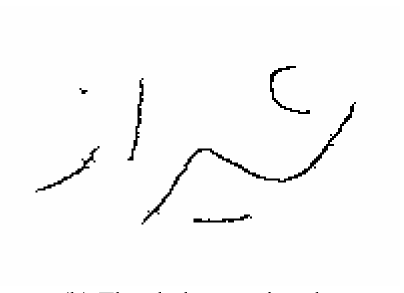
(d) The skeleton using the Zhang-Suen's algorithm

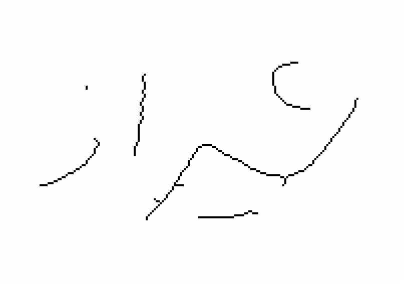
(e) The skeleton using the DTSA

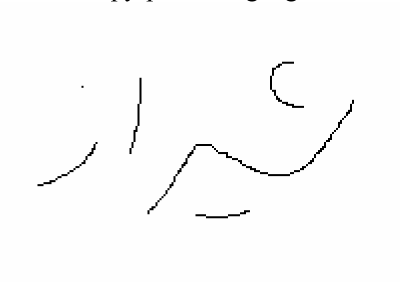
(f) The skeleton using the Huang et al.'s algorithm

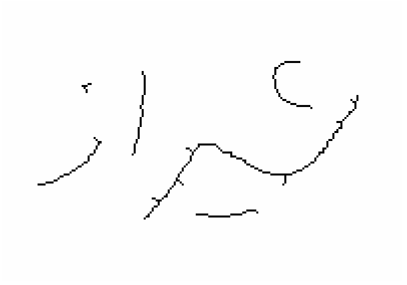
Figure 6.5. Applying the implemented skeletonization algorithm to the English character set.


(a) The input image


(b) The skeleton using the homotopy-preserving algorithm


(c) The skeleton using the SPTA


(d) The skeleton using the Zhang-Suen's algorithm


(e) The skeleton using the DTSA

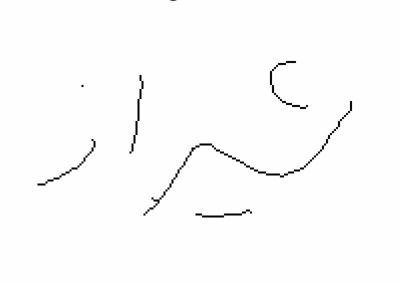

(f) The skeleton using the Huang et al.'s algorithm

Figure 6.6. Applying the implemented skeletonization algorithm to a Farsi word image with noisy border.

Overall, these experiments show the superiority of the Huang et al.'s algorithm over others, as verified by many other experiments.

6.7 Postprocessing

A skeletonization algorithm usually produces a distorted skeleton with some spurious branches which need a postprocessing step to be removed. The technique described here uses the maximum circle idea. Since the local features of the pattern are affected by the algorithm, the original pattern is also used to modify the skeleton.

Definition 6.1. A feature-point is a black pixel in the skeleton having a connectivity number other than two; i.e. p is a feature-point if and only if $C_n(p) \neq 2$.

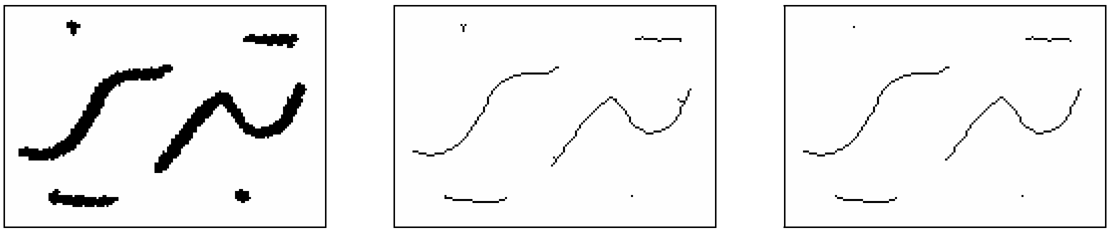
Definition 6.2. An end-point is a feature-point having a connectivity number of one; i.e. p is an end-point if and only if $C_n(p) = 1$. An end-point can be deleted without affecting the pattern connectivity.

The algorithm is as follows: first, for each end-point *ep*, the radius R_{ep} of the largest circle of black pixels within the original image that is centered at *ep* is evaluated (Algorithm 6.1). Then, the nearest non end-point *nep* to *ep* is found, and the link between *ep* and *nep* is removed if dist(*ep*,*nep*) < $R_{ep} + R_{nep}$. Where R_{nep} is the radius of the largest circle of black pixels within the original image that is centered at *nep*.

Figure 6.7 shows the advantage gained by the postprocessing step.

```
Let I[r,c] be the binary input image having R rows and C columns, and the
background is represented by zeros.
Let Center($r_c$,$c_c$) be the center of the largest circle of black pixels.


maxRadius = min(min($r_c$, R – $r_c$ – 1), min($c_c$, C – $c_c$ – 1));

if I[$r_c$,$c_c$] == 0
{
  return 0;
}

for r = 1 to maxRadius
{
  for $r_1$ = $r_c$ – r to $r_c$ + r
  {
    for $c_1$ = $c_c$ – r to $c_c$ + r
    {
      if $\sqrt{(r_1 - r_c)^2 + (c_1 - c_c)^2} \le r$ AND I[$r_1$,$c_1$] == 0
      {
        retrun r;
      }
    }
  }
}

return r;
```

Algorithm 6.1 Evaluating the radius of largest circle of black pixels at a point



(a) The input image

(b) The skeleton using the Huang et al.'s algorithm

(c) The skeleton after postprocessing

Figure 6.7. Postprocessing after skeletonization.

CHAPTER 7

STRUCTURAL FEATURES OF ARABIC/FARSI WORDS

7.1 Introduction

A method to extract structural features from Arabic/Farsi word images is presented in this chapter. Structural features are capable of tolerating many variations, but they are not robust to noise, and hard to extract. Since the recognition is based on 1D HMMs, the features must preserve the sequential characteristics of words, meaning that a 2D word image must be converted to a 1D signal so that the relative ordering of the characters is retained. The basic idea of the proposed method is based on the techniques described in (Khorsheed, 2000; Almuallim and Yamaguchi, 1987). The skeleton of a word image is decomposed into a number of links in a certain order. Then, each set of links that from a loop (cycle) is replaced with a special link representing the loop. Each link is then represented by a 10D feature vector. The features are the curvature of the link, its length relative to the word height, the position of the its two ends relative to the first row of the image, the connection type and four curved features. The features are irrespective of the baseline location, so the difficult and crucial problem of baseline detection is avoided

7.2 Preprocessing

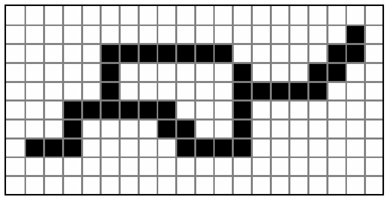
The preprocessing step has two duties: 1) normalization for word height; 2) skeleton modification. Before skeletonization the input word is resized to have a height of 128 pixels. This is done by detecting the word area that is the minimum rectangle containing the word in the input image. In order to be robust to noise, the upper side of the rectangle is set to be the first row having a horizontal white run-length with a length of higher than 2, and the lower side of the rectangle is set to be the last row having a horizontal white run-length with a length of higher than 2. The left side of the rectangle is set to be the first column having a vertical white run-length with a length of higher than 2, and similarly the right side of the rectangle is set to be the last column having a vertical white run-length with a length of higher than 2.

Before link extraction, the skeleton is modified to have as small pixels as possible. A pixel p is removed if matches any of the following templates or their rotations at angles 90^o, 180^o and 270^o:

0	1	x
1	*p*	0
x	0	0

x	1	x
1	*p*	1
x	0	x

These rules actually remove 4-connectivity of the pattern, and they must be applied sequentially in order to preserve 8-connectivity. Figure 7.1 illustrates how these rules modify a skeleton by removing some pixels.


(a) The input skeleton

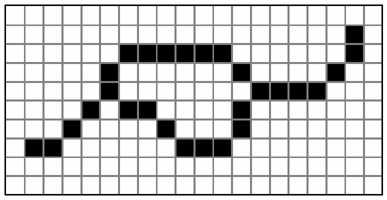


Figure 7.1. Skeleton modification by removing pixels that match the templates.

7.3 Link Extraction

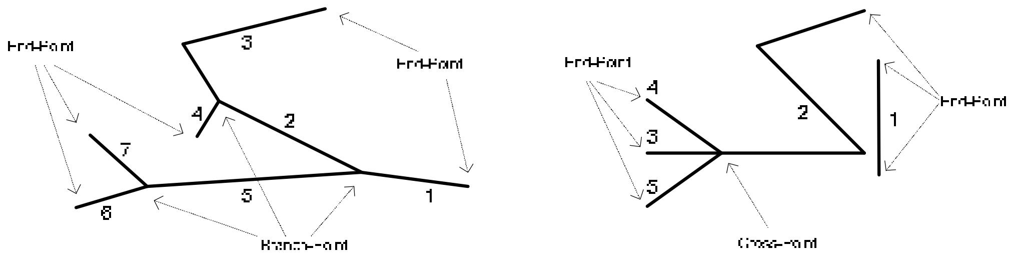
In speech recognition and online handwritten recognition, the input signal is one-dimensional itself, but here the 2D word image must be converted to a 1D observation sequence. This is done by tracing the skeleton of the word image to extract its links. A link is a set of neighboring pixels between two feature-points. So a link is extracted by starting from a feature-point and then moving from the current pixel to its adjacent until reaching another feature-point. The process is started from the right-most end-point. In order to extract the links in a canonical order, the following two rules are applied: 1) if fp_1 and fp_2 are two feature-points such that fp_1 is located to the right of fp_2, then all links branching from fp_1 must be extracted before any of the branching links from fp_2; 2) the first link that must be visited, from the links branching from a feature-point fp, is the one that makes the minimum angle with the current link (ending at fp) and the other branching links must be visited in a clockwise order.

Figure 7.2 shows two examples of visiting links in the canonical order. Different types of feature-points are also shown in the Figure.

Definition 7.1. A dot is a feature-point having a connectivity number of zero; i.e. p is a dot if and only if $C_n(p) = 0$.

Definition 7.2. A branch-point is a feature-point having a connectivity number of three; i.e. p is a branch-point if and only if $C_n(p) = 3$.

Definition 7.3. A cross-point is a feature-point having a connectivity number of four; i.e. p is a cross-point if and only if $C_n(p) = 4$.



(a) (b)

Figure 7.2. Two examples of visiting links in the canonical order. Lower-numbered links are visited before higher-numbered ones.

7.4 Loop Extraction

Loop extraction makes the number of strokes smaller, thus leading to lower computational cost and easier modeling. Loops are important distinctive features as a number of letters have loops inside. A loop can be of any of the following types: 1) simple-loop, which is a single link beginning from a feature-point and returning to the same point again; 2) multi-link-loop, which is a loop consisting of two or more links forming a closed path; 3) double-loop, which is a loop that contains one or more other loops (Figure 7.3). Simple-loop can be seen in letters 'صـ', 'ص', 'ضـ', 'ض', 'ط', 'ظ', 'فـ', 'ف', 'قـ', 'ق', 'مـ', 'م', 'و', 'ه', 'ـه' and sometimes 'جـ', 'ج', 'چـ', 'چ', 'حـ', 'ح', 'خـ', and 'خ'. Multi-link-loop can be seen in letters 'ـصـ', 'ـص', 'ـضـ', 'ـض', 'ط', 'ـط', 'ظ', 'ـظ', 'ـعـ', 'ـع', 'ـغـ', 'ـغ', 'ـفـ', 'ـف', 'ـقـ', 'ـق', 'ـمـ', 'ـم', 'ـو', 'ـه' and sometimes 'ـجـ', 'ـج', 'ـچـ', 'ـچ', 'ـحـ', 'ـح', 'ـخـ' and 'ـخ'. Double-loop can be seen in letters 'هـ' and 'ـهـ'. Simple-loops are straightforward to detect; Algorithm 7.1 is to find multi-link-loops and double-loops in a graph with vertices corresponding to feature-points of a word skeleton and edges corresponding to the links between the feature-points. We use the edge-list representation to describe a word graph.

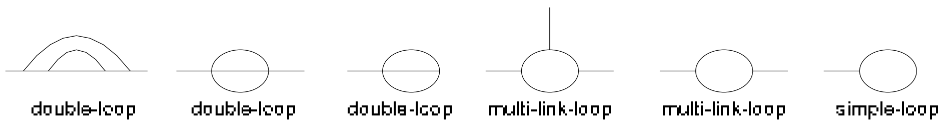


Figure 7.3. Examples of different types of loops.

```
Let G(V,E) be the graph with the set of vertices V, and the set of edges E.
Let L be a list for DFS, where L[0] denotes the first element.

for each vertex v in V
{
  L.push_front(v);
  while L is not empty
  {
    u = L.front(); // = L[L.size()-1].
    Let e(u,t) be an unvisited neighboring edge of u.
    if no such edge exists
    {
      L.pop_front(); // remove u from the list.
    }
    else
    {
      Mark e as visited.
      if there is an unvisited edge d between t and L[i], a vertex in L
      {
        Mark d as visited.
        Now, {(L[i],L[i+1]), (L[i+1],L[i+2]), ...,
          (L[L.size()-2],L.front()), (L.front(),t), (t,L[i])} is a cycle.
        Replace the cycle with a special cyclic edge c (an edge that
          represents the cycle).
        if the cycle has already contained a cyclic edge
        {
          c is marked as a double-cyclic edge (or a double-loop).
        }
      }
      L.push_front(t);
    }
  }
}
```

Algorithm 7.1. A DFS algorithm for detecting multi-link-loops and double-loops in a word graph.

7.5 Structural Features

After forming the word graph, each edge, corresponding to a link or a loop of the original word, is transformed into a 10D feature vector. The features have the following descriptions:

- Normalized length feature (f_1): The length of an edge (the number of its pixels) divided by the height of the word image (128). This feature is defined to be 2 for loops, and 0 for dots. f_1 is invariance against translation, rotation and scaling.

- Curvature feature (f_2): The curvature of an edge, defined as the proportion of the Euclidean distance between the two vertices of the edge by its actual length. Thus, the curvature becomes zero for a simple loop, and one for a straight line. This feature is defined to be 2 for multi-link-loops, 3 for double-loops, and 4 for dots. f_2 is invariance against translation, rotation and scaling.
- Slope feature (f_3): The slope of the line between the two vertices of an edge partitioned into 8 equal interval, labeled 1, 2, ..., 8. This feature is defined to be 0 for loops and dots. f_3 is invariance against translation and scaling.
- Connection type feature (f_4): The (connection) type of the two endpoints of an edge (to the previous and next edges). It has one of the following values:

Value	Type of beginning vertex	Type of ending vertex
0	end-point	end-point
1	end-point	branch-point
2	end-point	cross-point
3	branch-point	end-point
4	branch-point	branch-point
5	branch-point	cross-point
6	cross-point	end-point
7	cross-point	branch-point
8	cross-point	cross-point
9	when the edge is a dot	
10	when the edge is a loop	

  f_4 is invariance against translation, rotation and scaling.
- Endpoint distance feature (f_5): The normalized distance from the more distance vertex of an edge, from to the middle row of the word image, to the first row. This feature is defined to be 0 for loops, and helps determining whether a dot is above or below a character. f_5 is invariance against horizontal translation and scaling.
- Number of segments feature (f_6): The number of segments of the polyline fitted to an edge (Algorithm 7.2). f_6 is invariance against translation, rotation and scaling.
- Curved features (f_7-f_{10}): Percentage of pixels above the top feature-point, below the bottom feature-point, left of the left feature point, and right of

the right feature point respectively. These features are invariance against translation and scaling.

```
Let p be the set of vertices to be fitted by a polyline, where p[0] denotes
the first element.

first = 0; // index of the first point of the current line segment
last = 0; // index of the last point of the current line segment

for current = 1 to size(p)
{
  d = $\sum_{i=first}^{current}$ (perpendicular distance between p[i] and the straight connecting
             p[first] and p[current]);

  if d > (current – first + 1) * ERROR
  {
    p[first], p[first+1], ..., p[last] is a line segment.
    first = last;
  }
  last = current;
}

p[first], p[first+1], ..., p[last] is the last line segment.
```

Algorithm 7.2. Fitting a polyline to a set of points.

CHAPTER 8

HIDDEN MARKOV MODELS FOR HANDWRITTEN WORD RECOGNITION

8.1 Introduction

The output of a real-world process may be observed in the form of a continuous or discrete signal. A primary problem of interest is to build models for such real-world signals. A model for a signal is accompanied by several advantages. First, it provides the basis for a theoretical description of a signal processing system which can be used to process the signal to have a desired output. Second, a model can provide valuable information about the signal source without having to have the source available. Finally and most importantly, models actually work well and enable us to realize important practical systems (Rabiner, 1989).

There are several ways to model a signal depending on its type and properties. Generally, a signal model can be deterministic or stochastic (statistical). The deterministic models use some known properties of the signal and estimate parameter values of the model. On the other hand, in the statistical models, a parametric random process characterizes the signal. For applications such as speech recognition and handwritten recognition that are accompanied by uncertainty, stochastic models achieve better performance. The Hidden Markov Model (HMM), also referred to as Markov sources or probabilistic functions of Markov chains in the communication literature, is a widely used statistical model.

In this chapter, first we review the basic theory of Markov models, and then explaining HMMs. Finally, the theory is extended to continuous HMMs. All mathematical formulations needed to be implemented and some implementation issues are discussed.

8.2 Markov Models

An important class of stochastic processes is Markov processes, which has some special properties making them mathematically manageable. It is often desirable to analyze a sequence of random variables that are not independent, but rather the value of each variable depends on previous elements in the sequence. In a Markov process, the value of the current random variable is adequate to predict the value of future random variables i.e. future behavior of the process. In other words, future elements of the sequence are conditionally independent of past elements, given the present element. Let $X = (X_1, \ldots, X_T)$ be a sequence of random variable taking values in the finite state space S = $\{s_1, \ldots, s_N\}$. The Markov properties are:

$$P(X_{t+1} = s_k \mid X_1, X_2, ..., X_t) = P(X_{t+1} = s_k \mid X_t) \tag{8.1}$$

$$P(X_{t+1} = s_k \mid X_t) = P(X_2 = s_k \mid X_1) \tag{8.2}$$

The second property is called time invariance. If the sequence X has both Markov properties, it is said to be a Markov chain.

A Markov chain can be completely descried by the stochastic initial state vector $\prod$ and the stochastic transition matrix A:

$$p_i = P(X_1 = s_i) \tag{8.3}$$

$$a_{ij} = P(X_{t+1} = s_j \mid X_t = s_i) \tag{8.4}$$

Where $p_i \geq 0, \forall i$, and $\sum_{i=1}^{N} p_i = 1$, and $a_{ij} \geq 0, \forall i, j$, and $\sum_{j=1}^{N} a_{ij} = 1, \forall i$.

To illustrate the ideas, consider an example about weather prediction which is about trying to guess what the weather will be tomorrow based on a history of weather observations in the past. For simplicity, assume that there are three types of weather: *Sunny*, *Cloudy* and *Rainy*, and the weather lasts all day, i.e. it doesn't change from one state to another in the middle of the day. If we make the Markov assumption (which is not valid in real world), then the 3-state finite state machine of Figure 8.1 with arbitrary state transition probabilities represents a Markov chain. Note that the sum of probabilities of outgoing arcs from each state is 1. From Figure 8.1 it is clear that a Markov model can be taught of as a nondeterministic finite state machine with probabilities attached to arcs.

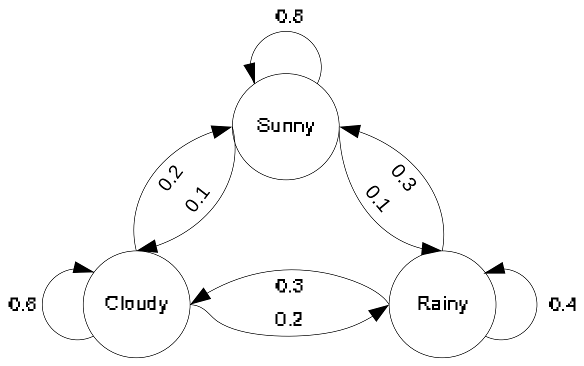


Figure 8.1. A Markov model for the weather prediction example.

Let s_1 = *Sunny*, s_2 = *Cloudy* and s_3 = *Rainy*, and the weather on the first day be Sunny. Then:

$$\prod = (1.0, 0.0, 0.0)$$

$$A = \begin{bmatrix} 0.8 & 0.1 & 0.1 \\ 0.2 & 0.6 & 0.2 \\ 0.3 & 0.3 & 0.4 \end{bmatrix}$$

The probability of a sequence of states $X_1, \ldots, X_K$ is easily calculated for a Markov chain:

$$\begin{aligned} P(X_1, ..., X_K) &= P(X_1)\, P(X_2 \mid X_1)\, P(X_3 \mid X_1, X_2) \ldots P(X_K \mid X_1, ..., X_{K-1}) \\ &= P(X_1)\, P(X_2 \mid X_1)\, P(X_3 \mid X_2) \ldots P(X_K \mid X_{K-1}) \\ &= p_{X_1} \prod_{t=1}^{K-1} a_{X_t X_{t+1}} \end{aligned} \quad (8.5)$$

So in the above example, the probability that the weather for the next seven days will be *Sunny*, *Sunny*, *Rainy*, *Rainy*, *Sunny*, *Cloudy*, *Sunny*, or more formally the probability of the observation sequence $O = s_1, s_1, s_3, s_3, s_1, s_2, s_1$, can be calculated as:

$$\begin{aligned} P(O \mid \text{Model}) &= p_1 P(s_1 \mid s_1)\, P(s_1 \mid s_1)\, P(s_3 \mid s_1)\, P(s_3 \mid s_3)\, P(s_1 \mid s_3)\, P(s_2 \mid s_1)\, P(s_1 \mid s_2) \\ &= p_1\, a_{11}\, a_{11}\, a_{13}\, a_{33}\, a_{31}\, a_{12}\, a_{21} \\ &= 1.0\,(0.8)\,(0.8)\,(0.1)\,(0.4)\,(0.3)\,(0.1)\,(0.2) \\ &= 1.536 \times 10^{-4} \end{aligned} \quad (8.6)$$

Generally, when we talk about Markov models, we mean first-order Markov models in which a history of size one is used to predict future behavior. But, sometimes the future states require a larger history in order to be predicted. In an n^{th} order Markov model, *n* previous states are used to predict the next state. In general, by elaborating the state space as a cross-product of the finite previous states, every n^{th} order Markov model can be encoded in a first-order Markov model. So theoretically, the first-order Markov assumption is not restrictive.

8.3 Hidden Markov Models

HMMs are powerful tools in the field of signal processing. Despite their limitations, variants of HMMs are still the most widely used technique in modern speech recognition systems. The similarity between speech and handwritten text, which both are made up of symbols with ambiguous boundaries and variations in appearance, has suggested extending the application of the HMMs to handwritten text recognition. The HMM does not model the whole pattern as a single feature vector; rather, it explores the relationship between consecutive segments of a pattern, since each segment is relatively smaller and easier to be characterized (He and Kundu, 1991).

A HMM can be considered as a nondeterministic finite state machine where each state is associated with a random function. Within a discrete period of time *t*, the model is assumed to be in some state and generates an observation by a random function of the state. Based on the transition probability of the current state, the underlying Markov chain changes to another state at time *t*+1. The state sequence that the model passes through is unknown, only some probabilistic function of the state sequence that is the observations produced by the random function of each state can be seen. A HMM can also be considered as a double stochastic process or a partially observed stochastic process. A HMM is characterized by the following elements:

N: The number of states of the model (8.7)

$S = \{s_1, s_2, ..., s_N\}$: The set of states (8.8)

$\prod = \{p_i = P(s_i \text{ at } t = 1)\}$: The initial state probabilities (8.9)

$A = \{a_{ij} = P(s_j \text{ at } t+1 \mid s_i \text{ at } t)\}$: The state transition probabilities (8.10)

M: The number of observation symbols (8.11)

$V = \{v_1, v_2, ..., v_M\}$: The set of possible observation symbols (8.12)

$B = \{b_i(v_k) = P(v_k \text{ at } t \mid s_i \text{ at } t\}$: The symbol emission probabilities (8.13)

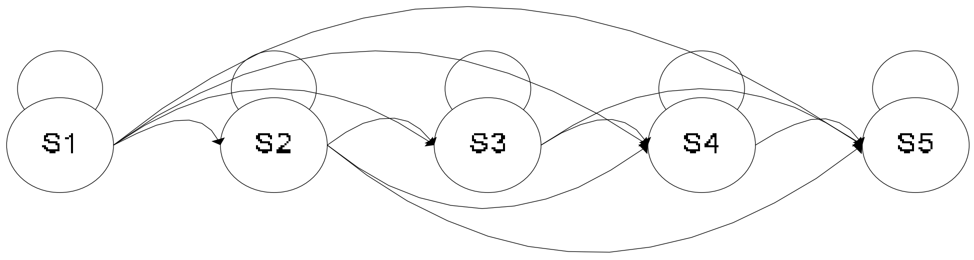
O_t: The observed symbol at time t (8.14)

T: The length of observation sequence (8.15)

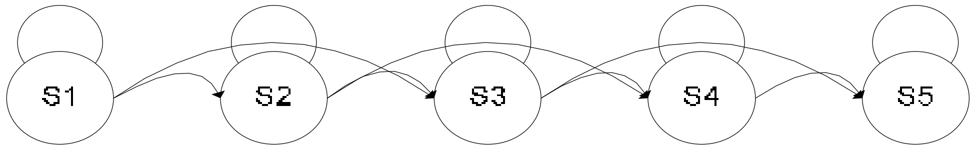
λ = (A, B, ∏): The compact notation to denote the HMM. (8.16)

Obviously, there are the following three constraints: $\sum_{i=1}^{N} p_i = 1$, $\sum_{j=1}^{N} a_{ij} = 1,\ \forall i$ and $\sum_{k=1}^{M} b_i(v_k) = 1,\ \forall i$.

The structure of the state transition matrix A determines the topology of the HMM. If $a_{ij} \neq 0\ \ \forall i, j$ meaning that each state in the model is reachable from any state within one transition, the model is called fully-connected or Ergodic. The widely used topology in speech/text recognition is the so called Left-to-Right (LR) or Bakis model in which lower numbered states account for observations occurring prior to higher numbered states. The temporal order in LR-HMMs is imposed by introducing structural zeros to the model in the form of the constraint ∏ = {1, 0, ..., 0} and $a_{ij} = 0,\ i > \mathrm{j}$ meaning that the model begins at the first (i.e. left most) state and at each time instant it can only proceed to the same or a higher numbered state. As a further constraint, in LR-HMM the number of forward jumps at each state is often limited in order to restrict large state changes, i.e. $a_{ij} = 0,\ \ j > i + \Delta$ for some fixed Δ (Figure 8.2).



(a) A 5-state Left-to-Right HMM



(b) A 5-state Left-to-Right HMM with maximum relative forward jump of 2

Figure 8.2. Left-to-Right HMMs.

The following example helps understand the application of HMMs. Suppose you were locked in a room for several days, and you were asked about the weather outside. The only available piece of information is whether the person who comes into the room giving your daily meal is carrying an umbrella or not, so $V = \{True, False\}$ for the observation of carrying umbrella. Let's assume $P(Umbrella \mid Sunny) = 0.1$, $P(Umbrella \mid Cloudy) = 0.3$ and $P(Umbrella \mid Rainy) = 0.7$. We want to draw conclusion form our observations (carrying an umbrella or not) about the weather outside as it is hidden from us. Let w_i be the weather condition on day i, and the boolean value of u_i mean whether you see an umbrella on the same day. Using Bayes' rule:

$$P(w_1,\dots,w_n \mid u_1,\dots u_n) = \frac{P(u_1,\dots u_n \mid w_1,\dots,w_n)P(w_1,\dots w_n)}{P(u_1,\dots,u_n)} \qquad (8.17)$$

The probability $P(w_1, \dots, w_n)$ is the same as the Markov model of the previous example, and $P(u_1, \dots, u_n)$ is the apriori probability of seeing a particular sequence of umbrella events. The probability $P(u_1,\dots,u_n \mid w_1,\dots,w_n)$ can be calculated as $\prod_{i=1}^{n} P(u_i \mid w_i)$ if we assume, for all i, given w_i, u_i is independent of all u_j and w_j for all $j \neq i$.

For the weather prediction, we can therefore omit the apriori probability $P(u_1, \dots, u_n)$ as it is independent of the weather. Based on the first-order Markov assumption, the likelihood L, a measure proportional to the probability, can be computed as:

$$\begin{aligned} &P(w_1,\dots, w_n \mid u_1, \dots, u_n) \propto \\ &L(w_1,\dots, w_n \mid u_1, \dots, u_n) = P(u_1,\dots,u_n \mid w_1,\dots, w_n)\, P(w_1, \dots, w_n) \\ &\qquad = \prod_{i=1}^{n} P(u_i \mid w_i) \prod_{i=1}^{n} P(w_i \mid w_{i-1}) \end{aligned} \qquad (8.18)$$

Suppose the day you were locked in was sunny. The next day, the person carried an umbrella into the room, and you would like to predict the weather on the next day.

First we calculate the likelihood assuming the next day to be sunny:

$$\begin{aligned} &L(w_2 = Sunny \mid w_1 = Sunny, u_2 = True) = P(u_2 = True \mid w_2 = Sunny)\,. \\ &\qquad P(w_2 = Sunny \mid w_1 = Sunny) = 0.1\ (0.8) = 0.08 \end{aligned} \qquad (8.19)$$

Then we calculate the likelihood assuming the next day to be cloudy:

$$\begin{aligned} &L(w_2 = Cloudy \mid w_1 = Sunny, u_2 = True) = P(u_2 = True \mid w_2 = Cloudy)\,. \\ &\qquad P(w_2 = Cloudy \mid w_1 = Sunny) = 0.3\ (0.1) = 0.03 \end{aligned} \qquad (8.20)$$

Finally for the next day to be rainy:

$$L(w_2 = Rainy \mid w_1 = Sunny, u_2 = True) = P(u_2 = True \mid w_2 = Rainy)\,.$$

$$P(w_2 = Rainy \mid w_1 = Sunny) = 0.7\ (0.1) = 0.07 \tag{8.21}$$

Thus, it is more likely that the next day was sunny.

8.4 The Three Fundamental Problems for HMMs

Most applications of HMMs need to solve the following problems:

Problem 1. Given a model $\lambda = (A, B, \prod)$, how do we efficiently compute $P(O \mid \lambda)$, the probability of occurrence of the observation sequence $O = O_1, O_2, ..., O_T$.

Problem 2. Given the observation sequence O and a model λ, how do we choose a state sequence $S = s_1, s_2, ..., s_T$ so that $P\ (O, S \mid \lambda)$, the joint probability of the observation sequence $O = O_1, O_2, ..., O_T$ and the state sequence given the model, is maximized. In other words, we want to find a state sequence that best explains the observation.

Problem 3. Given the observation sequence O, how do we adjust the model parameters $\lambda = (A, B, \prod)$ so that $P\ (O \mid \lambda)$ or $P\ (O, S \mid \lambda)$ is maximized. In other words, we want to find a model that best explains the observed data.

8.4.1 Solution to Problem 1

It deals with computing the probability that the model λ produces the observation sequence O. The most straightforward way to compute $P\ (O \mid \lambda)$ is to find $P\ (O \mid S, \lambda)$ for a fixed state S, multiply it by $P\ (S \mid \lambda)$, and then sum up over all possible state sequences of length T :

$$P(O \mid l) = \sum_S P(O \mid S, l).P(S \mid l) \tag{8.22}$$

Since $P(S \mid l) = p_{s_1} a_{s_1 s_2} a_{s_2 s_3} ... a_{s_{T-1} s_T}$ and $P(O \mid S, l) = b_{s_1}(O_1) b_{s_2}(O_2) .. b_{s_T}(O_T)$,

Equation (8.22) can be rewritten as:

$$P(O \mid l) = \sum_S p_{s_1} b_{s_1}(O_1) a_{s_1 s_2} b_{s_2}(O_2) ... a_{s_{T-1} s_T} b_{s_T}(O_T) \tag{8.23}$$

Computing the probability by Equation (8.23) is not practical since there are N^T state sequences, requiring $(2T\text{-}1)N^T$ multiplications and $N^T\text{-}1$ additions. Thus, an efficient procedure should be used instead. There are two alternatives: the forward procedure and the backward procedure.

The forward procedure calculates the forward variable $\alpha_t(s)$ for each state s defined as:

$$\alpha_t(s) = P(O_1, O_2, ..., O_t, s_t = s \mid \lambda) \tag{8.24}$$

That is the probability of the partial observation sequence up to time t and the state s, given the model λ. The following three-step procedure computes $\alpha_t(s)$ for all instances of time:

1. *Initialization*:

$$a_1(s) = p_s b_s(O_1),\ 1 \le s \le N \tag{8.25}$$

2. *Induction*:

$$a_{t+1}(r) = [\sum_{s=1}^{N} a_t(s) a_{sr}] b_r(O_{t+1}),\ 1 \le r \le N, 1 \le t \le T-1 \tag{8.26}$$

This calculates the forward probability of state r at time t+1 based on the joint probability of the previous forward variables from all states at time t and the transition probabilities from each of those states to state r. It is due to the fact that state r can be reached (with probability a_{sr}) independently from any of the N states at time t.

3. *Termination*:

$$P(O \mid l) = \sum_{s=1}^{N} a_T(s) \tag{8.27}$$

Calculating the forward variables over all states at all instances of time requires $N(N$-$1)(T$-$1)+(N$-$1)$ additions and $N + N(N+1)(T$-$1)$ multiplications, i.e. of the order of N^2T as compared to $2TN^T$ required for the direct method.

The backward procedure follows the same approach but in the opposite direction by calculating the backward variable $\beta_t(s)$ for each state s defined as:

$$\beta_t(s) = P(O_{t+1}, O_{t+2}, ..., O_T \mid s_t = s, \lambda) \tag{8.28}$$

That is the probability of the observation sequence from t+1 to T given the state s at time t and the model λ. Like $\alpha_t(s)$, $\beta_t(s)$ can be computed by the following three-step procedure for all instances of time:

1. *Initialization*:

$$b_T(s) = 1,\ 1 \le s \le N \tag{8.29}$$

2. *Induction*:

$$b_t(s) = \sum_{r=1}^{N} a_{sr} b_r(O_{t+1}) b_{t+1}(r),\ 1 \le s \le N, t = T-1, T-2, ..., 1 \tag{8.30}$$

3. *Termination*:

$$P(O \mid l) = \sum_{s=1}^{N} p_s b_s(O_1) b_1(s) \tag{8.31}$$

Computing $P(O \mid \lambda)$ using the backward variables also involves of the order of N^2T calculations.

8.4.2 Solution to Problem 2

Here we have to find the most likely state sequence (the hidden part of the model) associated with an observation sequence. The famous Viterbi algorithm is a dynamic programming approach to find the optimal path. It intermediately keeps the best possible state sequence at each instance of time for each of the *N* states, and finally it gives the best path for each of the N states as the last state for the observation sequence, from which the one with highest probability is selected.

The four-step Viterbi algorithm follows the same strategy as the forward procedure but it replaces summation with maximization (or minimization, depending on the optimality criterion). For a given the observation sequence $O = O_1, O_2, ..., O_T$ and the model λ, the algorithm involves the following steps:

1. *Initialization*:

$$d_1(s) = p_s b_s(O_1) \tag{8.32}$$

$$y_1(s) = 0,\ 1 \le s \le N \tag{8.33}$$

Where $\delta_t(s)$ denotes the accumulated weight when we are in state *s* at time *t*, and $\psi_t(s)$ represents the state at time *t*-1 which has the lowest cost (maximum probability) corresponding to the state transition to state *s* at time *t*.

2. *Induction*:

$$d_t(s) = \max_{1\le r\le N}[d_{t-1}(r)a_{rs}]b_s(O_t) \tag{8.34}$$

$$y_t(s) = \arg\max_{1\le r\le N}[d_{t-1}(r)a_{rs}],\ 1 \le s \le N, 2 \le t \le T \tag{8.35}$$

3. *Termination*:

$$P^* = \max_{1\le s\le N}[d_T(s)] \tag{8.36}$$

$$q_T^* = \arg\max_{1\le s\le N}[d_T(s)] \tag{8.37}$$

4. *Path Backtracking*:

$$q_t^* = y_{t+1}(q_{t+1}^*),\ t = T-1, T-2, ..., 1 \tag{8.38}$$

Now, $Q^* = \{q_1^*, q_2^*, ..., q_T^*\}$ is the optimal state sequence, and P^* is the joint probability of the observation sequence *O* and the optimal state sequence Q^*.

Like the forward and backward procedures, the complexity of the Viterbi algorithm is of the order of N^2T.

A direct implementation of the above algorithm does not take care about underflow. It is clear that the probabilities we are calculating involve multiplying together very small numbers, which will rapidly underflow the range of floating point numbers on a computer. To remedy this problem, the Viterbi algorithm is changed to work with logarithms. This not only solves the underflow problem, but also speeds up the computation, since addition is much faster than multiplication. A quick implementation of the Viterbi algorithm is highly desirable because it is a runtime algorithm, and not a training algorithm which can usually proceed offline. The efficient and practical version of the Viterbi algorithm is given below:

0. *Preprocessing*:

$$\tilde{p}_s = \log(p_s),\ 1 \le s \le N \tag{8.39}$$

$$\tilde{a}_{rs} = \log(a_{rs}),\ 1 \le r, s \le N \tag{8.40}$$

$$\tilde{b}_s(O_t) = \log(b_s(O_t)),\ 1 \le s \le N, 1 \le t \le T \tag{8.41}$$

1. *Initialization*:

$$\tilde{d}_1(s) = \tilde{p}_s + \tilde{b}_s(O_1) \tag{8.42}$$

$$y_1(s) = 0,\ 1 \le s \le N \tag{8.43}$$

2. *Induction*:

$$\tilde{d}_t(s) = \max_{1 \le r \le N}[\tilde{d}_{t-1}(r) + \tilde{a}_{rs}] + \tilde{b}_s(O_t) \tag{8.44}$$

$$y_t(s) = \arg\max_{1 \le r \le N}[\tilde{d}_{t-1}(r) + \tilde{a}_{rs}],\ 1 \le s \le N, 2 \le t \le T \tag{8.45}$$

3. *Termination*:

$$P^* = \max_{1 \le s \le N}[\tilde{d}_T(s)] \tag{8.46}$$

$$q_T^* = \arg\max_{1 \le s \le N}[\tilde{d}_T(s)] \tag{8.47}$$

4. *Path Backtracking*:

$$q_t^* = y_{t+1}(q_{t+1}^*),\ t = T-1, T-2, ..., 1 \tag{8.48}$$

Now, $Q^* = \{q_1^*, q_2^*, ..., q_T^*\}$ is the optimal state sequence, and $\exp(P^*)$ is the joint probability of the observation sequence O and the optimal state sequence Q^*.

You may notice that the Viterbi algorithm only involves multiplications, but the forward/backward algorithm involves additions too. Logarithms can still be used to prevent floating point underflow, here we need to calculate $\log(x+y)$ which can be achieved by the following technique (Manning and Schütze, 1999):

```
if y – x > log big
  return y;
else if x – y > log big
  return x;
else
```

```
    return min(x,y) + log(exp(x - min(x,y)) + exp(y – min(x,y)));
```

Where *big* is suitable large constant like 10^{30}.

8.4.3 Solution to Problem 3

There are two general approaches for estimating the model parameters (training) depending on the probability that is chosen for maximization. The segmental k-means (Juang and Rabiner, 1990) algorithm adjusts the parameters so that $P\ (O,\ Q^{*}|\ \lambda)$ is maximized, where Q^{*} is the optimal state sequence corresponding to the observation sequence O. The Baum-Welch algorithm (Rabiner, 1989) adjust the parameters to increase $P\ (O\ |\ \lambda)$ until a maximum value is reached, here $P\ (O\ |\ \lambda)$ involves summing up $P\ (O,\ S\ |\ \lambda)$ over all possible state sequences S, meaning that the algorithm does not focus on a particular state sequence. The segmental k-means algorithm is often preferred, because it requires much less computation as compared to the Baum-Welch algorithm, and also in text recognition applications, both modeling and decoding must be performed on the observation datasets and the criterion $P\ (O,\ Q^{*}|\ \lambda)$ seems quite natural for both these tasks.

8.4.4 The Segmental K-Means Algorithm

The segmental k-means algorithm requires a number of observation (training) sequences. Let there are w number of such sequences. Each sequence $O = O_1, O_2, ..., O_{T_i}$ consists of T_i observation vectors, so we have $\sum_{i=1}^{w} T_i$ observation vectors. Instead of w number of such sequences, if one long sequence is given, it can be segmented into an arbitrary number of short sequences. Each observation symbol O_i is assumed to be a vector of dimension of one or higher; and all observation vectors must be of equal dimension. The algorithm consists of the following steps:

1. Randomly choose N observation vectors $C_1,C_2,...,C_N$, and assign each of the remaining observation vectors to one of these N vectors from which its Euclidean distance is minimum. Therefore N clusters, each called a state, numbered from 1 to N are formed. The notation $O_t \in s$ means that the t^{th} observation symbol O_t of an observation sequence is assigned to state s. This initial choice of clustering does not influence the final HMM, but it can decide the number of iterations for training. To make the initial choice of clusters as widely distributed as possible a good strategy

when $w \geq N$ is to choose C_1 as the first observation vector of the first sequence, C_2 as the second observation vector of the second sequence and so on (Dugad and Desai, 1996). This step provides a good initialization for the complete training procedure.

2. Calculate the initial and the transitions probabilities:

$$\hat{p}_s = \frac{\text{number of occurrences of } \{O_1 \in s\}}{\text{total number of occurrences of } O_1},\ 1 \leq s \leq N \tag{8.49}$$

$$\hat{a}_{rs} = \frac{\text{number of occurrences of } \{O_t \in r \text{ and } O_{t+1} \in s\}}{\text{total number of occurrences of } \{O_t \in r\}},\ 1 \leq r, s \leq N,\ 1 \leq t \leq T_i - 1 \tag{8.50}$$

3. Calculate the mean and covariance matrix for each state:

$$\hat{m}_s = \frac{1}{N_s} \sum_{O_t \in s} O_t,\ 1 \leq s \leq N \tag{8.51}$$

$$\hat{V}_s = \frac{1}{N_s} \sum_{O_t \in s} (O_t - \hat{m}_s)^T (O_t - \hat{m}_s),\ 1 \leq s \leq N \tag{8.52}$$

4. Calculate the probability distribution for each observation vector for each state:

$$\hat{b}_s(O_t) = \frac{1}{(2p)^{\frac{D}{2}} |\hat{V}_s|^{\frac{1}{2}}} \exp(-\frac{1}{2}(O_t - \hat{m}_s)\hat{V}_s(O_t - \hat{m}_s)^T) \tag{8.53}$$

It has been proved that the algorithm converges to the state-optimized likelihood function for a wide range of density functions including Gaussian. Here, the Gaussian density is optionally chosen.

5. Use the Viterbi algorithm with the new probabilities to find the optimal state sequence Q^* for each training sequence. An observation vector is reassigned a state if its original assignment is different from the corresponding estimated optimal state, i.e. assign O_t to s if $q_t^* = s$.

6. If any vector is reassigned a new state in Step 5, then use the new state assignment and repeat Step 2 to Step 6; otherwise, stop.

8.4.5 The Baum-Welch Algorithm

The Baum-Welch algorithm is an Expectation-Maximization (EM) algorithm. The EM algorithm is a widely used approach to learning in the presence of unobserved (hidden) variables. It searches for a maximum likelihood hypothesis by iteratively re-estimating the expected values of the hidden variables given its current hypothesis, then recalculating the maximum likelihood hypothesis using these expected values for the hidden variables. In other words, the current hypothesis is used to estimate the unobserved variables, and then the expected values of these variables are used to calculate an improved hypothesis. It can be proved that the algorithm converges to a local maximum hypothesis (Mitchell, 1997).

An initial hypothesis (HMM) can be constructed in any way, but a reasonable initial estimate is obtained by using the first four steps of the segmental k-means algorithm. First, we should introduce some concepts and formulas that will be used in the final formulas. Consider $\gamma_t(s) = P(s_t = s \mid O, \lambda)$ that is the probability of being in state s at time t given the observation sequence O and the model λ. Using the Bayes law we have:

$$g_s(t) = \frac{P(s_t = s, O \mid l)}{P(O \mid l)} = \frac{a_t(s) b_t(s)}{P(O \mid l)},\ 1 \le s \le N \tag{8.54}$$

Where $\alpha_t(s)$ and $\beta_t(s)$ are the forward and backward variables.

We also define $\xi_t(r,s) = P(s_t = r, s_{t+1} = s \mid O, \lambda)$ that is the probability of being in state r at time t and making a transition to state s at time t+1. Using the Bayes law and the causality property of Markov chain, it can be shown that:

$$x_t(r,s) = \frac{a_t(r) a_{rs} b_s(O_{t+1}) b_{t+1}(s)}{P(O \mid l)},\ 1 \le r, s \le N \tag{8.55}$$

If $\gamma_t(s)$ is summed up from t = 1 to T, the expected number of times state s is visited is obtained, and if it is summed up only to T-1, the expected number of transitions out of state s is obtained. Similarly, if $\xi_t(r,s)$ is summed up from t = 1 to T-1, the expected number of transitions from state r to s is obtained:

$$\sum_{t=1}^{T-1} g_t(s) \text{ = expected number of transitions from state } s,\ 1 \le s \le N \tag{8.56}$$

$$\sum_{t=1}^{T-1} x_t(r,s) \text{ = expected number of transitions from state } r \text{ to state } s,\ 1 \le r,s \le N \tag{8.57}$$

$\gamma_t(r)$ and $\xi_t(r,s)$ can be related by summing up $\xi_t(r,s)$ over s:

$$g_t(r) = \sum_{s=1}^{N} x_t(r,s),\ 1 \le r \le N \tag{8.58}$$

The Baum-Welch re-estimation formulas are now defined as follows:

$$\hat{p}_s = g_1(s),\ 1 \le s \le N \tag{8.59}$$

$$\hat{a}_{rs} = \sum_{t=1}^{T-1} x_t(r,s) \Bigg/ \sum_{t=1}^{T-1} g_t(r),\ 1 \le r,s \le N \tag{8.60}$$

$$\hat{b}_s(v_k) = \sum_{t=1, O_t=v_k}^{T} g_t(s) \Bigg/ \sum_{t=1}^{T} g_t(s),\ 1 \le s \le N \tag{8.61}$$

The re-estimation formula for p_s is simple the probability of being in state *s* at time 1. The formula for a_{rs} is the ratio of expected number of transitions from state *r* to state *s* to the expected number of times making a transition out of state *r*. The formula for $b_s(v_k)$ is the ratio of the expected number of times of being in state *s* and observing symbol v_k to the expected number of times of being in state *s*.

8.5 Continuous Hidden Markov Models

Thus far, HMMs have been applied for process with discrete observation sequences, i.e. all observation vectors belong to a finite alphabet $V = \{v_1, v_2, ..., v_M\}$. In such a case, the model is a Discrete Hidden Markov Model (DHMM). The discrete observations can be the indices of codebook obtained by Vector Quantization (VQ) which is a clustering technique for producing an approximation of distribution of a multi-dimensional signal in a codebook. VQ is responsible for loosing some information from the signal. The loss is due to the quantization error (distortion) that can be reduced but not be eliminated by increasing the codebook size (number of clusters).

A Continuous Hidden Markov Model (CHMM) is an extension of DHMM to overcome the distortion problem. CHMM has more parameters than DHMM, thus requiring more memory and more deliberate techniques to initialize the model as it may easily diverge with randomly selected initial parameters.

In CHMM the parameter *B* is represented differently as here there is not a finite set of observation symbols *V*. The probability density function of an observation vector o_t in each state is considered to be a multivariate Gaussian mixture (other distributions are also valid, but the multivariate Gaussian mixture is general and proved to be promising):

$$b_i(o_t) = \sum_{m=1}^{M} c_{im} \mathrm{N}(o_t; m_{im}, \Sigma_{im})$$

$$= \sum_{m=1}^{M} \frac{c_{im}}{\sqrt{(2p)^K \mid \Sigma_{im} \mid}} \exp(-\frac{1}{2}(o_t - m_{im}) \Sigma_{im}^{-1} (o_t - m_{im})^T) \qquad (8.62)$$

where:

c_{im}: The mth mixture gain coefficient in state *i* (8.63)

μ_{im}: The mean of the mth mixture in state *i* (8.64)

$\sum_{im}$: The covariance of the mth mixture in state *i* (8.65)

M: The number of mixtures used (8.66)

K: The dimensionality of the observation space (8.67)

The following constraints have to be satisfied to ensure the consistency of the model:

$$c_{im} \geq 0,\ \forall i, m,\ \sum_{m=1}^{M} c_{im} = 1,\ \forall i \text{ and } \int_{-\infty}^{\infty} b_i(o_t) do_t = 1,\ \forall i$$

The covariance matrix $\sum$ can be simplified by using a diagonal matrix with elements representing the variance of each mixture. This approximation reduces the computational cost to a great extent, but the number of mixtures should be increased to make the model work better.

In the case of multi-mixture CHMMs, the re-estimation formulas have to be modified. Let $\gamma_t(i,m)$ be the probability of being in the mth mixture of state *i* at time *t*:

$$g_t(i,m) = \frac{a_t(i) b_t(i)}{\sum_{s=1}^{N} a_t(s) b_t(s)} \frac{c_{im} \mathrm{N}(o_t; m_{im}, \Sigma_{im})}{\sum_{k=1}^{M} c_{ik} \mathrm{N}(o_t; m_{ik}, \Sigma_{ik})} \qquad (8.68)$$

It should be clear that $\gamma_t(i,m) = \gamma_t(i)$ when M=1.

The re-estimation formulas for c_{im}, μ_{im} and $\sum_{im}$ are now defined as follows:

$$\hat{c}_{im} = \frac{\text{expected number of times of being in the } m^{\text{th}} \text{mixture of state } i}{\text{expected number of times of being in state } i}$$

$$= \frac{\sum_{t=1}^{T} g_t(i,m)}{\sum_{t=1}^{T} \sum_{m=1}^{M} g_t(i,m)} \qquad (8.69)$$

$$\hat{m}_{im} = \frac{\sum_{t=1}^{T} g_t(i,m) o_t}{\sum_{t=1}^{T} g_t(i,m)} \qquad (8.70)$$

$$\hat{\Sigma}_{im} = \frac{\sum_{t=1}^{T} g_t(i,m)(o_t - m_{im})(o_t - m_{im})^T}{\sum_{t=1}^{T} g_t(i,m)} \qquad (8.71)$$

8.6 Training and Recognition

The recognition system is trained and evaluated on a dataset of 100 city names of Iran. Thus a pattern recognition problem with 100 classes is considered. Most samples in the dataset were automatically generated by a Java program drawing input string with different fonts, sizes and orientations on output image. The dataset contains 150 samples for each word. The complete list of words and a few sample images generated by the program are shown in Appendix A.

Since the lexicon size is limited (100), a holistic approach based on model discriminant CHMM is chosen as the recognition engine, i.e. each word in the lexicon is modeled by a separate CHMM (Figure 8.3). The main advantage of the model discriminant scheme is that if a new word is added, the recognition system can simply be updated by adding the new word model to the system knowledgebase. But it has the major drawback of using a predefined lexicon which limits the recognition outputs to the lexicon words. Although a large lexicon of size ten thousands covers almost all words in a language, but such a large lexicon requires much memory and causes a severe delay in producing the ranked word list as the Viterbi algorithm has to be executed for each word model λ_i. To overcome the memory and speed problems, an alternative is to build a single HMM for all words, where each character is modeled by a small group of the HMM states, and a word is represented by a path through the model. This approach is called path discriminant HMM as a pattern is classified to the word which has the maximum path probability over all possible paths. Previous researches (Khorsheed, 2000) prove that a path discriminant (single-HMM) scheme achieves less accuracy than a model discriminant (multi-HMM) scheme for a same lexicon.

The number of states of a word model is set to be the size of the shortest observation sequence of the training instances of the word. A Bakis structure is selected for all HMMs, with minimum relative forward jump of 0 (loop to current state) and maximum relative forward jump of 2. The maximum allowed number of densities in each state is set to 10. No limit is imposed on the number of training iterations, i.e. the training procedure continues until convergence.

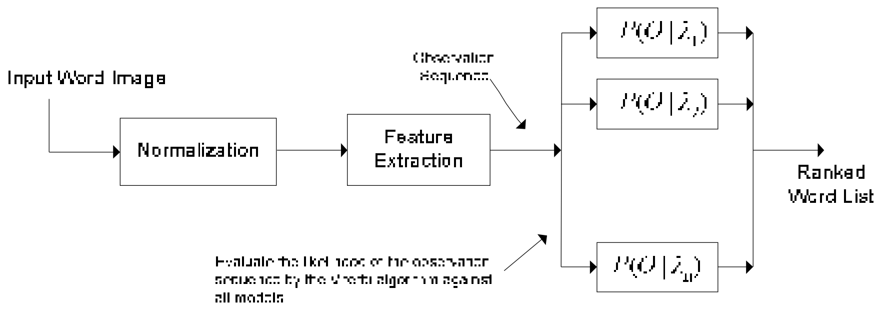


Figure 8.3. The block diagram of the handwritten recognition system.

Figure 8.4 shows an overview of the complete segmentation-recognition system.

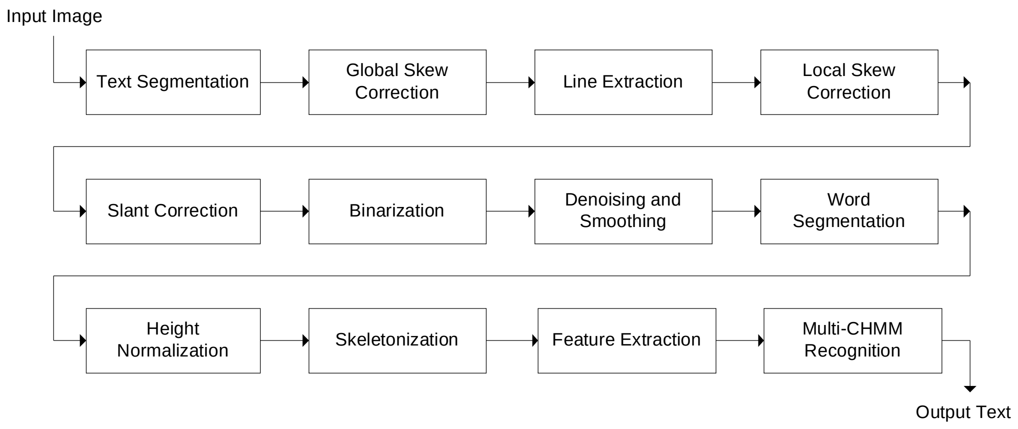


Figure 8.4. An overview of the complete segmentation-recognition system.

8.7 Experimental Results

Here some experimental results for the isolated word recognition system are presented. The multi-HMM recognition system can provide an N-best list of hypotheses rather than a single hypothesis. The N-best list is generated by sorting the entire probabilities $P(O \mid \lambda_i)$. Given an N-best list of possible hypotheses, a system may use other knowledge to find the correct hypothesis. It is said that a word image is N-best recognized when the N-best list includes the correct word hypothesis for the minimum value of N. Obviously the N-best recognition rate increases with N, and reaching 100% when N equals to the lexicon size in worst case. Some of the words in the following figures have overlapped and connected characters, contaminated with noise. It is observed that some characters are broken into parts. Sometime loops are not present in such characters as 'م' and 'و' that normally have loops, but loops are formed in characters that should not have them. All of these artifacts decrease the recognition rate, as the proposed system use structural features.

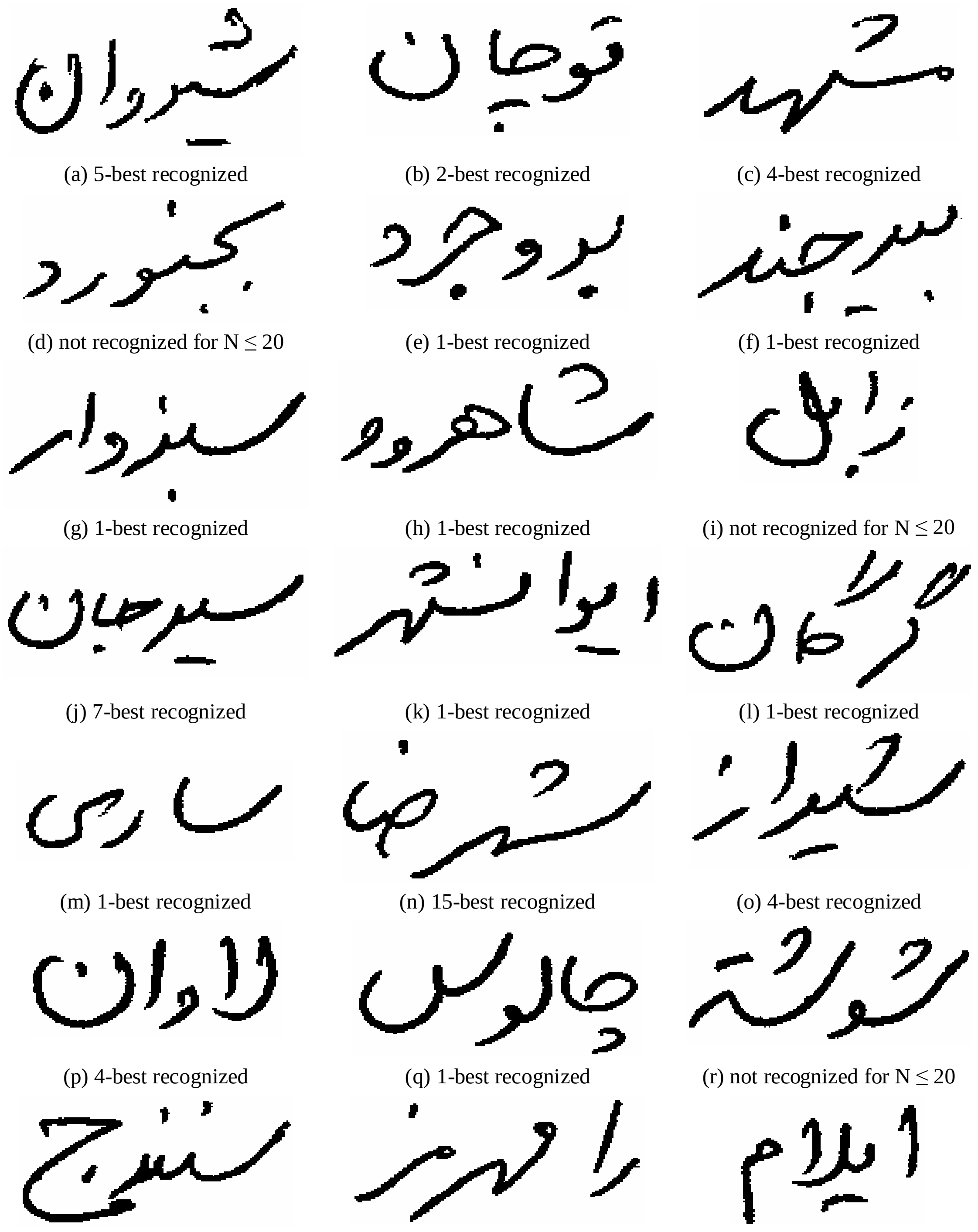
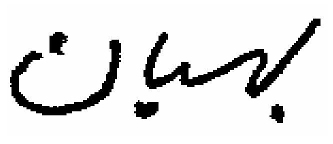
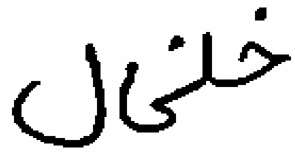
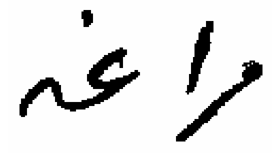
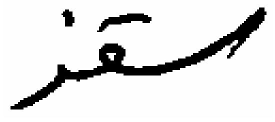

(a) 5-best recognized
(b) 2-best recognized
(c) 4-best recognized
(d) not recognized for N ≤ 20
(e) 1-best recognized
(f) 1-best recognized
(g) 1-best recognized
(h) 1-best recognized
(i) not recognized for N ≤ 20
(j) 7-best recognized
(k) 1-best recognized
(l) 1-best recognized
(m) 1-best recognized
(n) 15-best recognized
(o) 4-best recognized
(p) 4-best recognized
(q) 1-best recognized
(r) not recognized for N ≤ 20
(s) 1-best recognized
(t) 1-best recognized
(u) 1-best recognized

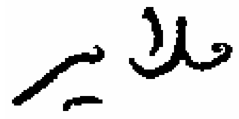
Figure 8.5. Examples of handwritten words used to evaluate the system performance.

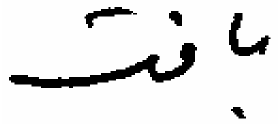

(a) not recognized for N ≤ 20

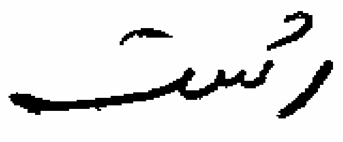

(b) 6-best recognized

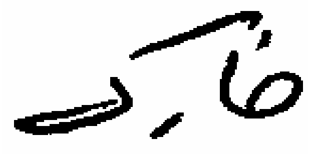

(c) 1-best recognized

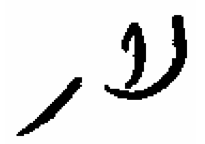

(d) 1-best recognized

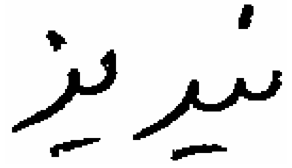

(e) 1-best recognized

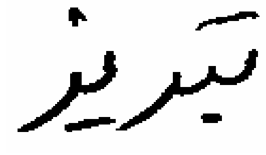

(f) 1-best recognized

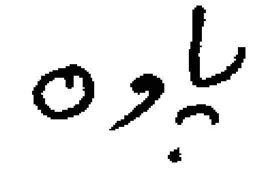

(g) 1-best recognized

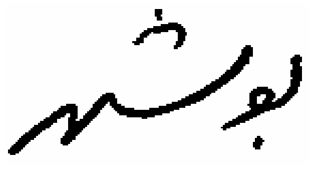

(h) 4-best recognized

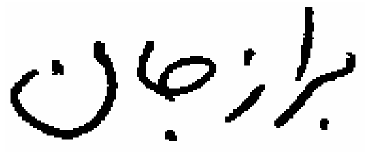

(i) 1-best recognized

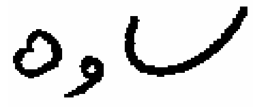

(j) 1-best recognized

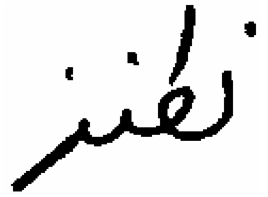

(k) 1-best recognized

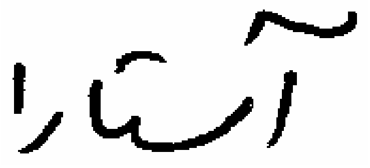

(l) 1-best recognized

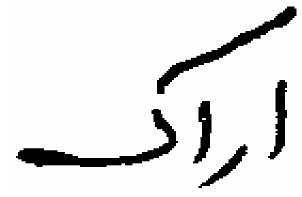

(m) 1-best recognized

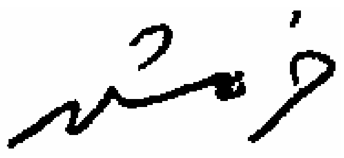

(n) 1-best recognized

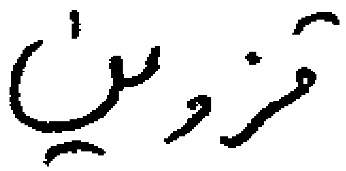

(o) 8-best recognized

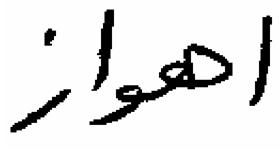

(p) 2-best recognized

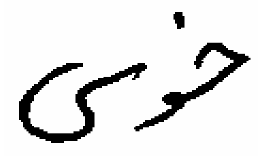

(q) 2-best recognized

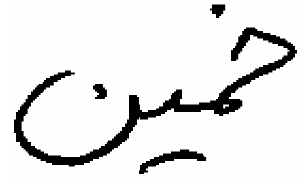

(r) 3-best recognized


(s) not recognized for N ≤ 20


(t) 1-best recognized


(u) 1-best recognized


(v) not recognized for N ≤ 20


(w) 1-best recognized

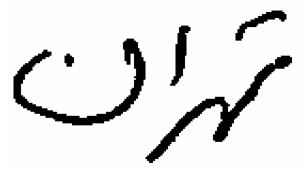
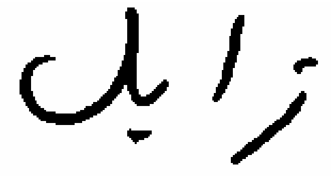

(x) 1-best recognized

Figure 8.6. Examples of handwritten words used to evaluate the system performance.

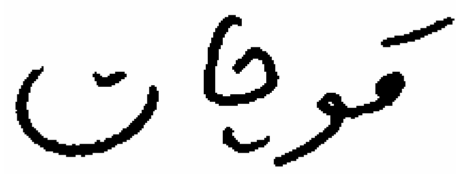
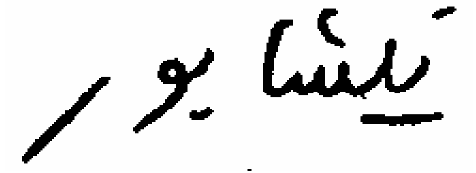
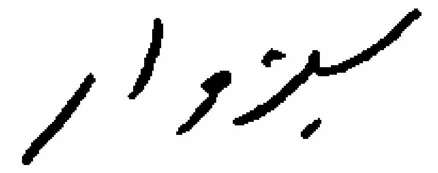
(a) 1-best recognized

(b) 1-best recognized

(c) 6-best recognized

(d) 1-best recognized

(e) 1-best recognized

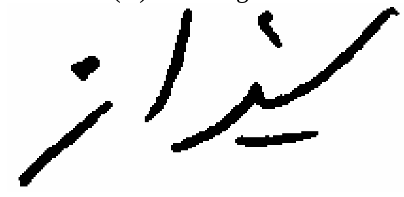
(f) 2-best recognized

(g) 1-best recognized

(h) 1-best recognized

(i) 1-best recognized

(j) not recognized for N ≤ 20

(k) 2-best recognized

(l) 1-best recognized

(m) not recognized for N ≤ 20

(n) 1-best recognized

(o) 4-best recognized

(p) 1-best recognized

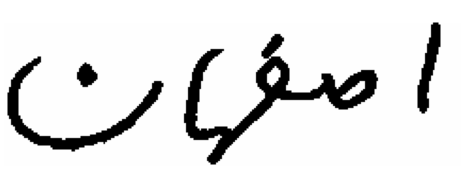
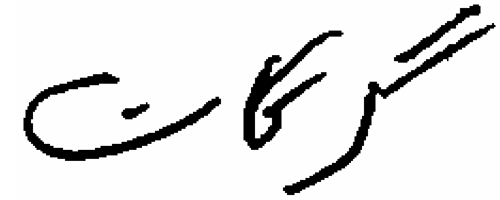
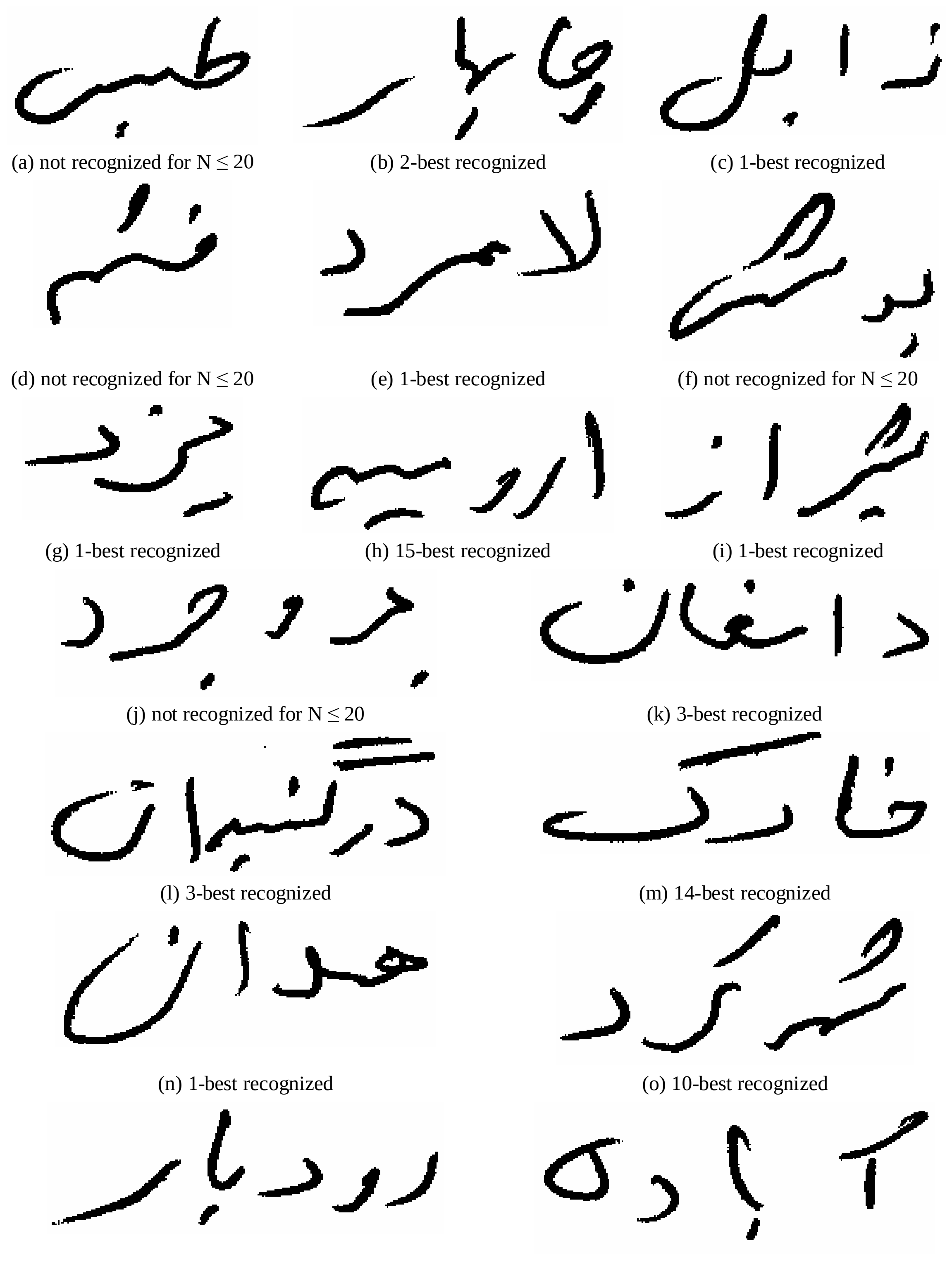
(q) 1-best recognized

(r) 1-best recognized

(s) 1-best recognized

Figure 8.7. Examples of handwritten words used to evaluate the system performance.



(a) not recognized for N ≤ 20

(b) 2-best recognized

(c) 1-best recognized

(d) not recognized for N ≤ 20

(e) 1-best recognized

(f) not recognized for N ≤ 20

(g) 1-best recognized

(h) 15-best recognized

(i) 1-best recognized

(j) not recognized for N ≤ 20

(k) 3-best recognized

(l) 3-best recognized

(m) 14-best recognized

(n) 1-best recognized

(o) 10-best recognized

(p) 1-best recognized

(q) 3-best recognized

Figure 8.8. Examples of handwritten words used to evaluate the system performance.

Conclusion

A complete offline recognition system for Farsi handwritten words was presented. To the best of our knowledge, this work was the first to use continuous hidden Markov models with structural features to recognize Farsi handwritten words. In addition to feature extraction and recognition, other parts of a complete recognition system, including text segmentation, binarization, skew correction, slant correction and skeletonization were addressed.

A new machine learning approach based on the naive Bayes classifier, which is fast both in training and application phase, was developed for text segmentation. It was shown that excellent results could be obtained by this simple classifier. Lack of large amount of proper training data usually restricts practicality of the modern AI methods. To overcome this problem, a simple procedure for generating the required training data from a set of 8 hand-segmented images was presented.

Four different algorithms for document image binarization were compared and contrasted: the Otsu's global method, the Niblack's local method, the Wu and Manmatha's method and the Liu and Srihari's method. The last two methods are designed specially for document image binarization, and performing better than the first two general-purpose methods, particularly in the presence of textured, shaded or noisy backgrounds. Excluding Niblack's, the other methods are quite fast, and even suitable for real-time applications.

Different skew and slant correction algorithms were surveyed for handwritten documents, and the problem of multiple skews was dealt with in a two-stage process. The first stage correct the global skew, and after extracting text lines, in the second stage, the skew of each line is corrected locally. It was shown that the projection profile based method for correcting global skew was robust and practical to be used in real systems, and since this method was rather slow, some techniques were proposed to speed it up. It was shown that the same technique utilized for skew correction could be applied to remove the slant of handwritten words.

Five different skeletonization algorithms were compared and contrasted: the SPTA, the Zhang-Suen's algorithm, the DTSA, the Ji and Piper's homotopy-preserving algorithm, and the Huang et al.'s algorithm. The main focus was on preserving text characteristics, such as not removing dots, obtaining well-connected skeletons of unitary thickness, and robustness with respect to border noise. It was shown that the Zhang-Suen's and the homotopy-preserving algorithms are not suitable for recognition. Among the other three algorithms, Huang et al.'s is the most robust, as it produces skeletons with the smallest number of spurious branches. It is also quite fast and practical. All of the surveyed skeletonization algorithms were iterative; however, it is worthy to survey non-iterative and indirect methods in the context of text recognition. A simple and effective skeleton post-processing technique was also described.

There exist two main types of features: statistical and structural. Structural features are capable of tolerating many variations, but not robust to noise, and hard to extract. On the other hand, statistical features are robust to noise, and easy to extract, but with the disadvantage of requiring a large set of training instances to attain well-trained classifiers. Structural features were used in this study since they have been less studied for offline handwritten recognition. The features were extracted from the graph representing the skeleton of an input word image. The loops and edges of the graph were visited in a canonical order, and then each one was represented by a 10D feature vector. So, each input word image was represented by a 1D observation sequence, being appropriate for 1D HMM-based classifier. The 10 features used to describe the edges or loops were independent of the baseline location, so the difficult and crucial problem of baseline detection was avoided.

The recognition was based on continuous hidden Markov models (CHMMs). Unlike discrete hidden Markov models (DHMMs), CHMMs do not quantize observation vectors, so they don't involve the distortion problem of DHMMs. Since the lexicon on which the system was intended to be trained was limited, a few hundred words, a model discriminant recognition scheme was chosen, i.e. each word in the lexicon was modeled by a separate CHMM. This scheme has two main advantages: 1) if a new word is added, the recognition system can simply be updated by adding the new word model to the system knowledgebase. When a neural network is used, for example, once a new class is added, the whole network must be retrained, which is a time-consuming procedure; and 2) the recognition can take advantage of being executed on a parallel computer, so the recognition delay can be kept constant with increasing the number of classes (the lexicon size).

There is no publicly available dataset for Farsi handwritten word images, and it is not wise to compare different systems evaluated on different datasets. The executable version of training, recognition and evaluation modules of the proposed system is provided on the thesis webpage: http://pasargad.cse.shirazu.ac.ir/~mhaji/handrec. So it can be trained and evaluated on different datasets, and simply compared with others'. The proposed method achieved a maximum recognition rate of about 82% on a small lexicon, containing word images of 100 cities of Iran. The striking aspect of the recognition system is its excellent generalization performance, as seen in our experiments, when multi-font machine-printed word images were used for training, the recognition ability could be generalized to handwriting.

To improve the recognition rate and dealing with large lexica, further research could be carried out in the following areas: 1) comparing and combining different classifiers; 2) combining statistical and structural features; 3) using more advanced HMMs; and 4) using lexicon pruning techniques to limit candidate words. The problem of recognizing handwritten text, with a performance comparable to human's, seems so difficult that it will remain unsolved, unless much more elaborate techniques are developed.

REFERENCES

Adab, M. (2001). "Simultaneous Segmentation and Recognition of Farsi/Arabic Printed Text", *M. Sc. Thesis*, Department of Control Engineering, Amir Kabir University, Tehran, Iran.

Ahmed, M. and Ward, R. K. (2000). "An Expert System for General Symbol Recognition", *Pattern Recognition*, vol. 33, pp. 1975-1988.

Ahmed, P. (1995). "A Neural Network Based Dedicated Thinning Method", *Pattern Recognition Letters*, vol. 16, pp. 585-590.

Almuallim, H. and Yamaguchi, S. (Sep. 1987). "A Method of Recognition of Arabic Cursive Handwriting", *IEEE Trans. on PAMI*, vol. 9(5), pp. 715-722.

Al-Yousefi, H. and Udpa, S. S. (Aug. 1992). "Recognition of Arabic Characters", *IEEE Trans. on PAMI*, vol. 14(8), pp. 853-857.

Amin, A. (1998). "Off-Line Arabic Character Recognition: The State of the Art", *Pattern Recognition*, vol. 31(5), pp. 517-530.

Amin, A. and Mari, J. (1989). "Machine Recognition and Correction of Printed Arabic Text", *IEEE. Trans. on Systems, Man and Cybernetics*, vol. 19(5), pp. 1300-1306.

Arica, N. and Yarman-Vural, F. T. (2000). "One-Dimensional Representation of Two-Dimensional Information for HMM Based Handwriting Recognition", *Pattern Recognition Letters*, vol. 21, pp. 583-592.

Avanindra and Subhasis Chaudhuri (February 1997). "Robust Detection of Skew in Document Images", *IEEE Trans. Image Processing*, vol. 6(2), pp. 344-349.

Bloomberg, D. S., Kopec, G. E. and Dasari, L. (Feb. 1995). "Measuring Document Image Skew and Orientation", *Proceedings of IS&T/SPIE EI'95 Conference 2422: Document Recognition II*, pp. 302-316.

Chaddha, N., Sharma, R., Agrawal, A. and Gupta, A. (1995). "Text segmentation in mixed-mode images", *Proceedings of the 28th Asilomar Conference on Signals, Systems and Computers*, vol. 2, pp. 1356-1361.

Changming Sun and Deyi Si (1997). “Skew and Slant Correction for Document Images Using Gradient Direction”, *IEEE International Conference on Document Analysis and Recognition*, vol. 1, pp.142-146.

Chen, D., Bourlard, H. and Thiran, J. (Dec. 2001). “Text Identification in Complex Backgrounds Using SVM”, *Proceedings of the International Conference on Computer Vision and Pattern Recognition*, pp. 621-626.

Choi, H. and Baraniuk, R. G. (Sep. 2001). “Multiscale Image Segmentation Using Wavelet-Domain Hidden Markov Models”, *IEEE Trans. on Image Processing*, vol. 10(9), pp. 1309-1321.

Dehghan, M., Faez, K., Ahmadi, M. and Shridhar, M. (2001). "Handwritten Farsi (Arabic) Word Recognition: A Holistic Approach Using Discrete HMM", *Pattern Recognition*, vol. 34, pp. 1057-1065.

Dehghan, M., Faez, K., Ahmadi, M. and Shridhar, M. (2001). "Unconstrained Farsi Handwritten Word Recognition Using Fuzzy Vector Quantization and Hidden Markov Models", *Pattern Recognition Letters*, vol. 22, pp. 209-214.

Deng, S. and Latifi, S. (2000). "Fast Text Segmentation Using Wavelet for Document Processing", *Proceedings of the 4th WAC, ISSCI, IFMIP*, Maui, Hawaii, USA, pp. 739-744.

Domingos, P. and Pazzani, M. (1997). "On the Optimality of the Simple Bayesian Classifier under Zero-One Loss", *Machine Learning*, vol. 29, pp. 103-130.

Dugad, R. and Desai, U. B. (May 1996). "A Tutorial on Hidden Markov Models", *Technical Report No. SPANN-96.1*, Indian Institute of Technology, Bombay, India.

El-Sheikh, T. and Guindi, R. (1988). "Computer Recognition of Arabic Cursive Script", *Pattern Recognition*, vol. 21(4), pp. 293-302.

Erlandson, E., Trenkle, J. and Vogt, R. (1996). "Word-Level Recognition of Multifont Arabic Text Using a Feature-Vector Matching Approach", *Proceedings of the SPIE*, vol. 2660-08.

Fletcher, L. A. and Kasturi, R. (Nov. 1988). “A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images”, *IEEE Trans. on PAMI*, vol. 10(6), pp. 910-918.

Gersho, A. and Gray, R. M. (1992). *Vector Quantization and Signal Compression*, Kluwer Academic Publishers.

He, Y. and Kundu, A. (Nov. 1991). "2-D Shape Classification Using Hidden Markov Model", *IEEE Trans. on PAMI*, vol. 13(11), pp. 1172-1184.

Huang, L., Wan, G. and Liu, C. (2003). "An Improved Parallel Thinning Algorithm", *Proceedings of the Seventh International Conference on Document Analysis and Recognition (ICDAR 2003)*, pp. 780-783.

Ivanov, D., Kuzmin, E. and Burtsev, S. (2000). "An Efficient Integer-Based Skeletonization Algorithm", *Computers and Graphics*, vol. 24, pp. 41-51.

Jain, A. K. (1989). *Fundamentals of Digital Image Processing*, Englewood Cliffs, Prentice Hall.

Jain, A. K. and Farrokhnia, F. (1991). “Unsupervised Texture Segmentation Using Gabor Filters”, *Pattern Recognition*, vol. 24, pp. 1167-1186.

Ji, L. and Piper, J. (1992). "Fast Homotopy-Preserving Skeletons Using Mathematical Morphology", *IEEE Trans. on PAMI*, vol. 14(6), pp. 653 - 664.

Jiang, H. F., Han, C. C. and Fan, K. C. (1997). "A Fast Approach to the Detection and Correction of Skewed Documents", *Pattern Recognition Letters*, vol. 18(7), pp. 675-686.

Jie Xi, Xian-Sheng Hua, Xiang-Rong Chen, et al. (August 2001). “A Video Text Detection and Recognition System”, *Proceedings of ICME 2001*, Waseda University, Japan, pp. 1080-1083.

Juang, B. H. and Rabiner, L. R. (Sep. 1990). "The Segmental K-Means Algorithm for Estimating the Parameters of Hidden Markov Models", *IEEE Trans. on Acoustics, Speech and Signal Processing*, vol. 38(9), pp. 1639-1641.

Kapur, J. N., et al. (1985). "A New Method for Gray-Level Picture Thresholding Using the Entropy of the Histogram", *Computer Vision, Graphics and Image Processing*, vol. 29, pp. 273-285.

Kavallieratou, E., Fakotakis, N. and Kokkinakis, G. (2000). "A Slant Removal Algorithm", *Pattern Recognition*, pp. 1261-1262.

Khorsheed, M. (June 2000). "Automatic Recognition of Words in Arabic Manuscripts", *Ph.D. Thesis*, Churchill College, University of Cambridge.

Kim, G. and Govindaraju, V. (April 1997). "A Lexicon Driven Approach to Handwritten Word Recognition for Real-Time Applications", *IEEE Trans. on PAMI*, vol. 19(4), pp. 366-379.

Kittler, J. and Illingworth, J. (1985). "On Threshold Selection Using Clustering Criteria", *IEEE Trans. on Systems, Man and Cybernetics*, vol. 15, pp. 652-655.

Lachiche, N. and Flach, P. (2003). "Improving Accuracy and Cost of Two-Class and Multi-Class Probabilistic Classifiers using ROC Curves", *Proceedings of the 20th International Conference on Machine Learning (ICML-2003).*

Li, J. and Gray, R. M. (Oct. 1998). “Text and Picture Segmentation by the Distribution Analysis of Wavelet Coefficients”, *Proceedings of IEEE International Conference on Image Processing*, Chicago, Illinois, vol. 3, pp 790-794.

Liu, Y. and Srihari, S. N. (May 1997). “Document Image Binarization Based on Texture Features”, *IEEE Trans. on PAMI*, vol. 19(5), pp. 540-544.

Lu, Z., Bazzi, I., Kornai, A., Makhoul, J., Natarajan, P. and Schwartz, R. (1999). "A Robust Language-Independent OCR System", *Proceedings of 27th AIPR Workshop: Advances in Computer-Assisted Recognition, SPIE Proceedings*.

Manning, C. D. and Schütze, H. (1999). *Foundations of Statistical Natural Language Processing*, The MIT Press.

Mitchell, Tom M. (1997). *Machine Learning*, McGraw-Hill.

Motawa, D., Amin, A. and Sabourin, R. (Sep. 1999). "Segmentation of Arabic Cursive Script", *Proceedings of the 5th International Conference on Document Analysis and Recognition, ICDAR99*, pp. 625-628.

Naccache, N. J. and Shinghal, R. (1984). "SPTA: A Proposed Algorithm for Digital Pictures", *IEEE Trans. on Systems, Man and Cybernetics*, vol. SMC-14(3), pp. 409-418.

Nadadur, D. and Haralick, R. (2000). "Recursive Binary Dilation and Erosion Using Digital Line Structuring Elements in Arbitrary Orientations", *IEEE Trans. on Image Processing*, vol. 9(5), pp. 749-759.

Najman, L. (2004). "Using Mathematical Morphology for Document Skew Estimation", *Proceedings of SPIE conference on Document Recognition and Retrieval*, pp. 182-191.

Niblack, W. (1989). *An Introduction to Digital Image Processing*, Prentice Hall, Englewood Cliffs, pp. 115-116.

Ohya, J., Shio, A. and Akamatsu, S. (Feb. 1994). “Recognizing Characters in Scene Images”, *IEEE Trans. on PAMI*, vol. 16(2), pp. 214-224.

Okun, O., Pietikäinen, M. and Sauvola, J. (1999). "Document Skew Estimation without Angle Range Restriction", *International Journal on Document Analysis and Recognition*, pp. 132-144.

Otsu, N. (Jan. 1979). “A Threshold Selection Method from Gray Level Histograms”, *IEEE Trans. on Systems, Man and Cybernetics*, vol. 9, pp. 62-66.

Parhami, B. and Taraghi, M. (1981). "Automatic Recognition of Printed Farsi Texts", *Pattern Recognition*, vol. 14(6), pp. 395-403.

Pietikäinen, M. and Okun, O. (June 2001). “Text Extraction from Grey Scale Page Images by Simple Edge Detectors”, *Proceedings of the 12th Scandinavian Conference on Image Analysis*, Bergen, Norway, pp. 628-635.

Postl, W. (1986). "Detection of linear oblique structure and skew scan in digitized documents", *Proceedings of International Conference on Pattern Recognition*, pp. 687-689.

Procter, S., Illingworth, J. and Mokhtarian, F. (August 2000). "Cursive Handwritten Recognition Using Hidden Markov Models and a Lexicon-Driven Level Building Algorithm", *IEE Proceedings on Vision, Image and Signal Processing*, vol. 147(4), pp. 332-339.

Rabiner, L. R. (1989). "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition", *IEEE Proceedings*, vol. 77(2), pp. 257-286.

Rish, I. (2001). "An Empirical Study of the Naive Bayes Classifier", *Proceedings of IJCAI 2001 Workshop on Empirical Methods in Artificial Intelligence*.

Rondel, M. and Burel, G. (August 1995). “Cooperation of Multi-Layer Perceptrons for the Estimation of Skew Angle in Text Document Images”, *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 1141-1144.

Sajjadi, M. R. (Oct. 1996). "Skeletonization of Persian Characters", *M. Sc. Thesis*, Computer Science and Engineering Department, Shiraz University, Iran.

Seeger, M. and Dance, C. (2001). "Binarising Camera Images for OCR", *ICDAR 2001*, pp. 54–59.

Shapiro, L. G. and Stockman, G. C. (2001). *Computer Vision*, Prentice-Hall Inc.

Shridhar, M. and Kimura, F. (1995). “Handwritten Address Interpretation Using Word Recognition with and without Lexicon”, *IEEE International Conference on Systems, Man and Cybernetics*, vol. 3, pp.2341-2346.

Singh, R., Cherkassky, V. and Papanikolopoulos, N. (January 2000). "Self-Organizing Maps for the Skeletonization of Sparse Shapes", *IEEE Trans. on Neural Networks*, vol. 11(1), pp. 241-248.

Slavik, P. and Govindaraju, V. (March 2001). "Equivalence of Different Methods for Slant and Slew Corrections in Word Recognition Applications", *IEEE Trans. on PAMI*, vol. 23(3), pp. 323-326.

Spitz, A. L. (1998). "Analysis of Compressed Document Images for Dominant Skew, Multiple Skew and Logotype Detection", *Computer Vision and Image Understanding*, vol. 70(3), pp. 321-334.

Taylor, M. J. and Dance, C. R. (Sep. 1998). "Enhancement of Document Images from Cameras", *Proceedings of SPIE conference on Document Recognition*, pp. 230-241.

Trenkle, J., Gillies, A., Erlandson, E. and Schlosser, S. (Oct. 1995). "Arabic Character Recognition", *Proceedings of Symposium on Document Image Understanding Technology*, Maryland, pp. 191-195.

Trier, D. and Taxt, T. (March 1995). "Evaluation of Binarization Methods for Document Images", *IEEE Trans. on PAMI*, vol. 17(3), pp. 312-315.

Tsai, W. (1985). "Moment-Preserving Thresholding: A New Approach", *Computer Vision, Graphics and Image Processing*, vol. 29, pp. 377-393.

Tuceryan, M. and Jain, A. K. (1990). “Texture Segmentation Using Voronoi Polygons”, *IEEE Trans. on PAMI*, vol. 12, pp. 211-216.

Uchida, S., Taira, E. and Sakoe, H. (Sep. 2001). "Nonuniform Slant Correction Using Dynamic Programming", *IEEE International Conference on Document Analysis and Recognition*, pp. 434-438.

Unser, M. (Nov. 1995). “Texture Classification and Segmentation Using Wavelet Frames”, *IEEE Trans. on Image Processing*, vol. 4(11), pp. 1549-1560.

Vinciarelli, A. (2002). "A Survey on Off-Line Cursive Word Recognition", *Pattern Recognition*, vol. 35, pp. 1433-1446.

Vlontzos, J. and Kung, S. (1992). "Hidden Markov Models for Character Recognition", *IEEE Trans. on Image Processing*, vol. 1(4), pp. 539-543.

Wu, V. and Manmatha, R. (Jan. 1998). "Document Image Clean-Up and Binarization", *Proceedings of SPIE conference on Document Recognition*.

Wu, V., Manmatha, R. and Riseman, E. M. (1997). “Finding text in images”, *Proceedings of ACM International Conference on Digital Libraries*.

Wu, V., Manmatha, R. and Riseman, E. M. (1999). “Textfinder: An Automatic System to Detect and Recognize Text in Images”, *IEEE Trans. on PAMI*, vol. 21(11), pp. 1224-1229.

Yuan, Q. and Tan, C. L. (2000). “Page Segmentation and Text Extraction from Gray-Scale Images in Micro Film Format”, *SPIE Proceedings on Document Recognition and Retrieval*, vol. 4307, pp.323-332.

Yue Lu and Chew Lim Tan (August 2003). "Improved Nearest Neighbor Based Approach to Accurate Document Skew Estimation", *IEEE International Conference on Document Analysis and Recognition*, pp. 503-507.

Zhang, T. Y. and Suen, C. Y. (1984). "A Fast Parallel Algorithm for Thinning Digital Patterns", *Comm. ACM*, vol. 27(3), pp. 236-239.

Zimmermann, M. and Mao, J. (1999). "Lexicon Reduction Using Key Characters in Cursive Handwritten Words", *Pattern Recognition Letters*, vol. 20, pp. 1297-1304.

Appendix A

100 cities of Iran for which the training images were generated:

همدان	بوشهر	ملاير	شيروان
دزفول	برازجان	اردبيل	قوچان
اسفراين	قم	خلخال	مشهد
چالوس	تهران	مراغه	نيشابور
لار	محلات	مياندوآب	بجنورد
لامرد	ايذه	ماكو	بروجرد
نيريز	ساوه	سلماس	سبزوار
خارك	اراك	اروميه	بيرجند
كيش	كاشان	تبريز	زابل
لاوان	سمنان	سقز	زاهدان
طبس	ياسوج	بانه	ايرانشهر
قشم	اصفهان	سنندج	چابهار
نطنز	شهرضا	زنجان	كرمان
پاوه	ابرقو	ايلام	بم
كرج	يزد	خوي	جيرفت
آستارا	شيراز	خمين	زرند
سراوان	آباده	خرم آباد	رفسنجان
قزوين	داراب	كرمانشاه	سيرجان
دامغان	جهرم	اهواز	شاهرود
دوگنبدان	مرودشت	خرمشهر	بندرعباس
اشنويه	فيروزآباد	ماهشهر	بندرلنگه
فسا	شهركرد	آبادان	گرگان
رودبار	فردوس	رامهرمز	ساري
اردكان	مرند	بهبهان	رشت
شوشتر	بافت	گناوه	رامسر

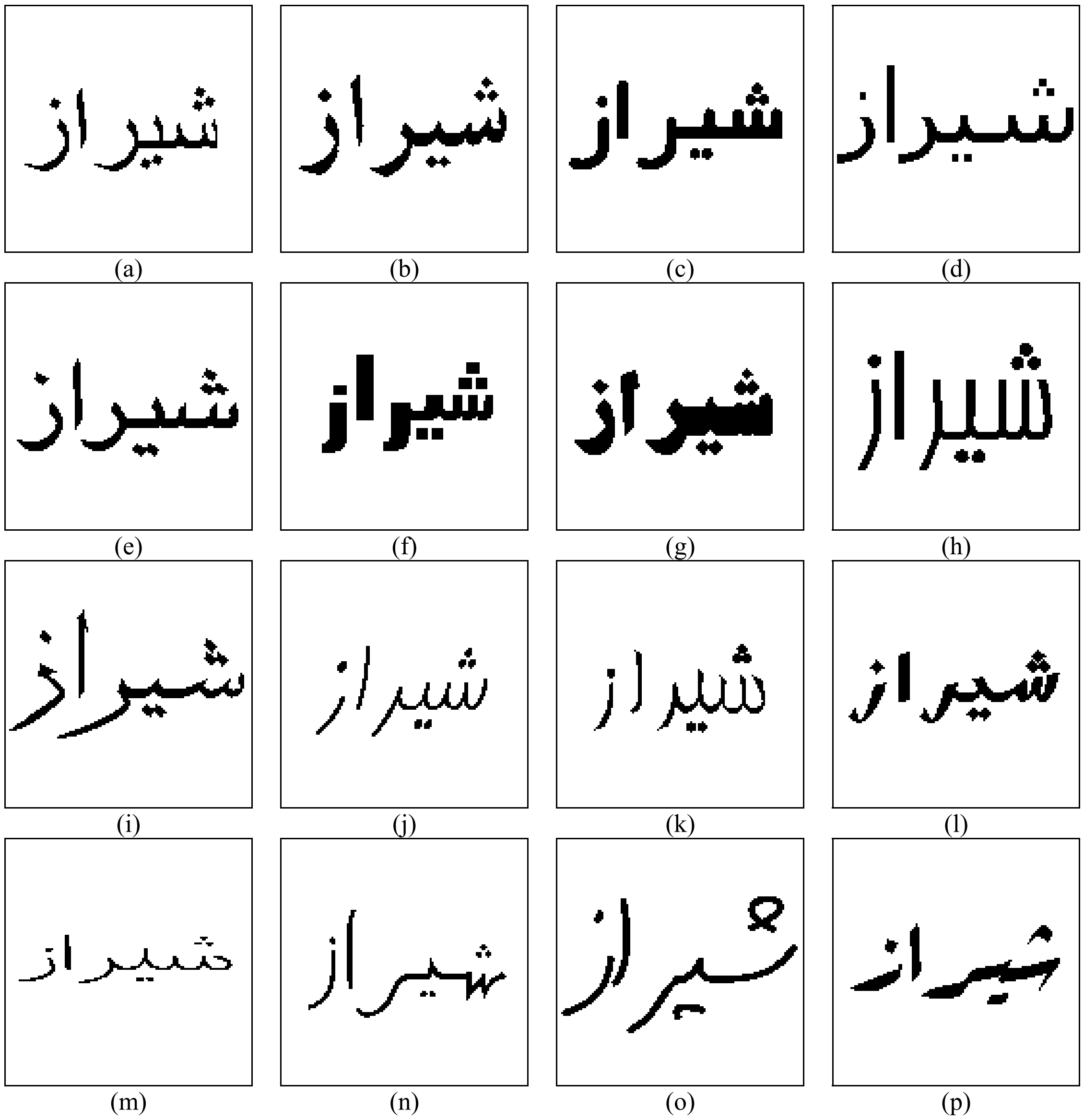


Figure A.1 Some training images of the word 'Shiraz' from the underlying dataset

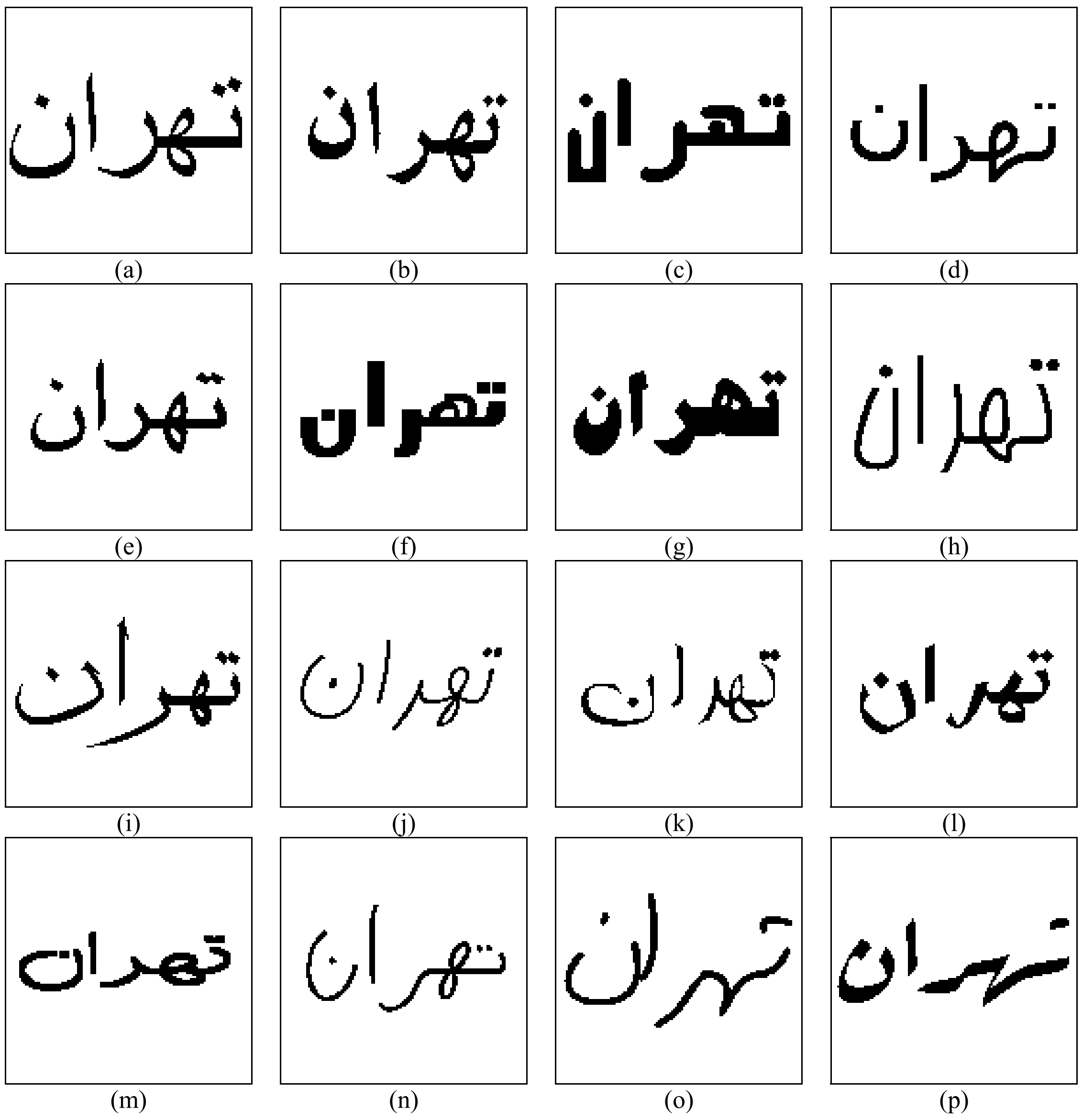


Figure A.2 Some training images of the word 'Tehran' from the underlying dataset

به نام خدا

تشخیص کلمات دستنویس فارسی با استفاده از مدلهای پیوسته پنهان مارکوف و ویژگیهای ساختاری

بوسیله:

محمد مهدی حاجی

پایان نامه

ارائه شده به معاونت تحصیلات تکمیلی به عنوان بخشی از فعالیتهای لازم برای اخذ درجه کارشناسی ارشد

در رشته ی:

مهندسی کامپیوتر گرایش هوش مصنوعی و رباتیک

دانشگاه شیراز

ارزیابی شده و تصویب شده توسط کمیته پایان نامه با درجه عالی

دکتر حسن اقبالی، استاد بخش مهندسی و علوم کامپیوتر (رئیس کمیته)................ [illegible]

دکتر سراج الدین کاتبی، استاد بخش مهندسی و علوم کامپیوتر (رئیس کمیته)................ [illegible]

دکتر احمد توحیدی، استاد بخش مهندسی و علوم کامپیوتر .... [illegible]

دیماه ۱۳۸۳

بسم الله الرحمن الرحيم

دانشگاه شیراز

تشخیص کلمات دستنویس فارسی با استفاده از مدلهای پیوسته پنهان مارکوف و ویژگیهای ساختاری

بوسیله:

محمد مهدی حاجی

استادان راهنما:

دکتر حسن اقبالی

دکتر سراج الدین کاتبی

رشته مهندسی کامپیوتر (هوش مصنوعی و رباتیک)

دیماه ۱۳۸۳