Copyright Warning & Restrictions

The copyright law of the United States (Title 17, United States Code) governs the making of photocopies or other reproductions of copyrighted material.

Under certain conditions specified in the law, libraries and archives are authorized to furnish a photocopy or other reproduction. One of these specified conditions is that the photocopy or reproduction is not to be "used for any purpose other than private study, scholarship, or research." If a, user makes a request for, or later uses, a photocopy or reproduction for purposes in excess of "fair use" that user may be liable for copyright infringement,

This institution reserves the right to refuse to accept a copying order if, in its judgment, fulfillment of the order would involve violation of copyright law.

Please Note: The author retains the copyright while the New Jersey Institute of Technology reserves the right to distribute this thesis or dissertation

Printing note: If you do not wish to print this page, then select "Pages from: first page # to: last page #" on the print dialog screen

ABSTRACT

DOCUMENT PREPROCESSING AND FUZZY UNSUPERVISED CHARACTER CLASSIFICATION

by
Shy-Shyan Chen

This dissertation presents document preprocessing and fuzzy unsupervised character classification for automatically reading daily-received office documents that have complex layout structures, such as multiple columns and mixed-mode contents of texts, graphics and half-tone pictures. First, the block segmentation algorithm is performed based on a simple two-step run-length smoothing to decompose a document into single-mode blocks. Next, the block classification is performed based on the clustering rules to classify each block into one of the types such as text, horizontal or vertical lines, graphics, and pictures. The mean white-to-black transition is shown as an invariance for textual blocks, and is useful for block discrimination.

A fuzzy model for unsupervised character classification is designed to improve the robustness, correctness, and speed of the character recognition system. The classification procedures are divided into two stages. The first stage separates the characters into seven typographical categories based on word structures of a text line. The second stage uses pattern matching to classify the characters in each category into a set of fuzzy prototypes based on a nonlinear weighted similarity function. A fuzzy model of unsupervised character classification, which is more natural in the representation of prototypes for character matching, is defined and the weighted fuzzy similarity measure is explored. The characteristics of the fuzzy model are discussed and used in speeding up the classification process.

After classification, the character recognition procedure is simply applied on the limited versions of the fuzzy prototypes. To avoid information loss and extra distortion, an topography-based approach is proposed to apply directly on the fuzzy prototypes to extract the skeletons. First, a convolution by a bell-shaped function is performed to obtain a smooth surface. Second, the ridge points are extracted by rule-based topographic analysis of the structure. Third, a membership function is assigned to ridge points with values indicating the degrees of membership with respect to the skeleton of an object. Finally, the significant ridge points are linked to form strokes of skeleton, and the clues of eigenvalue variation are used to deal with degradation and preserve connectivity. Experimental results show that our algorithm can reduce the deformation of junction points and correctly extract the whole skeleton although a character is broken into pieces. For some characters merged together, the breaking candidates can be easily located by searching for the saddle points. A pruning algorithm is then applied on each breaking position. At last, a multiple context confirmation can be applied to increase the reliability of breaking hypotheses.

DOCUMENT PREPROCESSING AND FUZZY UNSUPERVISED CHARACTER CLASSIFICATION

by
Shy-Shyan Chen

A Dissertation
Submitted to the Faculty of
New Jersey Institute of Technology
in Partial Fulfillment of the Requirements for the Degree of
Doctor of Philosophy

Department of Computer and Information Science

May 1995

APPROVAL PAGE

Document Preprocessing and Fuzzy Unsupervised Character Classification

Shy-Shyan Chen

Dr. Frank Y. Shih, Dissertation Advisor — Date
Associate Professor of Computer and Information Science, NJIT

Dr. Peter A. Ng, Dissertation Co-advisor — Date
Chairperson and Professor
of Computer and Information Science, NJIT

Dr. James A. M. McHugh, Committee Member — Date
Associate Chairperson and Professor
of Computer and Information Science, NJIT

Dr. DaoChaun Hung, Committee Member — Date
Assistant Professor of Computer and Information Science, NJIT

Dr. Nirwan Ansari, Committee Member — Date
Associate Professor of Electrical and Computer Engineering, NJIT

BIOGRAPHICAL SKETCH

Author: Shy-Shyan Chen

Degree: Doctor of Philosophy in Computer Science

Date: May 1995

Undergraduate and Graduate Education:

- Doctor of Philosophy in Computer Science,
  New Jersey Institute of Technology, Newark, NJ, 1995
- Master of Science in Mechanical Engineering,
  New Jersey Institute of Technology, Newark, NJ, 1989
- Bachelor of Science in Mechanical Engineering,
  National Cheng Kung University, Tainan, Taiwan, 1982

Major: Computer Science

Presentations and Publications:

D.T. Wang, C.S Wei, S.S. Chen, and P.A. Ng, "Cross Correlation of Sampled Boundary Distances - An Application to Object Recognition ," *Proc. 1st Int. Conf. System Integration*, Morristown, New Jersey, pp. 224-235, April 1990.

F.Y. Shih, S.S. Chen, D. hung, and P.A. Ng, "A Document Segmentation Classification and Recognition System," *Proc. 2nd Int. Conf. System Integration*, Morristown, New Jersey, pp. 258-267, June 1992.

S.S. Chen, F.Y. Shih, and P.A. Ng, "A Fuzzy Model for Unsupervised Character Classification," *Proc. 2nd Int. Conf. on Fuzzy Theory and Technology*, North Carolina, pp. 70-72, Oct. 1993.

S.S. Chen, F.Y. Shih, and P.A. Ng, "Fuzzy Typographical Analysis For Character Preclassification," *IEEE Trans. on SMC*, to appear.

S.S. Chen, F.Y. Shih, and P.A. Ng, "A Fuzzy Model for Unsupervised Character Classification," *Information Sciences Applications*, Vol. 2, No. 3, pp. 143-165, Nov. 1994.

S.S. Chen, F.Y. Shih, and P.A. Ng, "A Fuzzy Model for Skeletonization," {\m Proc. on the 3rd Int'l Conf. on Fuzzy Theory and Technology, North Carolina, Nov. 1994.

This dissertation is dedicated to
my mother and my family

ACKNOWLEDGMENT

First of all, the author would like to extend a warm thanks to his research advisors, Professor Frank Shih and Professor Peter Ng, for their valuable advising and guidance, constant support and encouragement. The author is so proud to co-author with them in many conferences, journals, and papers.

The author next appreciates the help and suggestions from Professors James McHugh, Dao-Chaun Hung, and Nirwan Ansari for serving as members of the committee.

The author also appreciates Professor David Wang who first encouraged me and helped me to enter the Department of Computer and Information Science. I'm glad to count him among my friends, too.

Next, the author wants to extend his sincere gratitude to all the helpful faculty and colleagues in the Department of Computer and Information Science, from whom, the author received encouragement. The author is grateful to the friends made during this research and would like to extend special thanks to them.

Finally, the author would like to extend sincere thanks to the author's mother and family for their never ending support, understanding, and encouragement during the years of advance studies. Heartily thanks to God who sent two angels, my son Jeffrey and my daughter Emilie to me during the lonely time. Special thanks to the author's father-in-law for his care of my two kids so the author could finish the research works. The author is grateful to his wife Vicky for her endless love and support in sharing her life with him.

TABLE OF CONTENTS

TABLE OF CONTENTS
(Continued)

TABLE OF CONTENTS
(Continued)

LIST OF TABLES

LIST OF FIGURES

LIST OF FIGURES
(Continued)

Figure **Page**

LIST OF FIGURES
(Continued)

Figure **Page**

CHAPTER 1

INTRODUCTION

Documents play a fundamental role in the communication of information. The rapid growing of knowledge and information exchange in today's office makes us busier for handling such a great deal of daily received documents. It needs more filing space for maintaining valuable documents and takes more time for retrieving a document from the filing room. Furthermore, the problem of loss, damage, or degradation of documents has been encountered in each office. Therefore, there is an urgent need to develop such a system which can automatically encode a huge amount of documents into computer processible forms in order to conquer the aforementioned problems. In a practical document processing system, the data of interests in any given text content take on a variety of formats such as character's sizes and fonts, graphics and pictures. The interpretation of graphics and pictures can be achieved by image processing and analysis alluded to the descriptive paragraphs. Therefore, the document processing system is the state-of-the-art enterprise of automating and integrating a wide range of processes and representations used for document perception. It integrates as parts many techniques involved in computer vision, image processing, artificial intelligence, and pattern recognition. In the last decades, there are many essential inventions and evolutions on computing equipments which make the document automation realizable. Firstly, the storage devices have been highly improved; e.g., HITFILE 650E optical disk filing system [53] can store up to 20,000 or 200,000 A4-size pages of compressed document image data per disk scanned at 400 dpi (dots per inch). Secondly, the speeds of the microprocessors and data communication have been tremendously increased. Thirdly, the high resolution digitizers have been successfully developed. Lastly, the powerful display facilities have been well manufactured. As the price of the equipments become inexpensive, we believe that document automation will be more widespread in the future.

1.1 System Overview

DPCS (Document Processing and Classification System) is designed for encoding a paper document into computer processible form to support the fundamental data which are used for document classification and information retrieval in TEXPROS (TEXt PROcessing System) [88], an intelligent document processing which includes filing and retrieval capabilities. In this section, the system requirements for DPCS are discussed. An overall organization of the system is proposed. The major functions and the input/output of each component of the DPCS are outlined.

1.1.1 System Requirements

A document can be interchanged either in an image form, which allows the document to be printed or displayed, or in a processible form [34], which allows document editing and layout revision. In today's office, documents can be generated by word processing software. The software reads document definitions in processible form, and produces a two-dimensional image form which is ready for printing or displaying. DPCS, which aims at the transformation of any information presented on paper into an equivalent symbolic representation, can somewhat be viewed as an inverse processing from an image form to a processible form. Note that a document of an image form can be mapped into various processible forms. That's still one of the active research issues in document analysis.

From a user's perception, a document is composed of a set of objects, which represent meaningful symbols to human comprehension, and the relationships among the objects. The intent of DPCS is to determine the physical layout structure and the contents of the objects. This result is then utilized to determine the logical meaning and the relationships of the objects for document classification and information extraction. DPCS must have the ability to extract the information from a document and store it in a form that can be recognized by the computer. The most significant information can be obtained from the content of text regions. DPCS, therefore, must first have the ability to separate

and encode characters correctly in the text. This corresponds to the classical optical character recognition (OCR). Secondly, some graphics may imply some important data, such as lines, that the user may want to know. Hence, DPCS must also be able to extract information from graphics regions. The third requirement for DPCS is to handle the photographic data, which includes the extraction of logos. Finally DPCS must be able to detect physical layout structure, which explicitly describes the internal organization of a document. Before the character recognition, graphic interpretation, and photographic processing can be executed, a processible image of a document must be ready, and therefore DPCS must be able to segment a document into several portions each of which consists of a single constituent such as text, graphics, and picture.

As a result of its functionalities, DPCS comprises six major subsystems:

1. Image acquisition.
2. Preprocessing.
3. Text recognition (equivalent to OCR).
4. Graphics interpretation.
5. Logo/picture processing.
6. Layout structure analysis.

The overall structure of the proposed system is shown in Fig. 1.1. The office documents are digitized and thresholded into binary images by a scanner or a facsimile. In order to encode information from a mixed-mode document which contains text, graphics and pictures, the first step is to segment the document image into individual blocks, namely, the text , graphics and picture blocks. The text blocks are further separated into isolated characters which are passed through a character recognition subsystem. The graphics blocks are further divided into text description and graphical primitives such as lines and curves. This text description can be dealt in the same way that the text blocks to be dealt with. The graphical primitives are further encoded into parametric

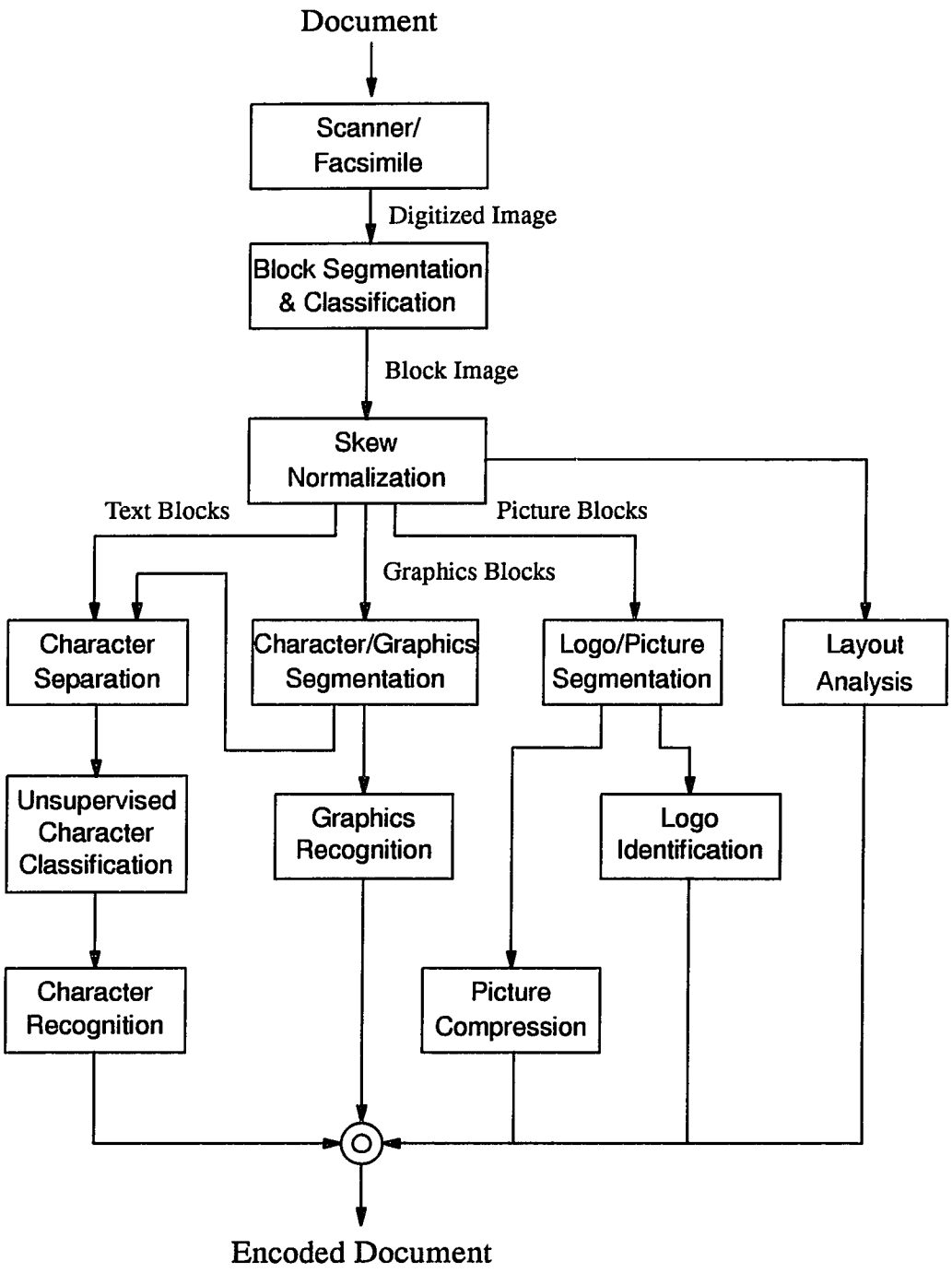


Figure 1.1 An overall organization of the proposed document processing system.

representations or symbolic words, or passed through a picture compressor. For instance, the logical AND-gate circuit diagram is converted into the name "AND." The logo picture discriminated from the half-tone picture is identified and transformed into its symbolic words. The picture blocks, consisting of high transitional density of white-to-black pixels, can be compressed for efficient storage using existing compression techniques. The physical layout structure is carried out by grouping the related blocks and detecting the inter-block relationships. In the remainder of this section, the function of each subsystem is briefly described. The input and output of each component are defined as well.

1.1.2 Image Acquisition

Upon receiving a physical document, the image acquisition subsystem converts it into a digitized document image. In order to be in a form suitable for computer processing, a continuous image function $f(x, y)$ must be digitized both in space and in amplitude, which are referred to as *image sampling* and *gray-level quantization*, respectively. Usually, the device for document image acquisition is an optical scanner which is connected to a computer and a scanning control software to specify user's requirements. Currently, desktop scanners are available and will be prevalent as their price continues to go down. Normally, these scanners can generate digitized images with 256 gray levels or colors, and 300 dot per inch (dpi), at least. Since a bi-level image, represented respectively by 1 and 0 for object and background, is commonly used for document analysis, binarization must be included to transform a gray-level into a binary format. This can be done while scanning the document, to select a global threshold and decide which value should be set to each pixel. When the background of a document (such as patterns on paper or a scened check) is not clear, an adaptive thresholding technique must be applied on the local features of the image.

1.1.3 Preprocessing

The preprocessing subsystem transforms a binary or gray-scale image into text, graphics, or logo/picture blocks.

Preprocessing, also known as document segmentation, separates different content portions. For example, graphics, lines, photographics, and text must be distinguished. From a user's perception, a document comprises a set of logical objects, i.e., blocks, such as title, author, text body, and other graphics. In the hierarchical structure of a document, objects which are close to each other are likely to be more related, and are linked together to comprise a block. This technique is known as block or area segmentation.

Each block is characterized by its basic geometric features, such as the coordinates of the upper-left corner, width, height, etc., and the derivative features, such as aspect ratio, density (number of black pixels in a block), centroid, number of crossing, etc. Note that the features of text blocks have some regularity and are invariant, which will be discussed in chapter 2. Based on the features associated with each block, a rule base is established for block classification. Examples of classes are text, graphics, pictures, vertical and horizontal lines.

In preprocessing, many block segmentation methods, such as recursive X-Y cut, are sensitive to skew. In our system, a improved run-length smoothing algorithm (RLSA) is applied and is tolerant to skew degree. Even though RLSA can segment correctly with a small skew angle, the blocks seem to be intersected each other, and therefore it is difficult to conduct the layout analysis. Another problem is that the block images occupy more space than they are normalized.

Previous works on skew normalization are based on two facts. One is that the projection profiles in horizontal and vertical directions are steepest than those in any other direction when the document is normalized. The other fact is that most of the characters within one text line are aligned with their base line. Using projection profiles, the document is rotated a small angle each time, say one degree, and the skew angle is detected

where the maximum variation of the projection profiles happens. In order to find the base line, a text line must be located first, and then the direction of the collinear is extracted by a technique, such as Hough transform.

1.1.4 Text Recognition

The text blocks obtained by the preprocessing subsystem are transformed into ASCII (American Standard Code for Information Interchange) coded characters by the text recognition subsystem. Generally, a text recognition consists of four stages -- segmentation, feature extraction, character recognition based on the features and a set of rules, and error correction. To facilitate character recognition, an unsupervised character classification is performed prior to it. The proposed text recognition subsystem consists of the following stages:

1. Character isolation.
2. Unsupervised character classification.
3. Character recognition.
4. Merged and ambiguous character recognition.
5. Dictionary checking.

Character isolation is a process to separate single characters from the text block. In general, the algorithms for character separation can be categorized into two types. One is the top-down method [23, 40] which is based on projection profiles obtained by projecting an image onto specific axes. The profiles show the spatial structure of the document image. The other is the bottom-up method [77] in which the image is first decomposed into fundamental connected components, such as characters, and then merged into words, lines, paragraphs, and so forth. Our approach is a combination of both methods, since a text block obtained from block segmentation contains a single text line.

A text line can be divided into words by detecting those valleys on the projection profile onto the x-axis with an appropriate spacing threshold. Characters can be further separated by setting a relatively smaller threshold. However, overlapped and merged characters (which may be caused by the font chosen, by the inadequacy of resolution of the scanning device or by the high brightness threshold we set to avoid breaking characters) can not be correctly isolated by projection profiles. Overlapped characters can be handled by connected component analysis. On the other hand, text lines are grouped into paragraphs based on the line spacing and other geometric features.

Hundreds of font styles and sizes are available for printed characters. Mathematical symbols, foreign characters, and other special typographic elements may also appear in the machine-printed documents. The tremendous range of possible input for the document makes it very expensive for the character classification or recognition. Typically, a document may contain a few hundred of characters (e.g., a memo or a short letter), thousands of characters (e.g., an article or short paper), millions of characters (e.g., a book). It becomes impractical to recognize every character in the document one by one. In reality, most documents are composed of only a few different font styles and sizes (typically, one to three). Each font contains 26 upper-case, 26 lower-case, and some other symbols. Therefore, the text of the document can be considered as a combination of a set of finite symbols. If the recognition problem can be reduced to such a set of symbols, then the complexity of OCR will be significantly reduced. Unsupervised character classification will not extract complex features and recognize them. Instead, it only classifies the whole character elements into a set of pattern categories. Thus, a simple technique such as pattern matching can be applied.

Unsupervised character classification reduces the character recognition problem to one of identifying the set of prototypes. The prototypes are sorted according to their number of occurrence in decreasing order. Prototypes are then recognized one by one according to certain rules. If an uncertainty of recognition is found for some prototype, the

classification halts and the next one is continued. As a result of recognition, characters with high reliability are recognized, and prototypes with uncertainty are labeled and further processing is needed.

After single character recognition, the remaining unrecognized prototypes include merged characters, broken characters, non-English symbols, etc. A more advanced technique is necessary to cater for these ambiguous problems. A splitting or grouping technique will be applied to the merged, or broken characters respectively. It performs a "hypothesize and test" cycle. In each cycle, entities are hypothesized by splitting a single object into pieces, or grouping different pieces into a single object. These hypotheses are tested by several evaluation algorithms and dictionary checking. Non-English symbols, such as Greek characters, can be stored in the rule base through learning. (Users must interactively enter the solution to the system at the first time.)

No dictionary of a living language can ever be complete. Therefore, dictionary checking is performed to reduce ambiguity and increase reliability of the final output. The errors may occur by mistyping or misclassification. The errors may be corrected according to the hypothesis with highest reliability. However, it is not possible to have 100% correctness. Moreover, the encoded data will be displayed, and those ambiguous words found in dictionary checking will be highlighted. Users can interactively edit or correct the errors due to classification or the originator.

1.1.5 Graphics Interpretation

The graphics interpretation transforms graphic block images into descriptions of symbolic representation. Graphics, which are mainly composed of a set of line drawings and some text, usually occur in technical and business reports. Always, graphics expresses information which is not easily described by description (such as an road map or an electronics drawing) or information which is succinct (such as diagram, flow chart, table, etc.). Therefore, graphics interpretation, when applied to office documents, does not

gather every logical meaning in the character description. For example, expressing a road map or a flow chart by a paragraph is more confusing than using its graphics representation. In such a case, the interpretation converts the contents into computer-coded forms in terms of physical attributes, such as a line with its end points and its width, a rectangle box with its thickness, etc.

In some other graphics, such as tables, the contents and the logical relationships among the contents may become significant for the users. Thus, the logical relationships must be extracted based on the physical structure. To obtain the graphics description, the following operations are performed on the graphics block image [8, 40]:

1. Separation of text strings from graphics: Unlike text region, character strings in graphics can be distributed irregularly, e.g., the label of y-axis may rotate 90δ. Therefore, separation of text from graphics elements is more complicated than block segmentation. Top-down approach is not appropriate because of the irregularity and the bottom-up technique [23] is better. Component generation is firstly applied to locate each connected component. Character strings are grouped one by one according to their homogeneity and collinearity.
2. Recognition of text parts: After the character strings are separated from graphics, it is passed to the character recognition system as described above. The only difference is that the rotation invariance must be considered, or the rotation angle of the strings must be detected.
3. Identification of line segments and their attributes: Graphics includes mostly line segments and some solid regions. Solid components can be extracted by the morphological opening, and encoded by their contours. The thin components are skeletonized, and the segments and nodes, such as end points, and joint point, are extracted.
4. Recognition of graphical primitives and their attributes: The graphics interpretation system aims at the transformation of a line drawing from a set of pixels into a set of

logical objects based on the user's comprehension. Logical conception is based upon the individual knowledge background and the field of application. However, no matter how different it is, they are all based on the physical graphical objects and their attributes. According to this fact, the system should support a set of primitives, such as lines, arcs, triangles, etc., on which the logical objects are based.

1.1.6 Logo/Picture Processing

The logo/picture subsystem converts picture block images into logo descriptions or compressed image with their geometric attributes.

Photographics appears in a document at times whenever they can give a clear and concise description at a glance. The only purpose in this stage is to condense the data as much as possible without losing details. Furthermore, a logo frequently appears in the cover page or a commercial letter which may occupy a lot of space, but contains very little significant information. This must be recognized based on the user's model base [73]. The user determines whether or not to add a picture which looks like a logo but not found in the model base.

1.1.7 Layout Structure Analysis

The layout structure analysis subsystem generates the structural representation of the physical layout of document from the description of blocks.

Any given document is characterized by its logical structure [34], which associates the content of the document with a hierarchy of logical objects. Examples of logical objects are title, abstract, author, date, paragraph, table, caption, etc. The logical structure is derived from the content and the internal organization of the document, corresponding to the layout structure. Typical layout objects are (text) line, block, frame, and page.

Layout objects are rectangular areas that do not overlap. Normally, the layout structure can be built as a tree structure, according to some rules, with the layout objects

forming the nodes, and the relationships between objects forming the edges in the graph. Basic objects are at the lowest hierarchical levels, the leaves of the tree. It is only through them that content portions can be directly associated by means of the attribute "references to content portions." Any intermediate node represents a composite layout object, which consist of components that may be other composite objects and/or basic object.

1.1.8 System Output

The overall output is designed to minimize the storage need, and to support quick retrieval for a browsing system. It should be possible to reconstruct the original image either partially or completely. In addition, the output format may to be extended to the filing system, browsing system, retrieval system, document classification system in TEXPROS, or other applications. The representation of the encoded document comprises:

- Document profile: document profile contains general information for handling the document as a whole. It may include the document ID, the original document image filename, the date, the number of pages, the size of each page, and other common information. The document type, editing and retrieval information, will be included when it is incorporated to a filing system.
- Layout structure: layout structure yields the hierarchical document architecture of the document. The attributes associated with each node are described, and the addresses of the content to which the leaves reference are specified. This is the information on which the document classification and browsing system are based.
- Encoded data: the encoded data are stored block by block, corresponding to the leaves of the layout structure. Note that only text, graphics, and logo blocks are included. Along with the layout structure, certain blocks can be quickly extracted in a browsing system.
- Image data: the original image data should be maintained in such a way that the users are allowed to browse through the image whenever it is requested. However,

the original image needs huge space, and it is impractical to keep it as a whole. In addition, keeping a set of block images does not reduce storage significantly. For text regions, only a set of fuzzy prototypes, which are obtained through an unsupervised character classification, are stored. For each text block, the index of characters, which correspond to the prototypes, are then sequentially recorded. The original block image can be approximately reconstructed upon the user's request. The original image of line drawing and logo are not maintained, since they can be reconstructed from their description or their model base. However, picture image is only stored in a compress format.

1.2 Review of Literature

1.2.1 TEXPROS

Document automation is a document processing which integrates a classification system for converting a human-readable document image into machine-readable codes, a document classification system for determining the document types, a document categorization system for filing the document, an information retrieval system for locating a document according to the user's query, a browsing system for viewing the document in either image form or encoded form, and a data management system for manipulating the huge amount of information.

Many systems have been developed to achieve parts of the document automation. Diamond [81] allows users to create, edit, and transmit multimedia documents. MULTOS (MULTimedia Office Server) [5, 67] is a distributed office system which extends the office document architecture (ODA) [34] by including a conceptual structure, and supports a well-defined query language and many query processing techniques. MINOS [15], an object-oriented multimedia information system, describes multimedia documents using a logical model and a physical model. It provides integrated facilities for creating complex document objects, and for extracting and formulating new information from

existing documents. MAIFIA (MAil-FIlter-Agent) system [51], based on the MULTOS document model, provides an automatic document classification. SMART [70] supports keyword based retrieval for bibliographic databases. Resumix [83] is concerned with the document categorization, which reads applicant's resumes, creates a resume summary, matches candidates to job openings.

TEXPROS is an intelligent document processing system built based on object-oriented programming and rule-based specification techniques. The system is a combination of filing and retrieval systems. TEXPROS has functional capabilities of automating (or semi-automating) the following activities [88]:

- document classification according to their types (e.g., memos, books, letters, papers, etc.);
- document categorization depending on the user's specification of the document filing organization;
- information extraction from the document based on the user's interest;
- information retrieval under the user's query;
- information browsing through the documents;
- synthesizing information from existing documents and either physically producing or storing the information as a view;
- document reproduction from the computer codes.

Based on the functional capabilities for which the system designs, most of them rely on the encoded information from the physical document. Therefore, another primary capability of TEXPROS is to efficiently and correctly transform the original documents into computer-recognized codes. The transformation needs to incorporate a classical OCR as well as a document analysis techniques such as the preprocessing, postprocessing, and layout structure analysis.

1.2.2 Review of OCR and Document Processing

Apart from the on-line character recognition, OCR is one of the research fields in off-line recognition, which deals with the recognition of optically processed characters rather than magnetically processed ones after the writing or printing is completed [39]. The history of OCR research is comparatively old in the field of pattern recognition. In the early days of the research dealing with pattern recognition, OCR is a major focus because characters were handy to be dealt with and were regarded as a problem that could be solved easily [MoS92]. Later on, a great deal of difficulties are surfaced, and many people switched their interests to the other subjects. So far as we know, the earliest idea of OCR could be Tausheck [Tau35], who obtained a patent on OCR in Germany in 1929, and Handel [Han33] did the same in the U.S in 1933. The principle of Tausheck's patent is a template/mask matching, which used an optical and mechanical template. Light passing through a set of mechanical masks is captured by a photodetector and scanned mechanically. When an exact match occurs, light fails to pass through, and so the machine recognizes the characters.

In the early age, OCR dealt with the isolated character images. At present, OCR, and document analysis in general, has a broad sense of dealing with the document as a whole. The field of document analysis can be traced back through a computer lineage that includes digital signal processing and digital image processing. Digital signal processing, initially fostered by the introduction of fast computers and algorithms such as the fast Fourier transform in the mid 60s, has its objective the interpretation of one-dimensional signals. In the early 70s, with larger computer memories and faster processors, image processing methods and systems were developed for analyzed two-dimensional signals. Specialized fields of image processing are associated with particular applications such as biomedical image processing, machine vision, and computer vision. In the mid to late 80s, document image analysis began to grow rapidly due to the hardware advances that supported fast processing at reasonable cost.

Commercial document analysis systems are now available for storing business forms, performing optical character recognition on typewritten text, and compressing engineering drawings [59]. Document analysis research continues to pursue more intelligent handling of documents, better compression, and faster processing. Several document processing and analysis systems [1, 3, 7, 14, 16, 44, 52, 71, 77, 95] have been developed in the last decade. Due to the variation of various documents, document analysis includes much more functions than before. Besides the character recognition, typically, these document analysis systems combine with segmentation techniques, normalization techniques, classification techniques, and postprocessing techniques. These systems are constrained for their application. Some of them are designed for Japanese documents [1, 3]. Most of them described their overall architectures and focus on some specific functions. There are still many open problems remained.

Many works also have contributed to the subareas in the document analysis. In area segmentation, close components which are more related are intended to be merged together. Wang et. al [91] introduced the run length smoothing algorithm (RLSA) and has been improved by Shih et. al [73]; a recursive X-Y cut (RXYC) proposed by Nagy also has been widely used [55, 56, 90]; Yamada and Hasuike [96] developed an algorithm based on enhanced border following algorithm; and Scherl et. al [72] used a method to transform document image to spatial frequency domain and discriminated by testing background and by histogram moments. Block classification is based on some invariant features among text, graphics, and picture images; the rule-based systems have been developed [21, 95, 73]; Wang and Srihari [90] classify newspaper based on texture analysis by studying a black-white pair run length matrix and a black-white-black run length matrix. Layout structure is essential to classify a document, and many techniques have been carried out in this field; ANASTASIL [17], developed by Dengel and Barth, use a hybrid knowledge-based system for document layout analysis; Dengel et. al. [18] presented the principles of the model-based document analysis system called ΠODA (paper

interface to ODA) for single-sided business letters in Germa; Nagy [55, 56] used x-y tree segmentation and labeling for a structure document and for classifying technical journals; Tsuji [84] developed a method to generate a syntactic tree structure; Bayer [4] built a frame system for interpretation of structured documents, which comprises a document representation language, a set of knowledge sources, an interpreter for this language (rule based), and a control algorithm; Tsujimoto and Asada [85] described the document processes consisting of a document analysis component to extract text, a document understanding component to obtain logical structure, and a character segmentation/recognition component. From roadmap to engineering drawing, graphical image is a problem with highest variation, and still have many unsolved problems. In this field, many research achieved for specific issues; Freeman [24] described the computer processing of line-drawing images; Fletch and Kasturi [23] developed an algorithm for text string separation from mixed text/graphics images based on grouping collinear components; Bow and Kasturi [8, 40] described a system for interpretation of line drawings; Ejiri et. al. [19] design a recognition system for engineering drawings and maps; an automaed conversion of engineering drawings to CAD form was presented by Filipski and Flandrena [22], Vaxiviere and Tombre [87]. Other areas also gained some contributions. Pavlidis [64] and White [94] developed dynamic thresholding techniques for character segmentation applied to scenic backgrounds of bankchecks. Zen and Ozawa [98] described a method to extract a fair document from handwritten mixed mode manuscript. Fujisawa et. al. [25] proposed a pattern-oriented segmentation method for tabular forms.

Research cannot exist without its applications. OCR has its use in many field. For instance, a user would like to excerpt a section from a magazine or business report. Another example is to help blind people reading. On a larger scale, libraries face the ongoing degradation of paper material and insufficient space to store books; manufacturing firms have the costly task of maintaining, location, and modifying their engineering drawing; and the postal system has the inefficiency for handling huge volume of mails.

Currently, the most important use of OCR stems from the general activities directed towards office automation dominating information processing [39], which is divided into three main application areas: text entry in office automation such as a desktop publishing, which integrates a scanner device into a word processing environment; data entry such as in banking applications, there are constrained paper formats, a limited character set, and high throughput requirements; and process automation such as the post automation for postal mail sorting. It is believed that OCR is at a historical juncture [62]. In the next decade, the applications of OCR must be widely spread in many other areas. One of the main streams should be the office document processing and information retrieval system.

The problems of efficiency, accuracy, and reliability in an OCR system have been encountered because hundreds of font styles, character sizes, special symbols, and different printing devices producing various qualities of images are to be dealt with. Another serious problem is the unavoidable noise while printing. Therefore, some postprocessing strategies such as dictionary checking and semantics understanding are necessary to combine with OCR [36, 39, 71, 76]. In this disstation, we focus on the document preprocessing and text recognition. Our intention is to reduce the complexity of the document processing, and improve the system performance.

1.2.3 Feature Extraction and Character Recognition

In document processing, a main subject is the conversion of textual regions to their representative computer codes. This procedure associated with a supervised classification is known as the character recognition. Apart from the unsupervised classification, which is associated with a clustering technique to group similar objects, supervised classification is an act of abstraction, which establishes a mapping from a normally high-dimensional feature space into a discrete space of class labels. This mapping, in the case of character recognition, is the inverse of printing. The dominant problem is to cope with the variability in character image appearance.

It is believed that feature extraction is one of the most difficult and key issues in pattern recognition. Many researches have been achieved in character recognition. Basically, most of the previous works are based on the binary character images. Impedovo *et al.* [39] classified the main techniques into four categories based on the selected features used for the classification – template matching and correlations, distribution of points, transformations and series expansions, and structural analysis.

(a) *Template matching and correlations.* Template matching is relatively old technique in pattern recognition since it is easy to implement. Back to 1929, Tausheck [80] proposed a template/mask matching which used an optical and mechanical template. When an exact match occurs, light fails to pass through the mechanical masks, and so the machine recognizes the characters. Mathematically speaking, the principle is the axiom of superposition.

In essence, template matching is to match an input character matrix against a set of templates, and the distance or similarity representing the correlationship between the input pattern and each template is calculated. The input pattern is then classified according to a minimum distance or a maximum similarity criterion. Some pixel-based correlation functions have been discussed in chapter 3. In order to reduce the complexity of calculating the correlation, some logic template matching methods were developed. One of the simplest is called the peephole method. A set of appropriate pixels are chosen for both black and white regions, and the input character is distinguished from other characters relying on the set of peepholes. The first announced OCR system using peephole method is called ERA [20], in which the total number of peepholes used was 100 to distinguish printed numerals. Iijima *et al.* [38] used 44 peepholes at 10 pixels/character to recognize 72 alphanumerals.

If a set of common peepholes can be selected for every character, a binary decision tree can be built up for the classification. The leaves of the decision tree represent the character classes, and each internal node of the decision tree represents a specific

peephole location having two branches, known as "black" and "white", leading to a terminal node. Wong *et al.* [95] presented this approach in 1982, in which each peephole in a character has three statuses – "white," "black," and "unreliable." When an unreliable peephole of a character is encountered, the character class will appear in both subtrees lead from this node. Fig. 1.2 illustrates an example of the decision tree, where P1 is "black" for class "A", "white" for class "B", and "unreliable" for class "C."

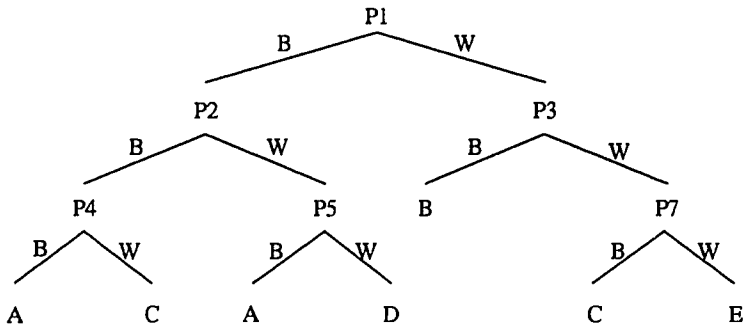


Figure 1.2 Illustration of a decision tree based on peephole method.

Template matching is not restricted on pixel-based operation. One of the variation is performed on the contour of the characters. To reduce the complexity, a cross correlation of sampled boundary distances was proposed by Wang *et al.* [92]. Samples of distances defined from a major axis to points located on the boundary of an object image and a correlation function was used to calculate the similarity.

(b) *Distribution of points*. This category comprises techniques that extract features from the statistical distribution of points. Representative techniques in this category are the use of moments, crossing counts, and zoned features.

Two-dimensional moments have been used for a number of image processing tasks and character recognition [86, 12, 43]. The moment of order $p + q$ of a continuous image function $f(x, y)$ is defined as

$$m_{pq} = \int\int x^p y^q f(x, y)dxdy \tag{1.1}$$

where p and q are non-negative integers. For digital images, the moment of order $p + q$ has the following form

$$m_{pq} = \sum_x \sum_y x^p y^q f(x, y) \tag{1.2}$$

In order to preserve the invariance to translation, it is recommanded to use the central moments by the following equation

$$M_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q f(x, y), \tag{1.3}$$

where $\bar{x} = \frac{m_{10}}{m_{00}}$ and $\bar{y} = \frac{m_{01}}{m_{00}}$ denote x- and y-coordinates of the centroid of the image. Furthermore, normalized central moments, which are invariant to both translation and scale of an image, can be derived from the central moments as [27]

$$\Gamma_{pq} = \frac{M_{pq}}{M_{00}^n}, \qquad \text{where } n = \frac{p+q}{2} + 1, \qquad p + q \geq 2. \tag{1.4}$$

The moments of high order are too sensitive to the noise, and low order moments are preferred to be applied for recognition. From the second and third moments, a set of seven invariant moments, which are invariant to translation, rotation, and scale change, have been derived as [35]:

$$\phi_1 = \Gamma_{20} + \Gamma_{02} \tag{1.5}$$

$$\phi_2 = (\Gamma_{20} - \Gamma_{02})^2 + 4\Gamma_{11}^2 \tag{1.6}$$

$$\phi_3 = (\Gamma_{30} - 3\Gamma_{12})^2 + (\Gamma_{03} - 3\Gamma_{21})^2 \tag{1.7}$$

$$\phi_4 = (\Gamma_{30} + \Gamma_{12})^2 + (\Gamma_{03} + \Gamma_{21})^2 \tag{1.8}$$

$$\begin{aligned}\phi_5 = {} & (\Gamma_{30} - 3\Gamma_{12})(\Gamma_{30} + \Gamma_{12})[(\Gamma_{30} + \Gamma_{12})^2 - 3(\Gamma_{03} + \Gamma_{21})^2 \\ & + (\Gamma_{03} - 3\Gamma_{21})(\Gamma_{03} + \Gamma_{21})[(\Gamma_{03} + \Gamma_{21})^2 - 3(\Gamma_{30} + \Gamma_{12})^2]\end{aligned} \tag{1.9}$$

$$\phi_6 = (\Gamma_{20} - \Gamma_{02})[(\Gamma_{30} + \Gamma_{12})^2 - (\Gamma_{03} + \Gamma_{21})^2] + 4\Gamma_{11}(\Gamma_{30} + \Gamma_{12})(\Gamma_{03} + \Gamma_{21}) \tag{1.10}$$

$$\phi_7 = (\Gamma_{03} - 3\Gamma_{21})(\Gamma_{30} + \Gamma_{12})[(\Gamma_{30} + \Gamma_{12})^2 - 3(\Gamma_{03} + \Gamma_{21})^2] \\ - (\Gamma_{30} - 3\Gamma_{12})(\Gamma_{03} + \Gamma_{21})[(\Gamma_{03} + \Gamma_{21})^2 - 3(\Gamma_{30} + \Gamma_{12})^2] \quad (1.11)$$

Peephole method is not always limited to a single pixel. Instead, it can be expanded into a slit or a window and becomes more flexible. Examples of this technique are crossing counting and zoning. In crossing counting technique [13, 33, 45], the crossing counts are measured in various directions and locations. Rohland [68] proposed this technique in 1954. In this method, Weeks [93] scanned in four directions, i.e., 0°, 45°, 90°, and 135°, for each of which six equally spaced and parallel lines are used to cover a character. This technique is often used by commercial system, such as CSL 2610 marketed by Computer Gesellschaft Konstanz [39], because it can be performed at high speed and requires low complexity.

Zoning technique can be regarded as a hybrid of template matching and structural analysis. The general idea is that pixelwise matching is replaced by subregionwise matching, and the matching objects are the so-called zoned features within the subregions. Specifically, zoning technique first divides the frame containing the character into subregions, in each of which some local features are detected. This method has been applied by Suen [78], who used the densities of points in the subregion as the features. NEC [58] obtained a U.K. patent with this approach, in which four kinds of directional features, i.e., vertical, horizontal, and two orthogonal diagonal directions, are detected.

(c) *Transformations and series expansions*. Transform theory has played a key role in image processing for a number of years, and has been applied in for image enhancement, restoration, encoding, and description. Fourier series expansion is the most popular and so naturally had been applied to character recognition systems. Fourier descriptors (FD) have been shown as a useful set of features to describe closed curves [97, 28, 65].

There are two kind of representations of FD's. The FD's given in [97] are defined as follows. Assume γ is a clockwise-oriented simple closed curve with parametric representation $(x(l), y(l)) = Z(l)$, where l is arc length and $0 \leq l \leq L$. Denote the angular direction

of γ at point l by the function $\theta(l)$. The cumulative angular function $\phi(l)$ is defined as the net amount of angular bend between starting point and point l, i.e., $\phi(l) = \theta(l) - \theta(0)$. Note that $\phi(L) = -2\pi$. The domain of definition $[0, L]$ of $\phi(l)$ simply contains absolute size information. A normalized variant $\phi^*(t)$, whose domain is $[0, 2\pi]$ and such that $\phi^*(0) = \phi^*(2\pi) = 0$, is defined as:

$$\phi^*(t) = \phi\left(\frac{Lt}{2}\pi\right) + t. \tag{1.12}$$

ϕ^* is invariant under translations, rotations, and changes of perimeter L. Note that $\phi^*(t) \equiv 0$ for a circle which is in some sense the most shapeless closed curve. Now ϕ^* is expanded as a Fourier series

$$\phi^*(t) = \mu_0 + \sum_{k=1}^{\infty}(a_k \cos kt + b_k \sin kt). \tag{1.13}$$

In polar form the expansion is

$$\phi^*(t) = \mu_0 + \sum_{k=1}^{\infty} A_k \cos(kt - \alpha_k) \tag{1.14}$$

where (A_k, α_k) are polar coordinates of (a_k, b_k). A_k and α_k are the FD's for curve γ and are known as the k^{th} harmonic amplitude and phase angle.

The other representation was proposed by Granlund [28] and developed by Persoon and Fu [65]. A point moving along the boundary generates the complex function $u(l) = x(l) + jy(l)$ which is periodic with period L. The FD's become

$$a_n = \frac{1}{L}\int_0^L u(l)e^{-j(2\pi/L)nl}dl \tag{1.15}$$

and

$$u(l) = \sum_{-\infty}^{\infty} a_n e^{j(2\pi/L)nl}. \tag{1.16}$$

(d) *Structural analysis.* Humans look at a character as a linelike object. Structural features are generally used to derive the global structure of a pattern. They describe the geometrical and topological properties of a character. Generally, structural analysis consists of three stages – Skeletonization, feature extraction, and classification. Ideally, a thinned line in Euclidean space is a line without width. However, this definition cannot be accepted in digital picture representation. Therefore a one-pixel width is used and the connectivity of a line needs to be defined. The most used features are strokes and curves in various directions, end points, corner points, junction points, intersection points, loops and their positions in the bounding box of a character.

One of the techniques in this class uses an attribute grammar to describe the general structure of a character and leads to a terminal node representing a character class. Fig. 1.3 illustrates an example of character generation by strokes. The generation diagram in Fig. 1.3 is similar to the state transition diagram that could be explained by a regular grammar, and can be implemented by a finite state automata. This approach was applied by Lee *et al.* [47], which employing attribute dependent regular programmed grammar for the recognition of Korean character.

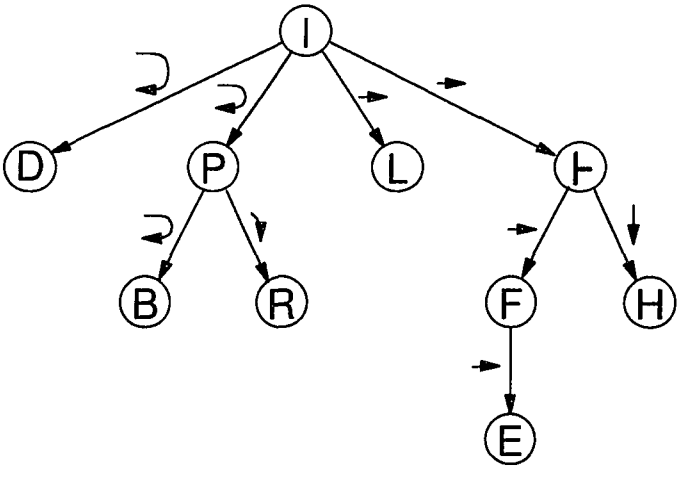


Figure 1.3 An example of character generation by strokes.

Another approach is stream following analysis. The strict description of stream following was given by Perotto [66]. Fig. 1.4 shows an example given by Perotto, which uses the order of raster scanning from top to bottom and from right to left. The description of the numeral "6" is labeled as shown at the bottom of Fig. 1.4, where labels P_1 and P_2 denote the two principles which are encountered while scanning, label D^2 denotes the branching of order 2 for P_2, and label U^2 denotes the state D merges to one segment at the position close to the left.

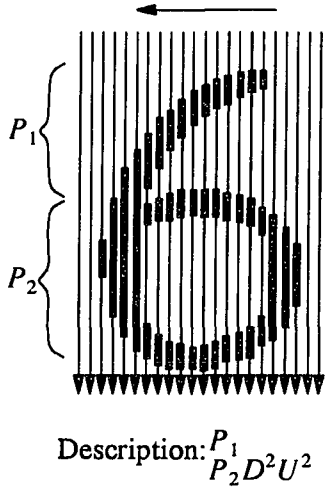


Figure 1.4 Illustration of the stream following method [66].

Siy and Chen [74] consider a character as a directed abstract graph, of which the node set consists of tips, corners, and junctions, and the branch set. The line segments of a character are fuzzily classified to branch types such as straight lines, circles or portions of circles, which in turn are divided into four, six, and five categories. The branch feature set are shown in Fig. 1.5, in which the branch features are labeled by the symbols under them. A character is characterized by its functional descriptions including the branch features and their corresponding nodes. In Fig. 1.6, for example, (a), (b), and (c) have the same functional descriptions, i.e., $F(a) = F(b) = F(c) = H(1,2)\ V(1,3)\ D(3,4)$. Other variants of the character 5, as shown in Fig 1.6(d) and (e), have the functional

descriptions $F(d) = H(1,2)\ P(1,3)\ D(3,4)$ and $F(e) = P(1,2)\ V(1,3)\ D(3,4)$, respectively. Thus, the character 5 in Fig 1.6 can be represented by

$$F(5) = F(5a) + F(5b) + F(5c) + F(5d) + F(5e)$$

$$= h(1,2)\ V(1,3)\ D(3,4) + H(1,2)\ P(1,3)\ D(3,4) + P(1,2)\ V(1,3)\ D(3,4).$$

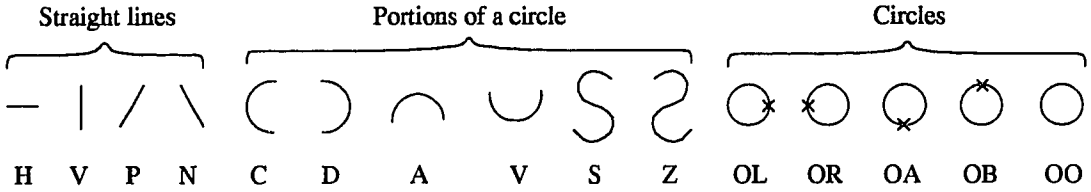


Figure 1.5 The branch feature set [74].

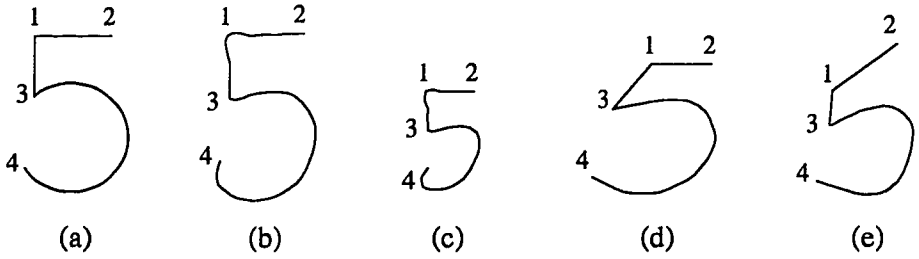


Figure 1.6 Some examples of character 5 [74].

Siy and Chen use a fuzzy logic for the measure of straightness and orientation to determine a branch's type. A measure of straightness for a branch is defined by

$$f_{SL} = \begin{cases} 1 - S/S_t, & \text{if } S < S_t \\ 0, & \text{if } S \geq S_t. \end{cases} \tag{1.17}$$

where S_t is the threshold least squares error. A given branch is classified as a portion of a circle if $0 \leq f_{SL} < 0.5$; or a straight line otherwise.

A new method of thinning using vectorization was developed by Pavlidis [63]. The method is based on a line adjacency graph (LAG), which can be performed directly on the run-length encoding. The resulting strokes and other shapes are mapped into binary features which are then fed into a statistical Bayesian classifier [2, 41]. Typical shape types which can be found by the LAG traversal are stroke, hole, arc, crossing point, and endpoint. It is called a "parameter-space" approach.

The parameterization of strokes is described as follows. Given two endpoints, a stroke can be represented as a 4-vector $< x, y, r, i >$, where $< x, y >$ represents the location of the stroke's center, and the complex number $< r, i >$ represents its length and orientation in a special way. The stroke length is normalized into the range [0, 0.5] and the angle between the stroke and the x-axis is doubled to give the angle of orientation $\tan^{-1}(i/r)$. Fig 1.7 shows the parameterization of the strokes of a "P."

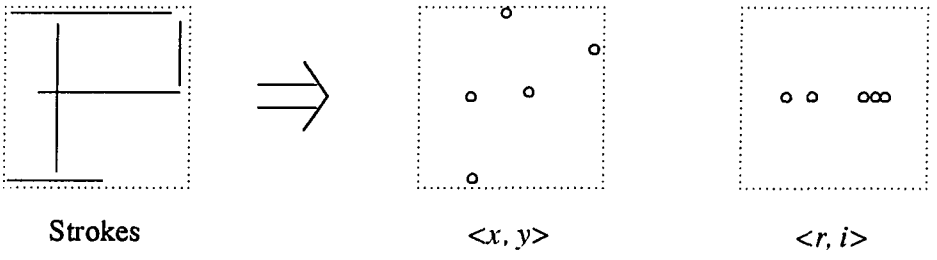


Figure 1.7 Parameterization of the strokes of a "P."

Template matching is robust in the global matching, but is sensitive to noise and character variation. Therefore, for multifont character recognition, Template matching needs to include all the fonts of characters in its model base. This makes the searching problem more complicated. Two-dimensional moments are invariant to translation, orientation, and scaling change. However, it is not invariant to font style change. In addition, to compute moments is time consuming. FD's are very useful for global feature extraction and powerful for distortion. However, they are not so good for local feature

extraction since they are insensitive to spurs on the boundary; e.g., they may not distinguish O and Q. Another problem is that the selection of starting point of a boundary tracing is crucial for the invariant properties of FD's.

From the nature of the characters, structural analysis seems closer to the perception of the human being. Especially for multifont and handwritten character recognition, the main advantage of structural analysis is its high flexibility to distortion and style variations. However, sometimes the extent of tolerance may not easy to be controlled, thus different characters may have similar structures. For example, stream following analysis is very simple and is very strong against variations of shape. Perotto called the description a morphotopological description. However, because of the simplicity, considerably different patterns may have the same descriptions, such as "U", "V", and "-", and can not be distinguished. We may need another raster scanning horizontally, or some other attributes to separate them. Furthermore, the extraction of the features for structural description is very difficult and is still a research topic.

1.2.4 Segmentation for Merged Character

Merged character is still a major problem in OCR which may occurs due to low resolution or smearing ink of a printing device when producing a document with small intra-character spacing. Especially when the text is printed in proportional spacing style, the characters within a word are close to each other. Even though the characters in the physical document are not touched, the character images are frequently merged together with such a small spacing. Kahan *et al.* [41] characterize the most frequent types of joins into two categories: serif joins and double-o joins. As shown in Fig. 1.8, serif joins are usually near the bottom or the top of the characters and are due to overlap of the extreme points of a pair of serifs. Double-o joins usually occur near the middle of the characters and are due to osculating convex contours. Joins which are not belong to these two types may also occur such as shown in Fig. 1.8(c).

(a) (b) (c)

Figure 1.8 (a) Serif join, (b) double-o join, and (c) other join.

General solution for merged character applies a "hypothesis and test" procedures, which mainly consists of three stages. First, the possible breaking positions are sought. Second, the candidates are recursively segmented and recognized. The possible candidates may be ranked by their reliabilities. At last, a linguistic context confirms the hypotheses. It is fundamental to locate the breaking positions properly. Vertical projection profile, skeletal features, and contour features have been used for touching character segmentation.

It is observed that the number of pixels at linking positions are relatively low in the vertical projection profile, denote as V, as shown in Fig. 1.9(b). However, it seems not sufficient since the horizontal strokes of a character also have a low number in V. A major difference between joins and thin horizontal lines is the quick change in V in the neighborhood of a join. Bsed on the sharp minimum of V, Kahan *et al.* [41] looks for a maximum in the second difference. That is

$$V_i' = (V_{i+1} - V_i) - (V_i - V_{i-1}) = V_{i+1} + V_{i-1} - 2V_i \tag{1.18}$$

Fig. 1.9(c) illustrates the functions V' that are computed for breaking the merged characters.

Tsujimoto and Asada [85] propose a break cost, which is a variation of vertical projection profile, as a metric to nominate break positions. Break cost (BC) evaluates the

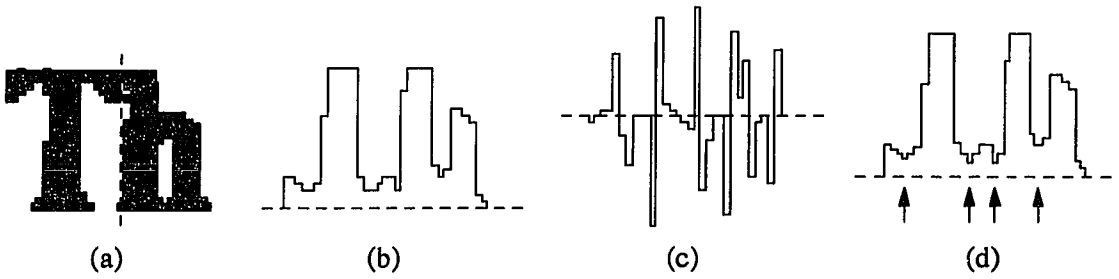


Figure 1.9 (a) Examples of merged characters (b) vertical projection profile (c) difference functions of projection profiles [41] (d) break cost function [85].

degree of contact for each pair of adjacent columns. It is calculated by accumulating the number of black pixels vertically in the image obtained after an AND operation between neighboring columns. Mathematically, $BC_j = \sum (x_{ij} \cap x_{i,j+1})$. Fig. 1.9(d) shows an example of break cost function, where the arrows indicate the candidates of break positions.

Another splitting algorithm uses skeletal features, such as end points, junctions, and stroke segments, as the primary splitting features. Mitchell and Gillies [54] proposed a splitter which is designed for splitting handwritten digit numerals and is restricted for two merged digits only. The splits roughly sort the stroke segments of the form into two groups representing the left and right digits. However, there are situations where the desired sorting results in stroke segments that are in both or neither groups. For example, the vertical stroke segment created by two touching zeros is really a part of both digits, but a stroke segment that bridges two digits is not a part of either digit. Then, some enumeration heuristics based on handwriting properties are used to limit the split number. For example,

- Horizontal strokes with three-way junctions at both ends are assumed to belong to neither split group, i.e., a bridge stroke.

After a split is generated, it is tested to see if the resulting form can possibly be a digit. Several pruning heuristics based on digit morphology properties are employed.

Finally, from the splits that survive the digit tests, the split which produces the largest digit stroke centroid separation is selected as the best candidate.

Contour analysis has also been used to detected the break positions. Fujisawa *et al.* [25] proposed an approach which firstly finds the contour of each connected component. The contour is separated into the upper and lower part as shown in Fig. 1.10(b) and (c), where separation between upper and lower contours is done at the leftmost and rightmost points, giving two lists of contour points:

$$C_U = Lb(x_{up}, y_{up}) \mid p = 1, \ldots, URb \tag{1.19}$$

$$C_L = Lb(x_{lq}, y_{lq}) \mid q = 1, \ldots, LRb \tag{1.20}$$

The two lists of upper/lower contour points C_U and C_L are then converted to single-valued functions $H_U(x)$ and $H_L(x)$, which represent the lowest and highest y-coordinates of the contour points at x respectively. An approximated measure of vertical width $H(x)$ is given as follows:

$$H(x) = |H_U(x) - H_L(x)| \tag{1.21}$$

The candidate locations is obtained by comparing the vertical width $H(x)$ with a threshold h_t. As shown in Fig. 1.10(d), this comparison is made in interval $[X_1, X_2]$ to limit the search range. Candidate locations are where the curve $H(x)$ and line h_t cross each other.

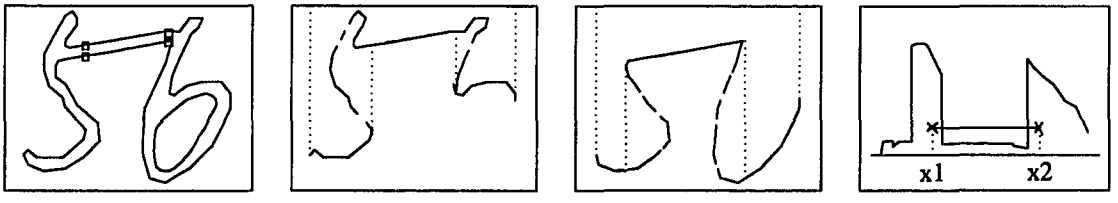


Figure 1.10 Break location candidates by contour analysis. (a) A contour of touching character and candidates for touching points. (b) Upper contour C_U and function $H_U(x)$. (c) Lower contour C_L and function $H_L(x)$. (d) Measure of vertical width $H(x)$.

Traditional techniques for splitting merged characters are simply based upon the "hypothesis and test" procedures. These techniques are weak by two reasons. First, in the hypothesis phase, existing algorithms for detecting break candidates are suitable for certain types of joins and not robust. Second, in the test phase, there may exist multiple choices which survive the segmentation and recognition, and the text context confirmation test. Usually, the one with highest reliability is then selected. However, the reliability ranking is still questionable.

1.3 Organization of the Dissertation

The remainder of this dissertation is organized as follows: Chapter 2 presents a block segmentation and classification technique. Chapter 3 presents a fuzzy typographical analysis for character preclassification. Chapter 4 proposes a fuzzy model for unsupervised character classification, which incorporates a fuzzy pattern matching technique. Chapter 5 presents a topography-based approach of skeletonization for fuzzy degraded character images. Chapter 6 summarizes the contributions of our research and briefly discusses the future research directions.

CHAPTER 2

PREPROCESSING

To retrieve information from a mix-mode document, the first step is to distinguish the text from non-text. This procedure is called preprocessing. Preprocessing, when applied to document analysis, associates the decomposition and preparation of the document into proper subordinate areas, or blocks, which are ready for subsequent processes. It is known as area segmentation, or block segmentation. In this chapter, we propose a two-step block segmentation and a rule-based block classification which is relatively independent of the change of text font style and size.

2.1 Block Segmentation

The block segmentation decomposes a document image into rectangular blocks each of which includes one of text, horizontal or vertical lines, graphics, or pictures. Several techniques for block segmentation have been developed [21, 23, 91]. A constrained run-length smoothing algorithm [91] is used to segment a document into areas of text, lines, and pictures. The graphics, except solid horizontal or vertical lines, is categorized into the same class as pictures. A rule-based block segmentation [21] consists of smearing the document image via the run-length smoothing algorithm, calculating the locations and statistical properties of connected components, and filtering out the image. The Bley algorithm [6] decomposes a connected component into subcomponents which makes it more complex in the recognition process and which is sensitive to text font and size variations. A robust algorithm for block segmentation [23] which uses the Hough transformation to group connected components together into logical character strings to be discriminated from the graphics, is relatively independent of changes in text's font, size, and string orientation.

The office documents are digitized and thresholded into black-and-white (or binary) images through a scanner or a facsimile, where white pixels are represented as 0's and black pixels as 1's. A run-length smoothing algorithm (RLSA) [21,95] can be applied to scan the binary sequences row-by-row or column-by-column. A set of adjacent 0's or 1's is called a *run*. The algorithm converts a binary input sequence f into an output sequence g by using the simple rule: the 0's in f are changed to 1's in g if the run length of 0's, i.e. the number of adjacent 0's, is less than or equal to a predefined threshold value C. For example, if the threshold $C = 5$ is chosen and the input is

$$f\colon\ 001100000010011100011000000100,$$

then the output will be

$$g\colon\ 001100000011111111111000000100.$$

The smoothing rule merges two runs of 1's together if the interval between them is not sufficiently spaced. Since the spacings of document elements are different horizontally and vertically, different values of C are used for processing the sequences row-by-row and column-by-column processing. With appropriate selection of C's, the merged runs will construct various blocks of a common data class. The original RLSA [95] consists of the following four steps:

(1) A horizontal smoothing is applied to the original document image by a predefined threshold C_h.

(2) A vertical smoothing is applied to the original document image by a predefined threshold C_v.

(3) The smoothing results of Step 1 and 2 are combined by a logical AND operation.

(4) An additional horizontal smoothing is applied to the output of Step 3 by a relatively small threshold C_a.

The selections of C_h, C_v and C_a in RLSA affect the resulting images. Too small C_h simply links the characters within a word but can not bridge the interword space. Too

large C_h, however, may cause text to be joined with non-text regions, or may cause text multi-columns concatenation. Similar effect occurs for the C_v selection. The relatively small threshold C_a of Step 4 is used to fill in the horizontal gaps between words in a text line. The original RLSA algorithm requires scanning the whole image four times. An improvement is accomplished to reduce the number of scannings from four to only two.

Let A and B denote the outputs of Step 1 and 2 respectively in the RLSA. Step 3 is to perform $A \cap B$, which is equivalent to $A - (\neg B)$. Therefore, the four steps of RLSA can be modified as follows: Step 1 and 4 remain unchanged, but Step 2 and 3 are merged into one step which is:

- If the run length of 0's in the vertical direction of the original image is greater than C_v, then reset the corresponding pixels in A to be 0's and leave A unchanged otherwise.

The three-step algorithm can be revised by carrying out the vertical smoothing prior to horizontal smoothing.

(1) A vertical smoothing is applied on the original document image by a predefined threshold C_v.

(2) If the run length of 0's in the horizontal direction of the original image is greater than C_h, then reset the corresponding pixels in the output of Step 1 to be 0's and leave those unchanged otherwise.

(3) An additional horizontal smoothing is applied to the output of Step 2 by a relatively small threshold C_a.

The three-step algorithm can be further improved by combining Step 2 and 3 of horizontal smoothing into one step. Suppose that C_a could be greater than, smaller than, or equal to C_h. The effect of C_a compared with C_h on the above three-step algorithm is analyzed below.

Firstly, for $C_h = C_a = C$ used in the Fisher's segmentation algorithm [21], if the number of horizontally consecutive 0's of the original image is greater than C, then the

corresponding pixels must be set to be 0's in Step 2 and will not be set to be 1's in Step 3. If the number is less than or equal to C, then the corresponding pixels remain unchanged in Step 2 and will change to 1's in Step 3. It can be observed that Step 3 is equivalent to checking on the original image while changes take place on the output of Step 1. Since Step 3 is independent of Step 2, Step 2 and 3 can be merged together as follows:

- If the run length of 0's in the horizontal direction of the original image is greater than C, then set the corresponding pixels in the output of Step 1 to 0's; otherwise set them to 1's.

Note that the final results determined by checking on the original image horizontally are independent of Step 1 no matter what values are changed vertically in Step 1. Therefore, this is equivalent to making changes on the original image, so that its result is the same function as Step 1 of the original RLSA. As a result, the one-step algorithm (i.e. Steps 2-4 of RLSA are redundant) can replace the four-step RLSA algorithm when $C_a = C_h$.

Secondly, for $C_h < C_a$, if the number of horizontally consecutive 0's of the original image is between C_h and C_a, then the corresponding pixels which remain 0's in Step 2 will be converted to 1's in Step 3. Therefore, Step 2 is redundant and can be removed.

Thirdly, when $C_a < C_h$ (used in the RLSA algorithm [95]), if the number of horizontally consecutive 0's of the original image is between C_h and C_a, then the corresponding pixels in the output of Step 1 are checked whether they are subdivided into smaller segments, and then the spacings between segments are compared with C_a to determine whether 0's or 1's are to be assigned. Thus, the final improved algorithm consists of only two steps:

(1) A vertical smoothing is applied to the original document image by a predefined threshold C_v.

(2) If the run length of 0's in the horizontal direction of the original image (denoted by RL) is greater than C_h, then the corresponding pixels in the output of Step 1 are reset to 0's. If $RL \leq C_a$, then the corresponding pixels in the output of Step 1 are switched

to 1's. If $C_a < RL \leq C_h$ and if the run length of horizontally consecutive 0's in the output of Step 1 is less than or equal to C_a, then the corresponding pixels in the output of Step 1 are set to 1's.

Another commonly used block segmentation algorithm, known as RXYC (or recursive X-Y cuts), is based on the projection profile. At each step of the recursive process, the projection profile is computed along both horizontal and vertical directions. Then subdivision along the two directions is accomplished by making cuts corresponding to deep valleys, with width larger than a predetermined threshold, in the projection profile.

RLSA is better than RXYC for getting small blocks where each block includes just a text line in the text area, . This is because the RLSA smearing process is done within each text line. On the other hand, the RXYC has to be done several times to make each text line in a block by setting threshold smaller than the inter-line space. If large blocks such as paragraphs are needed, then RXYC is better than RLSA. A merging algorithm has to follow the application of RLSA for merging blocks corresponding to single text lines to form blocks corresponding to paragraphs.

In comparison of the running time, the two-step RLSA scans the image twice, while the number of scan for RXYC depends on the structure of the document. Another disadvantage of using RXYC is that it is more sensitive to skew than using RLSA. Furthermore, the skewed angle is easier to detected by the smoothed image after applying RLSA.

Another advantage of RLSA is its flexibility, when applying to some irregular document such as a graphics spanning two columns in a three-column document. Fig. 2.1 shows an example of such special cases which can only be segmented by RLSA. If RXYC is applied to this case, a deadlock occurs. Also the layout structure of this case is a graph instead of a tree.

A shortcoming of the RLSA method is that the resulting blocks may not be rectangular, and an component generation algorithm for finding the bounding rectangle of each connected component has to be used after applying the RLSA.

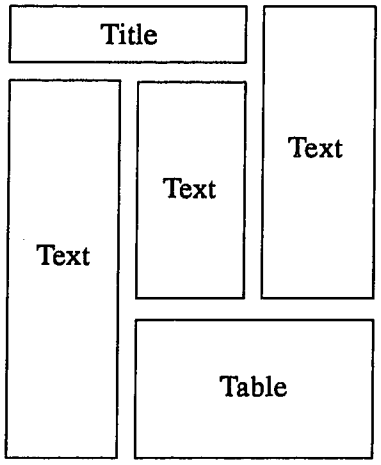


Figure 2.1 An example of document with non-hierarchical layout structure.

2.2 Block Classification

In the preceding section, the mixed-mode (mixture of text, graphics and pictures) document is segmented into blocks, each of which contains the single-mode content. This section presents the block classification algorithm to classify the blocks into one of the text, horizontal or vertical line, graphics and picture classes. Most existing block classification techniques are based on the discrimination of statistical local or global features. The commonly used features, such as the height of a block and the aspect ratio, are elementary for extracting the mostly popular text blocks. However, they are not sufficient in classifying the mixed-mode document blocks. Therefore, more complex features are needed to achieve higher reliability.

Since the text/nontext blocks tend to cluster in space with respect to some features, a threshold or a discriminant function is selected for separation. A two-dimensional plane consisting of mean value of the block's height versus mean run length of the black pixels is established to classify document blocks into text, nontext, horizontal line, and vertical line [95]. A rule-based classification uses the features such as height, aspect ratio, density, perimeter, and perimeter/width ratio [21]. A newspaper classification method creates

the black-white pair run-length matrix and black-white-black combination run-length matrix to derive three features: short run's emphasis, long run's emphasis, and extra long run's emphasis for clustering [90]. The distribution of the features used is dependent on the character's font and size and the image resolution. An inappropriately chosen threshold can lead into misclassification.

This section presents a robust block classification algorithm based on clustering rules. Let the origin of the document image be located at the upper-left corner. Each block is measured in terms of the following:

- Minimum x- and y-coordinates and the width and height of a block $(x_{\min}, y_{\min}, \Delta x, \Delta y)$.
- Number of black pixels corresponding to the block of the original image (N).
- Number of horizontal transitions of white-to-black pixels corresponding to the block of the original image (TH).
- Number of vertical transitions of white-to-black pixels corresponding to the block of the original image (TV).
- Number of columns in which any black pixel exists corresponding to the block of the original image (δx).

Since in most cases the projection profile of a block onto y-axis contains black pixels in each row, it is redundant since it measures the number of rows in which black pixels exist. The following features used in block classification can be easily calculated:

- Height of each block, $H = \Delta y$.
- Ratio of width to height (or aspect ratio), $R = \dfrac{\Delta x}{\Delta y}$.
- Density of black pixels in a block, $D = \dfrac{N}{\Delta x \Delta y}$.

- Horizontal transitions of white-to-black pixels per unit width, $TH_x = \frac{TH}{\delta x}$.
- Vertical transitions of white-to-black pixels per unit width, $TV_x = \frac{TV}{\delta x}$.
- Horizontal transitions of white-to-black pixels per unit height, $TH_y = \frac{TH}{\Delta y}$.

Note that a text block constructed from the segmentation algorithm contains a text line only. Since typical office documents often contain the text with mostly one popular size and font, the mean value of all the block-heights in a document is approximately equal to that most popular text's block-height. The ratio of width to height R can be used to detect the blocks, such as horizontal or vertical lines. Both the mean horizontal transition TH_x and the mean vertical transition TV_x play important roles in text and nontext discrimination. Both features are independent of variant text's fonts and sizes as long as the ratio of the width to height of a character is not varied significantly.

Let $TH_x^{\max}$ and $TH_x^{\min}$ denote the maximum and minimum TH_x's values of all characters, respectively. (and similar notations $TV_x^{\max}$ and $TV_x^{\min}$ are used.) Intuitively,

$$TH_x^{\min} \le TH_x \le TH_x^{\max}, \text{ and} \tag{2.1}$$

$$TV_x^{\min} \le TV_x \le TV_x^{\max} \tag{2.2}$$

Let H_m be the average height of the most popular blocks, which dominate the text of a document. The rule-based block segmentation algorithm is described as follows:

- Rule 1: if $c_1 H_m < H < c_2 H_m$, the block belongs to text.
- Rule 2: if $H < c_1 H_m$ and $c_{h1} < TH_x < c_{h2}$, the block belongs to text.
- Rule 3: if $TH_x < c_{h3}$, $R > c_R$, and $c_3 < TV_x < c_4$, the block is a horizontal line.
- Rule 4: if $TH_x > \frac{1}{c_{h3}}$, $R < 1/c_R$, and $c_3 < TH_y < c_4$, the block is a vertical line.
- Rule 5: if $H > c_2 H_m$, $c_{h1} < TH_x < c_{h2}$, and $c_{v1} < TV_x < c_{v2}$, the block belongs to text.

- Rule 6: if $D < c_5$, the block belongs to graphics.
- Rule 7: otherwise, the block belongs to a picture.

Rule 1 is used to extract most of the text lines. Rule 2 is used to capture the text lines with smaller sizes, such as footnotes or remarks. Rules 3 and 4 determine the horizontal and vertical lines, respectively. Rule 5 discriminates the text with larger sizes, such as titles and headings. The remaining blocks are therefore classified as nontextual. Rule 6 is used to distinguish the graphics and pictures by analyzing the block density. The graphics which contains a combination of line drawings and characters has lower density than the picture in most cases.

2.3 Parameters Adaptation

For our experiment, 100 documents with character size varying from 6 to 15 points were used. For each document, the mean values of height H_m and the standard deviation sd are derived from blocks of the most-popular height. The ratios of sd_H/H_m are distributed within the range of 0.027 and 0.044 with an average 0.034. For reliability, the tolerance of text height is selected to be six times of the average ratio, i.e. 0.2. Therefore, $c_1 = 1 - 0.2 = 0.8$ and $c_2 = 1 + 0.2 = 1.2$. This allows most of the text blocks to be extracted except those with significantly different sizes (e.g., title of text).

A set of text blocks with character sizes varying from 6 to 36 points mixed with four commonly used fonts – Roman, Italic, Boldface, and Courier were collected. Results of the mean values of TH_x and TV_x for different fonts and sizes are shown in Fig. 2.2. For all fonts, the mean value is slightly decreased as the character size decreases because the gaps between objects are relatively small and could be filled up easily in printing or scanning. The mean values of Roman, Italic, and Boldface fonts are almost the same. The mean value of TH_x of Courier font is slightly lower than the others because other fonts are printed in a proportional style, while Courier font is printed in a fixed pitch so that it consists of less vertical-like strokes within a unit length of text. On the contrary, the mean

value of TV_x of Courier font is slightly higher than the other fonts because horizontal serifs of Courier font are longer. The standard deviation of TH_x and TV_x for different fonts and sizes is less than 0.1. Therefore, the mean white-to-black transitions are reliable for text extraction with variant fonts and sizes. As a result with tolerance consideration, $c_{h1} = 1.2$, $c_{h2} = 3.0$, $c_{v1} = 1.2$, and $c_{v2} = 2.6$ are selected.

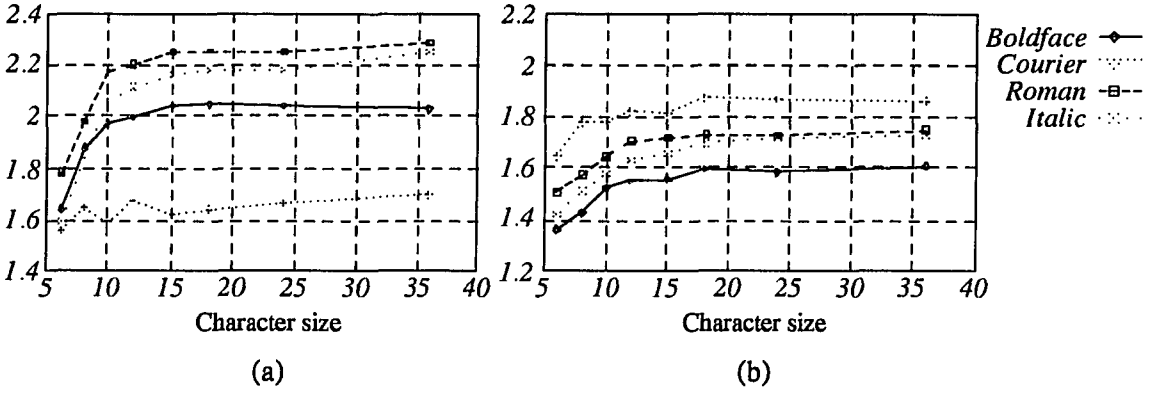


Figure 2.2 Mean values of (a) TH_x and (b) TV_x for different fonts and sizes.

Different resolutions (dots per inch) of a scanner were also experimented. As shown in Fig. 2.3, the mean values of TH_x and TV_x are slightly changed. The standard deviation is lower than 0.1, so that the mean transitions are reliable with respect to the scanning resolution.

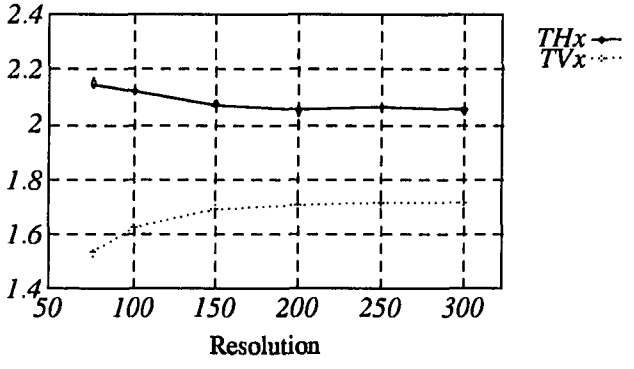


Figure 2.3 Mean values of TH_x and TV_x for different resolutions.

Results of TH_x and TV_x for blocks of text, vertical lines, graphics, and pictures are shown in Fig. 2.4, where a great deal of graphics and pictures are out of the display range using the scale. The big cluster centered at (1.82, 1.60) represents text blocks, and the cluster centered at (0.12, 1.00) corresponds to vertical lines. Note that for vertical lines, TV_x is always very close to 1, and the standard deviation is 0.0004, that means TV_x is very stable for vertical lines extraction. Similarly, for horizontal lines, TH_y is very close to 1, and is stable for horizontal lines extraction. Therefore, $c_3 = 0.95$ and $c_4 = 1.05$ are selected. Disregard the line width, the aspect ratio of a horizontal line R approximates $\cot\theta$, where θ denotes the tilt angle of the scanned document. Therefore, c_R is set to 5, that allows the document to be tilted up to 11.3 degrees. For a horizontal line, TH_x is within the range of $\pm\tan\theta$. Therefore, c_{h3} is selected as 0.2.

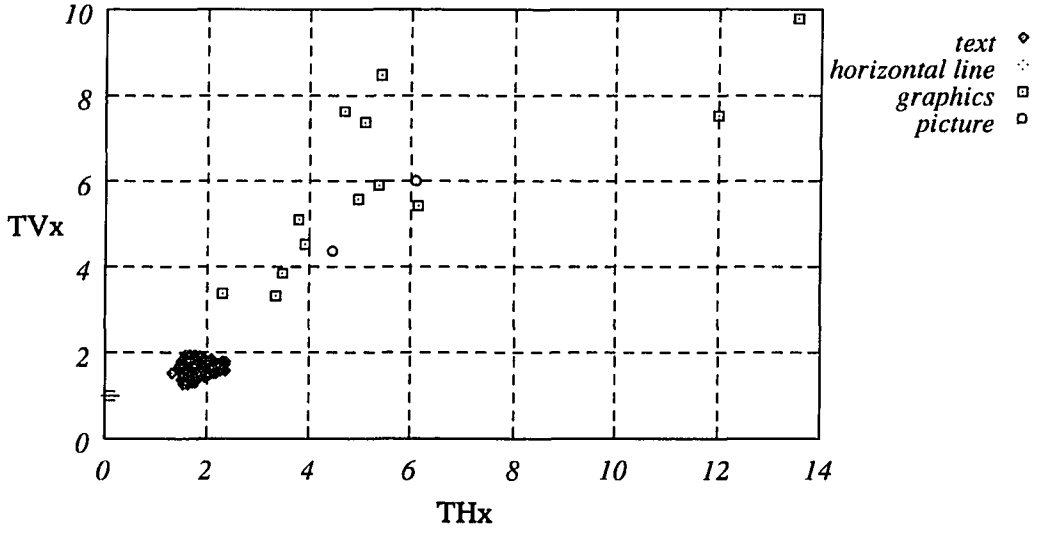


Figure 2.4 The projected $TH_x - TV_x$ plane.

Graphics and pictures are sparsely distributed on the $TH_x - TV_x$ plane, therefore it is not suited to use the mean transitions for discrimination. In principle, graphics composed of lines has lower density than pictures. From experiments, the mean value of density for graphics is 0.061 and the standard deviation is 0.033, while that for pictures is 0.853 and

the standard deviation is 0.108. In practice, the value in the range of [0.15, 0.25] is sufficient to separate graphics and pictures. Therefore, $c_5 = 0.2$ is selected.

2.4 Experimental Results

Our document block segmentation and classification algorithms were implemented on a SUN SPARC workstation under UNIX operating system. Document images are captured by a DEST PC 3000 scanner at the resolution of 300 dots per inch. We have randomly selected 100 documents which are the mixture of text, graphics, and pictures. Our algorithms perform the block segmentation and classification successfully. The following shows an example of the document used and results.

Fig. A.1a in Appendix A shows a document image that contains text with different fonts and sizes, a flow-chart, a picture, and two horizontal lines. Figs. A.1b and A.1c illustrate the resulting images after steps 1 and 2 of our improved two-step smoothing algorithm. Theoretically, we need only 2 steps and therefore save almost 50% computational time. Our step one is identical to the step two of the original algorithm. However, our step two is more complicated, and therefore takes about 10%-15% overhead. Generally, our algorithm takes about 60%-65% of the computational time of the original algorithm. For example, the document shown in Fig. A.1a takes 3.7432 seconds to perform the block segmentation by original algorithm, and takes 2.3682 seconds by our algrithm. Therefore, it save 37% computational time in this example. The result of our block classification applied on Fig. A.1a is shown in Table A.1, where classes t, p, g, and h denote text, picture, graphics, and horizontal line, respectively. The result of applying Wong et al.'s algorithm [95] is shown in Table A.2 for comparison, where classes t, h, n denote text, horizontal line, and nontext, respectively. Note that in Table A.2, block 38 which should be a horizontal line is misclassified as text.

Figs. A.2 and A.3 show the projected TH_x -H plane and TV_x -H plane of our algorithm, and Fig. A.4 shows the projected R-H plane of Wong et al.'s algorithm. Our

algorithm has the following three advantages over their algorithm. First, the features of TH_x and TV_x are more suitable in classifying text and non-text blocks than the feature of R. It is observed that TH_x and TV_x are more convergent than R for text blocks. Furthermore, the range of R of graphics blocks is mixed with that of text blocks. However, this effect does not occur in TH_x and TV_x.

Second, our method is better in classifying solid lines than Wong et al.'s algorithm [95]. A thin line could be misclassified as text due to its spur by using their algorithm. From observation, TV_x of a horizontal line is convergent to 1.0, and is also for TH_y of a vertical line. Thirdly, TH_x and TV_x are independent of character sizes and fonts, while R is not. When the character size becomes larger, for example, twice, R will be as twice as the original one. Moreover, if a certain font of a character has the thickness twice, R tends to increase twice again. In many documents, the headline or the title has greater size and thicker font. These will double the influence of the value R. Wong's algorithm restricts the text size up to three times of the most popular text blocks; this reduces its generality. Since our features are independent of the character's font and size, this limitation is conquered.

CHAPTER 3

FUZZY TYPOGRAPHICAL ANALYSIS FOR CHARACTER PRECLASSIFICATION

In this chapter, we investigate the problems of character isolation, and present a fuzzy logic approach to efficiently preclassify characters according to the typographical structures of textual blocks.

3.1 Introduction

The overall procedures of our document processing system is shown in Fig. 3.1. The office documents are digitized and preprocessed to extract the textual blocks as described in chapter 2. First, the individual character units are constructed based on the connected components and their relations. The typographical categorization divides the characters into seven categories based on their typographical structures such as upper zone, middle zone, and lower zone. The unsupervised character classification adopts the fuzzy matching technique to further classify them into a limited set of distinct fuzzy prototypes which will be presented in Chapter 4. The optical character recognition is employed to recognize the set of fuzzy prototypes. Finally, the postprocessing intends to correct the errors by means of dictionary checking or semantics understanding.

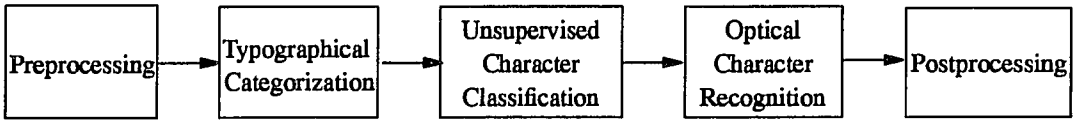


Figure 3.1 Overall procedure of our document processing system.

As illustrated in Fig. 3.2, a text line can be regarded as being composed of three stripes: the upper, middle, and lower zones. they are delimited by the top line, the upper baseline, the baseline, and the underline. The height of middle zone, being the major part of the text line, is about twice as high as the other two zones and can be further split by a mid-line.

Figure 3.2 Typographical structure of a text line.

Typographical analysis can detect the particular word structures such as subscripts or superscripts. On the other hand, some characters in the upper and lower cases with the same structures, such as "C" and "c", "O" and "o", can be distinguished. Explicitly, the baseline appears in a text line. The upper baseline may not present in the case of a short text composed of ascenders only, and the top line and the underline may not exist if only centered characters appear. Therefore, it is essential to locate the baseline. The baseline can also be used for document skew normalization and for determining interline spacing to be more computationally efficient than the traditional Hough and Fourier transform approaches [32, 57].

Luca and Gisotti [50] used the typographical analysis in a word structure for character classification. However, if a word is too short containing insufficient character elements, the structure analysis is quite unstable. In addition, the tolerance for variations in the reference lines was not incorporated. Our baseline detection algorithm based on a line of text is more reliable and efficient than the one based on a single word [50]. The remaining virtual reference lines are extracted by a clustering technique [30]. To allow the unpredictable noise and deformation, the tolerance analysis is included. To ensure the robustness and flexibility, a fuzzy-logic approach [42, 61] is used to assign a membership

to each typographical category for ambiguous classes. A linear mapping function is adopted and its boundary conditions are derived to preserve the continuity.

The chapter is organized as follows. Section 2 discusses the problem of character isolation. Section 3 describes the character typographical structure. Section 4 presents the baseline detection algorithm. Section 5 analyzes the tolerance. Section 6 describes the fuzzy typographical categorization. Experimental results and discussion are given in section 7.

3.2 Character Isolation

Character isolation is used to extract individual character components from the segmented text lines. First, the connected components are detected. Then, the characters with multiple components are constructed by grouping the related components, and words are built up by the intercharacter space checking. A projection histogram on x-axis over this spacing contains two significant peaks: one for the interword spacing and the other for the intraword spacing.

The first step for building characters from a textual block image is to extract each connected components. This can be achieved by a connected component growing algorithm which is described as follows:

(1) The textual block image is scanned from top to bottom and left to right until an unmarked object pixel, which is considered as the starting pixel, is encountered. The scanning ordering is consistent with the writing habit of English language.

(2) The textual block image is considered as a graph with the pixels representing the nodes. A link is connected if two foreground pixels are 8-neighboring. The depth-first search (DFS) or breadth-first search can be applied from the starting pixel, and the pixels are marked when they are traversed.

(3) Continue the scanning from the starting pixel until an unmarked object pixel, which is considered as the new starting pixel, is encountered and go to step (2). The procedures are repeated until every object pixel is marked.

When the growing algorithm is performed, the x- and y-coordinates of the top-left and bottom-right corners of the smallest enclosed rectangle of the component and the number of foreground pixels of the component are recorded. Two issues of the relationship of a character unit and a connected component are observed:

(1) Multiple connected components may construct a single character. A character may be broken into several components due to bad printing quality. Or, by nature, a character is made up of several constituents. Examples of these characters in English are character "i" which is composed of a base stroke and a dot, and several punctuation symbols such as "?" and ";". Some special notations or diacritic characters even contain three components. For example, the percentage symbol "%" and the second derivative $\ddot{x}$.

(2) One connected component represents multiple characters. This may occur by intention such that the characters "f" and "i" are set up in one component as "fi", or by accident due to the low resolution of a printer or a scanner.

Two approaches can be used to solve the first problem. The first is to group the components into one single image, which is treated as an entity for the subsequent processes. While the second treats the components independently. Each of them is passed to the character classifier and then is combined to one character according to some specified rules. The advantage of the first approach is that the recognition phase is simpler than that of the second approach. Therefore, The components must be grouped together as early as they could. The multiple components, which construct a single character by definition, normally have the characteristics of overlapping in vertical projection. Based on this assumption, the projections of components which overlap in a certain degree will be considered as a single unit. In our experiments, two components are grouped together if

either one overlapped with the other one more than 50%. However, the broken characters may happen arbitrarily, and we may need to apply the second approach. Alternatively, a more complicated method is needed to group those pieces into a character.

For the second issue, since the ligatures are intentionally typed as an entity, they should be treated as special character objects, and no attempt should be made to separate them into their constituents. Hence, the classifier should include ligatures in the set of classes. Merged characters as well as broken characters have to be further processed in the postprocessing.

Note that the character isolation is affected by the document printing style. If the text is typed with a fixed pitch such as the typewritten style, there is no ligature at all. Furthermore, the broken character grouping and the merged character splitting become simpler. On the contrary, when the text is printed with the proportional spacing, character isolation becomes more difficult. Therefore, prior to further processing the spacing style is detected. If the fixed spacing is used, the character isolation is straight-forward.

3.3 Character Typographical Structure Analysis

According to the stripes the characters intercept, we classify characters into primary and secondary classes, which are in turn divided into four and three categories as follows:

(1) Primary class: This class consists of larger symbols including most of the Latin alphabets.

(a) Ascender: Characters intercept fully the upper and middle zones, such as A, B, and d.

(b) Descender: Characters intercept fully the lower and middle zones, such as g, q, y, and p.

(c) Center: Characters lie fully in the middle zone only, such as a, c, e, m, and x.

(d) Full-range: Characters span all the three zones, such as j, (,), {, and f.

(2) Secondary class:

(a) Subscript: Symbols locate around the baseline, such as ".", ",", "_", and the shifted characters as subscripts.

(b) Superscript: Symbols lie around the upper baseline, such as ", ~, ^, and the shifted characters as superscripts.

(c) Internal: Characters locate around the mid-line and only partially intercept the middle zone, such as "-".

Typographical structure analysis aims at categorizing each character unit into one of the seven categories. Mostly, the baseline appears explicitly in each text line. The upper baseline may not present in the case of a short text composed of ascenders only, and the top line and underline may not exist if only centered characters appear. Therefore, it is essential to locate the baseline. However, the typographical structure analysis is sensitive to skew even in a very little angle. Despite the image was skew-normalized, the text line may not be exactly horizontal due to some error while printing or scanning. Next, we will present an efficient algorithm to precisely locate the baseline.

3.4 Baseline Detection

In order to correctly analyze the typographical structure, the virtual baseline must be detected. Fortunately, the disalignment is very small and the line is nearly horizontal. An efficient approach to locating the precise baseline is given as follows:

(1) Let a text line (denoted as **T**) of n character units (ch_i) appear in the text line from left to right sequentially. That is, $\mathbf{T} = \{ch_1, ch_2, \ldots, ch_n\}$. Let $p_i = (x_i, y_i)$ denote the x- and y-coordinates of the bottom-center point p_i at the bounding box of the i^{th} character, the set of those bottom-center points is $\mathbf{P} = \{p_1, p_2, \ldots, p_n\}$. Because most of the characters are of ascenders or centered, they are aligned on the baseline. Therefore, **P** provides the basis to find out the virtual baseline.

(2) Let $\mathbf{Y}' = \{y'_1, y'_2, \ldots, y'_{n-1}\}$ be the set of all the slopes, where the slope of the line segment $\overline{p_i p_{i+1}}$ is $y'_i = \dfrac{y_{i+1} - y_i}{x_{i+1} - x_i}$. Mostly, the slopes are near zero. That is, the contiguous characters ch_i and ch_{i+1} are frequently aligned on the baseline or underline.

(3) Let $\mathbf{Y}'_{\mathbf{mp}}$ denote the set of the most-popular slope obtained by clustering analysis from the set $\mathbf{Y}'$. That is

$$\mathbf{Y}'_{\mathbf{mp}} = \{y'_i \,|\, |y'_i - y'_{mp}| \le \varepsilon, i = 1, 2, \ldots, n-1\}, \tag{3.1}$$

where y'_{mp} denotes the most-popular slope, and ε denotes the clustering factor. The initial slope approximation of the whole baseline is derived as follows:

$$m_{appr} = \frac{\sum\limits_{y'_i \in Y'_{mp}} y'_i \Delta x_i}{\sum\limits_{y'_i \in Y'_{mp}} \Delta x_i}, \qquad \text{where } \Delta x_i = x_{i+1} - x_i \ . \tag{3.2}$$

(4) A line can be expressed in the form $y = mx + b$, where m is slope and b is the intercept of the line with y-axis. Let $\mathbf{B}$ denote the set of intercepts where the lines pass through the points in $\mathbf{P}$ with the slope m_{appr}. That is

$$\mathbf{B} = \{b_i \,|\, b_i = y_i - m_{appr} x_i, i = 1, 2, \ldots, n\}, \tag{3.3}$$

where b_i denotes the intercept of a line with the slope m_{appr} passing through p_i. Since the baseline slope is approximately equal to m_{appr}, the bottom-center points of the characters which are aligned on the baseline will be collinear and tend to cluster with respect to b. Some other small clusters, which may be sparsely distributed, represent some characters which are aligned in the same orientation as baseline. For example, "g" and "y" are clustered on the underline. Similar to $\mathbf{Y}'_{\mathbf{mp}}$, the set of the most-popular intercepts can be derived as

$$\mathbf{B}_{\mathbf{mp}} = \{b_i \,|\, |b_i - b_{mp}| \le \delta\}. \tag{3.4}$$

(5) $\mathbf{B_{mp}}$ represent all the points located at the baseline $\mathbf{P_{bl}}$, where

$$\mathbf{P_{bl}} = \{p_i \mid b_i \in \mathbf{B_{mp}}\}. \tag{3.5}$$

In order to obtain the precise baseline expressed by $y = m_{bl}x + b_{bl}$, a linear regression is performed on the points in $\mathbf{P_{bl}}$ by using the least-square-error approach [10]. The equations are

$$m_{bl} = \frac{\begin{vmatrix} \sum_{p_i \in \mathbf{P_{bl}}} x_i y_i & \sum_{p_i \in \mathbf{P_{bl}}} x_i \\ \sum_{p_i \in \mathbf{P_{bl}}} y_i & N_{\mathbf{P_{bl}}} \end{vmatrix}}{\begin{vmatrix} \sum_{p_i \in \mathbf{P_{bl}}} x_i^2 & \sum_{p_i \in \mathbf{P_{bl}}} x_i \\ \sum_{p_i \in \mathbf{P_{bl}}} x_i & N_{\mathbf{P_{bl}}} \end{vmatrix}} \tag{3.6}$$

and

$$b_{bl} = \frac{\begin{vmatrix} \sum_{p_i \in \mathbf{P_{bl}}} x_i^2 & \sum_{p_i \in \mathbf{P_{bl}}} x_i y_i \\ \sum_{p_i \in \mathbf{P_{bl}}} x_i & \sum_{p_i \in \mathbf{P_{bl}}} y_i \end{vmatrix}}{\begin{vmatrix} \sum_{p_i \in \mathbf{P_{bl}}} x_i^2 & \sum_{p_i \in \mathbf{P_{bl}}} x_i \\ \sum_{p_i \in \mathbf{P_{bl}}} x_i & N_{\mathbf{P_{bl}}} \end{vmatrix}}, \tag{3.7}$$

where $N_{\mathbf{P_{bl}}} = \sum_{p_i \in \mathbf{P_{bl}}} x_i^0$, denotes the number of the elements in $\mathbf{P_{bl}}$.

3.5 Tolerance Analysis

Typographical analysis is a statistical approach to locate the virtual reference lines in a text line. Due to the unavoidable noise in an image, a variable tolerance must be allowed in the baseline detection algorithm and in the typographical categorization described in the next section. The tolerance is defined as $\tau = 2\delta$ as illustrated in Fig. 3.3, where the shaded areas denote the tolerance of the reference lines. Suppose that the tolerance of

each reference line is equal. Let H denote the height of the text line including the tolerance, and H_l denote the height between baseline and underline. Assume $H_l = 4\delta$. Since the heights of the upper zone and the lower zone are the same and approximately half of the middle zone, $H = 18\delta$ can be obtained.

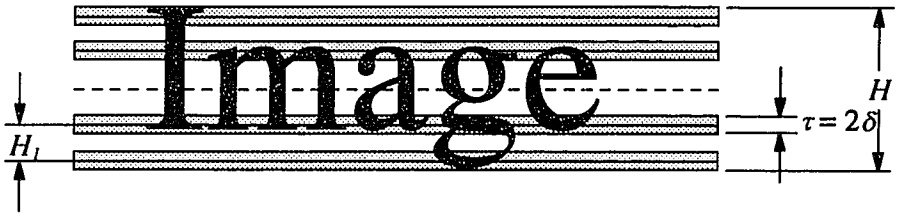


Figure 3.3 Tolerances of the reference lines

Let $\mathbf{Q} = \{q_1, q_2, \ldots, q_n\}$ be the set of points with $q_i = (x_i, y_i)$ representing the upper-center x- and y-coordinates of the character ch_i. The set of intercepts of the lines passing through the points in $\mathbf{Q}$ with y-axis can be defined as:

$$\mathbf{B_Q} = \{b_i \mid b_i = y_i - m_{appr} x_i, \text{ where}(x_i, y_i) \in \mathbf{Q}\}. \tag{3.8}$$

The height of the text line is derived as

$$H = \frac{1}{\sqrt{1 + m_{appr}^2}} (\max\{\omega \mid \omega \in \mathbf{B_P}\} - \min\{\omega \mid \omega \in \mathbf{B_Q}\}). \tag{3.9}$$

When m_{appr} is very small,

$$H \approx \max\{\omega \mid \omega \in \mathbf{B_P}\} - \min\{\omega \mid \omega \in \mathbf{B_Q}\}. \tag{3.10}$$

The tolerance $\delta = \frac{H}{18}$ is then computed and used in Step (4).

For example of an image with the resolution 300 dpi (where 1 dot = 1 pixel), the height of a text line is equal to $\frac{s \times 300}{72}$ pixels, where s denotes a font size, because the point size of 72 is 1 inch high. If the point size is 12 and $H = 50$ pixels, the upper and lower zones are approximately equal to 12 pixels and $\tau = 6$.

Assume that the distance between centers of the adjacent characters is no more than twice of the text height. Let ϕ denote the angle of the virtual baseline with respect to x-axis. If two adjacent characters, say ch_i and ch_{i+1}, are baseline-aligned, the line $\overline{p_i p_{i+1}}$ may exist in the range of the orientations from $\phi + \theta$ to $\phi - \theta$ as shown in Fig. 3.4, where

$$\theta = \tan^{-1} \frac{\tau}{2H} = \tan^{-1} \frac{\delta}{H}. \tag{3.11}$$

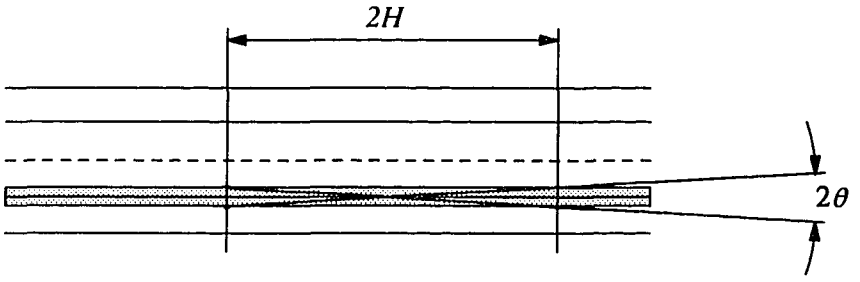


Figure 3.4 Tolerance of the slope of the baseline.

In other words, the slope varies from $m_{\phi+\theta}$ to $m_{\phi-\theta}$, where

$$m_{\phi+\theta} = \tan(\phi + \theta) = \frac{\tan\phi + \tan\theta}{1 - \tan\phi \tan\theta}, \tag{3.12}$$

$$m_{\phi-\theta} = \tan(\phi - \theta) = \frac{\tan\phi - \tan\theta}{1 + \tan\phi \tan\theta}. \tag{3.13}$$

When ϕ and θ are small, the term $\tan\phi \tan\theta$ is negligible. Eqs. (12) and (13) can be simplified as

$$m_{\phi+\theta} \approx \tan\phi + \tan\theta = m + \varepsilon, \tag{3.14}$$

$$m_{\phi-\theta} \approx \tan\phi - \tan\theta = m - \varepsilon, \tag{3.15}$$

where

$$\varepsilon = \tan\theta = \frac{\delta}{H} = \frac{1}{18}. \tag{3.16}$$

3.6 Fuzzy Typographical Categorization

The detection of other reference lines can be easily achieved by deriving from the baseline. First, the upper baseline is detected by projecting in parallel to the baseline the set of upper-center points **Q** of the characters whose sizes are larger than a threshold. The threshold is used to exclude the characters in the secondary class. Since the centered characters have the smallest height in the primary class, the threshold is selected as $\frac{H}{2} - \frac{H}{18} \times 2 = \frac{7}{18} H$ by considering the tolerance. The projected points onto y-axis tend to have two clusters. The most popular cluster corresponds to the upper baseline, while the other cluster corresponds to the top line. These results can be verified if the height of upper zone approximates to a half of the height of middle zone. In other words, $b_{ub} - b_{tl} \approx (b_{bl} - b_{ub}) / 2$.

Similarly, the underline can be detected by projecting to the y-axis the set of bottom-center points of the characters whose sizes are larger than the above threshold and whose bottom-center points are not located at the baseline.

The projected locations of upper-center points would appear only one cluster if a textual block contains only ascenders (e.g., a title with capital letters) or centered characters (e.g., the last line of a paragraph with a few characters). The block is hypothesized as the ascender if it does not exist in a paragraph or the number of characters in the block is reasonably large.

A character is assigned to one of the seven typographical categories based on its location listed in Table 3.1, where y_q and y_p denote y-coordinates of the upper-center and lower-center points of a character, respectively. The tolerance zones of the top line, upper baseline, mid-line, baseline, and underline are denoted as r_1, r_3, r_5, r_7, and r_9, respectively, and the ranges in between are denoted as r_2, r_4, r_6, and r_8 as illustrated in Fig. 3.5. The seven categories are denoted as $\uparrow$ (superscript), $\downarrow$ (subscript), A (ascender), D (descender), C (center), F (full-range), and I (internal). Most of the characters belong to

the deterministic classes, denoted as (y_q,y_p)'s pair such as (1,7), (1,9), (2,7), and (2,9), which are classified as the ascender, full-range, center, and descender, respectively. For other classes containing more than one category, an uncertainty for each class is detected. In Table 3.1, the category enclosed in parentheses represents a weak class.

Table 3.1 The decision table for typographical categorization.

$y_p \setminus y_q$	r_1	r_2	r_3	r_4	r_5	r_6	r_7	r_8	r_9
r_1	↑	-	-	-	-	-	-	-	-
r_2	↑	↑	-	-	-	-	-	-	-
r_3	↑	↑	↑	-	-	-	-	-	-
r_4	↑	↑	↑, (I)	↑, I	-	-	-	-	-
r_5	↑	↑	↑, I	I, (↑)	I	-	-	-	-
r_6	↑, A	↑, A, C	I, C, (↑)	C, I	I, (↓)	I, ↓	-	-	-
r_7	A	A, C	C	C, I, (↓)	I, ↓	↓, (I)	↓	-	-
r_8	A, F	A, C, F, D	C, D	C, D, ↓	↓	↓	↓	↓	-
r_9	F	F, D	D	D, ↓	↓	↓	↓	↓	↓

Figure 3.5 The illustration for the tolerance ranges used in Table 3.1.

Due to unpredictable noise or deformation, a character may be detected in an ambiguous position, for example y_q in r_1 and y_p in r_8, it could be an ascender or a full-ranged character. In this case, the memberships are assigned with the degrees of the character belonging to typographical categories.

Definition 1: The fuzzy typographical categorization of a character α is a list of ordered pairs such that $\alpha = \{(\Omega, \chi(\Omega))\}$, where $\Omega \in \{\uparrow, \downarrow, \text{A, D, C, F, I}\}$, and $\chi(\Omega)$, ranging in [0, 1], represents the grade of membership the character belongs to the category Ω.

Essentially, the memberships are characterized by size and position of characters, which in turn are determined by y_p and y_q. The size is $y_p - y_q$ and the position can be indicated by $y_p + y_q$. Let $y_0, y_1, \ldots, y_9$ be the y-coordinates which delimit $r_1, r_2, \ldots, r_9$ respectively as shown in Fig. 3.5. For simplicity, the tolerance ranges r_i, $i = 1, \ldots, 9$, are normalized and denoted as r_i'. The normalized y-coordinate of y_j, $j = p, q$, in r_i' will be $y_j' = \dfrac{y_j - y_{i-1}}{y_i - y_{i-1}}$ by linear interpolation. For example, the memberships in the class (1,8) are given as:

$$\chi_{(1,8)}(\text{A}) = 1 - y_p' = \frac{y_8 - y_p}{y_8 - y_7}, \qquad \chi_{(1,8)}(\text{F}) = y_p' = \frac{y_p - y_7}{y_8 - y_7}, \tag{3.17}$$

where (1,8) denotes the decision class with $y_q \in r_1$ and $y_p \in r_8$. Note that the membership functions in eq. (3.17) is continuous with $\chi_{(1,7)}(\Omega)$ and $\chi_{(1,9)}(\Omega)$.

Similarly, the membership functions $\chi_{(2,7)}(\Omega)$, $\chi_{(2,9)}(\Omega)$, and $\chi_{(3,8)}(\Omega)$ can be defined as:

$$\chi_{(2,7)}(\text{A}) = 1 - y_q' = \frac{y_2 - y_q}{y_2 - y_1}, \qquad \chi_{(2,7)}(\text{C}) = y_q' = \frac{y_q - y_1}{y_2 - y_1}, \tag{3.18}$$

$$\chi_{(2,9)}(\text{D}) = y_q' = \frac{y_q - y_1}{y_2 - y_1}, \qquad \chi_{(2,9)}(\text{F}) = 1 - y_q' = \frac{y_2 - y_q}{y_2 - y_1}, \tag{3.19}$$

and

$$\chi_{(3,8)}(\text{D}) = y_p' = \frac{y_p - y_7}{y_8 - y_7}, \qquad \chi_{(3,8)}(\text{C}) = 1 - y_p' = \frac{y_8 - y_p}{y_8 - y_7}. \tag{3.20}$$

Now, the membership in the most ambiguous decision class (2,8) will be determined by the characteristics of the typographical categories and the boundary conditions which are ensured the continuity with the memberships in classes (1,8), (2,7), (2,9), and (3,8). In

class (2,8), the character could be one of the ascender, descender, centered, and full-ranged characters. First, let us discuss the membership of the full-ranged character, which has the following boundary conditions:

$$\chi_{(2,8)}(\mathrm{F})_{y_q=y_1} = y'_p, \tag{3.21}$$

and

$$\chi_{(2,8)}(\mathrm{F})_{y_p=y_8} = 1 - y'_q. \tag{3.22}$$

The membership of the full-ranged character is characterized by the size only. Let $s = y'_p - y'_q$ and $t = y'_p + y'_q$. The membership function, which is linear with respect to the size, has the following characteristics:

$$\frac{\partial \chi_{(2,8)}(\mathrm{F})}{\partial s} = c_1, \quad \text{when } \chi_{(2,8)}((\mathrm{F}) > 0, \tag{3.23}$$

where c_1 is a positive constant, and

$$\frac{\partial \chi_{(2,8)}(\mathrm{F})}{\partial t} = 0. \tag{3.24}$$

From eqs. (3.21)-(3.24), the membership function of the full-ranged character can be derived as:

$$\chi_{(2,8)}(\mathrm{F}) = \begin{cases} y'_p - y'_q & \text{if } y'_p > y'_q \\ 0 & \text{otherwise.} \end{cases} \tag{3.25}$$

Similarly, the membership function of the centered character, which also simply depends on the character size, can be formulated as:

$$\chi_{(2,8)}(\mathrm{C}) = \begin{cases} y'_q - y'_p & \text{if } y'_p < y'_q \\ 0 & \text{otherwise.} \end{cases} \tag{3.26}$$

The memberships of the ascender and descender are more complicated since they depend on size and position. For the ascender, the membership function has the following boundary conditions:

$$\chi_{(2,8)}(\mathrm{A})_{y_q=y_1} = 1 - y'_p, \tag{3.27}$$

$$\chi_{(2,8)}(\mathrm{A})_{y_p=y_7} = 1 - y'_q. \tag{3.28}$$

In the other hand, the membership function should have the following characteristics:

$$\frac{\partial \chi_{(2,8)}(\mathrm{A})}{\partial t} = c_2, \tag{3.29}$$

and

$$\frac{\partial \chi_{(2,8)}(\mathrm{A})}{\partial s} = \begin{cases} c_3 & \text{if } y'_p < y'_q \\ -c_3 & \text{otherwise,} \end{cases} \tag{3.30}$$

where c_2 and c_3 are positive constants.

From eqs. (3.27)-(3.30), the membership function can be obtained:

$$\chi_{(2,8)}(\mathrm{A}) = \begin{cases} 1 - y'_q & \text{if } y'_p \le y'_q \\ 1 - y'_p & \text{otherwise.} \end{cases} \tag{3.31}$$

Similarly, the membership function of the descender can be derived as:

$$\chi_{(2,8)}(\mathrm{D}) = \begin{cases} y'_q & \text{if } y'_p \ge y'_q \\ y'_p & \text{otherwise.} \end{cases} \tag{3.32}$$

The membership functions in class (2,8) are continuous together with neighboring categories since the boundary conditions are considered. The membership functions in class (2,8) are shown in Fig. 3.6. Note that the sum of the memberships is equal to one.

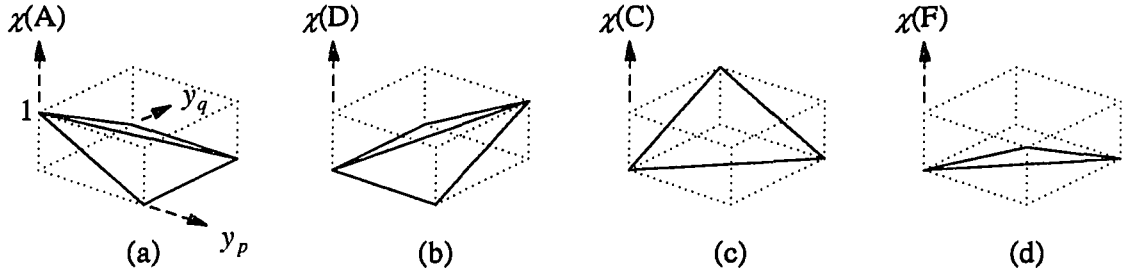


Figure 3.6 The membership functions in class (2,8) of (a) ascender, (b) descender, (c) center, and (d) full-range.

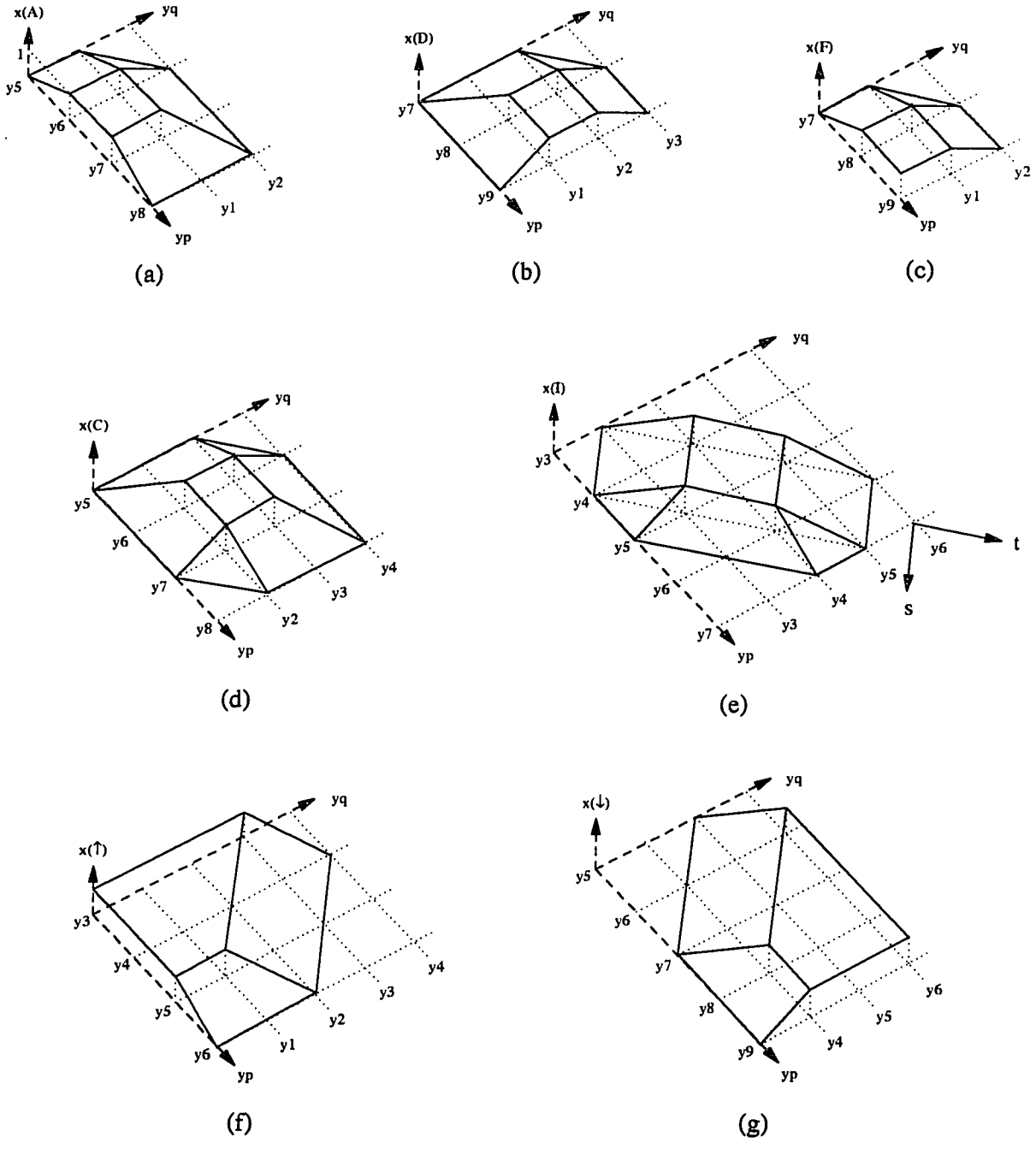


Figure 3.7 The membership functions of (a) ascender, (b) descender, (c) full-range, (d) center, (e) internal, (f) superscript, and (g)subscript.

The membership functions of other ambiguous decision classes can be similarly derived. The entire membership functions for each typographical category are shown in Fig. 3.7. Note that the membership functions of the internal, centered, and full-ranged characters are diagonally symmetric in the normalized (y_p,y_q)-plane. In addition, the membership functions of the ascender and superscript are diagonally symmetric with those of the descender and subscript.

3.7. Experimental Results and Discussion

The textual blocks extracted from chapter two are used as the input. In this stage, the typographical structure of each text line is analyzed independently. The result of applying our virtual reference lines detection is shown in Fig. 3.8. Various character sizes from 5 to 12 points, each consisting of 500 text lines, have been tested. The baseline is 100% correctly detected for the character size larger than 5 points. For 5-point characters, 2.5% error rate for baseline detection is found because the merged characters are highly increased. As shown in Table 3.2, 48.65% of characters are touching together. Besides, the touching characters may contain more characters as their sizes decrease. In our experiment, 26.28% of merged characters contain more than three characters, which makes the slope approximation inaccurate.

Beyond the seven basic typographical categories, one more category, semi-ascender, which corresponds to the class with $y_q \in r_2$ and $y_p \in r_7$ as listed in Table 3.1, has been included. This category includes character "t". In the experiment of a sample document with the 7-point character size, 78 input patterns which are character "t" in Roman or Italic fonts are classified as semi-ascenders, and 10 input patterns are classified as ascenders due to noise.

When the baseline is correctly located, the overall ability for typographical categorization is 100%, in which more than 99% of the characters are classified from the deterministic classes. According to the experimental results, there are a few special symbols

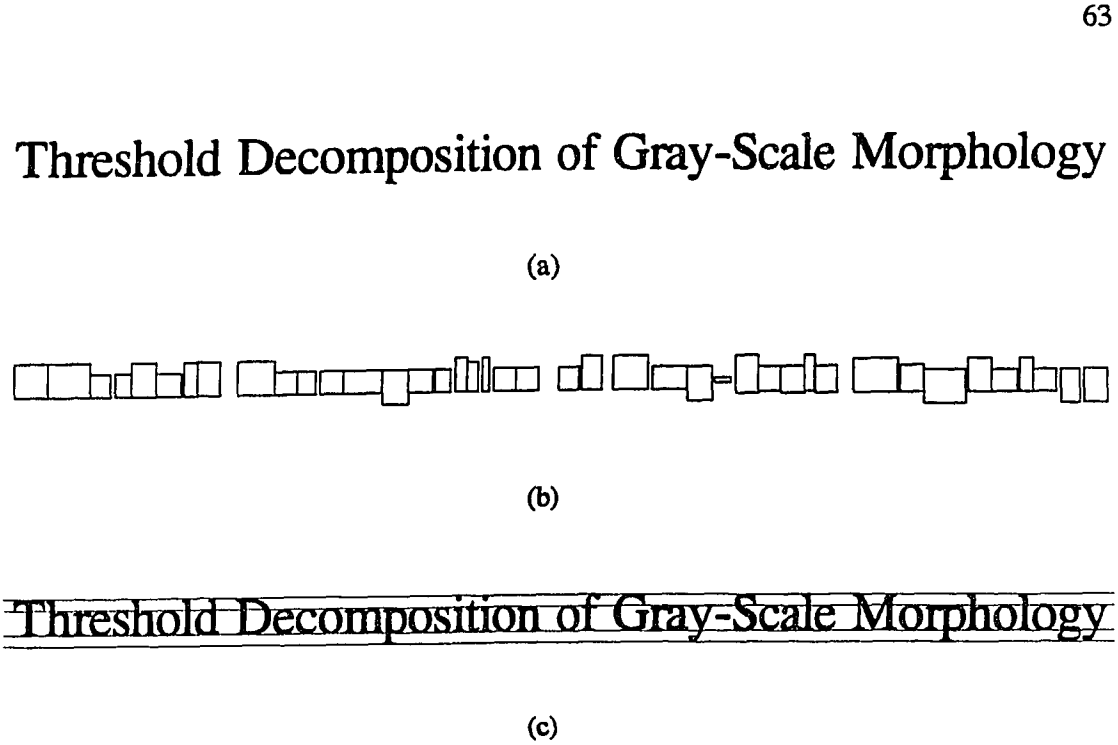


Figure 3.8 The typographical analysis of a textual block. (a) A sample text image, (b) the bounding boxes for each corresponding character, (c) the virtual reference lines of the text line.

Table 3.2 Experimental Results from Our Algorithm

Character size	Error rate for baseline detection	Ambiguous character	Merged character	Ave. Run time (ms/ch)
5	2.5%	0.50%	48.65%	85
6	0%	0.27%	28.23%	82
7	0%	0.10%	17.17%	84
8	0%	0.21%	17.59%	84
9	0%	0.19%	13.39%	81
10	0%	0.62%	5.56%	81
11	0%	0.68%	1.90%	79
12	0%	0.41%	1.45%	80

which may fall into the ambiguous classes. Sometimes, for example, the symbols "/", "(", ")", "[", and "]" are detected in class (1, 8) and ";" is detected in class (3, 8). When the fuzzy logic is applied, "/" is classified as ascender, ";" is classified as descender, and the remainders are classified as full-range if the category with highest membership is selected. Other ambiguous characters happen when merged characters contain these symbols or some characters belonging to the secondary class. For example, merged character "t," and "t;" have been detected as belonging to class (2, 8). This typographical knowledge may help to separate them in the recognition phase.

FTA has been tested on 4,000 text lines containing 307,221 characters. The average running time (denoted as t) is independent of the number and size of characters. When FTA is executed to each text line individually, t = 160.5 ms/ch (microsecond per character). However, when a whole document is processed, the initial slope of the baseline need not be estimated every time. Instead, the slope of the previous text line is used. Therefore, the running can be reduced to t = 82 ms/ch. In other word, for one second, FTA can process 12,195.12 characters. Comparing to the speed of character recognition, it is worthy of preclassifying the characters by FTA so that OCR can be simplified and improved.

One problem has to be considered is the tolerance estimation. When the character size is small, the tolerance will be relatively small. Therefore, a small noise may induce the misclassification. For example, the text line shown in Fig. 3.9 is firstly estimated with 21.6 pixel-high and the tolerance δ = 1.2. Therefore, δ = 2 is enforced when the derived tolerance is smaller than 2. However, the tolerance of top line and upper baseline will be overlapped in the case of the text height which is smaller than 16, i.e., smaller than a 4 point-size character. Fortunately, it is rarely used for normal documents. The results show that the text line can be correctly categorized for the character size as small as 6 points.

Figure 3.9 A text line with a small character size

Another problem is occurred if a text line includes different sizes of characters. This special case is illustrated in Fig. 3.10, where the characters belonging to the string "IEEE MEMBER" are categorized as centers. In this example, every word contains only a single type of characters, i.e., the ascender or the center.

Figure 3.10 A special case of text contains different sizes of characters

FTA can also be extended to handwritten characters. Particularly, it is significant to determine that a character should be an upper- or lower-case. Several examples are illustrated in Fig. 3.11. For the first example of the left column, the skew angle is also detected. For the second example of the left column, the word could be recognized as "RosemARiE" and would be corrected as "Rosemarie" by FTA.

Figure 3.11 Sample sets of handwritten characters for fuzzy typographical analysis.

CHAPTER 4

A FUZZY MODEL FOR UNSUPERVISED CHARACTER CLASSIFICATION

In this chapter, we present a fuzzy logic approach to efficiently perform the unsupervised character classification for improvement on robustness, correctness, and speed of a character recognition system.

4.1 Introduction

Traditionally, OCR employs the rule-based or supervised approach [1, 3, 7, 16, 44, 52, 71, 77] to convert the digital document image into ASCII codes in order for a computer to further editing, manipulation, and storage. The problems of efficiency, accuracy, and reliability have been encountered because hundreds of font styles, character sizes, special symbols, and different printing devices producing images of different qualities are to be dealt with. Another serious problem is the unavoidable noise while printing. Therefore, some postprocessing strategies such as dictionary checking and semantics understanding are necessary to combine with OCR [36, 39, 71, 76].

The character classification is performed prior to character recognition in order to extract a set of representative prototypes in which the similar patterns of characters are grouped together. Therefore, the character recognition procedure becomes simpler and faster if it has to apply on the limited versions of the fuzzy prototypes. Compared to the existing OCR systems, our classification reduces the scope of characters to be recognized significantly, and our fuzzy model is more robust. Wong *et al.* [11, 95] proposed a decision network using a binary tree in which each leaf denotes a prototype and the most distinguishable pixel among character patterns is placed at each node for decision making leading to the closest prototype. However, the reliability of the pixel at each node is

questionable since noise or some variations can happen. In addition, their methodology in searching for the most distinguishable pixel is unclear.

For each typographical category, the fuzzy unsupervised character classification classifies its characters into a set of fuzzy prototypes. Initially, the set of fuzzy prototypes associated with each category is empty. Given a category of characters, the first input of the category is set to the first element of the corresponding fuzzy prototypes set. The following input character is matched against the first element. If it is matched, the input character is grouped to that element; otherwise it is placed as the second element of the set of fuzzy prototypes. The procedure is repeated until all inputs are classified.

This chapter is organized as follows. Section 2 discusses the advantages of the fuzzy character classification. Section 3 presents the similarity measurement. Section 4 discusses the statistical fuzzy model for classification. Section 5 proposes the similarity measure in fuzzy model. Section 6 describes the matching algorithm. Section 7 presents the classification hierarchy. Section 8 elaborates the preclassifier for grouping the fuzzy prototypes. Section 9 provides experimental results.

4.2. Advantages of Unsupervised Fuzzy Character Classification

Our unsupervised character classification combines the topological and statistical approaches to group all the input characters into a relatively small set of categories, namely prototypes. It can simplify the character recognition problem with the following advantages:

(1) It reduces the recognition scope into a set of prototypes, and therefore reduces the processing time. As mentioned above, there exists hundreds of font styles and different character sizes in today's documents. These variations make the character recognition task to be much more complicated. It has been suggested that recognition of individual characters can be improved by combining multiple independent feature sets/classifiers, so that the weakness of one is compensated for by the strength of the

other [7, 9, 37, 79]. However, it also increases the running time. If the recognition is performed on every individual characters, the speed efficiency of the recognition will be reduced significantly. Therefore, a character classification prior to recognition is critical to simplify the recognition task.

(2) The special-font words can be detected easily, thus it helps to extract key words which are used in an information retrieval system. Usually, Roman font is applied for the main body of the text in a document. Most of the authors intend to emphasize some keywords by using a different font such as **bold-face** or *Italic*. The counts of the characters which constitute the prototypes provide the information of the majority and minority of characters in the document. Thus, the characters with different fonts can be separated, and the keywords could be captured from the words which are composed of the minor set of characters. Without the classification, an OCR system must be able to distinguish the fonts for every characters and therefore the performance will be reduced significantly. Moreover, even though we include the techniques to recognize the character fonts to the limited version of prototypes, the efficiency of the system will not be considerably affected.

Unsupervised character classification simplify the characters in the whole document into a set of prototypes. However, what is the appropriate representative image pattern for each prototype? If we use the first character image of each prototype as its representative image [95]. Should the first one of a prototype be a noisy image and be misrecognized, all the characters belong to this prototype are misclassified. That why the fuzzy logic is applied with the additional advantage:

(3) The fuzzy logic reduces the noise in the character images, and therefore increases the recognition rate. The noise often occurs on the boundary of the character to produce missing or extra pixels. Conventional approaches require a smoothing or enhancement algorithm to remove the noise. Sometimes, pixels are difficult to determine whether they belong to noise or not. In essence, pattern classification refers to the

problem of fuzziness. With the help of fuzzy logic, the membership of each pixel in a character pattern is assigned, and noise can be equivalently removed. Fig. 4.1 illustrates five images of character "a." If 8-connectivity is used for the object, the first and the fifth images appear to have two loops because the insufficient presence of the low resolution of the scanner. The recognition of these characters based on topological features is difficult since they have two holes [41]. When fuzzy classification is applied, the noise can be removed, and the correct topological features can be extracted.

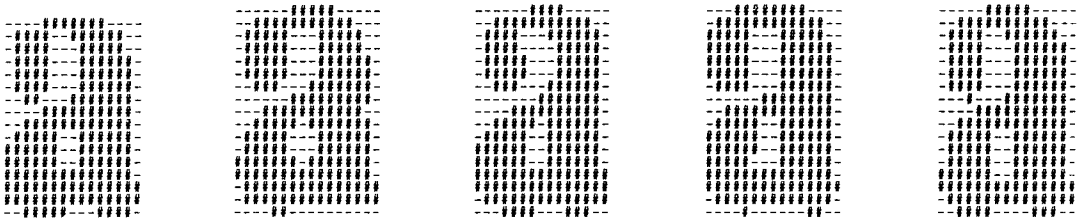


Figure 4.1 Character images of "a."

Incorporating fuzzy logic in the unsupervised character classification yields the increasing reliability for single character recognition. Furthermore, there is one more advantage in improving the recognition of the ambiguous characters.

(4) The recognition of ambiguous characters such as merged or broken characters, can be easily performed, thus it increases reliability. Recognition of ambiguous characters is still remaining as unsolved problem, and the merged characters are encountered frequently in a document. For example, "t" and "h" often touch with each other. Firstly, classification will assign those merged characters into one category, and the splitting algorithm will be applied only once to the representative prototype. Secondly, the linking pixels of these merged characters have lower membership values and these characters are said to be weakly linked when the fuzzy model is employed. Therefore, the splitting locations of characters are easy to detect.

Thirdly, the candidates of the break position can be searched by traditional splitting algorithms or by a partial match with the recognized prototypes. In particular, the possible break position can be verified using the dictionary checking not only to a single word but also to all corresponding words in a document.

4.3 Similarity Measurement

Similarity is an abstract concept of fuzziness which provides a quantitative measure to the relationship between two variables. Given two patterns A and B. Let $|A|$ denote the cardinality of set A. The similarity measurement or correlation coefficient, denoted as $\zeta(A, B)$, can be expressed in the following ways [26, 69, 73]:

$$\zeta(A, B) = \frac{|A \cap B|}{|A \cup B|}, \tag{4.1}$$

$$\zeta(A, B) = \frac{\sum a_{ij} b_{ij}}{\sqrt{\sum a_{ij}^2 \sum b_{ij}^2}}, \tag{4.2}$$

$$\zeta(A, B) = \frac{\sum (a_{ij} - a'_{ij})\, b_{ij}}{\sqrt{\sum a_{ij}^2 \sum b_{ij}^2}}, \tag{4.3}$$

$$\zeta(A, B) = \frac{\sum (a_{ij} - \bar{a})(b_{ij} - \bar{b})}{\sqrt{\sum (a_{ij} - \bar{a})^2 \sum (b_{ij} - \bar{b})^2}}, \tag{4.4}$$

where a'_{ij} denotes the complement of a_{ij}, $\bar{a} = \sum a_{ij} / |A|$ and $\bar{b} = \sum b_{ij} / |B|$. Assume $A = \{a_{ij} \mid 1 \le i \le cols, 1 \le j \le rows\}$ and $B = \{b_{ij} \mid 1 \le i \le cols, 1 \le j \le rows\}$ are images with the size $m = cols \times rows$. If they are binary images, let 0 or 1 represent background and foreground, respectively. Eqs. (4.1)-(4.3) are specifically for binary images. In order to preserve the symmetricity, i.e., $\zeta(A, B) = \zeta(B, A)$, eq. (4.3) is modified as

$$\zeta(A, B) = \frac{\sum (a_{ij} b_{ij} - \frac{1}{2} a'_{ij}\, b_{ij} - \frac{1}{2} a_{ij} b'_{ij})}{\sqrt{\sum a_{ij}^2 \sum b_{ij}^2}}. \tag{4.5}$$

The weight ½ is added to normalize the similarity ranging between -1 and 1. Eq. (4.4) is a general form for gray-scale images, which is known as the Pearson product moment correlation, .

Let us discuss the characteristics of the forementioned similarity functions. Let $n_A = |A|$ and $n_B = |B|$ denote the cardinality of sets A and B, respectively. Also let $x = |A \cap B|$ denote the number of common elements in A and B. Let m denote the total number of elements in the domain of A and B, i.e. $m = |A| + |A'| = |B| + |B'| = cols \times rows$. Eqs. (4.1), (4.2), (4.5), and (4.4) can be rewritten respectively as

$$\zeta = \frac{|A \cap B|}{|A \cup B|} = \frac{x}{n_A + n_B - x}, \tag{4.6}$$

$$\zeta = \frac{\sum a_{ij} b_{ij}}{\sqrt{\sum a_{ij}^2 \sum b_{ij}^2}} = \frac{x}{\sqrt{n_A n_B}}, \tag{4.7}$$

$$\zeta = \frac{\sum (a_{ij} b_{ij} - \frac{1}{2} a'_{ij}\, b_{ij} - \frac{1}{2} a_{ij} b'_{ij})}{\sqrt{\sum a_{ij}^2 \sum b_{ij}^2}} = \frac{2x - \frac{n_A}{2} - \frac{n_B}{2}}{\sqrt{n_A n_B}}, \tag{4.8}$$

$$\zeta = \frac{\sum (a_{ij} - \bar{a})(b_{ij} - \bar{b})}{\sqrt{\sum (a_{ij} - \bar{a})^2 \sum (b_{ij} - \bar{b})^2}} = \frac{mx - n_A n_B}{\sqrt{(mn_A - n_A^2)(mn_B - n_B^2)}}. \tag{4.9}$$

Eq. (4.6) is a parabola function and eqs. (4.7)-(4.9) are linear functions. Fig. 4.2 shows the relationship of correlation coefficient and the percentage of the intersection when $n_A = n_B = n$.

Eq. (4.9) is not suitable for binary image matching since the number of the domain image m affects the value of ζ. Eq. (4.8) seems more appropriate because it measures the differences of the equality, $\frac{x}{\sqrt{n_A n_B}}$, and the inequality, $\frac{n_A + n_B - 2x}{2\sqrt{n_A n_B}}$. However, if the weight of the inequality is a constant, the calculation of inequality is redundant since it is implied from the equality.

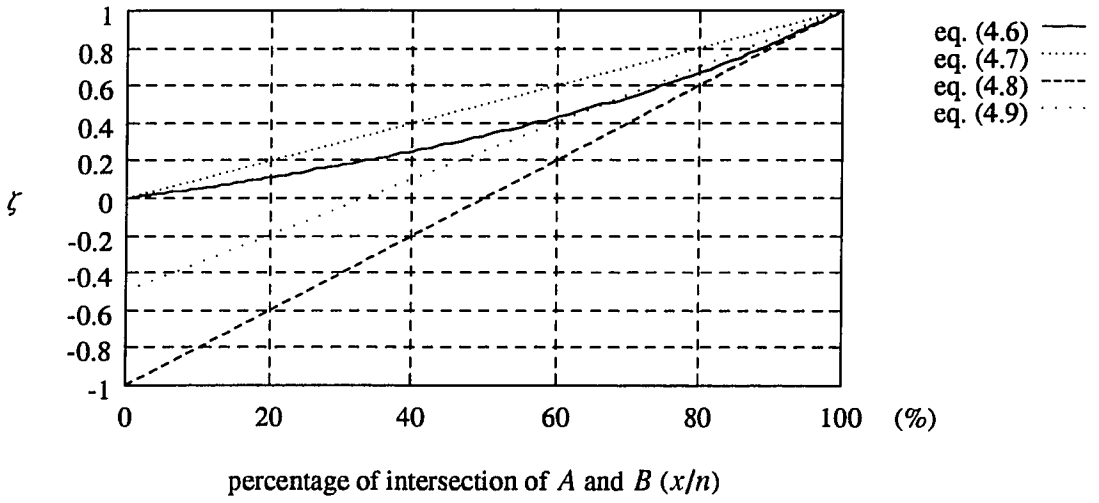
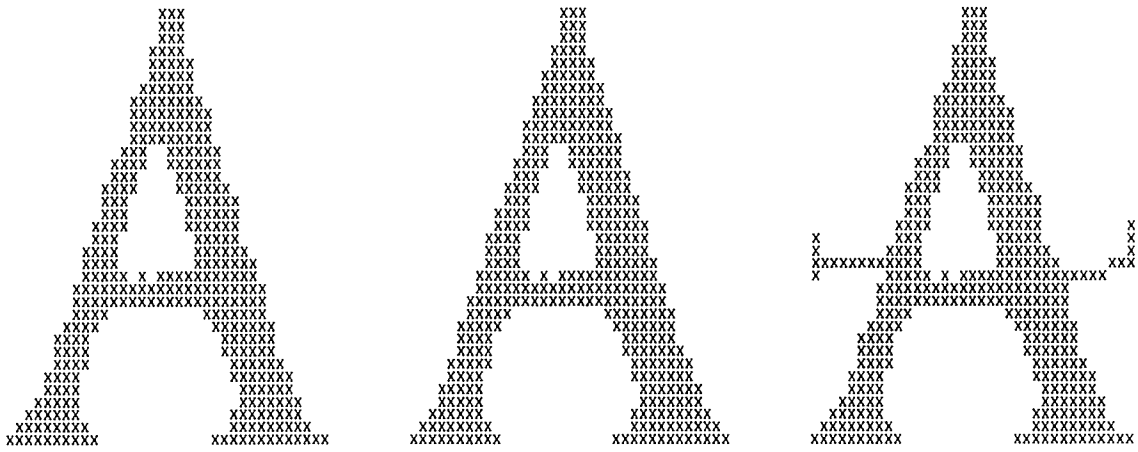


Figure 4.2 The relationship of ζ and $\frac{x}{n}$ when $n_A = n_B = n$.

A major problem occurs in eqs. (4.6)-(4.9). Since their matching is based on the percentage of intersection of the two patterns, the location of inequality between the two patterns is not taken into account. By observing three images A_1, A_2, and A_3 in Fig. 4.3, A_2 and A_3 have the same number of foreground pixels, and the total number of common pixels of A_1 and A_2 is the same as that of A_1 and A_3. From human's perception, A_1 and A_2 look more similar than A_1 and A_3 since the differences of A_1 and A_2 are the boundary pixels, but the differences of A_1 and A_3 are significant noise. However, the results show that the correlation of A_1 and A_2 is identical to that of A_1 and A_3 by applying the forementioned functions.

It is concluded that the similarity measure must include the location differences. The differences lying along the pattern's boundaries should be considered less significant than lying far away from the object. We propose a nonlinear weighted similarity function which can be expressed as

$$\zeta(A, B) = \frac{\sum(a_{ij}b_{ij} - \frac{1}{2}\omega_{ij}b_{ij} - \frac{1}{2}v_{ij}a_{ij})}{\sqrt{\sum a_{ij}^2 \sum b_{ij}^2}}, \tag{4.10}$$



A_1 A_2 A_3

(a)

	$\zeta(A_1, A_2)$	$\zeta(A_1, A_3)$
eq. (4.6)	0.941799	0.941799
eq. (4.7)	0.970463	0.970463
eq. (4.8)	0.940477	0.940477
eq. (4.9)	0.958100	0.958100

(b)

Figure 4.3 (a) Sample images A_1, A_2, and A_3 (b) Correlation coefficients by eqs. (4.6)-(4.9).

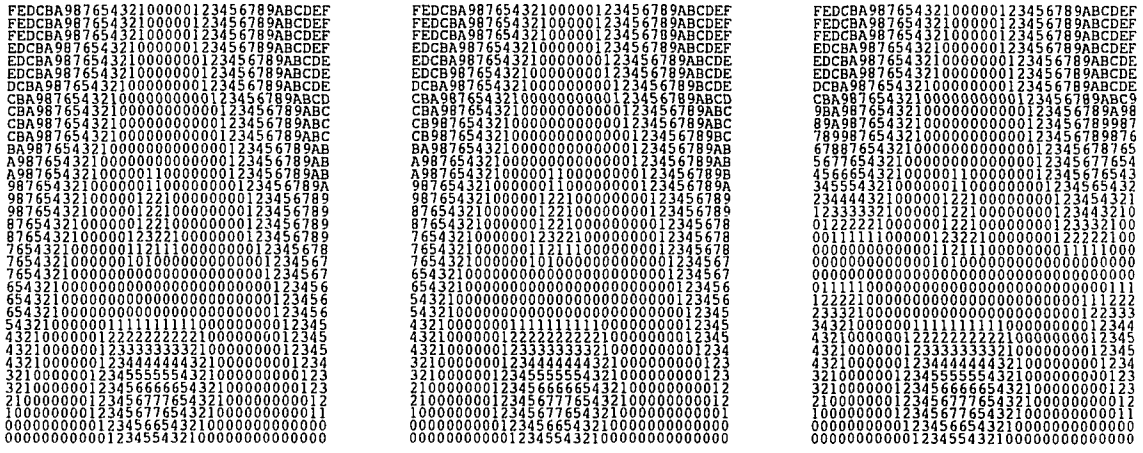
where ω_{ij} and v_{ij} are the weights representing a distance measure of pixel (i, j) to objects A and B, respectively. Since the 8-connectivity for objects and 4-connectivity for background are used, we adopt a simple city-block distance measure. It can be formulated as

$$\omega = \{\omega_{ij} \,|\, \omega_{ij} = \max\,(\, d_4(a_{ij}) - 1, 0\,)\}, \tag{4.11}$$

$$v = \{v_{ij} \,|\, v_{ij} = \max\,(\, d_4(b_{ij}) - 1, 0\,)\}, \tag{4.12}$$

where $d_4(a_{ij})$ denotes the city-block distance of pixel a_{ij} from the object A. Note that 1 is subtracted because a pixel along the contour of A is considered as a reasonable tolerance of noise. Fig. 4.4 shows the weights of the patterns in Fig. 4.3(a) and their correlations by eq. (3.10). Note that $\zeta(A_1, A_2)$ has the same value as eq. (4.7) is applied, but

$\zeta(A_1, A_3)$ decreases because the noisy pixels away from the boundary receive higher weights in the subtraction of eq. (4.10).



ω_1 ω_2 ω_3

(a)

$$\zeta(A_1, A_2) = 0.970463$$
$$\zeta(A_1, A_3) = 0.828710$$

(b)

Figure 4.4 (a) The weights of A_1, A_2, and A_3 (b) Their correlation coefficients by eq. (4.10).

4.4 Statistical Fuzzy Model for Classification

The set of fuzzy prototypes is constructed based on statistical analysis in grouping similar patterns into a single class. An image of a fuzzy prototype is a matrix of pixels with each element being associated with a membership representing the degree of the pixel which belongs to the object. The definitions, propositions, and theorems pertaining to fuzzy matching are described below.

Definition 1: Let E^2 be two-dimensional Euclidean space. A fuzzy model λ in E^2 is a matrix of ordered pairs such that $\lambda = \{(p_{ij}, \chi_{ij})\}$, where p_{ij} represents a pixel and χ_{ij}, ranging in [0, 1], represents the grade of membership of the pixel belonging to the object.

Note that the memberships of the pixels outside the matrix are considered as 0. In addition, a binary image can be regarded as a special case in fuzzy model whose membership has values only 1 or 0. In reality, a fuzzy model is more natural to represent the fuzziness of the image boundary. The concept of the fuzzy model is different from the gray-scale image whose element represents the gray levels in brightness.

Proposition 1: Assume a fuzzy prototype λ is composed of a set of merged binary images $\{A_1, A_2, \ldots, A_m\}$. The membership of each pixel in λ is computed as

$$\chi_{ij} = \frac{\sum_{a_{ij} \in A_1}^{A_m} a_{ij}}{m}. \tag{4.13}$$

The properties of the fuzzy model extended from the crispy model are also fuzzy. Some properties pertaining to character classification are discussed in this section.

Definition 2: Let $\lambda = \{(p_1, \chi_1), (p_2, \chi_2), \ldots, (p_n, \chi_n)\}$ be a fuzzy set. The *cardinality* of λ, denoted as σ_λ, is a fuzzy number and can be formulated as: $\sigma_\lambda = \{(i, \psi_i) \mid i = 0, 1, 2, \ldots, n\}$, where ψ_i denotes the membership of the cardinality being equal to i.

From probability theory, it is plausible to formulate ψ_i as follows:

$$\psi_0 = \chi'_1 \chi'_2 \cdots \chi'_n = \prod_{i=1}^{n} \chi'_i \tag{4.14}$$

$$\psi_1 = \sum_{i=1}^{n} \left[\chi_i \prod_{\substack{j=1,\ldots,n \\ j \neq i}} \chi'_j \right], \cdots, \quad \psi_{n-1} = \sum_{i=1}^{n} \left[\left(\prod_{\substack{j=1,\ldots,n \\ j \neq i}} \chi_j \right) \chi'_i \right], \quad \psi_n = \prod_{i=1}^{n} \chi_i. \tag{4.15}$$

Eq. (4.15) can be expressed in a general form as

$$\psi_m = \sum_{i_1 \neq i_2 \neq \cdots \neq i_m} \left[\chi_{i_1} \chi_{i_2} \cdots \chi_{i_m} \prod_{\substack{j \neq i_1 \\ j \neq i_2 \\ \cdots \\ j \neq i_m}} \chi'_j \right]. \tag{4.16}$$

Note that the fuzzy cardinality has the property that $\sum_{i=0}^{n} \psi_i = 1$. The expected value of the cardinality can be derived as

$$E(\sigma_\lambda) = \sum_{i=1}^{n} i\psi_i = \sum_{i=1}^{n} \chi_i. \tag{4.17}$$

The detail of the derivation is omitted here. It is significant to simplify the cardinality of a fuzzy set to a unique value, i.e. the expected value of the fuzzy cardinality, which is the sum of the membership of the fuzzy set.

Example: Let $\lambda = \{ (p_1, 1/2), (p_2, 1), (p_3, 1/4) \}$. The fuzzy cardinality $\sigma_\lambda = \{ (0, 0), (1, 3/8), (2, 1/2), (3, 1/8) \}$. The expected value of cardinality is $0 \times 0 + 1 \times 3/8 + 2 \times 1/2 + 3 \times 1/8 = 1/2 + 1 + 1/4 = 7/4$.

Proposition 2: The cardinality of a fuzzy prototype λ in E^2 is equal to the sum of the membership values. That is, $\sigma_\lambda = \sum \chi_{ij}$.

Similar to the derivation in cardinality, the centroid of a fuzzy prototype can be simplified to be a unique number.

Proposition 3: The centroid of the first moment of a fuzzy prototype λ in E^2 is formulated as

$$x_{c_\lambda} = \frac{\sum j\chi_{ij}}{\sum \chi_{ij}}, \quad y_{c_\lambda} = \frac{\sum i\chi_{ij}}{\sum \chi_{ij}}. \tag{4.18}$$

Proposition 4: The width of a fuzzy prototype λ in E^2 is the summation of the maximal memberships in columns, and the height of a fuzzy prototype λ in E^2 is the summation of the maximal memberships in rows. That is

$$w_\lambda = \sum_j (\max_i(\chi_{ij})), \quad h_\lambda = \sum_i (\max_j(\chi_{ij})). \tag{4.19}$$

Theorem 1: If a fuzzy prototype λ is composed of a set of merged binary images $\{A_1, A_2, \ldots, A_m\}$ with widths $\{w_1, w_2, \ldots, w_m\}$ and heights $\{h_1, h_2, \ldots, h_m\}$, then the width of λ is less than or equal to the average width of the set of the images, and the height of λ is less than or equal to the average height of the set of the images. Mathematically,

$$w_\lambda \le \sum_{i=1}^{m} \frac{w_i}{m}, \quad h_\lambda \le \sum_{i=1}^{m} \frac{h_i}{m}. \tag{4.20}$$

[*Proof*]:

The proof is given by the induction hypothesis. For case when $m = 1$, we have $w_\lambda = w_1$, that satisfies eq. (4.20).

Assume that $m = x$ also satisfies eq. (4.20) such that $w_\lambda \le \frac{1}{x}\sum_{i=1}^{x} w_i$. Let $\alpha_j = \max_i(\chi_{ij}^{\lambda})$ denote the maximal membership value of j^{th} column in the fuzzy prototype λ. Let $\beta_j = \max_i(\chi_{ij}^{A_{x+1}})$ denote the maximal membership value of j^{th} column in A_{x+1}, which is either 1 or 0. Let ρ_j denote the maximal membership value of j^{th} column in the new fuzzy prototype λ'. When a new image A_{x+1} is merged, the following inequality is derived:

$$\rho_j \le \frac{x\alpha_j + \beta_j}{x+1}.$$

The width of the new fuzzy prototype will be

$$w_{\lambda'} = \sum \rho_j \le \frac{\sum x\alpha_j + \beta_j}{x+1}$$

$$= \frac{x}{x+1}\sum \alpha_j + \frac{1}{x+1}\sum \beta_j$$

$$= \frac{x}{x+1} w_\lambda + \frac{1}{x+1} w_{x+1}$$

$$\leq \frac{x}{x+1}\left(\frac{1}{x}\sum_{i=1}^{x} w_i\right) + \frac{1}{x+1} w_{x+1}$$

$$= \frac{1}{x+1}\sum_{i=1}^{x+1} w_i.$$

The proof for the height can be similarly derived.

QED

Since the fuzzy prototype is a group of similar patterns, the difference between w_λ and $\frac{1}{m}\sum_{i=1}^{m} w_i$ is small. Thus, the latter can be applied to approximate the width of the fuzzy prototype for simplicity.

4.5 Similarity Measure in Fuzzy Model

The similarity measure between two binary patterns discussed in Section 3.5.1 can be extended to the fuzzy model. Similar to eq. (4.10), the similarity measure of two fuzzy prototypes is proposed below.

Proposition 5: Let $\lambda_1 = \{(p_{ij}, \chi_{ij}^{(1)})\}$ and $\lambda_2 = \{(p_{ij}, \chi_{ij}^{(2)})\}$ be two fuzzy prototypes in E^2. Let $\gamma_{\lambda_1} = \{\gamma_{ij}^{(1)}\}$ $\gamma_{\lambda_2} = \{\gamma_{ij}^{(2)}\}$ represent the weight functions associate with λ_1 and λ_2, respectively. The similarity measure of λ_1 and λ_2 is defined as

$$\zeta(\lambda_1, \lambda_2) = \frac{\sum(\chi_{ij}^{(1)} \wedge \chi_{ij}^{(2)} - \frac{1}{2}\gamma_{ij}^{(1)}\chi_{ij}^{(2)} - \frac{1}{2}\gamma_{ij}^{(2)}\chi_{ij}^{(1)})}{\sqrt{\sum \chi_{ij}^{(1)^2}\sum \chi_{ij}^{(2)^2}}}, \tag{4.21}$$

where $\wedge$ is the symbol for minimum representing the intersection on fuzzy sets. That is

$$\lambda_1 \cap \lambda_2 = \{(p_{ij},\ \chi_{ij}^{(1)} \wedge \chi_{ij}^{(2)})\} \tag{4.22}$$

The $\chi_{ij}^{(n)^2}$, where $n = 1, 2$, denotes the self-intersection $\chi_{ij}^{(n)} \wedge \chi_{ij}^{(n)}$.

The reason why the denominator $\chi_{ij}^{(n)} \wedge \chi_{ij}^{(n)}$ is used instead of $\chi_{ij}^{(n)} \times \chi_{ij}^{(n)}$ comes from the viewpoint of fuzzy properties. Consider a membership, $\chi_{ij} = 0.8$, which represents a concept that 80% of the area in a pixel p_{ij} belongs to the object (i.e. has value "1"), and 20% of the area belongs to the background (i.e. has value "0"). Therefore, χ_{ij}^2 should be carried out as $0.8 \times 1^2 + 0.2 \times 0^2 = 0.8$ instead of $0.8^2 = 0.64$. Moreover, the first term of the numerator is $\chi_{ij}^{(1)} \wedge \chi_{ij}^{(2)}$ instead of $\chi_{ij}^{(1)} \times \chi_{ij}^{(2)}$ because two fuzzy subsets are equal if their membership functions are equal. That is

$$\lambda_1 = \lambda_2 \quad \text{iff} \quad \chi_{ij}^{(1)} = \chi_{ij}^{(2)}. \tag{4.23}$$

Therefore,

$$\zeta(\lambda_1, \lambda_1) = \frac{\sum(\chi_{ij}^{(1)} \wedge \chi_{ij}^{(1)} - \frac{1}{2}\gamma_{ij}^{(1)}\chi_{ij}^{(1)} - \frac{1}{2}\gamma_{ij}^{(1)}\chi_{ij}^{(1)})}{\sqrt{\sum \chi_{ij}^{(1)^2} \sum \chi_{ij}^{(1)^2}}} = 1, \tag{4.24}$$

but not

$$\zeta(\lambda_1, \lambda_1) = \frac{\sum(\chi_{ij}^{(1)} \times \chi_{ij}^{(1)} - \frac{1}{2}\gamma_{ij}^{(1)}\chi_{ij}^{(1)} - \frac{1}{2}\gamma_{ij}^{(1)}\chi_{ij}^{(1)})}{\sqrt{\sum \chi_{ij}^{(1)^2} \sum \chi_{ij}^{(1)^2}}} \leq 1. \tag{4.35}$$

Eq. (4.21) is the extended form of eq. (4.10) except that $\chi_{ij}^{(1)}$ and $\chi_{ij}^{(2)}$ are in the interval [0, 1], and $\gamma_{ij}^{(1)}$ and $\gamma_{ij}^{(2)}$ are real numbers instead of integers. Therefore, it is the superset of eq. (4.10). The weight function γ_λ which represents a fuzzy distance measure from the object is also fuzzy. Similarly, it should be derived from the weight function ω of all the patterns comprising the prototype.

Proposition 6: Assume that λ, a fuzzy prototype, is composed of a set of merged binary images $\{A_1, A_2, \ldots, A_m\}$ associated with the weight functions $\{\omega_{A_1}, \omega_{A_2}, \ldots, \omega_{A_m}\}$, which are obtained from eq. (4.11). The weight function, γ, of λ is defined as

$$\gamma_{ij} = \frac{\sum_{\omega_{ij} \in \omega_{A_1}}^{\omega_{A_m}} \omega_{ij}}{m}. \tag{4.26}$$

where ω_{ij}'s are the elements in $\omega_{A_1}, \omega_{A_2}, \ldots, \omega_{A_m}$.

4.6 Matching Algorithm

Given two fuzzy patterns A and B, the way to find the best matching is to shift A around B. Calculate the correlation coefficient for every position, and select the highest value. But, it is not efficient. If two patterns are similar, they should have similar geometric properties. If two patterns are dissimilar, it does not make sense to find the best matching. A simple way is to calculate the centroids of both patterns, and the similarity measure is calculated by matching the centroids. Some allowance must be considered due to noise. The algorithm is described below.

(1) Calculate c_A and c_B, which represent the centroids of A and B. That is,

$$c_A = \left(\frac{\sum j \chi_{ij}^{(A)}}{\sum \chi_{ij}^{(A)}}, \frac{\sum i \chi_{ij}^{(A)}}{\sum \chi_{ij}^{(A)}}\right) \text{ and } c_B = \left(\frac{\sum j \chi_{ij}^{(B)}}{\sum \chi_{ij}^{(B)}}, \frac{\sum i \chi_{ij}^{(B)}}{\sum \chi_{ij}^{(B)}}\right). \tag{4.27}$$

(2) Compute $\zeta(A, B)$ with minimum distance $\overline{c_A c_B}$. Mathematically speaking, $\zeta_{\alpha\beta}(A, B)$ is derived by shifting pattern A with (α, β):

$$\zeta_{\alpha\beta}(A, B) = \frac{\sum(\chi_{ij}^{(A)} \wedge \chi_{i+\alpha, j+\beta}^{(B)} - \frac{1}{2}\gamma_{ij}^{(A)}\chi_{i+\alpha, j+\beta}^{(B)} - \frac{1}{2}\gamma_{i+\alpha, j+\beta}^{(B)}\chi_{ij}^{(A)})}{\sqrt{\sum \chi_{ij}^{(A)^2} \sum \chi_{ij}^{(B)^2}}}, \tag{4.28}$$

where $\alpha = \text{round}(x_{c_B} - x_{c_A})$ and $\beta = \text{round}(y_{c_B} - y_{c_A})$, which are approximately the x and y components of $\overline{c_A c_B}$. If the correlation coefficient is ζ is higher than the threshold, say 0.9, A and B are considered to be the same.

(3) If ζ is in critical range, say 0.8 to 0.9, then the values of $\zeta_{\alpha\beta}(A, B)$ are also calculated with α and β in the range of:

$$|\alpha - (x_{c_B} - x_{c_A})| \leq 1 \text{ and } |\beta - (y_{c_B} - y_{c_A})| \leq 1 \tag{4.29}$$

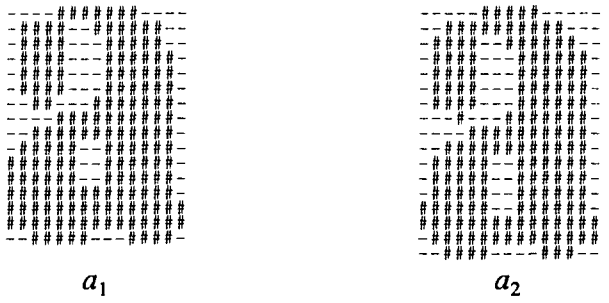
Normally, there are three more points to be matched in this step, if the values of $x_{c_B} - x_{c_A}$ and $y_{c_B} - y_{c_A}$ are not integers. If there is a match, i.e. higher than 0.9, A and B are set to the same class. Otherwise, the two patterns are considered as distinct objects.

Step (3) represents the fuzzy reasoning since the similarity is ambiguous in a critical range. The similarity between two similar patterns could be measured in this range because of the distortion of the centroid of the image due to noise, and a better match could be obtained by shifting one pixel for pattern A. Fig. 4.5 illustrates an example of two primitive images. The ambiguous similarity calculated in Step 2 is shown in Fig. 4.5(b), and a better match found by Step 3 is shown in Fig. 4.5(c). Note that Figs. 4.5(b) and (c) show the superposition of images a_1 and a_2 in their matching positions. The symbols "#", "1", and "2" are used to respectively represent the common pixel of two images, the pixel in image a_1 only, and the pixel in image a_2 only.

4.7 Classification Hierarchy

It is inefficient if the matching is performed against all the character prototypes in sequential. In this section, a hierarchical-tree approach is proposed as illustrated in Fig. 4.6. The leaves in the classification tree represent the primitive patterns, i.e., the raw character images. The intermediate nodes represent the fuzzy model of prototypes which comprise their descendant subtrees. The root is finally the set of prototypes which is composed of the characters of the whole document.

There are two advantages for the hierarchical approach. First, the searching time for matching is less in comparing with sequential approach. Consider two fuzzy subsets of



a_1 a_2

(a)

$\zeta = 0.888892$ $\zeta = 0.951571$

(b) (c)

Figure 4.5 (a) The sample images a_1 and a_2 (b) A critical similarity is measured (c) A good matching is found by shifting a_1 one pixel down.

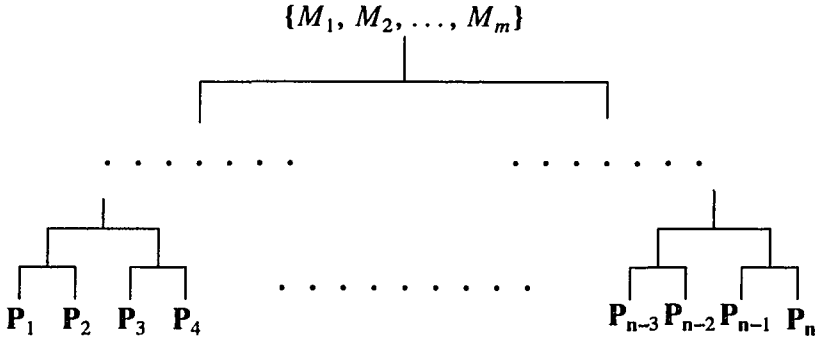


Figure 4.6 The general view of hierarchical classification.

prototypes, X and Y, which consist of the same set of n prototypes. Let the two fuzzy subsets X and Y be constructed from two primitive subsets A and B, respectively, each containing m primitive images. Assume the searching times for matching an input

through a set of n distinct prototypes is n. Therefore, to merge fuzzy sets X and Y requires $\frac{n(n+1)}{2}$ times since the matched prototypes are removed from the set sequentially. Let $T(A)$ and $T(B)$ denote the total searching times in classifying A into X and B into Y, respectively. The worst case in $T(A)$ or $T(B)$ is the first n patterns are distinct and the sequential classification is applied, of which the searching times for the total m patterns are 0, 1, ..., $n-1$, n, n, ..., n, that sum up to be $\frac{n(n-1)}{2} + n(m-n)$. The comparison of searching times using hierarchical and sequential methods is expressed as

$$T_{hier} = T(A) + T(B) + n + (n-1) + \cdots + 1 \tag{4.30}$$

$$\leq T(A) + \frac{n(n-1)}{2} + n(m-n) + \frac{n(n+1)}{2} \tag{4.31}$$

$$= T(A) + nm = T_{seq}, \tag{4.32}$$

where T_{seq} and T_{hier} denote the searching times required for sequential and hierarchical approaches, respectively.

Fig. 4.7 shows an example of 16 inputs with 3 categories by (a) sequential classification, and (b) hierarchical classification. The number under each input in (a) represents the searching time for the classification, and the number under each node in (b) represents the times for merging two sets of fuzzy prototypes.

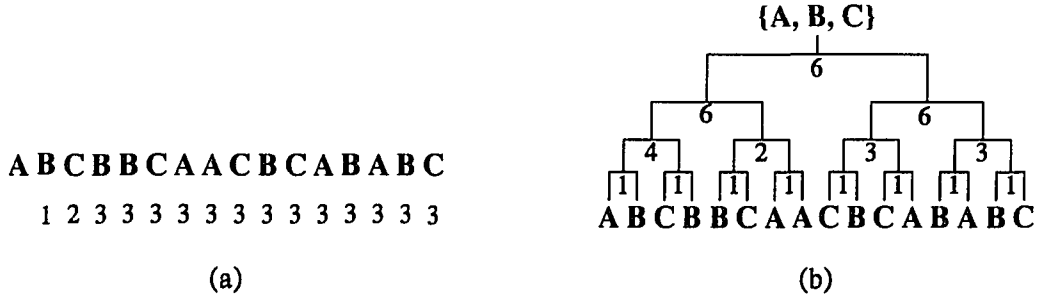


Figure 4.7 Illustration of searching times (a) $T_{seq} = 42$ (b) $T_{hier} = 38$.

Another advantage of the hierarchical classification is the capability of being processed in parallel. Two kinds of parallelism can be carried out. Firstly, each node in classification can be performed in parallel. Secondly, the merging of two fuzzy subsets can be implemented in parallel since all elements can be processed at the same time.

A textual block after segmentation typically represents a line of characters. Usually, the sizes of characters in a line are the same. Therefore, the text line is considered as the processing unit in classification. Firstly, the characters within a text line are classified, and the set of fuzzy prototypes of each line is hierarchically grouped. This facilitates the comparison of the character sizes. If the sizes of two text lines are obviously different, the merged prototype set is split into two disjoint subsets correspondingly and the matching is not performed. The hierarchy of our classification is shown in Fig. 4.8, where TL_i denote the fuzzy set of prototypes of a text line.

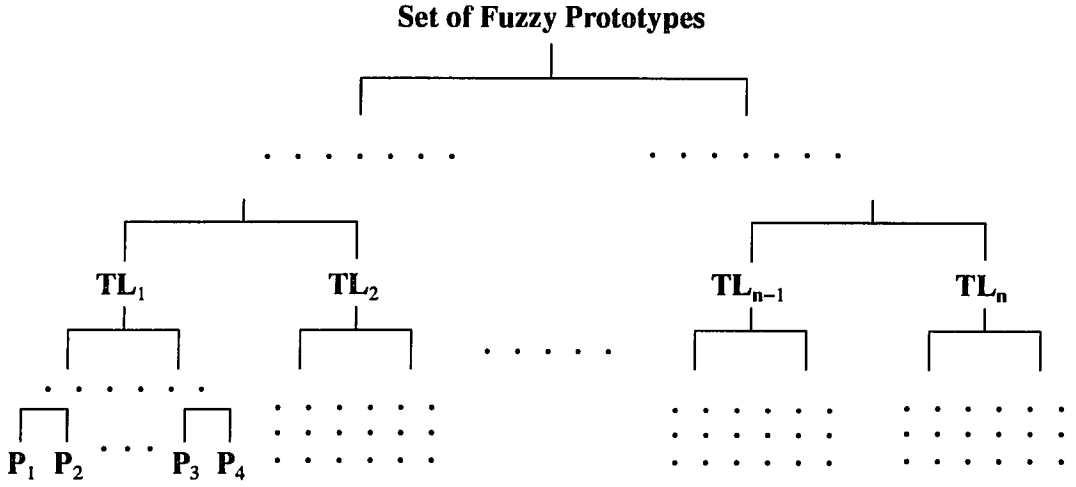


Figure 4.8 The hierarchy of our classification.

4.8 Preclassifier for Grouping the Fuzzy Model

For the lower levels in the hierarchical classification tree, the representative fuzzy subsets of prototypes contain only a few elements. Therefore, it is simple to search similar objects in two subsets. However, in the higher levels it becomes impractical to perform the matching one by one since the size of the libraries increases. To reduce computation, therefore, a preclassifier is used to find out the possible matching prototypes.

Casey *et al.* [11] applied a binary decision network with each interior node representing a comparison of a specific pixel location to the input character and having two branchs, which are known as "black" and "white", leading to a terminal node which represents a possibly matched prototype. However, the reliability is questionable since the reliable pixels of each prototype are different. In addition, a noise may produce the problem in decision making.

To solve the aforementioned problems, a rule-based preclassifier is used. Considering two fuzzy patterns λ_1 and λ_2, the similarity measure is computed using eq. (4.21). The correlation coefficient, ζ, can be divided into two terms of equality measure E and inequality measure I.

$$E = \frac{\sum \chi_{ij}^{(1)} \Lambda \chi_{ij}^{(2)}}{\sqrt{\sum \chi_{ij}^{(1)^2} \chi_{ij}^{(2)^2}}} = \frac{\sigma_{\lambda_1 \cap \lambda_2}}{\sqrt{\sigma_{\lambda_1} \sigma_{\lambda_2}}} \tag{4.33}$$

$$I = \frac{\gamma_{ij}^{(1)} \chi_{ij}^{(2)} + \gamma_{ij}^{(2)} \chi_{ij}^{(1)}}{2\sqrt{\sigma_{\lambda_1} \sigma_{\lambda_2}}} \tag{4.34}$$

Let ζ_t be the threshold of the similarity measure, λ_1 be the input model, and λ_2 be the fuzzy variable in the library in which λ_1 is searching for a matching. For $\sigma_{\lambda_1} > \sigma_{\lambda_2}$, the best equality measure happens if $\lambda_1 \supset \lambda_2$, i.e., $\chi_{ij}^{(1)} \geq \chi_{ij}^{(2)}$. Therefore, λ_2 is a possible matching prototype only if

$$E(\lambda_1, \lambda_2) = \frac{\sigma_{\lambda_1 \cap \lambda_2}}{\sqrt{\sigma_{\lambda_1} \sigma_{\lambda_2}}} = \frac{\sigma_{\lambda_2}}{\sqrt{\sigma_{\lambda_1} \sigma_{\lambda_2}}} = \sqrt{\frac{\sigma_{\lambda_2}}{\sigma_{\lambda_1}}} \geq \zeta_t \tag{4.35}$$

Similarly, for $\sigma_{\lambda_1} < \sigma_{\lambda_2}$, λ_2 is a possible matching prototype only if

$$\sqrt{\frac{\sigma_{\lambda_1}}{\sigma_{\lambda_2}}} \geq \zeta_t \tag{4.36}$$

Therefore, the first rule is concluded using eqs. (4.35) and (4.36).

Rule 1: λ_2 is a possible matching prototype of λ_1 iff $\zeta_t^2 \sigma_{\lambda_1} \leq \sigma_{\lambda_2} \leq \frac{\sigma_{\lambda_1}}{\zeta_t^2}$.

Since the similar prototypes posses similar features, additional heuristic rules based on the features of the prototypes are described as follows:

Rule 2: Two fuzzy prototypes are impossible to be matched if the difference between their widths exceed a threshold w_t.

Note that the height is not taken into consideration since the prototypes in the same typographical category have the similar heights.

Rule 3: Two fuzzy prototypes are impossible to be matched if the difference of two prototypes in the total number of columns of the left or right region to the centroid is greater than a threshold c_1.

Rule 4: Two fuzzy prototypes are impossible to be matched if the difference of two prototypes in the total number of rows of the upper or lower region to the centroid is greater than a threshold c_2.

In our system, each category of prototypes are sorted by their cardinalities. The set of prototypes to be possibly matched are extracted by Rule 1. Rules 2, 3, and 4 filter out the prototypes which are impossibly matched. Finally, a rough estimation of the similarity measure based on the projection profile is applied to extract the prototypes to be possibly matched prior to the two-dimensional pattern matching.

Rule 5: Let γ_i^{λ} denote the summation of the membership values on i^{th} column of fuzzy prototype λ. Two fuzzy prototypes λ_1 and λ_2 are possibly matched iff the following condition holds:

$$\zeta_t \sqrt{\sigma_{\lambda_1} \sigma_{\lambda_2}} \leq \sum(\gamma_i^{\lambda_1} \wedge \gamma_{i+\alpha}^{\lambda_2}) \text{ for } |\alpha - (x_{c_B} - x_{c_A})| \leq 1, \tag{4.37}$$

where $\wedge$ denotes the symbol of minimum.

4.9 Experimental Results and Discussion

The unsupervised character classification is performed on each text line and a fuzzy subset of prototypes is generated correspondingly. Note that a fuzzy set contains 8 subsets of the typographical categories including the semi-ascender. The subsets with the similar height are grouped hierarchically. Finally, several fuzzy sets of prototypes corresponding to different sizes of characters are produced. The set of fuzzy prototypes for the main text of the document shown in Fig. 2.5 and their features are illustrated in Table 4.1, including 25 ascenders, 6 descenders, 23 centers, 1 full-ranges, 2 subscripts, 2 internals, and 2 semi-ascenders. Fig. 4.9 shows the details of a few prototype examples, where the notations "#" and "-" represent the membership greater than 0.95 and less than 0.05 respectively, and an integer i represents the membership between $\frac{i}{10} - 0.05$ and $\frac{i}{10} + 0.05$. Obviously, the fuzzy prototypes are more reliable to be recognized since each pixel is statistically analyzed and the pixels with low membership values can be considered as noises and removed. Some of the merged characters are illustrated in Fig. 4.10. It is observed that the membership values of the linking pixels are lower when more patterns are included, and they can be separated. Those, which do not have lower values on the linking pixels because the prototype contains too few patterns, can be split more easily by means of splitting algorithm, partial matching, and dictionary look-up.

The problem in this stage is the selection of a suitable threshold for the similarity measure. For bigger character size, the boundary between distinct character categories is

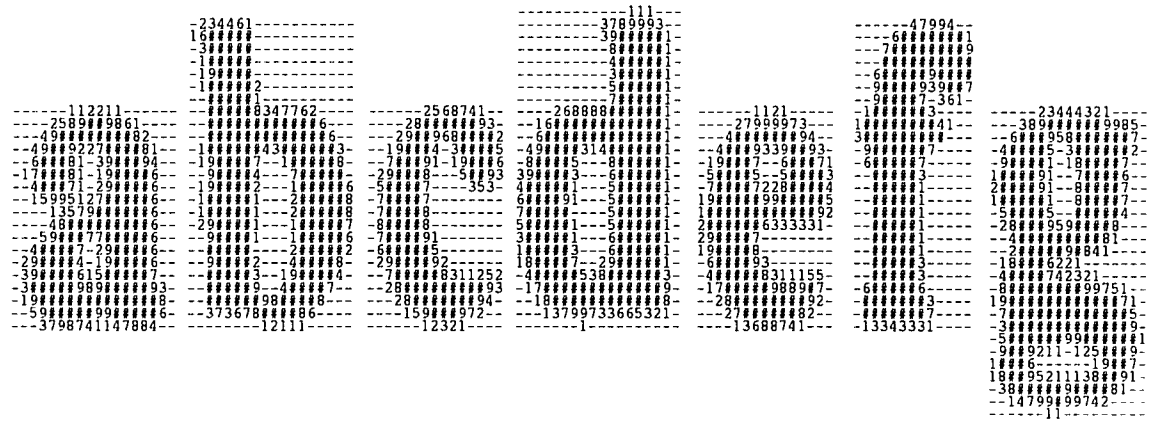


Figure 4.9 Examples of some fuzzy prototypes.

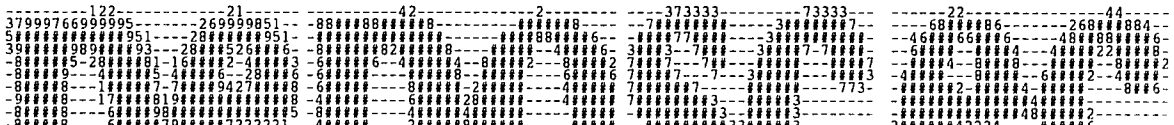


Figure 4.10 Examples of fuzzy prototypes with merged characters.

clear. However, when the characters are smaller, the scope of different categories may be overlapped. There are two reasons. Firstly, for smaller size the cardinality of the patterns is relatively small. However, the tolerance of noise, which may be caused by scanning or printing, is not affected by the character sizes. Therefore, the possible minimal equality measure for the same category will be lower. Secondly, the inequality measure may be smaller for different categories since the different part of two different patterns in smaller size is smaller, and the weight is linear to the distance from the object. For example, the inequality measure between character "i" and "l" in smaller size is usually less than that in bigger size for the same font because the latter contains more pixels of difference and

Table 4.1 The fuzzy set of prototypes for the main text of the document in Fig. 2.5.

Ascenders			
No.	Prototype	No. of patterns	Cardinality
1	i	96	114.24
2	t	10	121.00
3	l	66	130.03
4	*l*	1	147.00
5	f	7	165.43
6	I	2	176.50
7	*T*	1	191.00
8	V	1	195.00
9	*b*	1	199.00
10	L	1	201.00
11	*A*	1	203.00
12	T	3	208.33
13	S	1	216.00
14	k	3	216.67
15	A	1	223.00
16	b	9	223.56
17	*d*	1	225.00
18	d	36	236.97
19	h	41	240.85
20	G	1	259.00
21	fi	1	271.00
22	B	2	318.00
23	R	1	345.00
24	M	1	369.00
25	ffi	1	434.00

Descenders			
No.	Prototype	No. of patterns	Cardinality
1	y	27	144.67
2	p	36	242.69
3	g	33	244.52
4	pe	13	391.31
5	gy	1	399.00
6	po	5	402.00

Full-range			
No.	Prototype	No. of patterns	Cardinality
1	j	1	152.00

Centers			
No.	Prototype	No. of patterns	Cardinality
1	*r*	1	92.00
2	*s*	2	103.50
3	*c*	1	107.00
4	v	5	114.80
5	r	77	119.57
6	s	76	125.11
7	c	46	124.74
8	*e*	2	136.00
9	*x*	1	137.00
10	e	108	145.99
11	x	1	146.00
12	o	100	161.63
13	*a*	1	171.00
14	a	88	182.75
15	*n*	1	184.00
16	u	19	189.11
17	w	17	195.91
18	n	75	197.16
19	sc	3	260.33
20	ec	5	275.6
21	oc	2	284.00
22	m	31	293.16
23	*rm*	1	366.00

Subscripts			
No.	Prototype	No. of patterns	Cardinality
1	.	10	28.80
2	,	9	45.22

Internals			
No.	Prototype	No. of patterns	Cardinality
1	-	19	35.26
2	—	2	91.00

Semi-ascenders			
No.	Prototype	No. of patterns	Cardinality
1	*t*	2	107.00
2	t	76	117.25

has a higher weight. Therefore, the threshold must be adaptive to the character size. One approach is to emphasize the inequality measure such as

$$E - I \geq \zeta_t + cI \quad \text{or} \quad E - (c+1)I \geq \zeta_t, \tag{4.38}$$

where c is an coefficient. For the sample image, $c = 4$ and $\zeta_t = 0.86$ are adopted in classifying the main text with about 30-pixel high (point size 7). By emphasizing the inequality measure, the different categories are guaranteed to be separated, which is the principle of the classification. While the patterns belonging to the same category may be split due to the noise, and the fuzzy set may contain more prototypes than it really has. We can perform the similarity comparison again for the final set. Since the noise effect is reduced for the fuzzy set, the similar prototypes being separated before probably will be merged. In the main text of the sample document, there are 12 prototypes which are finally merged.

CHAPTER 5

SKELETONIZATION FOR FUZZY DEGRADED CHARACTER IMAGE

In this chapter we present a topography-based approach of skeletonization for fuzzy images which is capable to skeletonize degraded images.

5.1 Introduction

Most of the existing algorithms for skeletonization are carried out on binary images [46, 49, 75] where a thresholding on gray-scale images is required. To avoid information loss and extra distortion, an topography-based approach is proposed to apply directly on fuzzy or gray-scale images with tolerance to degradation. First, a convolution by a bell-shaped function is performed to obtain a smooth surface. Second, the ridge points are extracted by rule-based topographic analysis of the structure. Third, a membership function is assigned to ridge points with values indicating the degrees of membership with respect to the skeleton of an object. Finally, the significant ridge points are linked to form strokes of skeleton, and the clues of eigenvalue variation are used to deal with degradation and preserve connectivity. Experimental results show that our algorithm can reduce the deformation of junction points and correctly extract the whole skeleton although a character is broken into pieces. For some characters merged together, the breaking candidates can be easily located by searching for the saddle points. A pruning algorithm is then applied on each breaking position. At last, a multiple context confirmation can be applied to increase the reliability of breaking hypotheses.

Our approach for skeletonization, based on topography, can directly deal with the fuzzy character prototypes as well as gray-scale images. The approach competes favorably with the one by Wang and Pavlidis [89] for the following advantages:

(1) A fuzzy membership is assigned for each ridge point to indicate the degree of membership with respect to the skeleton.

(2) A precise location of the ridge point is sought on each crack of a pixel.

(3) The resulting skeleton is represented by a set of skeletal strokes which can be traced easily by their eigenvectors.

Besides, our approach distinguishes from traditional thinning algorithms [46, 49, 75] in the following aspects:

(1) *Object distortion avoidance*. Since our algorithm applies directly on fuzzy images without an α-cut, information loss and object distortion can be prevented. For example, a fuzzy image shown in Fig. 5.1(a) apparently indicates a horizontal line, where 4, 5 and # denote memberships of 0.4, 0.5 and 1, respectively. If an α-cut with $\alpha = 0.5$ is applied, the binary image shown in Fig. 5.1(b) is quite noisy compared to the original.

(a) (b)

Figure 5.1 A fuzzy image and the corresponding binary image by an α-cut with $\alpha = 0.5$.

(2) *Less deformation on junctions of a skeleton.* Conventional algorithms usually produce a deformed skeleton around junctions. Some algorithm adds knowledge rules to reduce the deformation [60]. However, the knowledge-based approach is quite heuristic. Our topography-based model produces less deformed skeletons along junctions.

(3) *Separation of the skeleton of merged characters.* The skeleton of merged characters by our method are the union of skeletons of individual characters and bridge strokes. The possible breaking positions can be easily located based on topographic features.

(4) *Linkage of skeletons of broken characters.* Our algorithm can link broken components of a character together in a reasonable way to form a complete skeleton.

The remainder of this chapter is organized as follows. Section 5.4 introduces the topographic structure analysis. Section 5.5 presents the transformation of a fuzzy image into a smooth function. Section 5.6 describes the extraction of ridge lines and skeleton. Section 5.7 presents the procedures for handling degraded character images. Section 5.8 provides experimental results.

5.2 Topographic Structure Analysis

5.2.1 Directional Derivatives and Eigenvectors

In this section, we review three-dimensional geometry [60, 82] and introduce the mathematical notations used. Let $g(x, y)$ denote the intensity function of an image in Cartesian coordinates. A three-dimensional surface can be constructed if the function value $g(x, y)$ is viewed as z-axis. The gradient of $g(x, y)$, ∇g, is a vector field such that

$$\nabla g = \frac{\partial g}{\partial x}\mathbf{i} + \frac{\partial g}{\partial y}\mathbf{j}, \tag{5.1}$$

where $\mathbf{i}$ and $\mathbf{j}$ are unit vectors in x- and y-axes, respectively. Note that ∇g points in the direction in which $g(x, y)$ increases at the maximum rate. Let $\mathbf{u}$ be a unit vector of orientation θ counterclockwise from x-axis, i.e., $\mathbf{u} = (\cos\theta, \sin\theta)$.

The directional derivative of $g(x, y)$ in the direction of $\mathbf{u}$ is defined as

$$g'_{\mathbf{u}} = \nabla g \cdot \mathbf{u} = \frac{\partial g}{\partial x}\cos\theta + \frac{\partial g}{\partial y}\sin\theta. \tag{5.2}$$

The second derivative of $g(x, y)$ in the direction of $\mathbf{u}$ is defined as

$$g''_{\mathbf{u}} = \mathbf{u}\mathbf{H}\mathbf{u}^T = g_{xx}\cos^2\theta + 2g_{xy}\sin\theta\cos\theta + g_{yy}\sin^2\theta, \tag{5.3}$$

where g_{xx}, g_{xy}, and g_{yy} denote $\frac{\partial^2 g}{\partial x^2}$, $\frac{\partial^2 g}{\partial x \partial y}$, and $\frac{\partial^2 g}{\partial y^2}$, respectively, and $\mathbf{H}$ is a Hessian

matrix defined as

$$\mathbf{H} = \begin{bmatrix} g_{xx} & g_{xy} \\ g_{xy} & g_{yy} \end{bmatrix}.$$

Since the second derivative of $g(x, y)$ has the form of $\mathbf{uHu}^T$, the eigenvalues of the Hessian matrix are the extrema of the second directional derivative and the eigenvectors are the directions in which the second directional derivative is extremized. Furthermore, the two eigenvectors are orthogonal because $\mathbf{H}$ is symmetric. Therefore, the eigenvectors can be obtained by setting $\frac{dg''_{\mathbf{u}}}{d\theta} = 0$. That is

$$\begin{aligned} \frac{dg''_{\mathbf{u}}}{d\theta} &= -2g_{xx}\cos\theta\sin\theta + 2g_{xy}(\cos^2\theta - \sin^2\theta) + 2g_{yy}\sin\theta\cos\theta \\ &= -g_{xx}\sin 2\theta + 2g_{xy}\cos 2\theta + g_{yy}\sin 2\theta = 0. \end{aligned}$$

We have

$$\tan 2\theta = \frac{2g_{xy}}{g_{xx} - g_{yy}}. \tag{5.4}$$

Two solutions can be obtained.

$$\theta_1 = \frac{1}{2}\tan^{-1}\frac{2g_{xy}}{g_{xx} - g_{yy}}, \quad \theta_2 = \frac{1}{2}\tan^{-1}\frac{2g_{xy}}{g_{xx} - g_{yy}} + \frac{\pi}{2}.$$

Let $\mathbf{w}_1$ and $\mathbf{w}_2$ denote the eigenvectors.

$$\mathbf{w}_1 = (\cos\theta_1, \sin\theta_1), \quad \mathbf{w}_2 = (\cos\theta_2, \sin\theta_2).$$

Replacing $\mathbf{u}$ in eq. (5.3) by $\mathbf{w}_1$ and $\mathbf{w}_2$, we obtain the two eigenvalues λ_1 and λ_2, respectively.

$$\lambda_1 = g''_{\mathbf{w}_1} = \frac{g_{xx} + g_{yy} + \sqrt{(g_{xx} - g_{yy})^2 + 4g_{xy}^2}}{2}, \quad \lambda_2 = g''_{\mathbf{w}_2} = \frac{g_{xx} + g_{yy} - \sqrt{(g_{xx} - g_{yy})^2 + 4g_{xy}^2}}{2}.$$

Note that from eq. (5.4), if $g_{xy} = 0$ and $g_{xx} - g_{yy} \neq 0$, the two eigenvectors are respectively in the horizontal and vertical directions and the corresponding eigenvalues are g_{xx} and g_{yy}. If $g_{xx} - g_{yy} = 0$ and $g_{xy} \neq 0$, then $\theta = \pm \frac{\pi}{4}$. If $g_{xx} - g_{yy} = 0$ and $g_{xy} = 0$, then the eigenvectors do not exist.

5.2.2 Mathematical Topographic Features

Topographic features of an image $g(x, y)$ can be derived from its gradient and the eigenvalues and eigenvectors of the Hessian matrix [31]. They can be categorized as follows.

(a) Peak: A peak is a point being a local maximum in all directions. It can be formulated as

$$||\nabla g|| = 0, \quad \lambda_1 < 0, \quad \lambda_2 < 0.$$

(b) Pit: A pit is a point being a local minimum in all directions. It can be derived from

$$||\nabla g|| = 0, \quad \lambda_1 > 0, \quad \lambda_2 > 0.$$

(c) Ridge: A ridge occurs when a local maximum exists in one of the eigenvectors; while the other is neither a local maximum nor a local minimum. Mathematically, a point is called a ridge if it satisfies

$$||\nabla g|| = 0, \quad \lambda_1 < 0, \quad \lambda_2 = 0,$$

or

$$||\nabla g|| = 0, \quad \lambda_1 = 0, \quad \lambda_2 < 0,$$

or

$$||\nabla g|| \neq 0, \quad \lambda_1 < 0, \quad \nabla g \cdot \mathbf{w}_1 = 0,$$

or

$$||\nabla g|| \neq 0, \quad \lambda_2 < 0, \quad \nabla g \cdot \mathbf{w}_2 = 0.$$

(d) Ravine: A ravine is analogous to a ridge except that a local minimum exists in one of the eigenvectors. Therefore, it satisfies

$$||\nabla g|| = 0, \quad \lambda_1 > 0, \quad \lambda_2 = 0,$$

or

$$||\nabla g|| = 0, \quad \lambda_1 = 0, \quad \lambda_2 > 0,$$

or

$$||\nabla g|| \neq 0, \quad \lambda_1 > 0, \quad \nabla g \cdot \mathbf{w}_1 = 0,$$

or

$$||\nabla g|| \neq 0, \quad \lambda_2 > 0, \quad \nabla g \cdot \mathbf{w}_2 = 0.$$

(e) Saddle: A saddle occurs when a local maximum and a local minimum exist respectively in two eigenvectors. It can be formulated as

$$||\nabla g|| = 0, \quad \lambda_1 \lambda_2 < 0.$$

(f) Flat: A point is called a flat if it satisfies

$$||\nabla g|| = 0, \quad \lambda_1 = 0, \quad \lambda_2 = 0.$$

(g) Hillside: A hillside is a point which does not belong in the preceding categories.

5.2.3 Ridge Lines and Skeleton

The set of ridge lines is defined as consisting of peak, ridge, and saddle points for the extraction of skeleton in an image. From the mathematical formulae of the preceding section, we can easily summarize that the ridge lines of an image $g(x, y)$ consist of the points which satisfy the conditions below. If $||\nabla g|| \neq 0$, then (1) $\lambda_1 < 0$ and $\nabla g \cdot \mathbf{w}_1 = 0$, or (2) $\lambda_2 < 0$ and $\nabla g \cdot \mathbf{w}_2 = 0$. If $||\nabla g|| = 0$, then $\lambda_1 < 0$ or $\lambda_2 < 0$. Note that $||\nabla g|| = 0$ implies $\nabla g \cdot \mathbf{w}_i = 0$, for $i = 1, 2$. Therefore, the conditions can be further simplified as

$$\lambda_i < 0, \quad \nabla g \cdot \mathbf{w}_i = 0, \qquad \text{for } i = 1 \text{ or } 2. \tag{5.5}$$

The concept of fuzzy logic is then applied to represent the membership grade of ridge lines belonging to the skeleton.

Proposition: The membership grade of a ridge point, denoted as $\nu(x, y)$, represents its strength pertaining to the skeleton of an object in the direction perpendicular to $\mathbf{w}_i$ and is defined as

$$\nu(x, y) = \frac{|\lambda_i|}{|\lambda_i| + |\lambda_j|}, \tag{5.6}$$

where λ_i and $\mathbf{w}_i$ are the eigenvalue and eigenvector satisfying the condition in eq. (5.5) and λ_j is the other eigenvalue.

A ridge line is called *significant* if it receives a high membership grade; otherwise, it is called *less-significant*. The significant ridges constitute the skeleton, and the less-significant ridges are discarded except those are bridges between significant ridges.

Example: Let a hemi-elliptical surface be defined as

$$g(x, y) = \begin{cases} a\sqrt{1 - \dfrac{x^2}{a^2} - \dfrac{y^2}{b^2}} & \text{if } \dfrac{x^2}{a^2} + \dfrac{y^2}{b^2} \le 1 \\ 0 & \text{otherwise,} \end{cases} \tag{5.7}$$

where a and b are constants and $a < b$. The hemi-elliptical surface is shown in Fig. 5.2(a). Let $\alpha = 1 - \dfrac{x^2}{a^2} - \dfrac{y^2}{b^2}$. The first and second partial derivatives of g with respect to x and y can be calculated as

$$\frac{\partial g}{\partial x} = -\frac{x}{a\sqrt{\alpha}}, \tag{5.8}$$

$$\frac{\partial g}{\partial y} = -\frac{ay}{b^2\sqrt{\alpha}}, \tag{5.9}$$

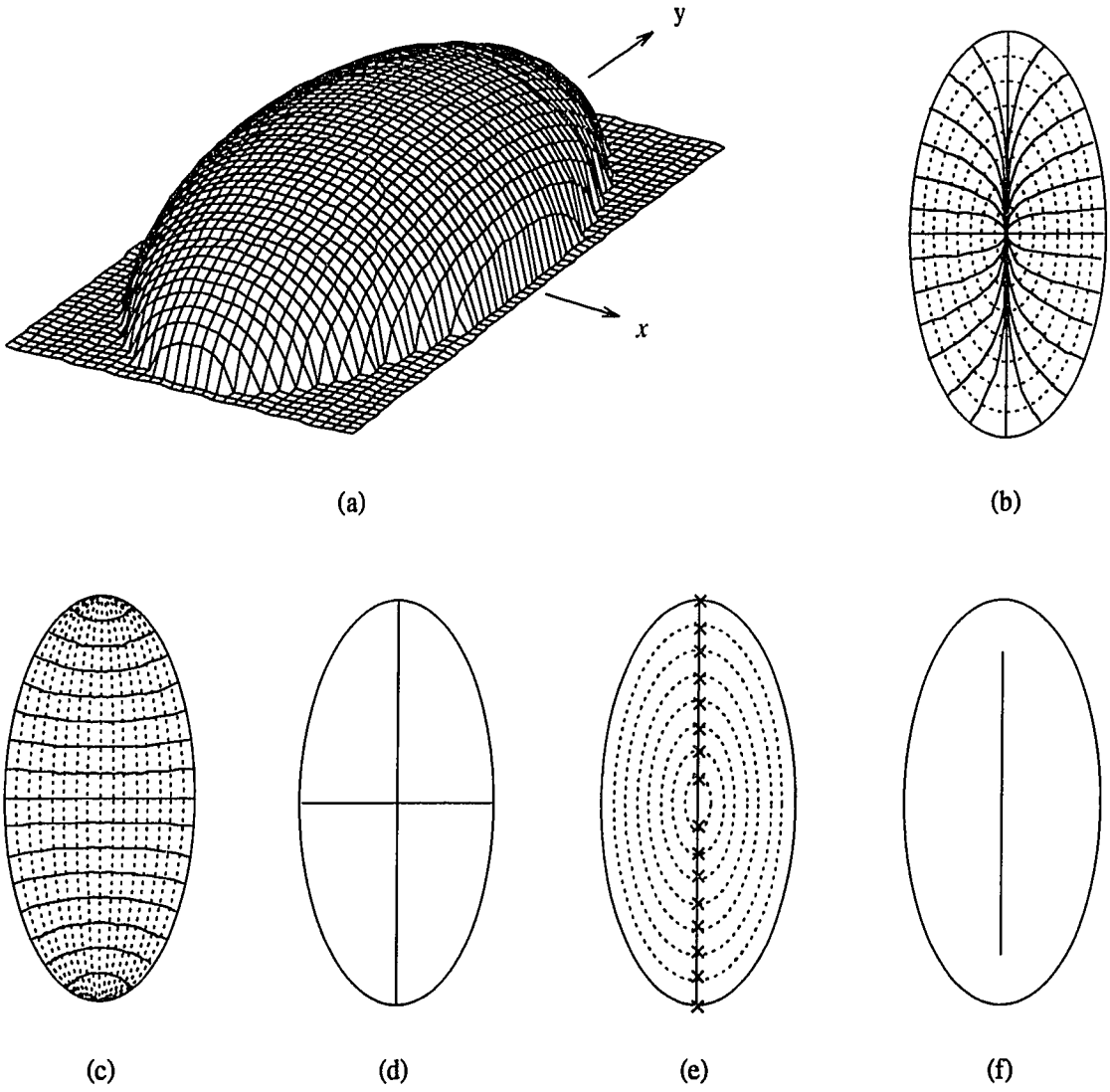


Figure 5.2 (a) a hemi-elliptical surface, (b) the gradient stream flows and the contour lines, (c) the stream flows of the eigenvectors, (d) the ridge lines based on eq. (5.5), (e) the local maximum of curvatures in each contour line, (f) the exact ridge of the hemi-elliptical surface by fuzzy logic.

$$\frac{\partial^2 g}{\partial x^2} = -\frac{1}{a}\alpha^{-1/2} - \frac{1}{a^3}x^2\alpha^{-3/2}, \tag{5.10}$$

$$\frac{\partial^2 g}{\partial x \partial y} = -\frac{1}{ab^2}xy\alpha^{-3/2}, \tag{5.11}$$

$$\frac{\partial^2 g}{\partial y^2} = -\frac{a}{b^2}\alpha^{-1/2} - \frac{a}{b^4}y^2\alpha^{-3/2}. \tag{5.12}$$

The gradient, eigenvalues, and eigenvectors can be easily computed using the equations in Section 5.4.1. The stream flow of gradient is shown in Fig. 5.2(b), where dotted lines represent the contour lines of $g(x, y)$, which are orthogonal to the gradient. The stream flow of the eigenvectors is shown in Fig. 5.2(c), where dotted lines represent the minor eigenvectors. As observed from Figs. 5.2(b) and 5.2(c), the gradient is perpendicular to one of the eigenvectors, i.e. $\nabla g \cdot \mathbf{w}_i = 0$, for $i = 1, 2$, along x- and y-axes. Following eq. (5.5), we can extract two ridge lines as shown in Fig. 5.2(d). However, the two ridge lines as being the skeleton of an ellipse are unintuitive to human's perception. Instead, a ridge line along the major axis should be observed as the skeleton. This can be obtained by selecting the local maximum of curvatures at each contour line, i.e., $g(x, y) = c$, where c is a constant in the range of $[0, g_{\max}]$, as shown in Fig. 5.2(e). However, its computation is costly.

Now, we adopt eq. (5.6) for computing fuzzy membership grades. Since that $\frac{\partial^2 g}{\partial x \partial y} = 0$ occurs at the two ridge lines, the two eigenvalues are $\frac{\partial^2 g}{\partial x^2}$ and $\frac{\partial^2 g}{\partial y^2}$. The membership function along x-axis is

$$\nu_{y=0} = \frac{\partial^2 g/\partial y^2}{\partial^2 g/\partial x^2 + \partial^2 g/\partial y^2} = \frac{a^2 - x^2}{a^2 + b^2 - x^2}, \quad -a \le x \le a, \tag{5.13}$$

and the membership function along y-axis is

$$\nu_{x=0} = \frac{\partial^2 g/\partial x^2}{\partial^2 g/\partial x^2 + \partial^2 g/\partial y^2} = \frac{b^2 - y^2}{a^2 + b^2 - y^2}, \quad -b \le y \le b. \tag{5.14}$$

We observed that the values of eq. (5.13) are always smaller than 0.5 because $a < b$. If an α-cut at 0.5 is applied, then the x-axis (i.e. $y = 0$) is discarded. Note that the two end points of the ridge line on $x = 0$ can be solved from $\frac{b^2 - y^2}{a^2 + b^2 - y^2} = 0.5$. That is $(0, \sqrt{b^2 - a^2})$ and $(0, -\sqrt{b^2 - a^2})$ which are the two foci of the ellipse, $\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$. The skeleton obtained from our fuzzy model is shown in Fig. 5.2(f). Furthermore, we should note that if $a = b$, eq. (5.7) becomes a hemi-spherical surface. The membership grades are smaller than 0.5 everywhere except at the origin which is equal to 0.5. Therefore, the skeleton of a circle vanishes to the center point.

5.3 Transformation of a Fuzzy Image

5.3.1 Transformation Function

Let $f(x, y)$ denote a fuzzy image. A point spread function $h(x, y)$ used for the smooth transformation is defined as

$$h(x, y) = C(x)C(y), \tag{5.15}$$

where

$$C(x) = \begin{cases} \frac{1}{\sigma}\cos^2(\frac{\pi x}{2\sigma}) & \text{if } |x| \le \sigma \\ 0 & \text{otherwise,} \end{cases}$$

and σ is a constant. The image $f(x, y)$ is convolved with the function $h(x, y)$ and the result is denoted by $g(x, y)$.

$$g(x, y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(u, v)h(x - u, y - v)dudv.$$

Assume that

$$f(x, y) = \begin{cases} \mu & \text{if } x_1 \le x \le x_2,\ y_1 \le y \le y_2 \\ 0 & \text{otherwise,} \end{cases}$$

where μ is a constant, $x_2 - x_1 \le 2\sigma$, and $y_2 - y_1 \le 2\sigma$. Therefore, we obtain

$$g(x, y) = \mu \int_{y_1}^{y_2} \int_{x_1}^{x_2} C(u-x)C(v-y)dudv$$

$$= \begin{cases} \mu\left[\frac{1}{2\pi}\sin\frac{(u-x)\pi}{\sigma} + \frac{u}{2\sigma}\right]_{u_1}^{u_2}\left[\frac{1}{2\pi}\sin\frac{(v-y)\pi}{\sigma} + \frac{v}{2\sigma}\right]_{v_1}^{v_2}, & \text{if } \begin{matrix} x_1 - \sigma \le x \le x_2 + \sigma, \\ y_1 - \sigma \le y \le y_2 + \sigma, \end{matrix} \\ 0, & \text{otherwise,} \end{cases} \quad (5.16)$$

where

$$u_1 = \begin{cases} x - \sigma & \text{if } x > x_1 + \sigma \\ x_1 & \text{otherwise,} \end{cases} \quad \text{and} \quad u_2 = \begin{cases} x + \sigma & \text{if } x < x_2 - \sigma \\ x_2 & \text{otherwise,} \end{cases}$$

$$v_1 = \begin{cases} y - \sigma & \text{if } y > y_1 + \sigma \\ y_1 & \text{otherwise,} \end{cases} \quad \text{and} \quad v_2 = \begin{cases} y + \sigma & \text{if } y < y_2 - \sigma \\ y_2 & \text{otherwise.} \end{cases}$$

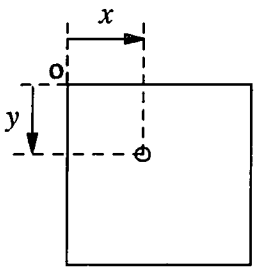


Figure 5.3 The coordinates within a pixel (i, j).

We select σ to be a positive integer. As shown in Fig. 5.3, let the upper left corner of a pixel be the origin. For the coordinates (x, y) within the pixel (i, j), where $0 \le x, y \le 1$, the convolution of $f(x, y)$ and $h(x, y)$ is

$$g_{ij}(x, y) = \sum_{m=-\sigma}^{\sigma}\sum_{n=-\sigma}^{\sigma} \mu_{i+m,j+n}\left[\frac{1}{2\pi}\sin\frac{(u-x)\pi}{\sigma} + \frac{u}{2\sigma}\right]_{u_1}^{u_2}\left[\frac{1}{2\pi}\sin\frac{(v-y)\pi}{\sigma} + \frac{v}{2\sigma}\right]_{v_1}^{v_2}, \quad (5.17)$$

where $\mu_{i+m,j+n}$ denotes the membership value of pixel $(i + m, j + n)$, and

$$u_1 = \begin{cases} m + x & \text{if } m = -\sigma \\ m & \text{otherwise,} \end{cases} \quad \text{and} \quad u_2 = \begin{cases} m + x & \text{if } m = \sigma \\ m + 1 & \text{otherwise,} \end{cases}$$

$$v_1 = \begin{cases} n + y & \text{if } n = -\sigma \\ n & \text{otherwise,} \end{cases} \quad \text{and} \quad v_2 = \begin{cases} n + y & \text{if } n = \sigma \\ n + 1 & \text{otherwise.} \end{cases}$$

Likewise, the first and second derivatives of $g_{ij}(x, y)$ can be derived as follows:

$$\frac{\partial g}{\partial x} = \sum_{m=-\sigma}^{\sigma} \sum_{n=-\sigma}^{\sigma} \mu_{i+m,j+n} \left[-\frac{1}{2\sigma} \cos \frac{(u-x)\pi}{\sigma} \right]_{u_1}^{u_2} \left[\frac{1}{2\pi} \sin \frac{(v-y)\pi}{\sigma} + \frac{v}{2\sigma} \right]_{v_1}^{v_2} \tag{5.18}$$

$$\frac{\partial g}{\partial y} = \sum_{m=-\sigma}^{\sigma} \sum_{n=-\sigma}^{\sigma} \mu_{i+m,j+n} \left[\frac{1}{2\pi} \sin \frac{(u-x)\pi}{\sigma} + \frac{u}{2\sigma} \right]_{u_1}^{u_2} \left[-\frac{1}{2\sigma} \cos \frac{(v-y)\pi}{\sigma} \right]_{v_1}^{v_2} \tag{5.19}$$

$$\frac{\partial^2 g}{\partial x^2} = \sum_{m=-\sigma}^{\sigma} \sum_{n=-\sigma}^{\sigma} \mu_{i+m,j+n} \left[-\frac{\pi}{2\sigma^2} \sin \frac{(u-x)\pi}{\sigma} \right]_{u_1}^{u_2} \left[\frac{1}{2\pi} \sin \frac{(v-y)\pi}{\sigma} + \frac{v}{2\sigma} \right]_{v_1}^{v_2} \tag{5.20}$$

$$\frac{\partial^2 g}{\partial y^2} = \sum_{m=-\sigma}^{\sigma} \sum_{n=-\sigma}^{\sigma} \mu_{i+m,j+n} \left[\frac{1}{2\pi} \sin \frac{(u-x)\pi}{\sigma} + \frac{u}{2\sigma} \right]_{u_1}^{u_2} \left[-\frac{\pi}{2\sigma^2} \sin \frac{(v-y)\pi}{\sigma} \right]_{v_1}^{v_2} \tag{5.21}$$

$$\frac{\partial^2 g}{\partial x \partial y} = \sum_{m=-\sigma}^{\sigma} \sum_{n=-\sigma}^{\sigma} \mu_{i+m,j+n} \left[\frac{1}{2\sigma} \cos \frac{(u-x)\pi}{\sigma} \right]_{u_1}^{u_2} \left[\frac{1}{2\sigma} \cos \frac{(v-y)\pi}{\sigma} \right]_{v_1}^{v_2} \tag{5.22}$$

5.3.2 Matrix Representation of the Transformation Function

By using eqs. (5.17)–(5.22), we have to perform a convolution for each x and y. The computation is quite expensive. A matrix representation can be achieved as follows. Let $\phi(x) = \frac{1}{2\pi} \sin \frac{x\pi}{\sigma} + \frac{x}{2\sigma}$ and substitute it into eq. (5.17). For $m = -\sigma$, we have

$$\phi(u-x)\Big|_{u_1}^{u_2} = \frac{1}{2\pi} \sin \frac{(-\sigma+1)\pi}{\sigma} \cos \frac{x\pi}{\sigma} - \frac{1}{2\pi} \cos \frac{(-\sigma+1)\pi}{\sigma} \sin \frac{x\pi}{\sigma} - \frac{x}{2\sigma} + \frac{1}{2\sigma}.$$

For $m = \sigma$, we have

$$\phi(u-x)\Big|_{u_1}^{u_2} = -\frac{1}{2\pi} \sin \frac{x\pi}{\sigma} + \frac{x}{2\sigma}.$$

Otherwise,

$$\phi(u-x)\Big|_{u_1}^{u_2} = \frac{1}{2\pi}\left(\sin\frac{(m+1)\pi}{\sigma} - \sin\frac{m\pi}{\sigma}\right)\cos\frac{x\pi}{\sigma} - \frac{1}{2\pi}\left(\cos\frac{(m+1)\pi}{\sigma} - \cos\frac{m\pi}{\sigma}\right)\sin\frac{x\pi}{\sigma}$$

$$+\frac{1}{2\sigma}.$$

Therefore, $\phi(u-x)\Big|_{u_1}^{u_2}$ can be represented in terms of $\cos\frac{x\pi}{\sigma}$, $\sin\frac{x\pi}{\sigma}$, and x. Similarly, $\phi(v-y)\Big|_{v_1}^{v_2}$ can be represented in terms of $\cos\frac{y\pi}{\sigma}$, $\sin\frac{y\pi}{\sigma}$, and y. By using the matrix representation, we have

$$g(x,y) = \mathbf{v}(x)^T\mathbf{K}\mathbf{v}(y) \tag{5.23}$$

$$\frac{\partial g}{\partial x} = \mathbf{v}(x)^T\mathbf{K}_x\mathbf{v}(y) \tag{5.24}$$

$$\frac{\partial g}{\partial y} = \mathbf{v}(x)^T\mathbf{K}_y\mathbf{v}(y) \tag{5.25}$$

$$\frac{\partial^2 g}{\partial x^2} = \mathbf{v}(x)^T\mathbf{K}_{xx}\mathbf{v}(y) \tag{5.26}$$

$$\frac{\partial^2 g}{\partial y^2} = \mathbf{v}(x)^T\mathbf{K}_{yy}\mathbf{v}(y) \tag{5.27}$$

$$\frac{\partial^2 g}{\partial x\partial y} = \mathbf{v}(x)^T\mathbf{K}_{xy}\mathbf{v}(y), \tag{5.28}$$

where T denotes the transpose of a vector, and $\mathbf{v}(x)$ and $\mathbf{v}(y)$ are vectors defined as

$$\mathbf{v}(x) = \begin{bmatrix} \cos\frac{x\pi}{\sigma} \\ \sin\frac{x\pi}{\sigma} \\ x \\ 1 \end{bmatrix}.$$

Note that $\mathbf{K}$, $\mathbf{K}_x$, $\mathbf{K}_y$, $\mathbf{K}_{xx}$, $\mathbf{K}_{yy}$, and $\mathbf{K}_{xy}$ are 4×4 matrices. Let $\mathbf{K}$

$$\mathbf{K} = \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ k_{31} & k_{32} & k_{33} & k_{34} \\ k_{41} & k_{42} & k_{43} & k_{43} \end{bmatrix},$$

where k_{ij}'s can be computed by summing up the corresponding terms in the convolution. Therefore, we have

$$\mathbf{K}_x = \begin{bmatrix} \frac{\pi}{\sigma} k_{21} & \frac{\pi}{\sigma} k_{22} & \frac{\pi}{\sigma} k_{23} & \frac{\pi}{\sigma} k_{24} \\ -\frac{\pi}{\sigma} k_{11} & -\frac{\pi}{\sigma} k_{12} & -\frac{\pi}{\sigma} k_{13} & -\frac{\pi}{\sigma} k_{14} \\ 0 & 0 & 0 & 0 \\ k_{31} & k_{32} & k_{33} & k_{34} \end{bmatrix}$$

$$\mathbf{K}_y = \begin{bmatrix} \frac{\pi}{\sigma} k_{12} & -\frac{\pi}{\sigma} k_{11} & 0 & k_{13} \\ \frac{\pi}{\sigma} k_{22} & -\frac{\pi}{\sigma} k_{21} & 0 & k_{23} \\ \frac{\pi}{\sigma} k_{32} & -\frac{\pi}{\sigma} k_{31} & 0 & k_{33} \\ \frac{\pi}{\sigma} k_{42} & -\frac{\pi}{\sigma} k_{41} & 0 & k_{43} \end{bmatrix}$$

$$\mathbf{K}_{xx} = -\frac{\pi^2}{\sigma^2} \begin{bmatrix} k_{11} & k_{12} & k_{13} & k_{14} \\ k_{21} & k_{22} & k_{23} & k_{24} \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}$$

$$\mathbf{K}_{yy} = -\frac{\pi^2}{\sigma^2} \begin{bmatrix} k_{11} & k_{12} & 0 & 0 \\ k_{21} & k_{22} & 0 & 0 \\ k_{31} & k_{32} & 0 & 0 \\ k_{41} & k_{42} & 0 & 0 \end{bmatrix}$$

$$\mathbf{K}_{xy} = \begin{bmatrix} \frac{\pi^2}{\sigma^2} k_{22} & -\frac{\pi^2}{\sigma^2} k_{21} & 0 & \frac{\pi}{\sigma} k_{23} \\ -\frac{\pi^2}{\sigma^2} k_{12} & \frac{\pi^2}{\sigma^2} k_{11} & 0 & -\frac{\pi}{\sigma} k_{13} \\ 0 & 0 & 0 & 0 \\ \frac{\pi}{\sigma} k_{32} & -\frac{\pi}{\sigma} k_{31} & 0 & k_{33} \end{bmatrix}$$

It is not practical to search for a ridge line within a pixel. Instead, we check whether a ridge line passes through the boundary of a pixel. For example, in the upper boundary (i.e., $y = 0$), $g(x, y)$ in eq. (5.23) can be simplified as

$$g_{y=0} = \mathbf{v}(x)^T \mathbf{K} \mathbf{v}(0) = \begin{bmatrix} \cos\frac{x\pi}{\sigma} & \sin\frac{x\pi}{\sigma} & x & 1 \end{bmatrix} \begin{bmatrix} k_{11} + k_{14} \\ k_{21} + k_{24} \\ k_{31} + k_{34} \\ k_{41} + k_{44} \end{bmatrix}.$$

Similarly, from eqs. (5.24) to (5.28) we have

$$\frac{\partial g}{\partial x} = \mathbf{v}(x)^T \mathbf{K}_x \mathbf{v}(0) = \mathbf{v}(x)^T \begin{bmatrix} \frac{\pi}{\sigma}(k_{21} + k_{24}) \\ -\frac{\pi}{\sigma}(k_{11} + k_{14}) \\ 0 \\ k_{31} + k_{34} \end{bmatrix}$$

$$\frac{\partial g}{\partial y} = \mathbf{v}(x)^T \mathbf{K}_y \mathbf{v}(0) = \mathbf{v}(x)^T \begin{bmatrix} \frac{\pi}{\sigma} k_{12} + k_{13} \\ \frac{\pi}{\sigma} k_{22} + k_{23} \\ \frac{\pi}{\sigma} k_{32} + k_{33} \\ \frac{\pi}{\sigma} k_{42} + k_{43} \end{bmatrix}$$

$$\frac{\partial^2 g}{\partial x^2} = \mathbf{v}(x)^T \mathbf{K}_{xx} \mathbf{v}(0) = \mathbf{v}(x)^T \left(-\frac{\pi^2}{\sigma^2}\right) \begin{bmatrix} k_{11} + k_{14} \\ k_{21} + k_{24} \\ 0 \\ 0 \end{bmatrix}$$

$$\frac{\partial^2 g}{\partial x \partial y} = \mathbf{v}(x)^T \mathbf{K}_{xy} \mathbf{v}(0) = \mathbf{v}(x)^T \begin{bmatrix} \frac{\pi^2}{\sigma^2} k_{22} + \frac{\pi}{\sigma} k_{23} \\ -\frac{\pi^2}{\sigma^2} k_{12} - \frac{\pi}{\sigma} k_{13} \\ 0 \\ \frac{\pi}{\sigma} k_{32} + k_{33} \end{bmatrix}$$

$$\frac{\partial^2 g}{\partial y^2} = \mathbf{v}(x)^T \mathbf{K}_{yy} \mathbf{v}(0) = \mathbf{v}(x)^T \left(-\frac{\pi^2}{\sigma^2}\right) \begin{bmatrix} k_{11} \\ k_{21} \\ k_{31} \\ k_{41} \end{bmatrix}$$

5.4 Extraction of Ridge Lines and Skeleton

5.4.1 Determination of the Parameter σ

To obtain a smooth function in eq. (5.15), the parameter σ must be determined. From previous discussion σ must be greater than or equal to a half of the maximal width of strokes in an image. Since our image is associated with fuzzy memberships in the range [0, 1], the two-pass distance transformation [69] can be modified as

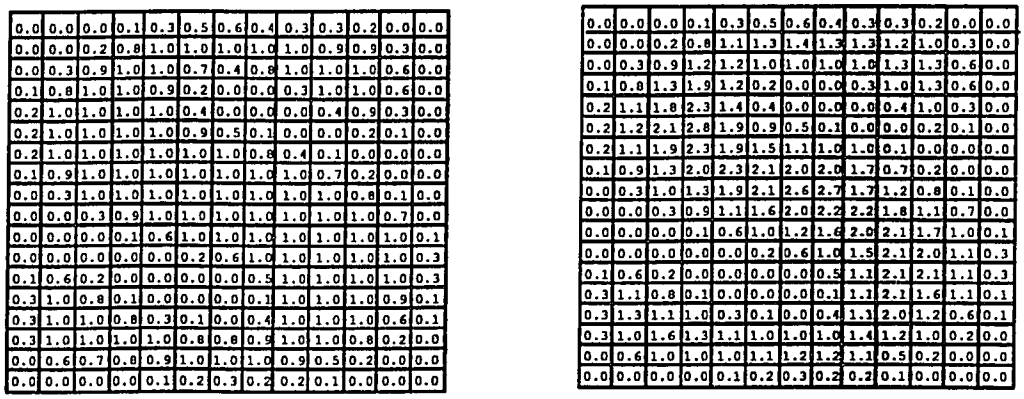
$$\chi'(p) = \begin{cases} \mu_p & \text{if } \mu_p < 1 \\ \min\limits_{q \in N_1} \chi'(q) + 1 & \text{if } \mu_p = 1 \end{cases}$$

$$\chi''(p) = \min_{q \in N_2} [\chi'(p),\ \chi''(q) + 1],$$

where $\chi'(p)$ and $\chi''(p)$ denote the outputs of a pixel $p = (x, y)$ after the first and second scans of an image, respectively. N_1 denotes the set of 8-neighbors that precede p in the raster scan. That is, $N_1 = \{(x-1, y+1), (x, y+1), (x+1, y+1), (x-1, y)\}$. N_2 denotes the remaining 8-neighbors of p. Fig. 5.4 illustrates the result of distance computation of a fuzzy image. The value of σ is therefore selected as the smallest integer which is greater than the maximal distance. For this example, the maximal distance is 2.8, and therefore $\sigma = 3$ is chosen.

5.4.2 Existence of Ridge Points within a Segment

Since $g(x, y)$ is an image after smooth, its gradient and eigenvectors vary continuously with respect to x and y. Each pixel with a nonzero function value has to be examined for the existence of a ridge line passing through it.



0.0	0.0	0.0	0.1	0.3	0.5	0.6	0.4	0.3	0.3	0.2	0.0	0.0
0.0	0.0	0.2	0.8	1.0	1.0	1.0	1.0	1.0	0.9	0.9	0.3	0.0
0.0	0.3	0.9	1.0	1.0	0.7	0.4	0.8	1.0	1.0	1.0	0.6	0.0
0.1	0.8	1.0	1.0	0.9	0.2	0.0	0.0	0.3	1.0	1.0	0.6	0.0
0.2	1.0	1.0	1.0	1.0	0.4	0.0	0.0	0.0	0.4	0.9	0.3	0.0
0.2	1.0	1.0	1.0	1.0	0.9	0.5	0.1	0.0	0.0	0.2	0.1	0.0
0.2	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.4	0.1	0.0	0.0	0.0
0.1	0.9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.2	0.0	0.0
0.0	0.3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.8	0.1	0.0
0.0	0.0	0.3	0.9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.7	0.0
0.0	0.0	0.0	0.1	0.6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	0.1
0.0	0.0	0.0	0.0	0.0	0.2	0.6	1.0	1.0	1.0	1.0	1.0	0.3
0.1	0.6	0.2	0.0	0.0	0.0	0.0	0.0	0.5	1.0	1.0	1.0	0.3
0.3	1.0	0.8	0.1	0.0	0.0	0.0	0.1	1.0	1.0	1.0	0.9	0.1
0.3	1.0	1.0	0.8	0.3	0.1	0.0	0.4	1.0	1.0	1.0	0.6	0.1
0.3	1.0	1.0	1.0	1.0	0.8	0.8	0.9	1.0	1.0	0.8	0.2	0.0
0.0	0.6	0.7	0.8	0.9	1.0	1.0	1.0	0.9	0.5	0.2	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.2	0.3	0.2	0.2	0.1	0.0	0.0	0.0

(a)

0.0	0.0	0.0	0.1	0.3	0.5	0.6	0.4	0.3	0.3	0.2	0.0	0.0
0.0	0.0	0.2	0.8	1.1	1.3	1.4	1.3	1.3	1.2	1.0	0.3	0.0
0.0	0.3	0.9	1.2	1.2	1.0	1.0	1.0	1.0	1.3	1.3	0.6	0.0
0.1	0.8	1.3	1.9	1.2	0.2	0.0	0.0	0.3	1.0	1.3	0.6	0.0
0.2	1.1	1.8	2.3	1.4	0.4	0.0	0.0	0.0	0.4	1.0	0.3	0.0
0.2	1.2	2.1	2.8	1.9	0.9	0.5	0.1	0.0	0.0	0.2	0.1	0.0
0.2	1.1	1.9	2.3	1.9	1.5	1.1	1.0	1.0	0.1	0.0	0.0	0.0
0.1	0.9	1.3	2.0	2.3	2.1	2.0	2.0	1.7	0.7	0.2	0.0	0.0
0.0	0.3	1.0	1.3	1.9	2.1	2.6	2.7	1.7	1.2	0.8	0.1	0.0
0.0	0.0	0.3	0.9	1.1	1.6	2.0	2.2	2.2	1.8	1.1	0.7	0.0
0.0	0.0	0.0	0.1	0.6	1.0	1.2	1.6	2.0	2.1	1.7	1.0	0.1
0.0	0.0	0.0	0.0	0.0	0.2	0.6	1.0	1.5	2.1	2.0	1.1	0.3
0.1	0.6	0.2	0.0	0.0	0.0	0.0	0.0	0.5	1.1	2.1	2.1	1.1	0.3
0.3	1.1	0.8	0.1	0.0	0.0	0.0	0.0	0.1	1.1	2.1	1.6	1.1	0.1
0.3	1.3	1.1	1.0	0.3	0.1	0.0	0.0	0.4	1.3	2.0	1.2	0.6	0.1
0.3	1.0	1.6	1.3	1.1	1.0	1.0	1.0	1.4	1.2	1.0	0.2	0.0
0.0	0.6	1.0	1.0	1.0	1.1	1.2	1.2	1.1	0.5	0.2	0.0	0.0
0.0	0.0	0.0	0.0	0.1	0.2	0.3	0.2	0.2	0.1	0.0	0.0	0.0

(b)

Figure 5.4 An example of a fuzzy image and its chessboard distance transformation.

A pixel is surrounded by 4 cracks (or grid segments) as shown in Fig. 5.3. For each crack, the gradients at two end points are calculated. Any end point must be a skeleton point if the gradient is zero. Otherwise, the eigenvectors and eigenvalues at the end point are calculated and if eq. (5.5) is satisfied, the end point is classified as a ridge point. If both end points are ridge points, they are linked. If only one end point is a ridge point, it is marked. If none is a ridge point, the following algorithm using a linear approximation is performed to determine whether a ridge point exists between the two end points.

First, the gradients at the two end points are checked. Two cases need to be considered.

Case I: The two gradients are in parallel. If they point in the same direction, the whole segment must belong to the hillside and therefore is discarded. If they point in the opposite directions, there exists a point within the segment which has zero-gradient. However, the zero-gradient point could be a ridge or a ravine point. As shown in Fig. 5.5(a), the two gradients are considered as pointing to each other, i.e. $\theta < 90°$, and the zero-gradient point will be a ridge point. Otherwise, as shown in Fig. 5.5(b), the zero-gradient point will be a ravine point.

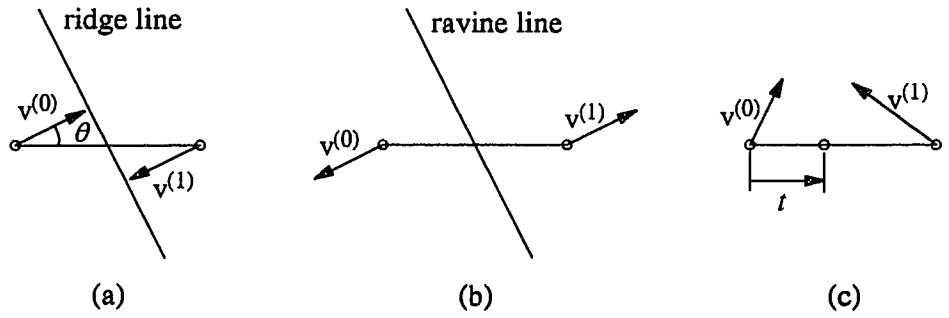


Figure 5.5 (a) Gradients are in parallel and point to each other, (b) gradients are in parallel and point away from each other, (c) gradients are not in parallel.

Case II: The two gradients are not in parallel. In this case, the gradient and eigenvectors are changed smoothly along the segment. In other words, let $\mathbf{v}^{(0)}$ and $\mathbf{v}^{(1)}$ denote the gradients and $\mathbf{w}_i^{(0)}$ and $\mathbf{w}_i^{(1)}$, where $i = 1, 2$, denote the eigenvectors at the two corresponding end points p_0 and p_1. By a linear approximation, the gradient and eigenvectors at $p_t = (1-t)p_0 + tp_1$, as shown in Fig. 5.5(c), can be derived as

$$\mathbf{v}^{(t)} = (1-t)\mathbf{v}^{(0)} + t\mathbf{v}^{(1)}, \quad \mathbf{w}_i^{(t)} = (1-t)\mathbf{w}_i^{(0)} + t\mathbf{w}_i^{(1)},$$

where $0 \le t \le 1$. A ridge point exists within the segment $\overline{p_0 p_1}$ if there exists $0 \le t \le 1$, so that $\mathbf{v}^{(t)}$ and $\mathbf{w}_i^{(t)}$ satisfy eq. (5.5). That is

$$\mathbf{v}^{(t)}\mathbf{w}_i^{(t)} = \left((1-t)\mathbf{v}^{(0)} + t\mathbf{v}^{(1)}\right)\left((1-t)\mathbf{w}_i^{(0)} + t\mathbf{w}_i^{(1)}\right)$$

$$= (1-t)^2\mathbf{v}^{(0)}\mathbf{w}_i^{(0)} + t(1-t)\mathbf{v}^{(0)}\mathbf{w}_i^{(1)} + t(1-t)\mathbf{v}^{(1)}\mathbf{w}_i^{(0)} + t^2\mathbf{v}^{(1)}\mathbf{w}_i^{(1)} = 0 \quad (5.29)$$

Therefore, the two solutions for t can be obtained. If one of the solution is a real number in the range of $0 \le t \le 1$, the algorithm described in the following section is applied to extract the correct ridge points.

5.4.3 Extraction of Ridge Points

Once the existence of ridge points within a segment has been detected, the gradient and eigenvectors at $p_t = (1-t)p_0 + tp_1$ are calculated. If eq. (5.5) is satisfied, point p_t is extracted as a ridge point. Otherwise, the segment $\overline{p_0 p_1}$ is split into two at point p_t. Existence of ridge points is again examined on these two segments, $\overline{p_0 p_t}$ and $\overline{p_t p_1}$, to determine which one the ridge point belongs to. The procedures repeat until a ridge point is extracted. Since the procedures are recursively applied, the calculations of the gradients, eigenvalues and eigenvectors by eqs. (5.18)-(5.22) are expensive. Therefore, a matrix representation described in Section 5.5.2 is applied to reduce computational time. According to our experiments, the average of the number of repetition is about 3 or 4.

5.4.4 Construction of Skeleton

At this stage, the ridge points are linked to form the skeleton of the fuzzy image. First, the significant ridge points are linked according to their positions and directions. Second, if the less-significant ridge points are isolated or they are connected to the end of a ridge line, they are discarded. The less-significant ridge points are considered important only when they play a role as a bridge between ridge lines. In essence, they appear around the significant ridge points such as corners or junctions. Besides, they may play an important role in broken characters to be linked.

5.5 Skeletonization for Degraded Character Images

Two problems exist in degraded character images: one, a single character is broken into multiple pieces, and the other, multiple characters are merged into a single connected component. In this section, we present our strategy in dealing with degraded images to correctly extract skeletons.

5.5.1 Broken Characters

When dealing with broken characters, two approachs can be used. One is to group the components into a single image and treat it as an entity for subsequent processes. The other is to treat the components independently. Each of them is passed through a classifier and then is combined together according to knowledge rules. However, the rules are quite heuristic and case-dependent. Therefore, the first approach makes the recognition phase simpler.

To illustrate how our algorithm works on broken characters, consider the broken pieces shown in Fig. 5.6(a). Its three-dimensional figure is shown in Fig. 5.6(b) if z-axis represents the gray level. Since our algorithm performs a convolution in the preprocessing, the two segments can be linked. The smoothed image is shown in Fig. 5.6(c), where the dark line represents the skeleton extracted using our ridge detection.

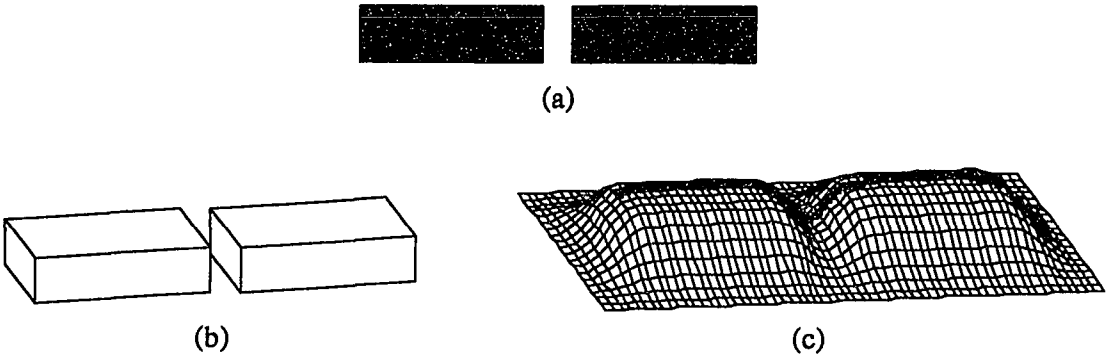


Figure 5.6 (a) A broken segment (b) its corresponding 3-D image (c) the smoothed image and its skeleton.

It was observed from Section 5.5 that our algorithm can link the gap width up to 2σ. Since a half of the stroke width is selected, the components can be linked with the maximal distance of the stroke width. This will be sufficient for most of the broken characters.

5.5.2 Merged Characters

Merged characters may occur due to low resolutions or smearing ink of printing devices. Most of the merged character segmentations [25, 41, 54] used vertical projection profile, skeletal features, and contour features. A splitting algorithm based on the sharp change in the vertical projection profile was proposed by Kahan *et al.* [41], where a maximum of the second difference in the vertical projection profile was selected as the possible breaking position. However, this method is limited on certain joins (e.g. serif joins and double-o joins). Some merged characters, such as those written in the Italic font, will be difficult to find. Tsujimoto and Asada [85] proposed a breaking cost, which is a variation of vertical projection profile, as a metric to nominate breaking positions. Break cost evaluates the degree of contact for each pair of adjacent columns. However, it has the same problem as Kahan's algorithm. Fujisawa *et al.* [25] proposed an approach using contour analysis. However, the breaking position may not be accurate, and some joins, such as serif join, may not be detected. Another splitting algorithm using skeletal features was proposed by Mitchell and Gillies [54]. The splits roughly sort the stroke segments of the form into two groups representing the left and right digits. However, it is restricted for merged characters consisting of two characters only. Besides, the desired sorting may result in stroke segments that are not in either group. Therefore, some enumeration heuristics incorporating the handwriting properties are used to limit the split number.

Our splitting algorithm, applied on the skeleton obtained in the preceding section, uses the gradient and eigenvalues to detect the breaking candidates. The algorithm consists of three stages:

(1) Searching for possible splitting positions.

(2) Pruning the linking strokes.

(3) Recognizing individual characters and context confirmation.

5.5.2.1 Searching for Possible Splitting Positions Essentially, joined positions of merged characters usually have less-pixel connections or lower membership values than the positions within a stroke. Therefore, after performing a convolution, the smoothed membership values at joined positions are relatively low. However, thin strokes may also have low membership values. The difference between the thin stroke and the joined position is that the membership values change smoothly within the thin stroke, while they change sharply around the joined positions. Thus, the joined position can be characterized as the local minimum of membership values with sharp curvature in the tangent of the ridge line. In other words, a joined position is a saddle point with high eigenvalues. Therefore, the first step to segment merged characters is to locate the saddle points.

The procedure for searching for saddle positions is straight forward. Since each stroke is composed of a set of ridge points, we simply trace each point within a stroke sequentially. Based on the features of a saddle point, the gradient of each point is checked. A saddle point is classified if its gradient is zero and its eigenvalue with respect to the eigenvector in parallel to the ridge line is negative. In most cases, the saddle points are not exactly at ridge points. Instead, a saddle point may exist within a ridge line segment. We simply check the gradients of the two end points of each ridge line segment. A saddle point exists within a ridge line if both gradients on the end points point outward. Mathematically speaking, It satisfies the following two conditions: (1) $\nabla g_1 \cdot \nabla g_2 \leq 0$ and (2) $\mathbf{w}_1 > 0$ and $\mathbf{w}_2 > 0$, where ∇g_1, ∇g_2 and $\mathbf{w}_1$, $\mathbf{w}_2$ denote the gradients and eigenvalues with respect to the eigenvector in parallel to the ridge line of the two end points. For simplicity, the larger eigenvalue of the two is selected as the eigenvalue of the saddle point.

Once the saddle points are located, the ones whose eigenvalues exceed a threshold are selected as possible breaking positions. The possible breaking positions are then ranked according to their eigenvalues and the membership values of the pixels where the saddle points are located. The membership values reflect the breaking cost of the breaking pixels. Therefore, the points with lower membership values are ranked as higher

possibilities. If the membership values of two points are identical, the eigenvalues are then checked. The eigenvalues reflect the breaking cost in the neighborhood of the breaking candidates. The higher eigenvalue represents the lower breaking cost. Therefore, the point with the higher eigenvalue is ranked as a higher possibility. After ranking, the splitting algorithm is then applied and the reliability is confirmed from the highest ranked candidate.

5.5.2.2 Pruning the Linking Strokes Kahan *et al.* [41] characterized the most frequent types of joins into two categories: serif joins and double-o joins, as shown in Figs. 5.7(a) and 16(b). When applying our skeletonization algorithm on a merged character with double-o join, there exists an extra stoke at the join position as shown by dotted line in Fig. 5.7(e). This extra stroke should not be a part of any character and must be entirely removed. However, the extra skeleton line for the serif join is a part of a stroke, as shown in Fig. 5.7(d), and only needs partially pruned. Other joins may have extra skeleton lines as a part of one character. For example, the extra skeleton line as shown in Fig. 5.7(f) is a part of one of the strokes belonging to character "y". In this example, the extra skeleton in the left-hand side of the saddle point needs to be removed, but the skeleton line in the right-hand side can only be partially pruned.

Our criteria for pruning the extra skeleton line is described below. First, the stroke which includes a breaking point is split into two at the saddle point. Each half is then checked independently. If the segment is short and the other end of the segment is a "T" junction, then the join must be an "O" type join. "O" type join can be double-o join or a convex contour osculating to another stoke (e.g. Fig. 5.7(e)). Therefore, the segment must be entirely removed. The judgement of the segment which is long or short depends on the value of σ. If the segment is long or the other end of the segment is not a "T" junction, then the extra segment is not an "O" type join. In this case, the segment contains an actual skeleton line and part of the extra line. To prune the extra skeleton, we

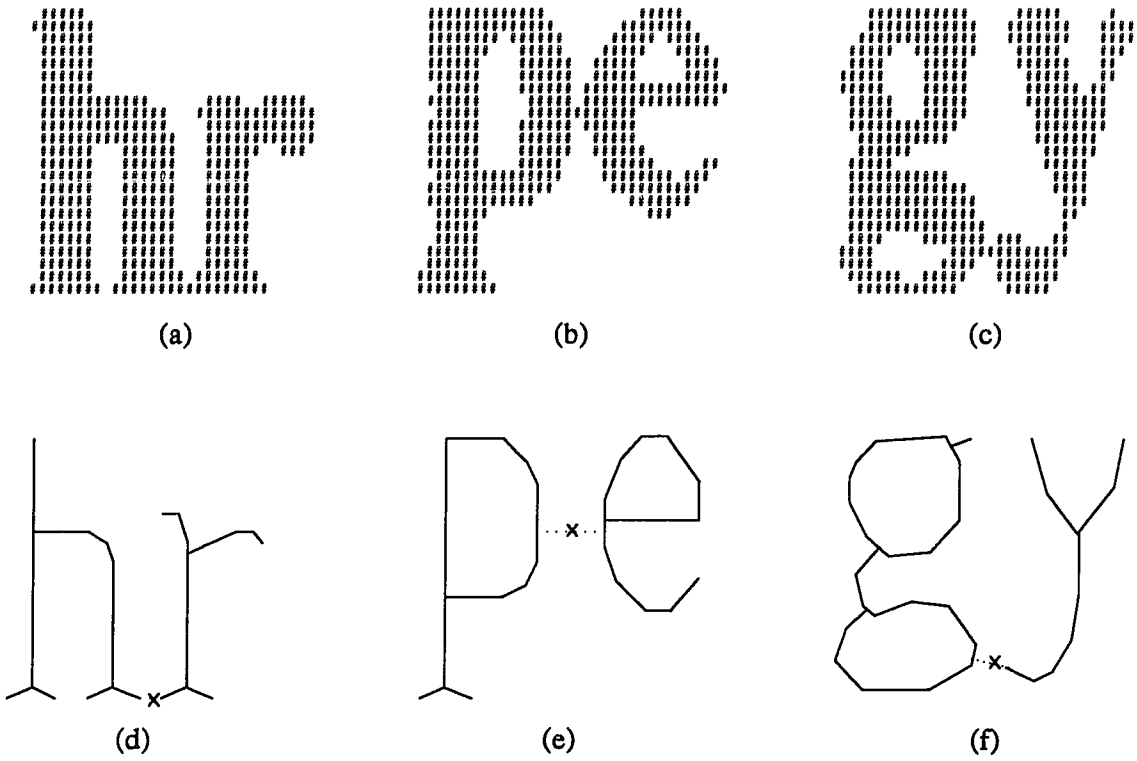


Figure 5.7 (a) Serif join, (b) double-o join, (c) other join, (d), (e), and (f) are the skeletons of (a), (b) and (c).

simply trace down from the saddle point to until a zero-crossing of the eigenvalue occurs. The segment from the saddle point to the point with a zero-crossing of the eigenvalue is then deleted.

5.5.2.3 Recognition of Individual Characters and Context Confirmation Once the pruning procedure is performed, the skeleton of the merged character is split into several skeletons of individual characters. For most of the existing splitting algorithms [25, 41, 85], each breaking hypothesis needs further processing on the separated character images to extract individual features. This increases the processing time. However, our method has the advantage that after the pruning procedure, no more processing is needed. Each individual skeleton is just passed through a character classifier. For example, the

skeletons of two merged characters shown in Fig. 5.7(e) are separated. The two skeletons are then passed through a character classifier [47, 63, 74]. If the recognition fails, the hypothesis of the breaking candidate is rejected and not considered. Otherwise, a context confirmation is then applied to the hypotheses which survive in the recognition to determine the reliability of the splitting hypotheses. The context confirmation is based on the corresponding words which contain the merged character. The reliability can be calculated by the number of words which survive in the dictionary checking divided by the total number of the words which contain the merged character. For the example of Fig. 5.7(b), there are six words of "operations", three words of "property", a single word of "inspection", "speed", "paper", and "superposition", which contain this merged character. The characters with the breaking candidate shown in Fig. 5.7(e) are recognized as "p" and "e" which will succeed for all the words by dictionary checking. Therefore, the reliability of the hypothesis is 1.

5.6 Experimental Results

A set of fuzzy character image prototypes which were obtained from preceding chapter were used as our input. Besides, binary and gray-scale images captured from a scanner were also included in our experiments. In this chapter, we focus on the skeletal processing which has been implemented in C language on a SUN workstation 4/490 under UNIX operating system. Splitting and pruning algorithm for merged characters have been developed and tested as well. In the following, we present test results on fuzzy, binary, and gray-scale images. Additionally, several handwritten character images were also include in our experiments.

Example 1: Fig. B.1(a) in Appendix B illustrates three fuzzy images which were obtained by unsupervised character classification from a document. The three images are built up by 88, 9, and 7 character images from the selected document, where "#" represents the membership greater than 0.95 and the number represents the closest integer of 10 times

of the membership. Since the fuzzy images are composed of a set of character images belonging to the same classes, the memberships around the contours of the images are quite ambiguous. Fig. B.1(b) shows the smooth surface patches. The skeletons of the fuzzy images are shown in Fig. B.1(c).

Example 2: This example shows the result when our algorithm is applied on a binary image. Fig. B.2(a) shows a binary image of character "T". If we use the thinning algorithm in [69] which deletes simple points successively from north, south, east, and west boundaries, the skeleton is shown in Fig. B.2(b). The skeleton by our method is shown in Fig. B.2(c). The deformation around the junction point becomes relatively small.

Example 3: This example illustrates the processing on the merged characters. As shown in Fig. B.3(a), two merged character images were constructed by the unsupervised character classification from a document. The first one is constituted by 5 original images, and the second one is simply by one image. Again, the convolution is performed on the fuzzy images, and smoothed images are obtained as shown in Fig. B.3(b). The entire skeletons of the merged character images are then extracted as shown in Fig. B.3(c), including the dotted lines.

To separate individual character skeletons, we trace on the skeleton lines and search for the saddle points. To avoid the trivial saddle points due to some noise, a threshold of the eigenvalue associated with the tangent of the skeleton line must be selected. From our experimental results, the range between 0.015 to 0.02 is distinguishable for those trivial saddle points. In our experiments, 0.015 is used. The crosses shown in Fig. B.3(c) show the saddle points with eigenvalues exceed the threshold. There are five and six candidates for the two examples, respectively. Among the set of saddle points, the original membership values of the pixels where the saddle points locate are first evaluated. For the first example, the membership of the saddle point at the linking stroke is 0.4. Therefore, this point is ranked by 1. For any other saddle point, which has membership 1, the

eigenvalues are further checked and ranked accordingly. Besides, it is observed that the eigenvalue of the saddle point 1 is also the highest. The eigenvalues for the five saddle points are 0.0627, 0.0507, 0.0482, 0.0465, and 0.0354, respectively. Since the second example is a single binary image, all the membership values at saddle points are 1. Therefore, only the eigenvalues are compared. The points are ranked by the numbers with the six eigenvalues 0.2903, 0.2438, 0.1929, 0.1630, 0.1156, and 0.0845, respectively.

The pruning algorithm is then applied on the first candidate of each skeleton. In the first example, the stroke containing the saddle point is short. The determination of a stroke which is short or long is based on the comparison of the stroke length and the value 2σ, which represents the maximum width of the image. The length of this stroke is 6.6103 which is quite close to 2σ, where $\sigma = 3$ in this example. Besides, the two end points of this stroke belong to a "T" junction which tells the join is due to two osculating vertical convex contours (i.e. double-o join). Therefore, the entire stroke is removed. For the second example, the stroke containing the saddle point is long. Therefore, the pruning algorithm is applied on the two segments on the left- and right-hand sides of the saddle point separately. The segment on the left-hand side is short and has a "T" junction at the end point. It is therefore erased. However, the segment on the right-hand side is a long stroke. Therefore, it is pruned from the saddle point to the point with zero-crossing of the eigenvalue. The dotted lines shown in Fig. B.3(c) are the extra skeletons which are removed.

As observed, the first candidates in these two examples are the correct hypotheses. The resulting split skeletons will be recognized directly without any further preprocessing for features extraction. The multiple context confirmation is then applied to evaluate the reliability of the hypotheses. This is specially useful when a fuzzy image is constructed by multiple images. For the first example, the recognized characters "p" and "o" will be confirmed by dictionary checking with the corresponding words containing the merged character. That is, the five words "important", "decomposition", "decomposed",

"superposition", and "decomposition" as shown in Fig. B.4 will be confirmed, which results a reliability of 1. Our splitting algorithm has been tested on 100 merged characters. There are 75 cases succeed at the first candidate.

Example 4: Fig. B.5(a) shows three broken characters which are captured from a poor-printed document. When the convolution is performed, the broken gaps are linked by relatively small membership values as shown in Fig. B.5(b). Therefore, connected skeletons are extracted as shown in Fig. B.5(c).

Example 5: When the input are gray-scale images, our algorithm can also directly apply on them without converting into binary images. Fig. B.6(a) illustrates two gray-scale images. The first one was captured from [89]. First, we convert the gray-levels into fuzzy memberships by a linear mapping. Mathematically, the memberships are calculated by the mapping function: $\mu = \frac{g - g_{\min}}{g_{\max} - g_{\min}}$, where g denotes the gray-level of a pixel, and $g_{\max}$ and $g_{\min}$ denote the maximum and minimum of gray-levels in the images. Therefore, even though the background of the images may not clear, the membership values on the background will be close to 0 if the background is shaded uniformly.

However, since the memberships on the background are not all zeros, the method for determining σ (Section 5.6.1) is not appropriate. Here, we simply apply an edge detection [69], and then a raster scan is applied to find the most popular run length. To allow a tolerance, the smallest integer greater than 1.2 times of the most popular run length is then selected as σ. After performing convolution and ridge detection, the results of skeletonization are shown in Fig. B.6(b).

Example 6: Lastly, we show a few test results for handwritten characters. To allow more flexibility of handwritten characters, larger σ can be used. In these test results as shown in Fig. B.7, σ is still selected as the smallest integer which is greater than the maximal distance. some small gaps small than up the stroke widths, such as the close loop of

character "a" may leave an opening due to the quick movement of writing, have been connected. We also experimented with selecting σ from the stroke width up to 3 times of stroke width. However, when the value of σ is increased, the deformations at junctions and corners are also increased. Additionally, some actual gaps may also be linked. According to our trial experiments, σ up to twice of the stroke width gave quite good results, and the the deformations are tolerable. Most linking positions of merged characters can be detected by our algorithm. However, the pruning procedure is more complicated since some long strokes containing breaking points do not belong to any character.

CHAPTER 6

CONCLUSIONS AND FUTURE RESEARCH

In this chapter we summarize the contributions of our research and briefly discuss further research directions.

6.1. Contributions of This Dissertation

The motivation of this dissertation is to provide the fundamental data and information for an office document management and filing system. The significant information and data which appear in a paper document must be captured and encoded. The unsupervised character classification has been experimented to be a suitable approach to simplify an OCR system. Fuzzy theory and technology have been investigated and used as a natural tool for the character classification and feature extraction.

The block segmentation based on a run-length smoothing algorithm [21, 95] has been developed into a concise two-step method. Therefore, the computational time can be reduced significantly almost 40% comparing to the original algorithm. The block classification has been developed based on the feature of mean transition which has been verified as invariance to character size, font style, and even though the scanning resolution. The mean transition TV_x for classifying vertical lines and TH_y for classifying horizontal lines has been examined in a statistic way and found to be a very stable features with standard deviations lower than 0.0005. When they are applied on single-font and fixed-sized documents, the standard deviations of the mean transitions are lower than 0.005. For multi-font, variable-sized documents, even the scanning resolution changes, the standard deviations are still less than 0.1. The parameters can be adaptively adjusted with the consideration of tolerance. These algorithms have been successfully experimented.

We have presented an efficient method for fuzzy typographical analysis (FTA) of a textual block to improve the character classification and recognition performance. Each text line is decomposed into three zones and the typographical categorization is performed based on word structures. An efficient baseline detection technique has been developed, and the tolerance for baseline detection and the typographical analysis have been discussed. Baseline detection can also be used for skew angle detection when a document is tilted. Another usage of baseline is to determine the interline spacing, which is useful for grouping textual blocks in a paragraph for layout structure analysis. FTA has been tested for different font sizes with satisfactory performance. The results show that FTA can correctly preclassify characters as small as 6-point, and is efficient to process more than 10,000 character per second. FTA can be extended to handwritten character to simplify the classification problem. Fuzzy consideration is incorporated to ensure the robustness for special cases. Partial differential equations are formulated as the constraints on the fuzzy membership functions. Their boundary conditions are considered to preserve the continuity. The fuzzy membership functions are then derived according to the character size and location within the typographical structure of a text line.

We have also presented a fuzzy model of unsupervised classification for preclassifying characters in the document processing system. A fuzzy model of prototypes is defined and several propositions of the features of the fuzzy model are given. The existing similarity equations for matching are investigated. However, those methods have a common problem because their matching is based on the percentage of intersection of two patterns. They can not distinguish noises nearby or far away from the boundary of an object. Therefore, we proposed a nonlinear weighted similarity function based on distance transformation. Furthermore, a similarity measure of the fuzzy model is proposed from the extension of the nonlinear similarity function. A simple matching algorithm is applied for grouping input images into the fuzzy prototypes. A hierarchical scheme for classification is proposed for the prototype grouping in order to save the computational

time as compared to the sequential grouping. The hierarchy also has the advantage for parallel processing. The characteristics of the fuzzy model are discussed and used in speeding up the classification process. The emphasis of inequality measure for small characters guarantees no misclassification, but a little redundancy is encountered on the fuzzy prototype set. This redundancy can be removed by self-grouping the final prototype set. The propositions and algorithms have been tested with satisfactory performance.

The fuzzy model of prototypes is verified that it can reduce the effect of noise. After classification, the character recognition which is simply applied on a smaller set of the fuzzy prototypes, becomes much easier and less time-consuming. Based on the prototypes which is free of noise, the recognition problem will be simplified and the speed as well as recognition rate will be increased. For ambiguous characters, probably as merged, the accuracy of postprocessing will be also improved. Since the scope of the proposed model is limited and the matching algorithm is easy to implement, the complexity of a document processing system is reduced.

It is believed that feature extraction is one of the most difficult and key issues in pattern recognition. A topographic approach for skeletonization on fuzzy images has been presented. The method works directly on a fuzzy image or a gray-scale image, so that it is less sensitive to noise and can avoid information loss and extra distortion. A convolution by a bell-shaped function is applied on the original images which is able to link broken characters in a certain degree. The coefficient σ used for convolution is determined by a distance computation which can reduce the deformation close to junctions. Topographic features are based on the first- and second-order directional derivatives on the transformed image. Ridge points are extracted by rule-based topographic analysis of the structure. A matrix representation of the transformation function is derived to save the computational time for locating the accurate ridge points. A membership function is then assigned to ridge points with values indicating the degrees of membership with respect to the skeleton of an object. The significant ridge points are linked to form strokes of

skeleton, and the clues of eigenvalue variation are used to deal with degradation and preserve connectivity. A splitting algorithm is proposed for merged characters based on the skeletons which includes three stages: 1. searching for significant saddle points as possible breaking positions, 2. pruning the linking strokes based on the type of joins, and 3. recognizing individual characters and applying multiple context confirmations. Our algorithm has another advantage that the separated individual character skeletons do not need further processing for feature extraction. The algorithm has been tested on 100 merged character images obtained from scanned documents as well as from unsupervised character classification; 75 of them the correct breaking positions are located from the first breaking candidate.

6.2. Directions of Future Research

In this section, we discuss further studies to improve our algorithms and some potential researches to which our current works can be applied.

1. *Typographical analysis for degraded text blocks*: The fuzzy typographical analysis yields the accuracy to the baseline detection. From our experimental results, 2.5% of errors are found for the baseline detection when the character size is reduced to 5 points. The major problem is that the merged characters are highly increased. One of the solution is to split the merged characters into several components. The rules for splitting in order to correctly detect the baseline of degraded text images is our further study.

2. *Applications on typographical analysis*: Our original intent for typographical analysis is to preclassify characters. However, the information can be used for other applications. For example, the skew angle of document image, interline spacing, and character size are precisely obtained to assist in skew normalization and layout structure analysis. This could be useful for determination of the orientations of captions contained in graphical blocks which usually use Hough transform in a time consuming fashion. How the technique can be transported in the handwritten characters recognition is another issue.

3. *Parallel processing for unsupervised character classification*: The mechanism of unsupervised character classification is simple and has the merit for parallel processing. The study for parallel implementation should be a direction which can highly increase OCR performance.

4. *Application of unsupervised character classification*: Automatic information retrieval is still an unsolved problem. In many documents, keywords are printed in special fonts, which can be easily separated by unsupervised character classification. Therefore, certain keywords can be extracted easily.

5. *Fuzzy theory and technology*: Fuzzy technology has been applied in many fields. In this dissertation, we discover that fuzzy theory and technology is quite natural and successful to interpret many ambiguous situations in classification and skeletonization. The applications of fuzzy technology still remains on many research topics in document processing and analysis. In this dissertation, we formulate two dimensional fuzzy membership functions. The technique can be extended to higher dimensional formulation.

6. *Applications of topographic analysis*: Unlike the edge of an image, topographic primal sketch is invariant with respect to monotonic image transformations [31]. There should be some high-level matching can be applied on the topographic structures in a computer vision system. The research issues include that what other transformation or curve fitting functions can create a better smoothed surface function, and how well the topographic structures can be performed in the three-dimensional object matching.

Statistics, fuzzy theory and technology, typography, topography, and morphological operations have been applied in our research. I strongly hope that this dissertation will be helpful not only in document processing, but also for some problems in computer vision and pattern recognition.

APPENDIX A

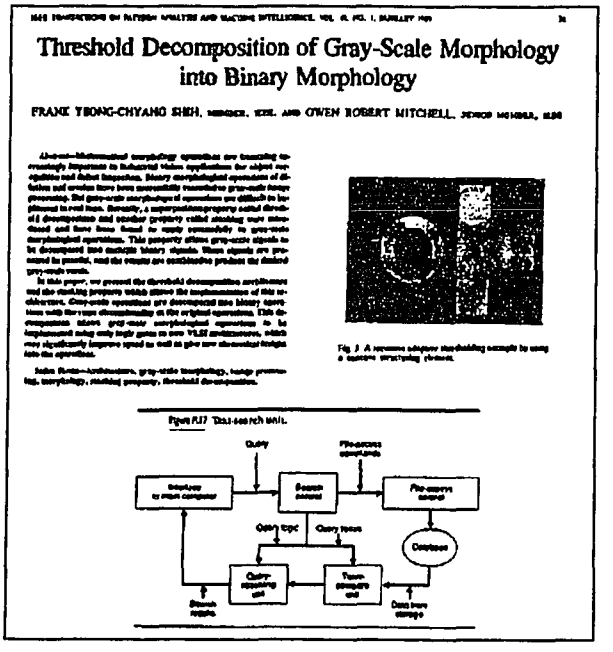
EXPERIMENTAL RESULTS OF CHAPTER 2

Table A.1 The Result of Block Classification of Our Algorithm

No.	$x_{\min}$	Δx	$y_{\min}$	Δy	δx	$\delta x/\Delta x$	*TH*	*TV*	TH_x	TV_x	*N*	*D*	*Class*
1	81	1591	38	23	1118	0.7027	2191	1676	1.96	1.50	9557	0.26117	t
2	2173	26	40	24	19	0.7308	47	33	2.47	1.74	197	0.31571	t
3	88	2101	127	101	1848	0.8796	3954	3210	2.14	1.74	43263	0.20388	t
4	676	936	247	94	836	0.8932	1888	1446	2.26	1.73	20493	0.23292	t
5	88	2101	408	45	1599	0.7611	3079	2707	1.93	1.69	18257	0.19310	t
6	88	1027	588	34	803	0.7819	1478	1166	1.84	1.45	8917	0.25537	t
7	77	1038	630	33	797	0.7678	1580	1210	1.98	1.52	9447	0.27579	t
8	77	1037	672	33	827	0.7975	1667	1254	2.02	1.52	10114	0.29555	t
9	1336	795	687	582	792	0.9962	19117	20627	24.14	26.04	338680	0.73198	p
10	77	1037	713	34	833	0.8033	1589	1286	1.91	1.54	9624	0.27296	t
11	77	1036	755	32	843	0.8137	1635	1292	1.94	1.53	10130	0.30556	t
12	77	1035	796	33	824	0.7961	1594	1206	1.93	1.46	9971	0.29193	t
13	76	1036	837	34	802	0.7741	1526	1177	1.90	1.47	9447	0.26820	t
14	76	1036	879	33	722	0.6969	1394	1078	1.93	1.49	8671	0.25363	t
15	76	1035	920	35	820	0.7923	1584	1263	1.93	1.54	9709	0.26802	t
16	75	1035	962	35	774	0.7478	1526	1208	1.97	1.56	9374	0.25877	t
17	74	1036	1004	34	811	0.7828	1577	1219	1.94	1.50	9765	0.27723	t
18	74	253	1046	31	208	0.8221	375	341	1.80	1.64	2365	0.30154	t
19	74	1035	1087	35	794	0.7671	1507	1163	1.90	1.46	9391	0.25924	t
20	72	1037	1129	34	820	0.7907	1606	1206	1.96	1.47	9764	0.27693	t
21	71	1038	1170	36	822	0.7919	1528	1217	1.86	1.48	9360	0.25048	t
22	71	1038	1212	34	809	0.7794	1608	1226	1.99	1.52	9706	0.27502	t
23	70	1039	1254	34	744	0.7161	1439	1125	1.93	1.51	8782	0.24860	t
24	70	1036	1294	34	802	0.7741	1581	1212	1.97	1.51	9536	0.27072	t
25	70	1039	1337	34	800	0.7700	1618	1261	2.02	1.58	9795	0.27727	t
26	1313	833	1351	38	634	0.7611	1315	1074	2.07	1.69	6102	0.19277	t
27	70	286	1378	32	232	0.8112	447	338	1.93	1.46	2794	0.30529	t
28	1314	462	1392	33	345	0.7468	677	566	1.96	1.64	3130	0.20530	t
29	70	1034	1444	35	823	0.7959	1443	1239	1.75	1.51	8820	0.24371	t
30	69	921	1485	35	747	0.8111	1450	1096	1.94	1.47	8698	0.26983	t
31	523	1350	1609	11	1350	1.0000	93	1350	0.07	1.00	11991	0.80747	h
32	649	461	1640	39	343	0.7440	672	554	1.96	1.62	3857	0.21453	t
33	647	145	1693	3	145	1.0000	17	145	0.12	1.00	371	0.85287	h
34	951	535	1737	32	213	0.3981	410	385	1.92	1.81	2241	0.13090	t
35	517	1359	1774	630	1359	1.0000	7286	7972	5.36	5.87	51075	0.05966	g
36	740	94	2413	25	83	0.8830	172	139	2.07	1.67	948	0.40340	t
37	1548	107	2417	29	94	0.8785	183	181	1.95	1.93	1054	0.33967	t
38	514	1350	2478	5	1349	0.9993	143	1349	0.11	1.00	2436	0.36089	h

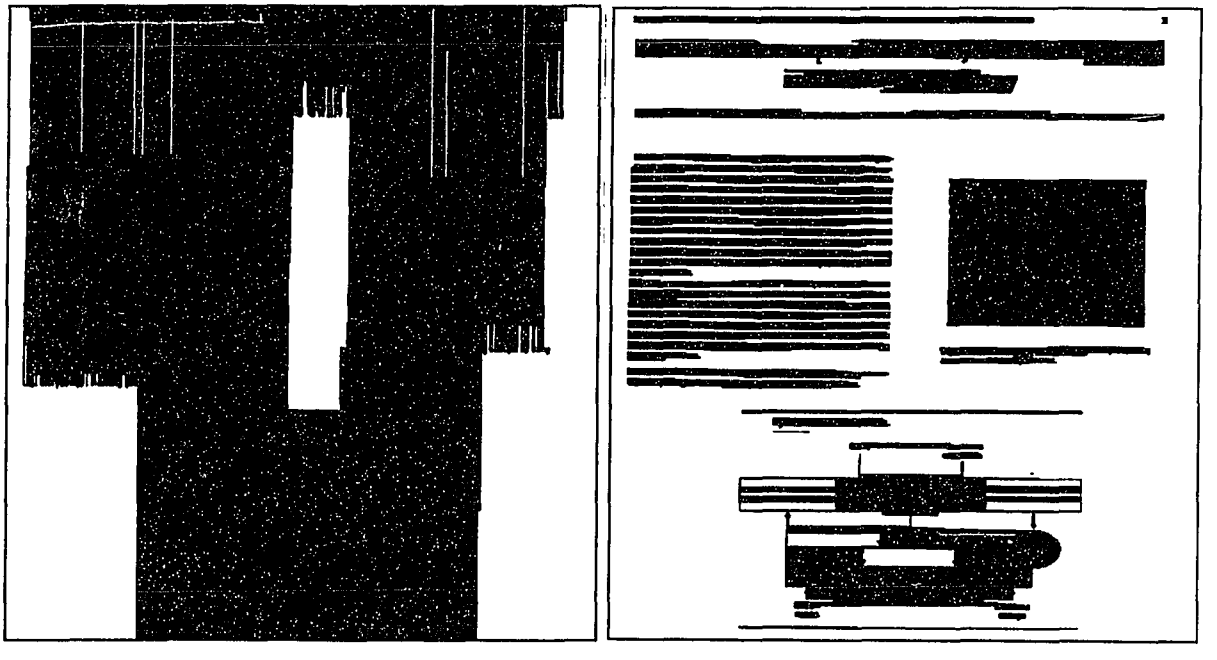
Table A.2 The Result of Block Classification of Wong's Algorithm

No.	x_{min}	Δx	y_{min}	Δy	*TH*	*N*	*N/TH*	$\Delta x/\Delta y$	*Class*
1	81	1591	38	23	2191	9557	4.36	69.17	t
2	2173	26	40	24	47	197	4.19	1.08	t
3	88	2101	127	101	3954	43263	10.94	20.80	t
4	676	936	247	94	1888	20493	10.85	9.96	t
5	88	2101	408	45	3079	18257	5.93	46.69	t
6	88	1027	588	34	1478	8917	6.03	30.21	t
7	77	1038	630	33	1580	9447	5.98	31.45	t
8	77	1037	672	33	1667	10114	6.07	31.42	t
9	1336	795	687	582	19117	338680	17.72	1.37	n
10	77	1037	713	34	1589	9624	6.06	30.50	t
11	77	1036	755	32	1635	10130	6.20	32.38	t
12	77	1035	796	33	1594	9971	6.26	31.36	t
13	76	1036	837	34	1526	9447	6.19	30.47	t
14	76	1036	879	33	1394	8671	6.22	31.39	t
15	76	1035	920	35	1584	9709	6.13	29.57	t
16	75	1035	962	35	1526	9374	6.14	29.57	t
17	74	1036	1004	34	1577	9765	6.19	30.47	t
18	74	253	1046	31	375	2365	6.31	8.16	t
19	74	1035	1087	35	1507	9391	6.23	29.57	t
20	72	1037	1129	34	1606	9764	6.08	30.50	t
21	71	1038	1170	36	1528	9360	6.13	28.83	t
22	71	1038	1212	34	1608	9706	6.04	30.53	t
23	70	1039	1254	34	1439	8782	6.10	30.56	t
24	70	1036	1294	34	1581	9536	6.03	30.47	t
25	70	1039	1337	34	1618	9795	6.05	30.56	t
26	1313	833	1351	38	1315	6102	4.64	21.92	t
27	70	286	1378	32	447	2794	6.25	8.94	t
28	1314	462	1392	33	677	3130	4.62	14.00	t
29	70	1034	1444	35	1443	8820	6.11	29.54	t
30	69	921	1485	35	1450	8698	6.00	26.31	t
31	523	1350	1609	11	93	11991	128.94	122.73	h
32	649	461	1640	39	672	3857	5.74	11.82	t
33	647	145	1693	3	17	371	21.82	48.33	h
34	951	535	1737	32	410	2241	5.47	16.72	t
35	517	1359	1774	630	7286	51075	7.01	2.16	n
36	740	94	2413	25	172	948	5.51	3.76	t
37	1548	107	2417	29	183	1054	5.76	3.69	t
38	514	1350	2478	5	143	2436	17.03	270.00	t



Threshold Decomposition of Gray-Scale Morphology into Binary Morphology

(a)



(b) (c)

Figure A.1 An example of our improved two-step smoothing algorithm (a) the document image, (b) the resulting image after step 1, (c) the resulting image after step 2.

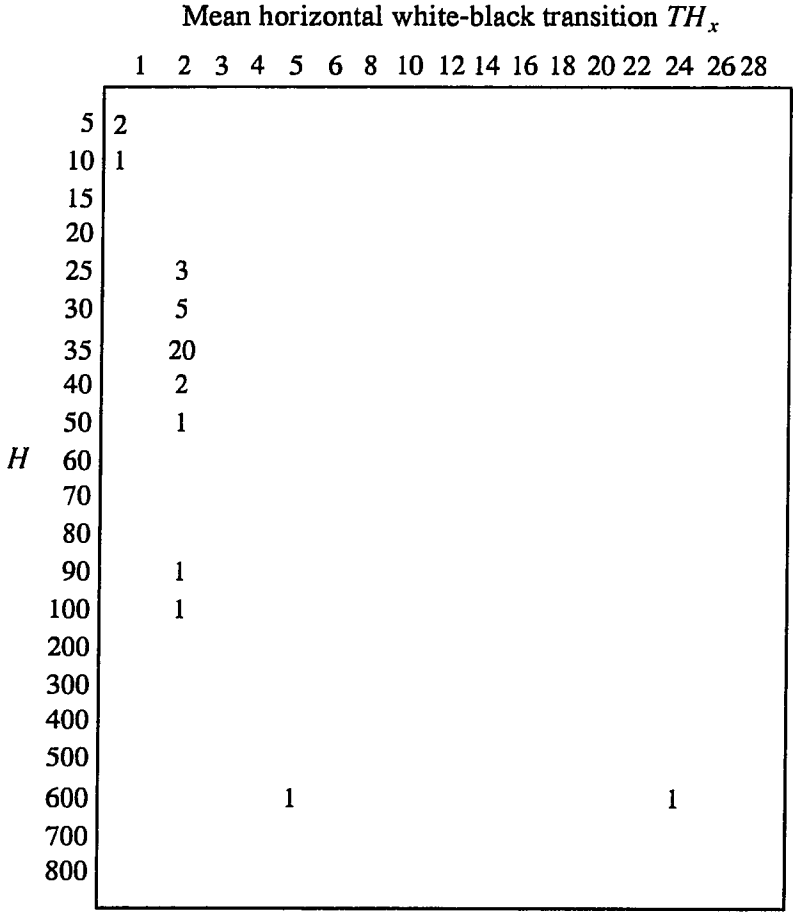


Figure A.2 The projected TH_x - H plane of our algorithm.

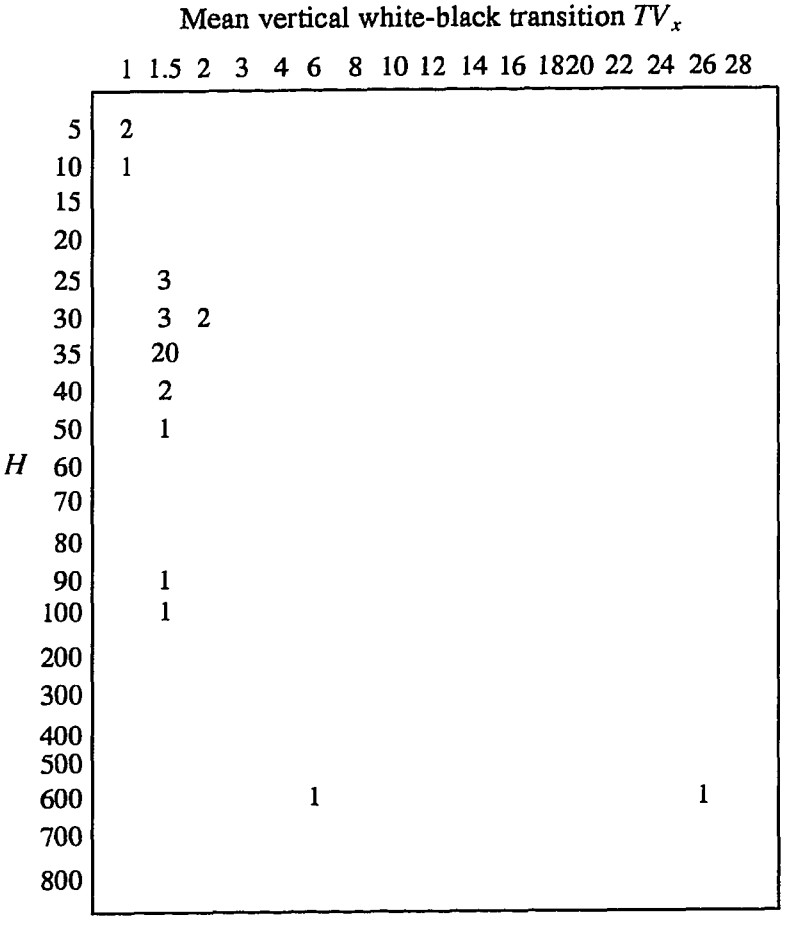


Figure A.3 The projected TV_x - H plane of our algorithm.

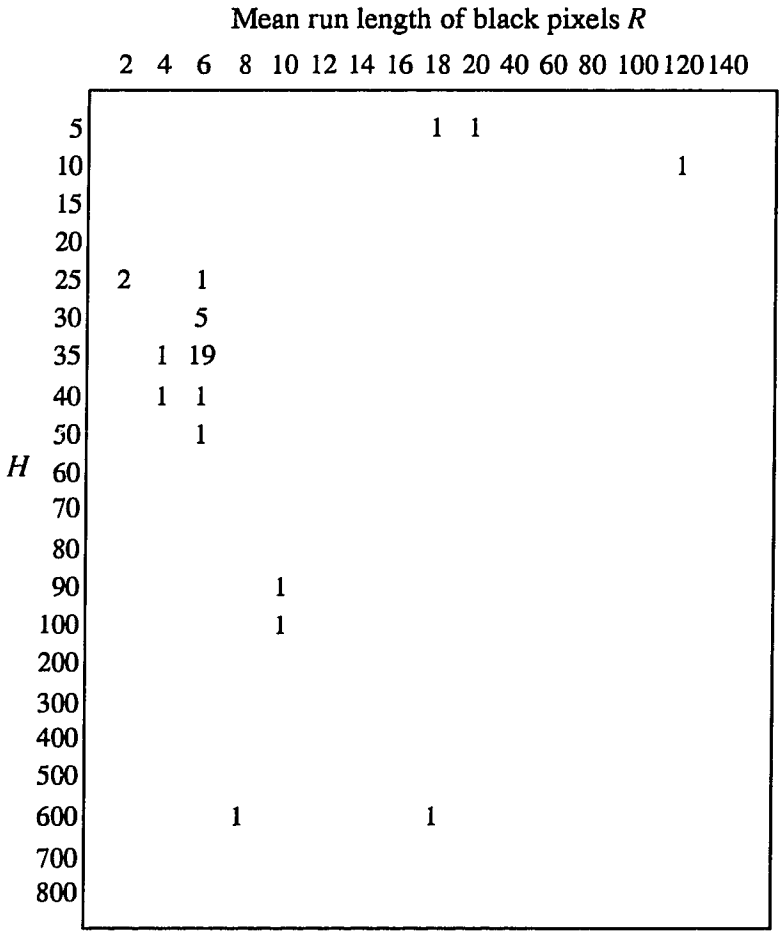


Figure A.4 The projected R - H plane of Wong's algorithm [WoC82].

APPENDIX B

EXPERIMENTAL RESULTS OF CHAPTER 5

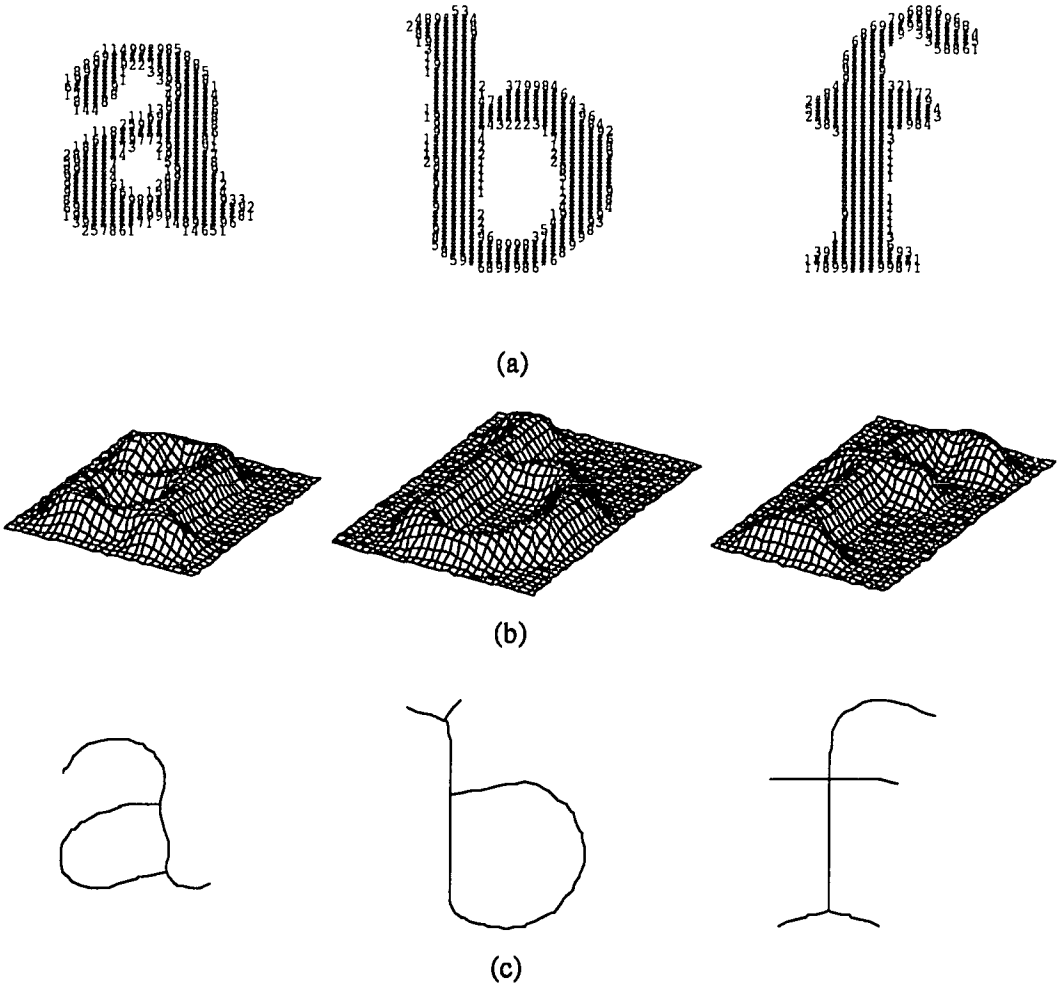


Figure B.1 (a) Examples of fuzzy images, (b) transformation of fuzzy memberships, (c) skeleton of fuzzy images.

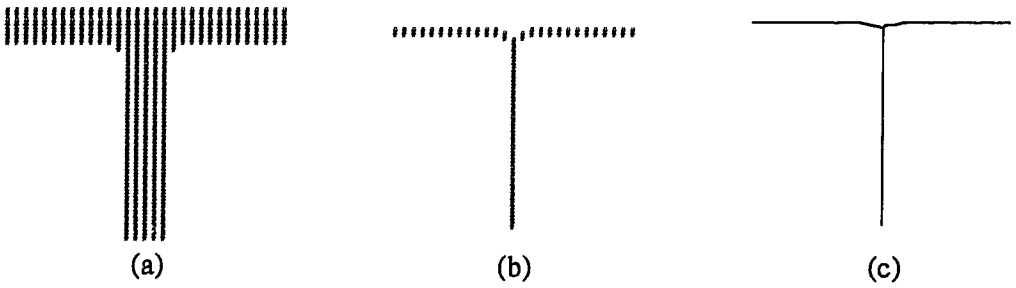


Figure B.2 (a) A binary image of character "T", (b) the skeleton extracted by [69], (c) the skeleton by our method.

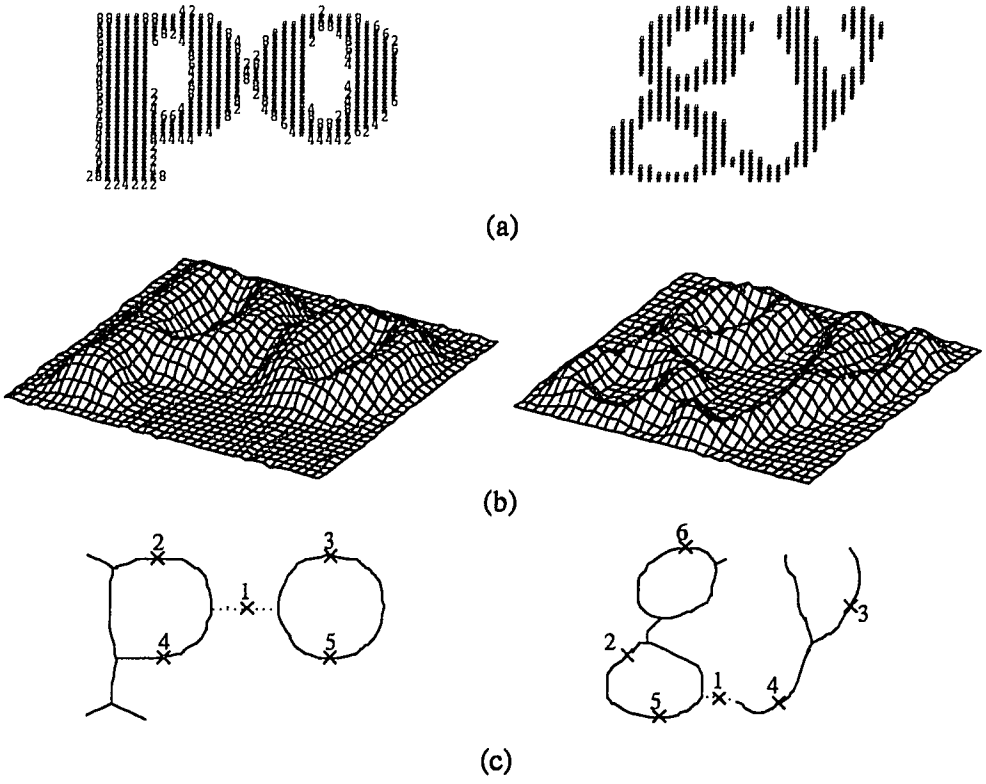


Figure B.3 (a) Examples of merged characters, (b) transformation of fuzzy memberships, (c) skeletons of merged character images.

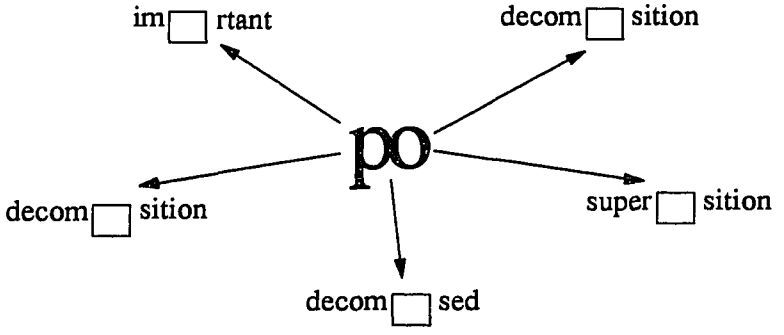


Figure B.4 Multiple context confirmation for merged character illustrated in Fig. 5.10.

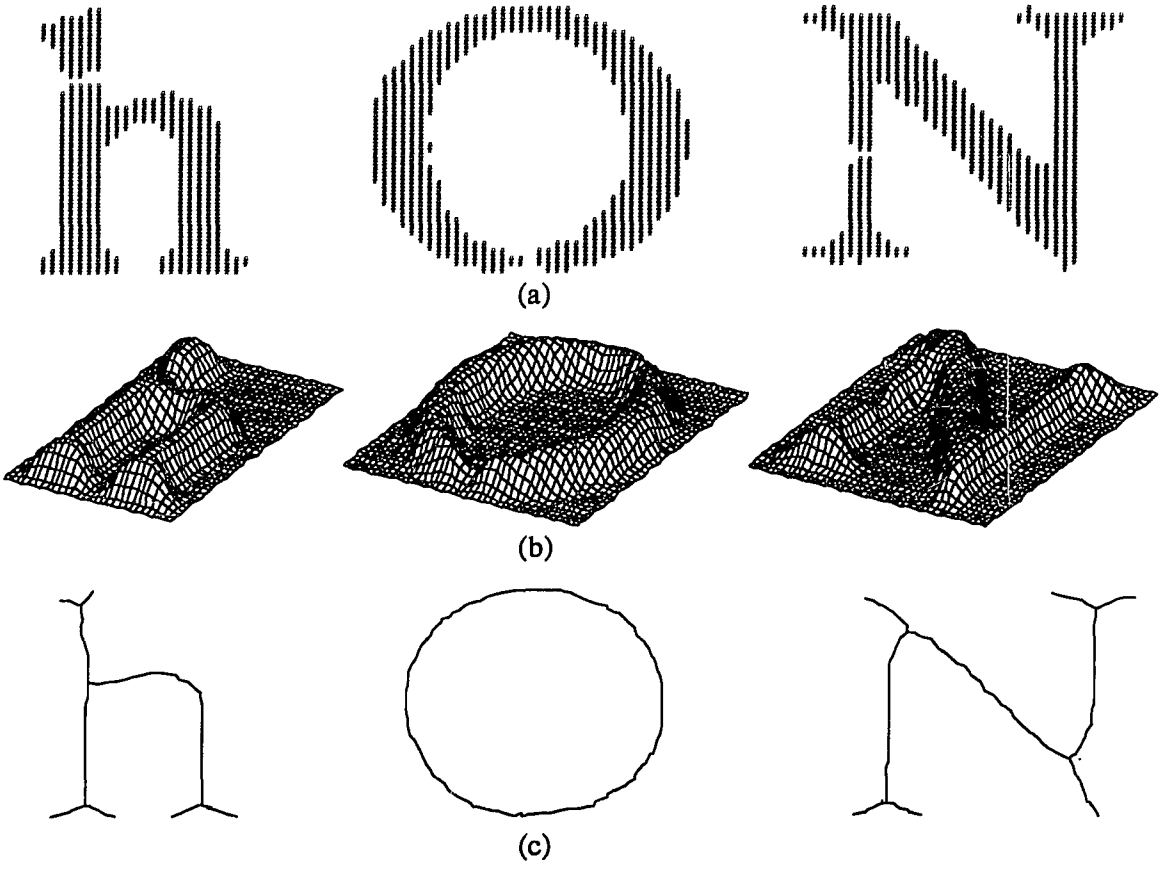


Figure B.5 (a) Examples of broken characters, (b) transformation of fuzzy memberships, (c) skeletons of broken character images.

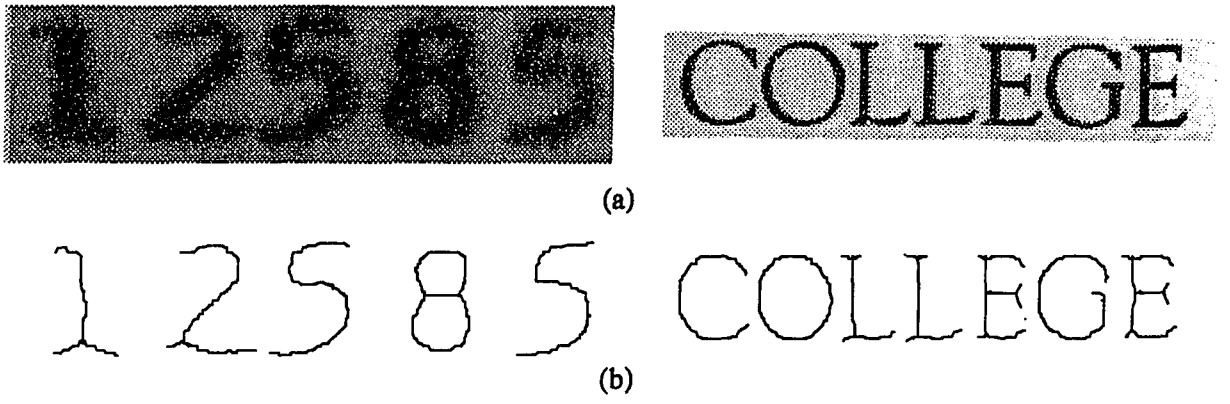


Figure B.6 (a) Two examples of gray-scale images, first one is captured from [89]. (b) The skeletons of gray-scale images.

Figure B.7 Examples of hand-written characters and their skeletons.

BIBLIOGRAPHY

[1] T. Akiyama and N. Hagita, "Automated entry system for printed documents," *Pattern Recognition*, vol. 23, no. 11, pp. 1141-1154, 1990.

[2] Henry S. Baird, "Feature identification for hybrid structural/statistical pattern classification," *Proc. IEEE Computer Vision and Pattern Recognition*, pp. 150-155, 1986.

[3] K. Banno, T. Kawamata, K. Kobayashi and H. Nambu, "Text recognition system for Japanese documents," *Proc. IEEE 9th Int. Conf. Pattern Recognition*, Rome, Italy, pp. 176-180, Nov. 1988.

[4] T. Bayer, "Interpretation of structured documents in a frame system," in *Pre-Proc. IAPR Workshop on SSPR*, pp. 47-56, June 1990.

[5] E. Bertino, F. Rabitti, and S. Gibbs, "Query processing in a multimedia document system," *ACM Transactions on Information Systems*, vol. 6, pp. 1-41, Jan. 1988.

[6] H. Bley, "Segmentation and preprocessing of electrical schematics using picture graphs," *Computer Vision, Graphics, Image Processing*, vol. 28, no. 3, pp. 271-288, 1984.

[7] M. Bokser, "Omnidocument technologies," *Proc. of the IEEE*, vol. 80, no. 7, pp. 1066-1078, July 1992.

[8] S.-T. Bow and R. Kasturi, "A graphics-recognition system for interpretation of line drawings," *Image Analysis Applications* (edited by R. Kasturi and M. M. Trivedi), Marcel Dekker, New York, NY, pp. 37-72, 1990.

[9] R. Bradford and T. Nartker, "Error correlation in contemporary OCR systems," in *Proc. 1st Int. Conf. Document Analysis and Recognition*, vol. 2, Saint Malo, France, pp. 516-523, 1991.

[10] R. L. Burden, J. D. Faires, and A. C. Reynolds, *Numerical Analysis*, 3rd Ed., Prindle, Weber & Schmidt, Boston, MA, 1985.

[11] R. G. Casey, S. K. Chai and K. Y. Wong, "Unsupervised construction of decision networks for pattern classification," *Research Report RJ 4264*, IBM Research Lab., San Jose, California, 1982.

[12] G. L. Cash and M. Hatamian, "Optical character recognition by the method of moments," *Computer Vision, Graphics, Image Processing*, vol. 39, pp. 291-310, 1987.

[13] T. W. Calvert, "Nonorthogonal projections for features extraction in pattern recognition," *IEEE Trans. Comput.* vol. 19, pp. 366-372, May 1970.

[14] R. G. Casey and K. Y. Wong, "Document-analysis systems and techniques," *Image Analysis Applications* (edited by R. Kasturi and M. M. Trivedi), Marcel Dekker, New York, NY, pp. 1-36, 1990.

[15] S. Christodoulakis, M. Theodoridou, F. Ho, M. Papa, and A. Pathria, "Multimedia document presentation, information extraction, and document formation in MINOS: A model and a system," *ACM Transactions on Information Systems*, vol. 4, pp. 345--383, Oct. 1986.

[16] G. Ciardiello, M.T. Degrandi, M.P. Roccotelli, G. Scafuro, and M.R. Spada, "An experimental system for office document handling and text recognition," *Proc. IEEE 9th Int. Conf. Pattern Recognition,* Rome, Italy, pp. 739-743, Nov. 1988.

[17] A. Dengel and G. Barth, "ANASTASIL: a hybrid knowledge-based system for document layout analysis," *Int. Joint Conf. of Artifitial Intellegience,* vol. 2, pp. 1249-1254, 1989.

[18] A. Dengel, R. Bleisinger, R. Hoch, F. Fein, and F. hones, "From paper to office document standard representation," *IEEE Computer*, vol. 25, no. 7, pp. 63-70, 1992.

[19] M. Ejiri, S. Kakumoto, T. Miyatake, S. Shimada, and K. Iwamura, "Automatic recognition of engineering drawings and maps," *Image Analysis Applications* (edited by R. Kasturi and M. M. Trivedi), Marcel Dekker, New York, NY, pp. 73-126, 1990.

[20] ERA, "An electronic reading automation," *Electronic Eng.*, pp. 189-190, Apr. 1957.

[21] J. L. Fisher, S. C. Hinds and D. P. D'Amato, "A rule-based system for document image segmentation," *Proc. IEEE 10th Int. Conf. Pattern Recognition*, Atlantic City, NJ, pp. 567-572, June 1990.

[22] A. J. Filipski and R. Flandrena, "Automated conversion of engineering drawings to CAD form," *Proc. of the IEEE*, vol. 80, no. 7, pp. 1195-1209, July 1992.

[23] L. A. Fletcher and R. Kasturi, "A robust algorithm for text string separation from mixed text/graphics images," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 10, no. 6, pp. 910-918, Nov. 1988.

[24] H. Freeman, 'Computer processing of line-drawing images," *Computing Surveys*, vol. 6, no. 1, March 1974.

[25] H. Fujisawa, Y. Nakano, and K. Kurino, "Segmentation methods for character recognition: from segmentation to document structrue analysis," *Proc. of the IEEE*, vol. 80, no. 7, pp. 1079-1092, July 1992.

[26] R. N. Goldman and J. S. Weinberg, *Statistics – An Introduction*, Prentice-Hall, Englewood Cliffs, NJ, 1985

[27] R. C. Gonzalez and P. Wintz, *Digital Image Processing*, 2nd ed., Addison-Wesley, Reading, MA, 1987.

[28] G. H. Granlund, "Fourier preprocessing for hand print character recognition," *IEEE Trans. Computers*, vol. C-21, pp. 195-201, 1972.

[29] P. W. Handel, "Statistical machine," U.S. Patent 1915993, June 1933.

[30] J. A. Hartigan, *Clustering Algorithms*, John Wiley & Sons, New York, NY, 1975.

[31] R. M. Haralick, L. T. Watson, and T. J. Laffey, "The topographic primal sketch," *Int. J. Robotics Res.*, vol. 2, pp. 50-72, 1983.

[32] S. C. Hinds, J. L. Fisher, and D. P. D'Amato, "A document detection method using run-length encoding and the hough transform," *Proc. IEEE 10th Int. Conf. Pattern Recognition*, Atlantic City, NJ, pp. 464-468, 1990.

[33] A. W. Holt, "Algorithm for a low-cost hand print reader," *Comput. Design*, pp. 85-89, Feb. 1974.

[34] W. Horak, "Office document architecture and office document interchange formats - a current status of international standardization," *IEEE Computer*, vol. 18, no. 10, pp. 50-60, Oct. 1985.

[35] M. K. Hu, "Visual pattern recognition by moment invariants," *IEEE Trans. Inform. Theory*, vol. 8, pp. 179-187, Feb. 1962.

[36] J. J. Hull, "Word shape analysis in a knowledge based system for reading text," in *Proc. 2nd IEEE Conf. Artificial Intell. Appl.*, Miami, FL, 1985.

[37] T. Ho, J. Hull, and S. Srihari, "Combination of structural classifiers," in *Proc. Workshop on Syntactic and Structural Pattern Recognition*, pp. 123-136, 1990.

[38] T. Iijima, Y. Okumura, and K. Kuwabara, "New process of character recognition using sieving method," *Information and Control Research*, vol. 1, no. 1, pp. 30-35, 1963.

[39] S. Impedovo, L. Ottaviano, and S. Occhinegro, "Optical character recognition -- a survey," *Int. J. of Pattern Recognition and Artificial Intelligence*, vol. 5, no. 1 & 2, pp. 1-24, 1991.

[40] R. Kasturi, S.-T. Bow, W. El-Masri, J. Shah, J. R. Gattiker, and U. B. Mokate, "A system for interpretation of line drawings," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 10, pp. 978-991, Oct. 1990.

[41] S. Kahan, T. Pavlidis and H. S. Baird, "On the recognition of printed characters of any font and size," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 9, no. 2, pp. 274-288, March 1987.

[42] A. Kandel, *Fuzzy Techniques in Pattern Recognition*, John Wiley & Sons, New York, NY, 1982.

[43] A. Khotanzad and Y. H. Hong, "Invariant image recognition by zernike moments," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 12, no. 5, pp 489-497, May 1990.

[44] H. Kida, O. Iwaki and K. Kawada, "Document recognition system for office automation," *Proc. IEEE 8th Int. Conf. Pattern Recognition*, Paris, France, pp. 446-448, Oct. 1986.

[45] S. K. Kwon and D. C. Lai, "Recognition experiments with handprinted numerals,"*Proc. Joint Workshop on Pattern Recognition and Artificial Intelligence*, pp. 74-83, June 1976.

[46] P. C. K. Kwok, "A thinning algorithm by contour generation," *Communications ACM*, vol. 31, pp. 1314-1324, Nov. 1988.

[47] K. H. Lee, K. B. Eom, and R. L. Kashyap, "Character recognition using attributed grammar," *Proc. IEEE Computer Vision and Pattern Recognition*, pp.418-423, 1988.

[48] B. Lindgren, "Machine recognition of human language. Part III-cursive script recognition," *IEEE Spectrum*, pp. 104-116, May 1965.

[49] B. Li and C. Y. Suen, "A knowledge-based thining algorithm," *Pattern Recognition*, vol. 24, no. 12, pp. 1211-1221, 1991.

[50] P. G. De Luca and A. Gisotti, "Printed character preclassification based on word structure," *Pattern Recognition*, vol. 24, no. 7, pp. 609-615, 1991.

[51] E. Lutz, H. V. Kleist-Retzow, and K. Hoernig, "MAFIA - An active mail-filter-agent for an intelligent document processing support," in *Multi-User Interfaces and Applications* (S. Gibbs and A. A. Verrijn-Stuart, eds.), North-Holland: Elsevier Science Publishers B. V., 1990.

[52] I. Masuda, N. Hagita, T. Akiyama, T. Takahashi, and S. Naito, "Approach to smart document reader system," *Proc. IEEE CVPR*, pp. 550-557, 1985.

[53] H. Masuzaki, N. Takahashi, and . Kurosu, "HITFILE 650E optical disk filing system," *Hitachi Review*, vol. 38, no. 5, pp. 257-264, 1989.

[54] B. T. Mitchell and A. M. Gillies, "A model-based computer vision system for recognizing handwritten ZIP codes," *Machine Vision and Applications*, pp. 231-243, 1989.

[55] G. Nagy, "Towards a structured-document-image utility," in *Pre-Proc. IAPR Workshop on SSPR*, pp. 293-309, June 1990.

[56] G. Nagy, "A prototype document image analysis system for technical journals," *IEEE Computer*, vol. 25, no. 7, pp. 10-22, 1992.

[57] Y. Nakano, Y. Shima and H. Fujisawa, "An algorithm for the skew normalization of document image," *Proc. IEEE 10th Int. Conf. Pattern Recognition*, Atlantic City, NJ, pp. 8-11, June 1990.

[58] NEC (Nippon Electric Company), "Inmprovements in or relating to character recognition apparatus," U.K. Patent 1124130, Aug. 1968.

[59] L. O'Gorman and R. Kasturi, "Guest editors' introduction: document image analysis systems," *IEEE Computer*, vol. 25, no. 7, pp. 5-8, 1992.

[60] P. V. O'Neil, *Advanced Engineering Mathematics*, 3rd ed., Wadsworth Pub. Co., Belmont, CA,Belmont, CA, 1991.

[61] Sankar K. Pal, *Fuzzy Mathematical Approach to Pattern Recognition*, John Wiley & Sons, New York, NY, 1986.

[62] T. Pavlidis and S. Mori, "Scanning the issue," *Proc. of the IEEE*, vol. 80, no. 7, pp. 1027-1028, July 1992.

[63] T. Pavlidis, "A vectorizer and feature extractor for document recognition," *Computer Vision, Graphics, Image Processing*, vol. 35, pp. 111-127, 1986.

[64] T. Pavlidis and G. Wolberg, "An algorithm for the segmentation of bilevel images," *Proc. IEEE Computer Vision and Pattern Recognition*, pp.570-575. 1986.

[65] E. Persoon and K. S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Trans. on Syst., Man, Cybern.*, vol. SMC-7, no. 3, pp. 170-179, Mar. 1977

[66] P. G. Perotto, "A new method for automatic character recognition," *IEEE Trans. Electron. Comput.*, vol. EC-12, pp. 521-526, Oct, 1963.

[67] F. Rabitti, "A model for multimedia documents," *Office Automation* (D. Tsichritzis, ed.), pp 227-250, 1985.

[68] W. S. Rohland, "Character sensing system," U.S. Patent 2877951, Mar. 1959.

[69] A. Rosenfeld and A. Kak, *Digital Picture Processing*, vol. 2, Academic Press, New York, NY, 1982.

[70] G. Salton, *Automatic Text Processing*, Addison-Wesley, Reading, MA, 1989.

[71] J. Schurmann, N. Bartneck, T. Bayer, J. Franke, E. Mandler, and M. Oberlander, "Document analysis - from pixels to contents," *Proc. of the IEEE*, vol. 80, no. 7, pp. 1101-1119, July 1992.

[72] W. Scherl, F. Wahl and H. Fuchsberger, "Automatic separation of text, graphic and picture segments in printed material," *Pattern Recognition in Practice*, E. S. Gelsema and L. N. Kanal, eds, pp. 213-221, North-Holland, Amsterdam, 1980.

[73] F. Y. Shih, S.-S. Chen, D. D. Hung and P. A. Ng, "A document segmentation, classification and recognition system," *Proc. of the 2nd Int. Conf. on System Integration*, Morristown, NJ, pp. 258-267, June, 1992.

[74] P. Siy and C. S. Chen, "Fuzzy logic for handwritten numeral character recognition," *IEEE Trans. Syst., Man, Cybern.*, pp. 570-575, Nov. 1974.

[75] R. M. K. Sinha, "A width-independent algorithm for character skeleton estimation," *Computer Vision, Graphics, Image Processing*, vol. 40, pp. 388-397, 1987.

[76] S. N. Srihari, Ed. *Computer Text Recognition and Error Correction*, Silver Spring, MD: IEEE Computer Science Press, 1985.

[77] S. N. Srihari and G. W. Zack, "Document image analysis," *Proc. IEEE 8th Int. Conf. Pattern Recognition*, Paris, France, pp. 434-436, Oct. 1986.

[78] C. Y. Suen, M. Berthod, and S. Mori, "Automatic recognition of handprinted characters - the state of the art," *Proc. IEEE*, vol. 68, no. 4, pp. 469-498, 1980.

[79] C. Suen, C. Nadal, T. Mai, R. Legault, and L. Lam, "Recognition of totally unconstrained handwritten numerals based on the concept of multiple experts," in *Proc. Int. Workshop on Frontiers in Handwritting Recognition,* pp. 131-140, Apr. 1990.

[80] G. Tauschek, "Reading machine," U.S. Patent 2026329, Dec. 1935.

[81] R. H. Thomas, H. C. Forsdick, T. R. Crowley, R. W. Schaaf, R. S. Thomlinson, V. M. Travers, and G. G. Robertson, "Diamond: A multimedia message system build on a distributed architecture," *IEEE comput.* 18, 12, pp. 65-78, 1985.

[82] G. B. Thomas and R. L. Finney, *Calculus and Analytic Geometry,* 7th ed., Addison-Wesley, Reading, MA, 1988.

[83] L. Tokuda, "Computers assist humans in human resources," in *AAAI Proc. 2nd Annual Conf. on Innovative Applications of Artificial Intelligence,* Washington, D. C., pp. 31-35, 1990

[84] Y. Tsuji, "Document image analysis for generating syntactic structure description," *Proc. IEEE 9th Int. Conf. Pattern Recognition,* Rome, Italy, pp. 744-747, Nov. 1988.

[85] S. Tsujimoto and H. Asada, "Major components of a complete text reading system," *Proc. of the IEEE,* vol. 80, no. 7, pp. 1133-1149, July 1992.

[86] N. D. Tucker and F. C. Evans, "A two-step strategy for character recognition using geometrical moments," *Proc. 2nd Int. Joint Conf. on Pattern Recognition,* pp. 223-225, Aug. 1974.

[87] P. Vaxiviere and K. Tombre, "Celesstin: CAD conversion of mechanical drawings," *IEEE Computer,* vol. 25, no. 7, pp. 46-54, 1992.

[88] J. T. L. Wang and P. A. Ng, "TEXPROS: An intelligent document processing system," *Int. J. of Software Engineering and Knowledge Engineering,* vol. 2, no. 2, pp. 171-196, 1992.

[89] L. Wang and T. Pavlidis, "Direct gray-scale extraction of features for character recognition," *IEEE Trans. Pattern Anal. Machine Intell.,* vol. 15, no. 10, pp. 1053-1067, Oct. 1993.

[90] D. Wang and S. N. Srihari, "Classification of newspaper image blocks using texture analysis," *Computer Vision, Graphics, Image Processing*, vol. 47, no. 4, pp. 327-352, 1989.

[91] F. M. Wahl, K. Y. Wong and R. G. Casey, "Block segmentation and text extraction in mixed text/image documents," *Computer Vision, Graphics, Image Processing*, vol. 20, no. 4, pp. 375-390, 1982.

[92] D. T. Wang, C.-S. Wei, S.-S. Chen, B.-C. Sung, T. H. Shiau, and P. A. Ng, "Cross correlation of sampled boundary distances - an application to object recognition," *Proc. of the 2nd Int. Conf. on System Integration*, Morristown, NJ, pp. 224-235, Apr. 1990.

[93] R. W. Weeks, "Rotating raster character recognition system," *AIEE Trans. Communications and Electronics*, vol. 80, pt. I, pp. 353-359, Sep. 1961.

[94] J.M. White and G.D. Rohrer, "Image thresholding for optical character recognition and other application requiring character image extraction," *IBM J. Res. Devel.*, Vol. 27, no. 4, pp. 400-411.

[95] K. Y. Wong, R. G. Casey and F. M. Wahl, "Document analysis system," *IBM J. Res. Develop.*, vol. 6, pp. 642-656, Nov. 1982.

[96] M. Yamada and K. Hasuike, "Document image processing based on enhanced border following algorithm," *Proc. IEEE 10th Int. Conf. Pattern Recognition*, Atlantic City, NJ, pp. 231-236, June 1990.

[97] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Trans. Computers*, vol. 21, no. 3, pp. 269-281, Mar. 1972.

[98] H. Zen and S. Ozawa, "Extraction of the fair document from mixed mode manuscript," *Proc. IEEE on Computer Vision and Pattern Recognition*, San Francisco, CA, pp. 544-549, June 1985.