Intelligent character recognition using hidden Markov models

This item was submitted to Loughborough University's Institutional Repository by the/an author.

Additional Information:

• A Doctoral Thesis. Submitted in partial fulfilment of the requirements for the award of Doctor of Philosophy of Loughborough University.

Metadata Record: https://dspace.lboro.ac.uk/2134/13786

Publisher: © Kamran Kordi

Please cite the published version.
This item was submitted to Loughborough University as a PhD thesis by the author and is made available in the Institutional Repository (https://dspace.lboro.ac.uk/) under the following Creative Commons Licence conditions.

For the full text of this licence, please go to: http://creativecommons.org/licenses/by-nc-nd/2.5/
<table>
<thead>
<tr>
<th>VOL. NO.</th>
<th>CLASS MARK</th>
</tr>
</thead>
<tbody>
<tr>
<td>28 APR 1995</td>
<td>Loan Copy</td>
</tr>
<tr>
<td>21 MAR 1997</td>
<td></td>
</tr>
<tr>
<td>25 APR 1997</td>
<td></td>
</tr>
<tr>
<td>23 MAY 1997</td>
<td></td>
</tr>
<tr>
<td>6 OCT 2000</td>
<td></td>
</tr>
</tbody>
</table>
Intelligent Character Recognition

using

Hidden Markov Models

by

Kamran Kordi, Bsc., Msc.

A thesis submitted in partial fulfilment of the requirements for the award of Doctorate of Philosophy of the Loughborough University of Technology.

1990

Supervisor: C.S. Xydeas
Department of Electronic and Electrical Engineering
Loughborough University of Technology
©Kamran Kordi 1990
ABSTRACT

Recognition of printed and hand printed characters has received much attention over the past decade as the need for automated 'document entry' systems assumes a commanding role in office automation. Although, present Optical Character Recognition (OCR) systems have reached a high degree of sophistication as compared to early systems, the design of a robust system which can separate text from images accurately and cope reliably with noisy input and frequent change of font is a formidable task. In this thesis, a novel method of character recognition based on Hidden Markov Modelling (HMM) is initially described. The scheme first describes a training set of characters by their outer contours using Freeman codes; next, the HMM method is applied to capture topological variation of the characters automatically, by looking at typical samples of the different characters. Fonts of similar topology can also be incorporated in one hidden Markov model. Once the model of a character in upright position is derived, the character can be recognized, even, when it has been rotated by multiples of 90 degrees. This technique is further extended to combine structural analysis/description of characters with hidden Markov modelling.
In this scheme, a character is first skeletonized and then split to primitives; each primitive is described by hidden Markov models while its corresponding position with respect to nodes (junctions) where the primitives meet, are recorded. This scheme is virtually font and size independent. A new document classification algorithm based on Fuzzy theory is also proposed which provides an indication of a document's contents in terms of 'text' and 'non-text' portions.
Acknowledgement

I wish to express my gratitude to my supervisor Prof. C.S. Xydeas for his guidance and encouragement throughout the course of the research. Thanks are also due to Dr. M.J.J Holt for his useful comments in the initial phase of the project. In addition, Prof I.R. Smith's kind permission to use the departmental facilities is acknowledged.

The final stage of writing this thesis was marred by the untimely death of my director of research Prof. A.P Clark; I wish to pay tribute (however posthumously) for his kind and courteous treatment of students under his direction. He was a refined man, and no doubt his absence will be felt among his students and friends alike.

Finally, I wish to thank my wife, Haydeh, for her resilience, courage and patience during many difficult years which were accompanied with personal tragedy; had it not been for her enormous sacrifice this thesis may not have been written.
# Table of Contents

**Chapter 1: Introduction**  
1

**Chapter 2: OCR systems, past and present**  
2.1: Introduction  
12  
2.2: Template matching and Correlation  
19  
2.3: Transformation and Series expansion  
30  
2.4: Structural Recognition of characters  
43  
2.5: Distribution of black/white pixels  
59  
2.6: Pre-processing techniques  
75  
2.7: Skew normalization  
76  
2.8: Filling/Smoothing  
86  
2.9: Thinning  
87  
2.10: Chu’s Thinning algorithm  
91  
2.11: Sinha’s Thinning algorithm  
96  
2.12: Segmentation of joined characters  
98

**Chapter 3: Document analysis and field segmentation**  
3.1: Introduction  
106  
3.2: Document segmentation in retrospect  
107  
3.3: Document classification using Fuzzy theory  
127  
3.4: Fuzzy sets, Fuzzy reasoning  
131  
3.5: Details of the scheme and results  
134
Chapter 4: Pre-processing techniques implemented/designed

4.1: Introduction 137
4.2: Detection of loops/holes in a character 137
4.3: Detecting baseline 139
4.4: Size normalization 140
4.5: The thinning algorithm 141

Chapter 5: Hidden Markov Modelling of Shape

5.1: Introduction 143
5.2: Theory of Hidden Markov Modelling 143
5.3: Considerations for implementation 151
5.4: Scaling 152
5.5: Finite training set 156
5.6: Types of Hidden Markov Models and training process 157
5.7: Character recognition using first order Hidden Markov Models 160
5.8: Experiments and results 165

Chapter 6: Structural recognition of characters using Hidden Markov Modelling

6.1: Introduction 168
6.2: The scheme’s concepts 170
6.3: Recognition process 171
6.4: The algorithm and its implementation 175
6.5: Results and discussion 177

Chapter 7: Suggested further work 181

Bibliography
No part of the work described herein, has been submitted to Loughborough University of Technology or any other institution for a degree.
Chapter 1

Introduction
I. Introduction

Recognition of printed and hand printed characters has been one of the earliest applications of pattern recognition methodologies and many of the statistical and structural approaches to more general problems of pattern recognition have been illustrated using examples of character recognition.

The origins of optical character recognition (OCR) date back to the Carey's invention of Retina Scanner in 1870 [Mantas, 1986], which was an image transmission system using a mosaic of photocells. Later, in 1890 Nipkow invented the sequential scanner which was a major breakthrough both for modern television and reading machines. In 1900 a Russian scientist, Tyurin successfully used character recognition as an aid to the visually handicapped.

The slow progress of OCR technology continued until late 1940s and early 50s, when the early versions of modern OCR appeared with the advent of digital computers and for the first time, OCR was realized as a data processing approach with particular application to the business world. From that perspective, David Shephard of INTELLIGENT MACHINE RESEARCH CO. can be considered as the pioneer of the development and building of commercial OCR equipment. Although, Modern commercial OCR machines first
came to the fore in the 70s, only over the past few years has the technology advanced sufficiently for OCR to be feasible commercially. There are three main application areas of character recognition technology:

- Data entry
- Text entry
- Process automation

Data entry first appeared with document reading for banking applications; its character set is extremely limited, which includes numerals and some special symbols. Paper format is very constrained with a limited number of code lines to be read. High throughput requirements and a remarkably high tolerance of bad printing quality to cope with distorted input are among the main features of a 'Data entry' system. A typical system throughput ranges up to 150000 documents/hour which results in more than 3000 character/second for the single code line. Typical single character error rates and reject rates are listed as 0.0001% and 0.01 % respectively.

The second major application of OCR is that of page readers for text entry. Restrictions on paper and the character set and size are no longer valid; instead constraints concerning type font and printing quality are of paramount importance. These machines are expected to handle documents
of normal typewriter page size and are meant to be used in a word processing
environment, for example in the newspaper and publishing industry. They
can scan the full alphanumeric character set with upper and lower case letters
plus punctuation marks but only if the document is typewritten in one of the
selected type fonts which the recognition system is designed for, such as OCR-
A, OCR-B, Gothic (fig.1.1) and if the type writing has been performed with
one shot carbon ribbon avoiding any topological distortion of characters. The
main justification of page readers is their cost effectiveness compared with
manual typing of characters. An ordinary copy typist is expected to produce
about five errors in a page of 800 words or about 0.1% error rate[Schurmann,
1982]. To make an OCR system commercially viable, it has to perform far
better than 0.1% error rate, particularly if it is vital to have absolutely ac-
curate text, since one incorrect symbol may render a piece of text useless, as
would happen with financial information. Page readers achieve single font
error and reject rates of 0.01% and 0.1% respectively. These rates very much
depend on the precision and the quality of the typewritten print. Over the
past 6 years, however, multifont/omnifont page readers have appeared in the
market; the leader in this field is Kurzweil's DISCOVER series. I have bench
tested one of the Kurzweil's products(DISCOVER 7320); the machine has a
Fig. 1.1: Samples of a) OCR A, b) OCR B alphabet.
confidence measure (0 - 50) which is usually set at 25 for an optimum performance. A simple test showed that, it heavily depends on dictionary look up tables; a type written page of text containing 2 - 3 upside down characters was fed into the scanner. The result was propagation of the misrecognized characters right to the end of the text. It means that some probability regarding frequency of recognised characters is collected as recognition is in progress and the information is used to recognise the incoming data. When the confidence measure was set to 50, the recognition rate dropped drastically with respect to the quality of paper and print as the system treated the probabilities of letter frequency with caution. Frequent changes of dissimilar fonts in a page decreases the performance rate of the OCR system; indeed all other OCR products that I am aware of, suffer from similar symptoms. With the advent of 'office automation', it was predicted that paper will go away in the electronic office; this has not yet materialised and as office automation increases productivity, more memos are written and distributed by people. Paper has a lot of desirable qualities like its high resolution as well as being acceptable to human beings; it is also easy to carry around. Nowadays office documents include text as well as nontext i.e. pictures, diagrams etc. One of the most pressing impediments in office automation (apart from designing a
high performance and robust OCR system) is separation of large chunks of text from non-text (e.g. graphics, etc.), so that, different parts of a document are coded according to their contents, thereby maximizing the compression ratio of documents.

The third major application is reading post codes which are mainly hand written. In this context a correct recognition rate as low as 60% is acceptable; since the volume of letters to be processed is usually high, a 60% correct recognition rate leaves about 40% of letters to be sorted manually, a clear improvement in letter sorting at a post office. Automation of postal letter sorting was one of the early commercial applications of OCR and there are fully automated systems in operation in U.S.A and Europe.

One last potential application worth mentioning which also has its roots in the origins of OCR is the reading aids for the visually impaired or blind. It combines OCR with speech synthesis and has received much acclaim for its humanitarian use. Kurzweil has produced such a system which I have seen in operation at Manchester municipal library. It is a remarkable aid for the blind to read text, but the actual problems with OCR and speech synthesis are yet unresolved and it is hoped that the gradual refinement of technology in future will produce a more reliable system. Different schemes that exist
under OCR term are depicted in figure 1.2; the definitions of major schemes not mentioned above, are as follows:

[I] On-line character recognition means recognition of single hand printed characters using a tablet (or a light pen) where not only the character image is provided, but also the timing information for each stroke is available.

[II] Hand drawn character recognition is the recognition of single hand printed characters which are unconnected.

[III] Script recognition is concerned with recognition of unrestricted hand-written characters which may be connected and cursive.

[IV] Multifont/Omnifont recognition is an OCR system which is expected to cope with virtually any printed fonts and sizes.

II. Preliminaries and thesis organisation

It is the general impression that recognition of printed and hand printed characters could not be easier; the computer recognizes printed or hand drawn characters on the paper and turns them into ASCII code. Humans can read by the age of four; surely computers can not possibly find it harder
Figure 1.2: Different areas covered by the term "Character recognition".
than man.? Rarely is there any technology that has been in the coming
to than OCR; every few years a breakthrough is announced and yet each break­
through runs out of steam as time passes. What are the problems.? The
fundamental problem is that the brain and eye are extremely good at pat­
tern recognition. Although we do not have a thorough understanding of
their reasoning principles, OCR techniques are designed to simulate and if
possible to emulate human reading ability in speed and performance based
on our latest theories explaining the gift we are endowed with at birth. If
one looks through a magnifying glass at actual printed marks on the paper,
they often look like a collection of inky dots, unrecognizable as the letters we
have known since our childhood. The brain not only detects similarities that
defy logical analysis, it has a wide range of auxiliary mechanism to correct
mistakes, so that, the sense of reading text passes unimpaired through what
may be a collection of black dots. There are different factors which may
distort a character prior to recognition; they can all be considered as 'noise',
but for the sake of clarity, the following terms are defined:

- Noise: Disconnected segments, bumps, gaps in lines, which are caused
  by quantization noise and the actual print quality.
• Distortion: Local variations, rounding of corners, improper protrusions, dilation and shrinkage.

• Translation: Movement of the whole character or its components.

• Rotation: Change in orientation.

• Style variation: Use of different shapes to represent the same character, serifs, slants, etc.

A cursory review of the available literature reveals that an immense effort has been directed towards recognition of hand printed characters rather than machine printed symbols. It is argued that capturing and processing the ever increasing volumes of data generated by mankind, e.g. mail, cheques, account sheets, transaction statements and computer programs are economically very attractive [Suen, 1982]. Although that is true, it is not possible to ignore the huge bulk of ever increasing machine printed documents which may need computer processing of some kind in office automated environments. At the first glance, recognizing machine printed symbols may look easier than hand drawn characters, as no significant visual variations in a character's shape are detected and machines are far more consistent in producing similar patterns than human beings. In practice, however, that is
not the case, printed characters are as difficult as hand printed characters to recognize, with one major difference: Style variation. Variation in writing style must be within a predefined bound known to the system, just like different fonts and sizes of printed alphabet which must be known to an OCR system. Until 1982, almost all commercial OCR systems used simple pattern matching techniques to recognize characters of predefined fonts and size, but since then, there has been a gradual change towards employing more of the sophisticated techniques used in recognition of hand drawn characters; a clear acknowledgement as how difficult recognition of printed characters can be. There are three main analysis/description layers (fig. 1.3) in any modern OCR scheme:

[1] Early feature classification which groups characters with similar tangible features as one category, e.g. letters with a descender and a loop, like 'p' and 'q'.

[2] Low level recognition of the character at bitmap level.

[3] Contextual post-processing, which looks at the frequency of letters, words and the semantics of English language to validate weak decisions at level two or predict a missing (rejected) character.
Fig. 1.3: Three main analysis/description layers of a modern OCR scheme.
In the following chapters a novel technique to recognize characters at low level is formulated. Hidden Markov Modelling (HMM) is used to model a character's shape; its training (learning) process is iterative and, thus needs little hand tuning. Once an H.M.M model of a character (pattern) in upright position is extracted, the model can also robustly recognize the same character (pattern) even if, it is rotated by multiples of 90 degrees. The H.M.M recognition scheme is further extended by using structural analysis/description of a character's skeleton to produce a more reliable recognition scheme. Unlike many OCR techniques it avoids simple pattern matching of bit maps, and makes an analysis of the topological features in the recognition process. A novel and efficient document classification technique based on Fuzzy theory is also proposed. Chapter two reviews major OCR systems available to date, as well as reviewing various pre-processing techniques. Chapter three discusses document segmentation and classification and introduces a new document classification based on Fuzzy theory. Chapter four describes pre-processing techniques designed/implemented during the course of this research work. Chapter five proposes a novel technique in character recognition based on hidden Markov modelling as well as deriving its relevant rotation invariant properties, while chapter six proposes a more sophisticated
system combining hidden Markov modelling and structural description of characters. Finally in chapter seven, further work in OCR is proposed; in particular a totally new approach for performance assessment/prediction of commercial OCR systems is outlined.
Chapter 2

OCR systems, Past and Present.
OCR systems, Past and present

2.1 Introduction

An OCR system is basically made of five major sub systems: a digital optical scanner, field segmentation and character extraction, a pre-processor, a feature detection/extraction, and recognition/decision logic. The block diagram of a typical OCR system is shown in figure 2.1. Input documents are read and digitized by an optical scanner using a thresholding technique optimal to the system to produce a bi-level version of the document. Optical scanners may be divided [Nagy, 1982] into 'flying spot' devices, where successive portions of the document are illuminated in turn and all of the reflected or transmitted light is collected to determine whether the illuminated spot is black or white, and 'flying aperture' devices, where the entire document is illuminated but light is collected only from a single spot. It is also possible to combine the two methods and both illuminate and observe only a single spot at a time; this expensive arrangement results in a greatly improved signal to noise ratio and therefore more accurate grey scale quantization. Scanners can also be crudely classified according to the mechanism used to address successive portions of the document: Mechanical, Television camera, Cath-
Figure 2.1: Block diagram of a typical OCR system.
ode ray tube and Solid state scanners are the most common; some scanners utilize hybrid combination of these basic types.

Raster scanning [Freedman, 1974] is commonly used in character recognition schemes, the pattern followed in raster scanning is illustrated in figure 2.2. The field of view is divided effectively into a series of horizontal strips and each element of each strip is examined in turn. The scanner thus examines each point within the field of view in a pre-determined sequence. Raster scanning can be accomplished either mechanically or electronically.

In the mechanical scanner, as illustrated in figure 2.3, the light reflected from a given point X in the field of view is focused on to the face of a photosensitive device such as a photo cell via a mirror and lens system. If the mirror is rotated mechanically, light from other points on the line \(X'X''\) can be focused similarly on to the photocell by using a second mirror, or by allowing the document to move past the read station (as in fig.2.3), the line being scanned can be made to move across the document and a raster scan can be obtained.

Mechanical scanning was popular in the early sixties, due to its low cost and compatibility with the relatively slow scan logic circuit then available. As the availability of inexpensive high speed logic circuitry increased,
Figure 2.2: Typical raster scanning pattern.

Figure 2.3: Mechanical Scanning.
higher speed electronic systems employing flying spot/videcon scanners became more common. The flying spot scanner depicted in figure 2.4 produces a high intensity spot of light which is focused by a lens to create a single point of illumination on the object containing the characters to read. A multiplier photo tube senses the light reflected from the object and produces an output that is dependent on the reflectivity of the point on which the spot is focused.

Important characteristics of optical scanners [Nagy,1982; Browne, et. al. 1986] can be classified by:

1) Resolution: The exact measurement of the two dimensional optical transfer function is complex, but for OCR purposes one may express resolution as the number of elements that can be resolved within the limits of the scan. In this form it is a more useful parameter in the selection of the scanning system. A typical resolution is 200/300 pixels per inch.

2) Scanning rate: Two factors effect the scanning rate, the time required to move from one scanning element to the next and the minimum value of the dwell time on each element or
Fig. 2.4: Flying spot scanner.
the time between successive scans of each elements or both. The minimum dwell time may be related to the correct operation of a photo site or to the signal to noise (S/N) ratio of the resulting signal.

3) Signal to noise ratio: The noise superimposed on the output of a scanning system arises from several sources. Noise is generated within the electronic circuits and there is a random function in the collection of photons. Both of these effects are worse when the light level is low. Noise may also be introduced when the sensitivity of the system or dc offset vary with the position of the scanning element. For this reason it is known as spatial fixed pattern noise, by its nature it can be measured and cancelled from processing.

4) Repeatibility: It is important to be able to return exactly to a previous spot on the document, for example to rescan a character or to repeat an experiment. Short term positional repeatability is generally more important than long term repeatability.

In the next stage of the process, shown in fig. 2.1, the document is
segmented into text, graphics and image regions. The text regions are further processed to separate individual characters for recognition. Once a character is extracted, it must be pre-processed in order to get rid of breaks in line segments as well as removing isolated dots and if needed, to normalize the character size.

Although it may be desirable to pre-process the entire document before any attempts are made to detect features of individual characters, commercially available OCR machines preprocess characters individually to save computer memory. At the last stage, the recognition algorithm extracts distinctive features from the preprocessed character and makes a decision to identify the character based on comparing the extracted features with a set of pre-defined criteria.

Previous surveys and reviews [Suen et al., 1977, 1980] have classified OCR techniques according to the vocabulary studied:

- Numerics
- Alphanumeric
- Fortran
- Katakana
CHARACTER

Structural Analysis

Global Analysis

Distribution of points
Global Features

Crossings and Distances
Characteristic Loci
n-Tuples
Moments
Zoning

End points and intersection of line segments
Strokes and bays in various Directions

Transformation
Template Matching

Figure 2.5: Character recognition techniques
It should be pointed out however, that many interesting research papers have appeared on other languages such as Chinese [Mori, et.al., 1980], Kanji [Ikeda, et.al., 1978] and Arabic [e.g. Amin, 1984]. OCR techniques can also be broadly classified according to the methodology in the following order as depicted in figure 2.5.

1- Global Analysis:

Global features

a) Template matching

b) Transformation and series expansion

Distribution of points

c) Crossings and Distances

d) Characteristic loci

e) n-tuples

f) moments

g) Zoning

2- Structural Analysis

a) Strokes and bays in various directions. e.g. \(\sqcap\sqcup\ominus\)
Figure 2.6: Multicategory classifier, Machine 0.
b) End points, intersection of line segments.

Unfortunately different input data as well different scanner resolutions have been used by research workers in this field which makes it impossible to compare all techniques with absolute accuracy. Having said that, it is my view that some techniques have been tested with sufficient volumes of real data, thus a performance comparison between certain techniques can be considered as relatively reliable but not conclusive. Fortunately, IEEE has made some data bases available as OCR standard data bases, the most widely used of which are due to Munson [1968] and Highleyman [1961, 1962] but use of consistent input data by all researchers is a goal not yet achieved.

Although, there is a plethora of published papers on character recognition, I shall focus on the most significant contributions; a detailed bibliography of all papers in this field can be found in Suen, et.al.[1978, 1980]. In the following sections, OCR systems are reviewed according to the methodologies used in designing their recognition logic. A review of pre-processing techniques is also introduced in subsequent sections.
2.2 Template matching and correlation

In its simplest form, template matching/correlation takes states (black and white) of those points which lie within the smallest white rectangular border enclosing a character (frame) to compare with all or part of a reference pattern. The output of this process is an array that gives the degree of match at each point (or a group of points) in the input pattern. Such methods are size dependent and input patterns must be normalized before they are matched against stored templates.

One of the earliest correlation techniques was proposed by McLaughlin et al. [1968]. Regarding each pattern as a point in a vector space, they attempt to map all points corresponding to translated (or scaled) versions of one pattern into a single point. Patterns which differ in other ways should map into distinct points and in some sense patterns which are similar should map into points that are close to one another. Given a real valued function $x(t)$, its $n$th order correlation function is defined by:

$$r_n(t_1, \ldots, t_n) = \int x(t)x(t + t_1) \cdots x(t + t_n)dt$$  \hspace{1cm} (2.1)

It is easy to see that this function is translation invariant in the sense that
x(t) and y(t) = x(t + \tau) have the same n th. order auto-correlation function. The use of these functions in pattern recognition was suggested by Horowitz and Shelton[1965], but their experiments were limited to \( n \leq 2 \) because of the high dimensionality of the n th. order auto-correlation space. If each \( \tau \) ranges over m values, n th. order autocorrelation space has dimension \( m^n \).

Mclaughlin et.al.[1968] extended Horowitz’s result to use nth order autocorrelations in pattern recognition without explicitly computing the autocorrelations and thus avoiding the above limitation.

Assuming that \( r_x(\tau_1, ..., \tau_n), r_y(\tau_1, ..., \tau_n) \) are autocorrelations of patterns x(t) and y(t), the following inner product in \( R_n \) can be derived:

\[
<r_x, r_y> = \int \cdots \int r_x(\tau_1 \cdots \tau_n) r_y(\tau_1 \cdots \tau_n) d\tau_1 \cdots d\tau_n
\]

\[
= \int \cdots \int x(t) x(t + \tau_1) \cdots x(t + \tau_n) dt \cdot \int y(v) y(v + \tau_1) \cdots y(v + \tau_n) dv d\tau_1 \cdots d\tau_n
\]

\[
= \int \int x(t)y(v)[\int x(t + \tau)y(v + \tau)d\tau]^n dvdt
\]

\[
= \int \int x(t)y(u + t)[\int x(s)y(u + s)ds]^n dudt
\]
where \( v = u + t \) and \( s = t + r \).

thus:

\[
<r_x, r_y> = \int \int x(s)g(s+u)ds^1du^{n+1}
\]  \hspace{1cm} (2.2)

Restricting eqn. 2.2 to finite dimensional spaces, \( t \) and \( \tau \) taking only a finite set of values, the discrete inner product between autocorrelation of patterns \( x_i(t) \) and \( x_j(t) \) is given by:

\[
<r_i, r_j> = \sum_{\tau} \left[ \sum_t x_i(t)x_j(t+\tau) \right]^{n+1}
\]  \hspace{1cm} (2.3)

where \( r_i \) and \( r_j \) are the corresponding column vectors representing the n-th order autocorrelation functions of patterns \( x_i(t) \) and \( x_j(t) \).

The angle between autocorrelation vectors is used to compute a measure of similarity between two patterns. In particular, let \( x_1(t), \ldots, x_c(t) \) be the reference patterns for \( c \) classes and let \( r_1, \ldots, r_c \) be the n-th order autocorrelations (where \( t \) is a discrete two dimensional vector corresponding to spatial coordinates and ranges over a finite set of values). Suppose \( r \) is the n-th order autocorrelation of a pattern \( x(t) \) which is to be classified. The angle between the autocorrelation vectors is calculated by:
for each class $i$ which gives the squared cosine of the angle between $r$ and $r_i$ in the $n$-th order autocorrelation space. Then $x(t)$ is assigned to a class $k$ such that $S_k \geq S_i$ $i = 1, 2, ..., c$.

Experiments [Maclaughlin et al., 1968] on type written characters show that a minimum of a 9th order autocorrelation function is necessary for a good recognition performance. Although only 3 mistakes were detected out of more than 1300 input characters, the character set was limited to letters 'a', 'e' and 's'.

Shimura [1973] describes a pattern matching technique which is based on a non parametric classification method. The problem is to classify input patterns into one of $c$ categories $c_1, ..., c_k, ..., c_c$. Let $X = (x_1, x_2, ..., x_n)'$ where "\'" indicates transpose, be the $n$-dimensional pattern vector of which the component is "1" or "0" (i.e. a binary image) and $Z(\mu) = (z_{1\mu}, z_{2\mu}, ..., z_{n\mu})$ be the template pattern vector of the category $c_\mu$. Assume that the vector $\bar{Z}(\mu) = (\bar{z}_{1\mu}, \bar{z}_{2\mu}, ..., \bar{z}_{n\mu})'$ is defined as follows:
\[ z_i^\mu = \begin{cases} 
1 & \text{if } z_i^\mu = 0 \\
0 & \text{if } z_i^\mu = 1 
\end{cases} \]

and called the complement pattern vector of the template pattern \( Z(\mu) \).

The traditional multiclassifier (or machine 0) is shown in figure 2.6. It's training process has been analyzed by Nilsson [1965] in detail. The weighted sum of the binary pattern is given by the accumulator and is represented in the form:

\[ g_\mu(x) = W^\mu X \]

\[ \mu = 1, 2, \ldots, \sigma \]

\[ W^\mu = (w_1^\mu, w_2^\mu, \ldots, w_n^\mu) \]

The category to which the input pattern belongs is category \( C_\alpha \) if the weighted sum \( g_\alpha(x) \) is maximum, this machine therefore requires \( n \times \sigma \) variable weighting devices. Shamura's classifier (or machine I, fig.2.7) transforms the input pattern \( X \) to \( X \oplus \bar{Z}(\mu) \) by the preprocessor, where the symbol \( \oplus \) indicates "exclusively OR" and \( \bar{Z}(\mu) \) is the complement pattern vector of \( Z(\mu) \). That is the vector \( X \oplus \bar{Z}(\mu) \) becomes \( (1, 1, \ldots, 1)' \) by the above transformation if the input pattern \( X \) is the template pattern of category \( C_\mu \), and
Figure 2.7: Learning classifier for character reading, Machine I.
Figure 2.8: Learning classifier for character reading, Machine II.
it becomes \((0,0,...,0)\)' if \(X\) is the complement pattern of the template.

The input of the maximum selector \(C_\mu\) is given by the switches shown in figure 2.6. Thus the maximum selector of this machine must have short term memories to select the maximum value of \(g_\mu(X), \mu = 1, 2, ..., \sigma\).

In comparison with machine zero, this machine is able to reduce the number of weighting devices by using the template pattern of each category. Let \(X_k = (x_{1k}, x_{2k}, ..., x_{nk})'\) be the pattern vector and \(W_k = (w_{1k}, w_{2k}, ..., w_{nk})\) be the weight vector at the \(k\)th iteration in the training sequence. After each presentation of input patterns, a decision is made whether or not to change the weight vector based upon the output response obtained.

A training rule is also defined in which changes of weight vector are made if and only if an error occurs. A more sophisticated version of machine I (called machine II) is also proposed by Shimura; it has the capability of finding out the template patterns by iterative sequence when they are unknown. The classification structure is the same as that of machine I, as shown in figure 2.8. The template pattern can be considered as a standard pattern and hence its vector is a mean vector of training patterns. The mean vector can be obtained by the following algorithm. Let \(n_k(\mu)\) be the number of the input patterns of the category \(c_\mu\) which are classified into the wrong
Figure 2.9: Pattern matching of character 'E'.
category until the k th. step and γ be the set of such patterns.

Parameters $\sigma_{k+1}^j$ are introduced as follows:

$$\sigma_{k+1}^j(\mu) = \frac{\sum \gamma x_{ijk}}{n_k(\mu)}$$

$\mu = 1, 2, \cdots, \sigma$

$j = 1, 2, \cdots, n$

The value $\sigma_{k+1}^j(\mu)$ may be determined sequentially according to the iterative rule:

$$\sigma_{k+1}^j(\mu) = \frac{(n_k(\mu) - 1)\sigma_k^j(\mu) + x_{jk}}{n_k(\mu)}$$

The j th. component of the template pattern vector $Z_{k+1}^\mu$ of $C_\mu$ at the k+1 st iteration is given by:

$$Z_{j,k+1}^\mu = \begin{cases} 1 & \text{if } \sigma_{k+1}^j(\mu) \geq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

and

$$Z_{k+1}(\mu) = (Z_{1,k+1}^\mu, Z_{2,k+1}^\mu, \cdots, Z_{n_K+1}^\mu)'$$

$\mu = 1, 2, \cdots, \sigma$
Note that the initial template vector $Z_t(\mu)$ is only the input pattern vector given to the machine initially. The interesting point about machine II is that, it can derive a new reference pattern (template) by averaging each element of the training samples iteratively, thereby introducing topological variations of each training pattern in the template.

A recognition rate (correct classification) of 84.5% of handwritten characters for both machines is also reported. A much simpler method developed along similar lines [Freedman, 1974] compares the candidate (input) pattern with a stored representation (or pattern) of each character directly and selects the closest match; figure 2.9 illustrates the application of this technique to identify character 'E' under ideal conditions.

Proper alignment of scanner output 'E' is seen to result in a complete registration. The best possible alignment of the input data 'F' and the 'L' templates on the other hand, yields poor registration. The bottom of 'E' is mismatched in the case of the 'F' template and the centre and the top of the 'E' are mismatched in the case of the 'L' template. Recognition is completed on the basis of the 'best match' using a criterion such as the minimum 'hamming' distance between the classifier input and the set of stored templates. This particular technique was implemented on OCR machines in the late
Figure 2.10: Example of Holt's pattern matching technique [1985].
fifties and the sixties, but it was only used on relatively clean data due to its susceptibility to noise and distortion. A more effective approach however is to 'X OR' a candidate pattern against a template; the resulting matrix contains the discrepancy between the two patterns (i.e. points that do not match).

The weighted Exclusive OR (WXOR) error map is obtained from the unweighted XOR map by replacing each error with the error count in the surrounding 3 X 3 neighbourhood. In this way greater weight is given to clustered errors which occur with distinct symbol patterns than sparse errors which occur along the common boundary of similar symbol patterns (fig.2.10).

In the pattern matching and substitution (PMS) scheme [Johnsen, et.al. 1983] a matching decision is determined by the WXOR map using only local criteria. A match is rejected if either (1) any error pixel has a weight of 5 or more, or (2) any error pixel has both (i) two neighbouring error pixels which are not adjacent to each other and (ii) a 3 X 3 neighbourhood in which all 9 pixels are the same colour in one of the two symbol patterns. Rule 1 rejects most distinct patterns, but rule 2 is included for pair of distinct symbols whose difference is manifested only in narrow strokes or gaps.

The Combined Symbol Matching (CSM) scheme [Pratt, et.al., 1980] de-
terminates a match by comparing the sum of the weighted errors (the WXOR count) against a sliding scale of the threshold values related to symbol size. The scale of threshold values is not quoted, but is said to be a non-linear function of the symbol's black pixel count, obtained from empirically determined look-up table. A refined version of the above techniques was proposed by Holt, et.al. [1986].

The weighted errors in this scheme are evaluated in a different way. The two symbol patterns are superimposed and elements which differ are indicated in an error map. Each element is then tagged in one of the two ways according to which of the symbol patterns has a black pixel in the corresponding position. Examples of these error maps are given in figure 2.11 where '+ ' denotes pixels which are black in the first symbol but not the second and '- ' denotes error pixels which are black in the second symbol but not the first. The weighted error map is then obtained as follows:

Each '-' error is given a weight equal to the number of '-' errors in a $3 \times 3$ neighbourhood centred on the element in question. Similarly each '+' error is weighted according to the number of the neighbouring '+' errors. Elements not tagged in the error map are given zero weight. The resulting weights are summed up over the whole error map. As in the CSM algorithm
Figure 2.11: 'Weighted Exclusive OR' pattern matching.
the matching decision is then made by comparing the weighted error count against an empirically determined sliding scale of threshold values.

Holt's scheme is a robust pattern matching technique most suited for recognition of printed characters.

Among other significant contributions on correlation and pattern matching are:

- the scheme devised by Golshan et al. [1970] based on physical measurement of vertical and horizontal features of a character and matching them against predefined features.

- The Kozlay's OCR system [1971] which relies on a series of correlator reference patterns generated for a 2 class discrimination. A feature ordering tree is used in the sequential decision process; class probabilities at each branch of the tree are also used to classify samples.
2.3 Transformation and series expansion

Transform theory has played a key role in image processing for a number of years and continues to be a topic of interest in this field while making inroads into the field of pattern/character recognition. The most useful property of transform theory in pattern recognition is its ability to compress data as well as highlighting significant features which can be used for recognition purposes. An important class of such transforms is the rotational transformations. The one dimensional discrete Fourier transform is one of the important rotational transformations which can be expressed in terms of the general relation [Gonzalez, 1987]:

\[ T(u) = \sum_{x=0}^{N-1} f(x)g(x, u) \]

where \( T(u) \) is the transform of \( f(x) \), \( g(x,u) \) is the forward transformation kernel and \( u \) assumes values in the ranges 0,1, ..., N-1. Similarly, the inverse transform is given by the relation:

\[ f(x) = \sum_{u=0}^{N-1} T(u)h(x, u) \]

where \( h(x, u) \) is the inverse transformation kernel and \( x \) assumes values
in the range 0,1,..., N-1. The nature of a transform is determined by the properties of its transformational kernel. For two dimensional arrays the forward and inverse transforms are given by the equations:

$$T(u,v) = \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x,y)g(x,y,u,v)$$

(2.4)

and

$$f(x,y) = \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} T(u,v)h(x,y,u,v)$$

(2.5)

where as above $g(x,y,u,v)$ and $h(x,y,u,v)$ are called the forward and inverse transformation kernels respectively. The forward kernel is said to be separable if

$$g(x,y,u,v) = g_1(x,u)g_2(y,v)$$

(2.6)

In addition, the kernel is symmetric if $g_1$ is functionally equal to $g_2$. In this case, eqn.(2.6) can be expressed in the form

$$g(x,y,u,v) = g_1(x,u)g_1(y,v)$$

(2.7)

Identical comments hold for their inverse if $g(x,y,u,v)$ is replaced by
h(x,y,u,v) in eqns. (2.6) and (2.7). The two dimensional Fourier transform is a special case of eqn. 2.4. It has the kernel \( g(x, y, u, v) = \frac{1}{N} \exp\left[-j\frac{2\pi(xu+yv)}{N}\right] \)

which is separable and symmetric since

\[
g(x, y, u, v) = g_1(x, u)g_1(y, v) = 
\frac{1}{N} \exp\left[-j\frac{2\pi u x}{N}\right] \frac{1}{\sqrt{\pi}} \exp\left[-j\frac{2\pi v y}{N}\right]
\]

It is easily shown that the inverse Fourier kernel is also separable and symmetric. A transform with a separable kernel can be computed in two steps, each requiring a one dimensional transform. First, the one dimensional transform is taken along each row of \( f(x, y) \) yielding:

\[
T(x, v) = \sum_{y=0}^{N-1} f(x, y)g_2(y, v)
\]

for \( x, v=0,1,2,\ldots,N-1 \). Next, the one dimensional transform is taken along each column of \( T(x,v) \); this results in the expression:

\[
T(u, v) = \sum_{x=0}^{N-1} T(u, v)g_1(x, u)
\]

for \( u,v = 0,1,2,\ldots,N-1 \). If the kernel \( g(x,y,u,v) \) is separable and symmetric eqn.2.4 can also be expressed in the following matrix form:
where $F$ is the $N \times N$ image matrix, $A$ is an $N \times N$ symmetric transformation matrix with elements $a_{ij} = g_1(i,j)$ and $T$ is the resulting $N \times N$ transform for values of $u$ and $v$ in the range $0, 1, 2, ..., N-1$. To obtain the inverse transform, eqn. 2.8 is pre-multiplied as well as post-multiplied by an inverse transformation matrix $B$, which yields the expression: $B \ T \ B = B \ A \ F \ A \ B$.

If $B = A^{-1}$, it then follows that

$$F = BTB$$

(2.9)

which indicates that the digital image $F$ can be recovered completely from its transform. If $B$ is not equal to $A^{-1}$, then use of eqn. 2.9 yields an approximation to $F$ given by the relation

$$\hat{F} = BAFAB$$

Transforms including Fourier, Walsh, Hadamard and discrete Cosine transforms can be formulated in the form of eqns. 2.8 and 2.9. An important property of the resulting transformation matrices is that they can be decomposed into products of matrices with fewer non-zero entries than the original.
matrix [Good, 1958], thereby reducing the computational effort. Rotational transformations provide a measure of correlation among feature dimensions. Andrews [1971] and Wending [1978] used Walsh-Hadamard transform for character recognition employing a Euclidean distance for final classification. Another interesting transformation is the Karhunen-Loeve transform which unlike other transforms is based on the statistical properties of a pattern and provides a feature space in which the largest variance eigen value and energy dimensions can be solved for feature classification purposes. To formulate the problem, suppose that an N x N image \( f(x, y) \) is to be classified and each sample image \( f_i(x, y) \) of the image array is expressed in the form of an \( N^2 \) dimensional vector \( x_i \) as follows:

\[
x_i = \begin{bmatrix}
x_{i1} \\
x_{i2} \\
\vdots \\
x_{ij} \\
x_{iN^2}
\end{bmatrix}
\]

where \( x_{ij} \) denotes the \( j \)th component of vector \( x_i \). The common way to construct such a vector is to let the first \( N \) component of \( x_i \) be formed from the first row of \( f_i(x, y) \) ; the second set of \( N \) components from the second row and so on. The covariance matrix of the \( x \) vectors is defined as:
\[ C_x = E[(x - m_x)(x - m_x)'] \]

where \( m_x = E[x] \) is the mean vector, \( E \) is the expected value; eqns. 2.10 and 2.11 can be approximated using the relations.

\[
m_x = \frac{1}{M} \sum_{i=1}^{M} x_i
\]

and

\[
C_x = \frac{1}{M} \sum_{i=1}^{M} (x_i - m_x)(x_i - m_x)'
\]

or equivalently

\[
C_x = \frac{1}{M} \left[ \sum_{i=1}^{M} x_i x_i' \right] - m_x m_x'
\]

The mean vector is of dimensionality \( N^2 \) and \( C_x \) is an \( N^2 \times N^2 \) matrix. Let \( e_i \) and \( \lambda_i \), \( i = 1, 2, \ldots, N^2 \), be the eigenvectors and corresponding eigenvalues of \( C_x \). It is assumed for convenience in notation that the eigenvalues have been arranged in decreasing order so that \( \lambda_1 \geq \lambda_2 \geq \ldots \geq \lambda_{N^2} \). A transformation matrix whose rows are the eigen vectors of \( C_x \) is given by:

\[
A = \begin{bmatrix}
    e_{11} & e_{12} & \cdots & e_{1N^2} \\
    e_{21} & e_{22} & \cdots & e_{2N^2} \\
    \vdots & \vdots & \ddots & \vdots \\
    e_{N^21} & e_{N^22} & \cdots & e_{N^2N^2}
\end{bmatrix}
\]
where $e_{ij}$ is the $j$th component of the $i$th eigenvector. The Karhunen-Loeve transform then consists simply of multiplying a centralized image vector, $(x - m_x)$ by $A$ to obtain a new image vector $y$, that is

$$y = A(x - m_x)$$

The covariance matrix of $y$ vectors is given by:

$$C_y = E[(y - m_y)(y - m_y)']$$

where $m_y = E[y]

$$E[y] = E[A(x - m_x)]$$

$$= AE[x] - Am_x$$

$$= 0$$

(1)

substituting eqns. 2.12 and 2.13 into eqn 2.14 yields:

$$C_y = AE[x - m_x](x - m_x)'A'$$

$$= AC_xA'$$

(2)
It can be shown (Lawley and Maxwell, 1963) that $C_y$ is a diagonal matrix with elements equal to the eigenvalues of $C_x$; that is:

$$C_y = \begin{bmatrix}
\lambda_1 & 0 & 0 & 0 & 0 \\
0 & \lambda_2 & 0 & 0 & 0 \\
0 & 0 & \lambda_3 & 0 & 0 \\
0 & 0 & 0 & \lambda_4 & 0 \\
0 & 0 & 0 & 0 & \lambda_{N^2}
\end{bmatrix}$$

The importance of this property is that, since the terms off the main diagonal are zero, the elements of $y$ are uncorrelated. In addition, each eigenvalue $\lambda_i$ is equal to the variance of the $i$th element of $y$ along eigenvector $e_i$. One of the major drawbacks of Karhunen-Loeve transform is the need to diagonalize a usually large covariance matrix. Krause, et al. [1973] converted the binary matrices of characters into a Karhunen-Loeve series from which 40 coefficients (eigenvalues) were used as features. Gudesen [1976] also extracted the eigenvectors from a modified covariance matrix of the whole character sample for subsequent use of a Bayes classifier. Although, implementation of transform techniques in character recognition seem to offer some improvement in recognition rate, but, since they have to operate on all pixels within the frame (bordering a character) they still impose a heavy computational burden on the overall recognition scheme. The most interesting of trans-
formation techniques, however, is the 'Fourier Descriptors' which has been
used widely for boundary description of a pattern in terms of tangent angle
versus arc length or a complex function $ax(l) + jay(l)$ where $ax(l)$ and $ay(l)$
denote the boundary coordinates. The former has been used by Brill[1968],
Zahn and Roskies[1972]; while the latter has been used by Grandlund[1972],
Richards et.al.[1974] and Persoon et.al. [1977]. Zhan et.al. [1972]'s Fourier
descriptor (FD) are defined as follows:

We assume $\gamma$ is a clockwise simple closed curve with $m$ parametric rep-
resentation $(x(l), y(l)) = z(l)$ where $l$ is arc length and $0 \leq l \leq L$. Denote
the angular direction of $\gamma$ at point $l$ by the function $\theta(l)$. Let $\sigma_0 = \theta(0)$ be
the absolute angular direction at the starting point $Z(0)$. We now define the
cumulative angular function $\phi(l)$ as the net amount of angular bend between
starting point and point $l$ as depicted in fig.2.12a . With this definition
$\phi(0) = 0$ and $\phi(l) + \sigma_0$ is identical to $\theta(l)$ except for a possible multiple of
$2\pi$. If the curve $\gamma$ winds in a spiral then $\phi(l)$ achieves values larger than $2\pi$
. Figure 2.12.b shows $\phi(l)$ for the shape depicted in fig.2.12.a . The domain
definition of $[0, L]$ of $\phi(l)$ simply contains absolute size definition and we wish
to normalize it to the interval $[0, 2\pi]$ which is standard for periodic functions.
Hence a normalized variant $\phi(t)$ is defined by:
Figure 2.12: a) Parametric representation of a plane curve with tangential direction; b) Tangent or velocity of plane curve.

Figure 2.13: A typical curve...
\[ \phi^*(t) = \phi\left(\frac{Lt}{2\pi}\right) + t \]

where \( t \) ranges from 0 to \( 2\pi \) such that \( \phi^*(0) = \phi(2\pi) = 0 \). Note that \( \phi \) is invariant under translation, rotations and changes of perimeter \( L \). We now expand \( \phi^* \) as a Fourier series:

\[
\phi(t) = A_0 + \sum_{k=1}^{\infty} \left( a_k \cos kt + b_k \sin kt \right)
\]

and in polar form, the expansion is:

\[
\phi^*(t) = A_0 + \sum_{k=1}^{\infty} A_k \cos(kt - \alpha_k)
\]

where \((\alpha_k, A_k)\) are polar coordinates of \((a_k, b_k)\). These numbers \( A_k \) and \( \alpha_k \) are the Fourier descriptors for curve \( \gamma \) and are known as the \( k \) th. harmonic amplitude and phase angle. In practice a contour is described by a set of discrete points, thus discrete Fourier transform of \( \phi^*(t) \) must be computed. A typical contour is depicted in fig.2.13. Let \( \phi(k) \) be the rotation to a boundary at the \( k \) th. point in comparison to the tangent at an initial point \( x(0), y(0) \) and \( l_k \), the arc length between these two points. These two quantities can be defined as follows:
\[ l_k = \sum_{i=1}^{k} \sqrt{[x(t_i) - x(t_{i-1})]^2 + [y(t_i) - y(t_{i-1})]^2} \]

\[ k = 1, 2, ..., N + 1 \]

\[ \phi(k) = \tan^{-1} \frac{y(t_{k+1}) - y(t_k)}{x(t_{k+1}) - x(t_k)} - \tan^{-1} \frac{y(t_l) - y(0)}{x(t_l) - x(0)} \]

\[ k = 1, 2, ..., N + 1 \]

Note that \( \phi(0) = 0 \), \( \phi(N + 1) = 2\pi \) and that the boundary can be reconstructed by \( \phi(k) \) versus \( l_k \). The Fourier transform of a set of discrete boundary points is given by [Zahn, 1972]:

\[ A_o = -\pi - \left( \frac{1}{L} \right) \sum_{k=1}^{N+1} l_k(\phi_k - \phi_{k-1}) \]

\[ A_n = -1/N\pi \sum_{k=1}^{N+1} (\phi_k - \phi_{k-1}) \sin(2\pi n l_k/L) \]

\[ B_n = \frac{1}{N\pi} \sum_{k=1}^{N+1} (\phi_k - \phi_{k-1}) \cos(\frac{2\pi n l_k}{L}) \]

so that \( \phi^*(t) = A_0 + \sum_{n=1}^{N+1} [A_n \cos(nt) + B_n \sin(nt)] \). Granlund’s formulation [1972] which was further developed by Persoon et.al. [1977] is defined by:
\[ G_n = \frac{1}{L} \int_0^L a(t)e^{-j2\pi nt/L} dt \]

where \( a(t) \) is the complex function \( ax(t) + jay(t) \). The length of the curve from an original point can serve as the parameter \( t \), therefore, the derivative of \( a(t) \) with respect to \( t \) is:

\[ |\frac{da}{dt}| = \sqrt{(\frac{da}{dt})^2 + (\frac{db}{dt})^2} = 1 \]

because \( dt = \sqrt{(dax)^2 + (day)^2} \). This means that the coefficients \( G_n \) are not independent, thus, there is redundancy in the description. In general the major advantage of Fourier transform descriptors is their ease of computation. Their major disadvantage is the difficulty in describing local information [Persoon, 1977]. They can also distinguish among symmetric curves only on the basis of the phase of the descriptors, which can not be computed reliably in many cases [Zahn, 1972]. In the Zahn's formulation \( A_k \) and \( \alpha_k \) describe not closed curves; \( \phi^* \) for polygonal curves contains discontinuity and therefore \( A_k \) will decrease slowly as \( k \) increases. Reconstruction of the curve \( (\gamma) \) requires numerical integration. Granlund's approach however, is expected to have a faster rate of convergence as well as the property that
its inverse transform always produces closed curves. Persoon et.al. [1977] used such descriptions for hand written numerals and machine parts. They normalize each set for scaling, rotation and starting point. A nearest neighbour rule using Euclidean distance is used for pattern classification. Series expansion/Transformation methods are generally noise tolerant and provide a robust performance in presence of global translation or rotation.
2.4 Structural Recognition of characters

The origin of structural recognition of characters can be found in the middle of 1950's with the development by Noam Chomsky of mathematical models of grammars related to his work in natural languages. One of the original goals of the work of linguists working in this area was to develop computational grammars capable of describing natural languages such as English. The hope was that, if this could be done, it would be a relatively simple matter to teach computers to interpret natural languages for the purposes of translation and problem solving. Although, it is generally agreed that these expectations have not been realised thus far, the spin-offs of research in this area have had significant impact on other fields such as compiler design, computer languages, automata theory and more recently pattern recognition and image processing. In pattern recognition the idea is to break a pattern (or in our case a character) to simpler strokes or primitives and somehow derive the relationship which these primitives must have with one another to form a particular pattern. Murray Eden [1961, 1962, 1968] showed for the first time that English alphabet (upper and lower case) can be formed by 18 primitives as depicted in fig.2.14 which can be reduced to 4 basic shapes as shown in fig.2.15. In practice, however, these four basic
Figure 2.14: 18 primitives proposed by Eden [1961].

Figure 2.15: 4 basic primitives proposed by Eden [1962].
shapes may not be detected and recognised due to noise and varying styles of writing as well as difficulties in segmenting the character. To make use of this method, strokes and curves facing different directions as depicted in fig.2.16 have been used. A few other researchers have also made use of end points and intersections of line segments as depicted in fig.2.17. In the subsequent sections the most important contributions in these two basic trends are considered.

2.4.1 Structural Analysis/Recognition using strokes and curves

Tou et al. [1972] propose a multilevel recognition scheme based on centring the thinned pattern on an octagonal grid shown in fig.2.18. Each octant is searched for a series of features which may describe a stroke (primitive). If no features are detected in all 8 octants, the pattern is rejected, otherwise, the detected features are examined to determine whether they represent a unique pattern. Should the first level to recognise the character fail, the second stage is then activated to look for more "specialised" features which discriminate between the patterns under consideration. The function of the second stage is to resolve the ambiguities created by a lack of information in the features of the first stage. These features are compiled experimentally. A correct recognition rate of 94 percent on all tested numerals and upper case
Figure 2.16: Strokes and curves used as primitives.

Figure 2.17: End points and intersection of lines used as primitives.
Figure 2.18: Tou's Octagonal grid [1972].
characters is reported; of the remaining characters 4 percent were rejected and 2 percent were misrecognised.

Unfortunately, the authors give no indication of the number of characters used in testing the scheme. Siy et.al.[1974] use fuzzy theory in their scheme to describe primitives. They distinguish 15 primitives as depicted in fig.2.19.

Each primitive is described by:

a) measure of straightness: The measure of straightness of a primitive is determined by fitting a straight line with the minimum least square error; and the primitive is represented by the line with the least square error.

b) Measure of orientation: If the classification of a primitive (to the class of a straight line, a portion of a circle or a circle with node) is known, then the measure of orientation can be used to further characterize this branch.

The slope of the best fit line is used as a measure of straight line orientation while the slope of the line connecting the end points of a primitive is used to quantify orientation of a "portion of circle". Both of the measures are modelled using fuzzy theory and they are based on empirical data. The block diagram of the scheme is shown in fig.2.20. In the learning phase the skeleton of the character is obtained using Hilditch 's thinning algorithm [1969], the pattern is then passed to the labelling process where various nodes
Figure 2.19: Siy’s proposed primitives.
Figure 2.20: Block diagram of the recognition system. a) Learning phase, b) Recognition phase.
defined as the collection of tips (points that have one neighbour), corners (points that have two neighbours and where one abrupt change of line direction occurs) and junction (points that have three or more neighbours) are detected and labelled. Subsequently, branches (primitives) connecting adjacent nodes are also labelled using measures of straightness and orientation. Once the labelling process is completed, the pattern is functionally coded to describe its shape using detected primitives. The functional codes are stored as prototypes for recognition. In the recognition phase the incoming pattern passes through all the stages detailed above, before being matched against reference codes already stored. The authors report a correct classification of 98.4 percent out of a total sample of 5000 handwritten numerals. Pavlidis [1976], Ali et al. [1977] and Pavlidis et al. [1979] propose a different approach in detecting primitives. Their scheme does not include any character thinning, instead the pattern’s contour is polygonal approximated using the split and merge [Horowitz, 1975, Pavlidis et al. 1974 and 1975] to eliminate noise. The pattern can now be described using four basic primitives:

1- Arcs which can be approximated by a quadratic curve called 'QUAD'.
2- Sharp protrusions or intrusions called 'TRUS'.
3- Long linear segments (line).
4- Short segments having no regular shape are called 'BREAK', which give rise to a pair of primitives namely:

5- Stroke: made of two lines whose angles differ from 180 degrees less than a threshold ($\theta$).

6- Corner: made of two lines which form an angle greater than a threshold ($\theta$).

A 4 x 3 grid superimposed on the pattern also provides location attributes of the primitives. In the recognition process, the boundary coded accordingly is matched against the strings already stored in the reference library; if no match is found, the string is assigned to the class which matches the string with the highest confidence level. The authors report an overall error rate of 5.37 percent and 4.49 percent of rejection on a total of 1320 hand written numerals. Pavlidis [1983] adopted a similar approach for thinned printed characters. The new proposal describes thinned and a polygonally approximated pattern using arcs (concave or convex), loops and strokes. Positions of points are determined with respect to the bounding rectangle of the skeleton. Three vertical and three horizontal zones are used for a total of nine positions. Initially, each character is placed in one of the following six classes.

(1) All branches form a single loop (e.g. 0, O, D)

47
(2) There is exactly one loop (e.g. a, b, d, p, q, P)

(3) There are exactly two loops (e.g. B, 8)

(4) There is a dot over the character (e.g. i, j)

(5) There is no node of degree different than two in a middle horizontal zone (e.g. l, c)

(6) All other characters.

The assignment to such classes is done sequentially. If a character is found to belong to more than one class, then it is placed in the one coming first in the above list. The algorithm then attempts to detect features which identify strokes, arcs, loop location, right angles or connected components as nodes of degree one (end points). These features are arranged in the form of binary vectors [Duda, 1979] which are used for classification. Nine fonts of excellent quality (phototype ready) were selected for the initial design. Six with serifs: Bemaul, Palatino, Italic, Times Roman, Times Bold, Times Italic; and three without serifs: Avantguarde Light, European, Helvetica Bold. The classes considered were the upper and Lower case alphabets, ten numerals and the ' & ' sign. For each character the contour was scaled into one of twelve sizes and then scan converted; thus there were 108 samples per character or a total of 3402 samples samples per each set. Half of the samples were
used as a design set and the rest as a test set. The experiment reported
show a correct recognition rate above 99%. Pavlidis’s ideas were further
developed by Baird[1986] and Kahan et.al. [1987]. Kahan et.al. describe
a fully engineered omnifont character recognition system. It is essentially
made of five stages as depicted in fig.2.21. A character is first extracted
from the document; it is then normalised to a pre-defined size to make the
recognition process size invariant. At the next stage the character is thinned
using the run length information as well as its Line Adjacency Graph (LAG)
to derive:

1- The number of holes.
2- Approximate hole location and size from subgraph matching on LAG.
3- Skeletal; concavities in the skeletal structure.
4- Crossings of strokes.
5- End points in the vertical direction.
6- The bounding box of the character, defined as the smallest upright
rectangle containing the character.

Fixed length binary vectors are used to describe shapes like strokes,
arcs, holes, etc. For example a stroke is described by the coordinates of its
end points and its centre as well as the stroke’s orientation defined in a spe-
Figure 2.21: Kahan, et.al.'s OCR scheme.
cial way [Kahan, et.al., 1987]. All these parameters are clustered in a feature space. Those strokes which are similar lie close to one another in the feature space whilst the dissimilar ones lie apart. A statistical Bayesian classifier using binary features was also employed for shape classification. The set of stroke clusters found over an entire alphabet numbers about 500 which can undergo a merging process to remove any redundancies. About 100 distinct stroke clusters were identified. The number of clusters overall, including those holes, arcs, etc. starts out at about 1500 and is reduced by merging to about 300. Should the recognition algorithm fail to recognize the character, other additional information such as document layout and linguistic information are employed to recognize the character. The authors report extensive experiments to test the system’s performance which are summarised here:

An alphabet of 70 characters were used including upper and lower case letters, numerals and &,(,),−,\$,,][,*,'" signs. Print quality was excellent (original phototype setter output), but samples at small point sizes (8 and 10 points) were used to simulate noisy images. A total 196000 character images was used: 40 samples of each 70 characters at seven point sizes for ten fonts. Since, the scanner resolution was 200 pels per inch (ppi), high recognition performance was expected on characters of size 12 points and
above. For most single fonts a top choice performance of above 99.5% has been achieved. At or below 10 point, performance falls to well below 98%. Performance also declines on mixture of dissimilar fonts to 97 percent for six fonts.

Another significant contribution was proposed recently by Lam et.al.[1988] in which relaxation matching was incorporated to increase reliably as well as efficiency. Although the method itself was tailored to recognize unconstrained hand written numerals, it can also be used in recognition of alphabet characters with some modifications in the algorithm's implementation. The scheme is made of 4 parts as depicted in fig.2.22.

a) Pre-processing: An input character is first smoothed and skeletonized; the skeleton is then approximated by line segments.

b) Feature extraction: The character is normalized in size and decomposed into its basic sub-patterns or primitives consisting of convex polygons and line segments. Features are extracted from each primitive.

c) Structural classification: Based on the configuration of primitives, patterns with simpler structures are classified by decision trees according to the primitives and their extracted features. Convex polygons and line seg-
Figure 2.22: Lam et.al.'s OCR scheme, 1988.
ments are used as primitives. The polygons are classified as EAST, NORTH, WEST, and SOUTH as shown in fig.2.23.

d) Relaxation matching: The patterns not recognized by the structural classifier are matched against a set of templates using a relaxation process and the distance between the pattern and each template is calculated. The pattern is then classified according to a minimum distance criterion.

In the training process about 5000 samples were used; of these about 1000 were used to select an initial set of masks for relaxation matching and the other 4000 (400 samples per class) were processed repeatedly to revise the rules for structural classification and at the same time to advance the mask selection process. The testing set consists of 2000 samples (200 per class) not used in the training process. The use of the two classification algorithms in conjunction resulted in an overall recognition, rejection and substitution rates of 96%, 0.85% and 3.15% respectively.

2.4.2 End points and intersection of lines and loops.

End points, intersection of lines and loops are not frequently used to describe a character on their own; they have, however, been employed as back up features to assist recognition in some OCR schemes [e.g. Kahan et.al. 1987]. In the following, important contributions on this technique
Figure 2.23: Primitives used in the Lam's scheme.
are discussed. M. Beun [1973] proposes an OCR system based on end points, intersection of lines and loops. The segmented character is first thinned and then the end points (points with only one black neighbour), junctions (points with three or more black neighbours) and their relative positions with respect to the centre of the encasing rectangle are defined. Each character is modelled using these attributes. The system reported is tailored to recognize handwritten numerals. The algorithm depicted in fig. 2.24 can be described as follows: Let E and F denote the number of end points and the number of junctions (for K points) respectively. If E and F are both equal to zero, then it must be numeral '0', or if (E=0) and (F=0), then it is '8'. If both E and F equal 1, then it must be either '6' or '9'. To resolve the ambiguity, the position of end points is examined. If it is above the junction then it ought to be '6' otherwise it is '9'. In the fourth category numerals '1', '2', '3', '5' and '7' are put in one category. Since they all have two end points (E=2) and no junction (F=0). To distinguish between these numerals, the total number of black points forming the skeleton are counted which is termed W (for weight). Next, the number of rows that the skeleton covers in the rectangle is counted denoted by H (for height). For all '1' numerals W=H which does not apply to the other numerals. The '1' are thus eliminated. To select the
<table>
<thead>
<tr>
<th>(E,F)-GROUP</th>
<th>TYPE OF MINERAL</th>
<th>PROCEDURE</th>
<th>RECOGNIZED AS</th>
</tr>
</thead>
<tbody>
<tr>
<td>(0,0)</td>
<td>0</td>
<td></td>
<td>0</td>
</tr>
<tr>
<td>(0,2)</td>
<td>8 8</td>
<td>Is (w) above (f)? yes (\rightarrow 6) no (\rightarrow 9)</td>
<td></td>
</tr>
<tr>
<td>(1,1)</td>
<td>6 9</td>
<td>Is (w) equal to (H)? yes (\rightarrow 1) no (\rightarrow 2)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Is (e_1) on the right-hand edge of the frame? yes (\rightarrow 3) no (\rightarrow 4)</td>
<td></td>
</tr>
<tr>
<td>(2,0)</td>
<td>1 2 3 5 7</td>
<td>Is (e_2) on the right-hand edge of the frame? yes (\rightarrow 5) no (\rightarrow 6)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Is (e_3) on the lower edge of the frame? yes (\rightarrow 7) no (\rightarrow 8)</td>
<td></td>
</tr>
<tr>
<td>(3,1)</td>
<td>2 4 5 3 4 7</td>
<td>Is (f) in the upper third of the right-hand half of the rectangle? yes (\rightarrow 9) no (\rightarrow 10)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Is (f) in the lower third of the rectangle? yes (\rightarrow 11) no (\rightarrow 12)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Is (f) to the right of the three end points? yes (\rightarrow 13) no (\rightarrow 14)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Are (e_1) and (e_2) above (f)? yes (\rightarrow 15) no (\rightarrow 16)</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>Is (e_3) on the right-hand edge of the frame? yes (\rightarrow 17) no (\rightarrow 18)</td>
<td></td>
</tr>
</tbody>
</table>

Figure 2.24: Beun's algorithm [1973].
are discussed. M. Beun [1973] proposes an OCR system based on end points, intersection of lines and loops. The segmented character is first thinned and then the end points (points with only one black neighbour), junctions (points with three or more black neighbours) and their relative positions with respect to the centre of the encasing rectangle are defined. Each character is modelled using these attributes. The system reported is tailored to recognize handwritten numerals. The algorithm depicted in fig. 2.24 can be described as follows: Let $E$ and $F$ denote the number of end points and the number of junctions (for $K$ points) respectively. If $E$ and $F$ are both equal to zero, then it must be numeral '0', or if ($E=0$) and ($F=0$), then it is '8'. If both $E$ and $F$ equal 1, then it must be either '6' or '9'. To resolve the ambiguity, the position of end points is examined. If it is above the junction then it ought to be '6' otherwise it is '9'. In the fourth category numerals '1', '2', '3', '5' and '7' are put in one category. Since they all have two end points ($E=2$) and no junction ($F=0$). To distinguish between these numerals, the total number of black points forming the skeleton are counted which is termed $W$ (for weight). Next, the number of rows that the skeleton covers in the rectangle is counted denoted by $H$ (for height). For all '1' numerals $W=H$ which does not apply to the other numerals. The '1' are thus eliminated. To select the
are discussed. M. Beun [1973] proposes an OCR system based on end points, intersection of lines and loops. The segmented character is first thinned and then the end points (points with only one black neighbour), junctions (points with three or more black neighbours) and their relative positions with respect to the centre of the encasing rectangle are defined. Each character is modelled using these attributes. The system reported is tailored to recognize handwritten numerals. The algorithm depicted in fig. 2.24 can be described as follows: Let E and F denote the number of end points and the number of junctions (for K points) respectively. If E and F are both equal to zero, then it must be numeral '0', or if (E=0) and (F=0), then it is '8'. If both E and F equal 1, then it must be either '6' or '9'. To resolve the ambiguity, the position of end points is examined. If it is above the junction then it ought to be '6' otherwise it is '9'. In the fourth category numerals '1', '2', '3', '5' and '7' are put in one category. Since they all have two end points (E=2) and no junction (F=0). To distinguish between these numerals, the total number of black points forming the skeleton are counted which is termed W (for weight). Next, the number of rows that the skeleton covers in the rectangle is counted denoted by H (for height). For all '1' numerals W=H which does not apply to the other numerals. The '1' are thus eliminated. To select the
other possibilities, the end points are numbered in the sequence in which they are encountered when the rectangle is scanned row by row, beginning with the top row and moving from left to right. The end points are given the symbols $e_1$ and $e_2$. If $e_1$ is at the right hand edge of the rectangle, the numeral must evidently be '5'. If, however, $e_2$ falls on the right hand edge of the frame, then the numeral is a '2'. If neither $e_1$ or $e_2$ are at the right hand edge of the frame, the decision is between '3' and '7'. Because of the constraints imposed on the types of numerals, we need in this case only to know whether $e_2$ is at the bottom edge of the frame. If this is so we then have a '7', otherwise, it is a '3'. In case (E=3) and (F=1) our choice is restricted to two different 'Fours' and also '2', '3', '5' and '7'. We first check whether F (the junction) lies in the right hand half of the upper third of the rectangle. If it does, then we have a '7'. Next, we see whether 'f' is in the lower third of the rectangle. In this way we identify the '2's. We then see if the three end points are all on the left of the junction which enables us to recognize the '3's. The '4's are found by checking whether $e_1$ and $e_2$ both lie above 'F'(or the junction). Any numeral that is not yet classified can only be a '5'. Practical implementation reported by the author suggest a 91.3% correct recognition, 2.67% mis-recognition and
5.96% rejection out of 10,000 numerals. Hunt [1972] proposes a more elaborate approach. In this scheme a character is first thinned and then its limbs are defined by horizontal lines (H), vertical lines (V), end points and junction points. For example Fig.2.25a depicts the thinned numeral '9' and Fig.2.25b shows the corresponding horizontal and vertical lines as well as end points and the junction. These contracted features are matched against a repertoire of features related to different characters and a final decision is made. According to the author, out of 8000 test characters, 0.5% were misclassified whilst 2.5% were rejected using good quality characters. Chain coding of character skeletons has also been used to detect features like cavities. Sue et al. [1976] propose an OCR system which extracts the skeleton of a numeral and then derives its chain code. During this process, the number of circles, branch points (junctions) and end points are simultaneously determined. After the list of code sequence of a numeral character is obtained, direction changes in code sequences are performed. Clockwise and counter clockwise cavities are also derived. To explain the process of skeleton code generation, the following terminologies are introduced:

(i) Inverse digital: The inverse of a digital is obtained simply performing \[ \text{Add 4 (mod 8)} \].
Figure 25.a: A thinned numeral.

Figure 2.25b: The corresponding horizontal and vertical lines, end points and junctions.
(ii) Path reversal: The path reversal is the process of tracing out a curve in the reverse direction. It is achieved by replacing each digital of a curve sequence by its inverse and by reversing the sequence.

(iii) Circle: A circle is a sequence of image cells forming a loop.

To describe a numeral, the starting point is first identified and tracing the skeleton is then performed. Since the original skeleton code is too sensitive to be used for feature extraction, "contraction coding" is introduced, which combines three digits into one digit as follows: For the consecutive three code digits \( a_n a_{n+1} a_{n+2} \), if the \( n \)th. digit equals to the \( (n+2) \)th. digit, select the \( n \)th. digit as the resultant digit in the contracted code, otherwise \( (n+1) \)th. digit is selected as the resultant digit. If the length of the skeleton code is not divisible by 3, then (i) if two digits are left, the second digit is retained; (ii) if one digit is left, the digit is discarded. Given the code sequence \( A(i) \), \( i = 1, 2, ..., n \) \( D(j) \) is defined by:

\[
D(j) = A(j+1) - A(j) \pmod{8}
\]

and such that

\[-4 \leq D(j) \leq 4 \quad j = 1, 2, ..., n - 1\]

(1) To detect clockwise direction \( D(j) > 0 \).

(2) If \( D(j) < 0 \), then it is counter clockwise.

(3) If \( D(j) = 0 \), then the direction in the \( j \)th. position is the same as the \( (j+1) \)th position.
The curvature code is also defined as a code whose elements are obtained by taking the sum of consecutive change elements of the same sign. To illustrate the algorithm, consider numeral '3' in fig.2.26.

Code sequence: 0 0 0 0 0 7 0 7 5 5 4 5 5 7 7 0 7 6 6 6 5 4 4 4 4 4 4 4 5

Contraction code: 0 0 1 3 1 1 6 4 4 4

Direction change code: 0 1 2 -2 0 1 2 0 0

Curvature code: 3 -2 3

The curvature code indicates that the unknown pattern has an upper clockwise cavity with a curvature 3, a lower clockwise cavity with a curvature 3 and an in-between counter clockwise cavity with a curvature -2. Other features such as number of circles, position of the branch points relative to their corresponding starting points and the number of end points are also derived. To classify the pattern, a tree structure is employed to eliminate unlikely candidates systematically. The simulated scheme achieves a correct recognition rate of upto 96.94%, an error rate of 0.96% and a rejection rate of 2.1% using a total of 810 samples. In a more recent contribution Chi-Heng et.al.[1980] describe a two stage scheme where the number of end points, the number of junctions of 3 branches and the number of junctions of 4 branches are counted and are used to pre classify the candidate pattern which will be
Figure 2.26: The numeral '3' and its corresponding chain codes

Code sequence: 000007075545577707666544454445
recognized in the second stage. The second stage of decision making involves describing the skeleton by a feature string which is constructed in the skeleton tracing process, via the identification of the following characteristics of each dot:

1. Point characteristic.

2. Direction characteristic.

3. Line segment characteristic.

A feature group of the above characteristics is obtained for each traced dot (pixel). By arranging the feature groups of dots of the skeleton successfully in tracing direction order, the feature string representing an input character is obtained. The dictionary for decision making is a bank of sequential logics divided into categories which are employed in direct matching of candidate strings. Unfortunately, the authors do not offer an explicit and precise explanation of the three characteristics mentioned above nor do they present results of any practical experimentation of their scheme. Note that, utilising end points, intersection of lines and loops to recognise characters can only produce a reasonable performance, should they be used on strictly constrained input data; otherwise, any unexpected shape variation will generally fail the final decision making logic.
2.5 Distribution of black/white pixels.

Statistical distribution of black/white pixels offer features of reduced dimensionality; they provide features such as the centre of gravity which can be used as a reference point. Different types of distribution are tailored to different recognition techniques.

2.5.1 Zoning

The frame containing a character is divided to non-overlapping (or sometimes overlapping) zones and densities of black/white pixels in each zone are utilised as recognition features. Hussain et.al.[1972] divides a $20 \times 20$ character matrix to $25$, $4 \times 4$ non-overlapping squares. A feature vector $x_i$ which represents the number of black points in each square is defined; thus $x_i$ could assume any integer value from $0$ to $16$ inclusive. A quantity $R$ is defined by:

$$R = \sum_{i=1}^{i=25} \ln[p(x_i/c_j)] + \ln[p(c_j)]$$

where $c_j$ is the character class to classify different characters $p(x_i/c_j)$ denotes the probability of feature vector $x_i$ conditioned on character class $c_j$ and $p(c_j)$ is a priori probability of the character class $c_j$.

2.5.2 n-Tuples

This method is essentially based on the occurrence of black and white
pixels and their joint occurrence as features. It was proposed by Bledsoe and Browning in 1959. Simulation of the maximum likelihood version was reported by Bledsoe and Bisson in 1962 which pointed out, that this is a method of approximating a higher order distribution by a product of lower order distributions. A comprehensive investigation of the method was conducted by Ullman [1969]; he concluded that, the 'non-weighted' version of the method worked better than the maximum likelihood weighted version, but it was achieved at an enormous storage cost (almost 3 megabytes!). An experimental investigation of mixed font printed characters was reported by Liu and Shelton [1966]. The basic procedure in this scheme is a minimum distance linear classification. An unknown Character's measurements are compared with a set of stored references and the unknown is classified as a member of the character class whose stored reference is at a minimum distance to the unknown character. Specifically, let $X = (x_1, x_2, \cdots, x_L)$ be the binary measurement vector of a given character. From a set of character samples called 'design data', we estimate for each character class $c_i$, the conditional probabilities $p(x_j = 1/c_i)$ for $j = 1, 2, \cdots, L$ and $i = 1, 2, \cdots, K$, where $K$ is the number of character classes being considered. These probabilities are then quantized at only three different levels based on the following threshold.
requirements:

\[ p'(x_j = 1/c_i) = -1 \text{ if } p(x_j = 1/c_i) \geq 0.8 \]

\[ p'(x_j = 1/c_i) = 1 \text{ if } p(x_j = 1/c_i) \leq 0.2 \]

\[ p'(x_j = 1/c_i) = 0 \text{ Otherwise} \]

Now, let \( p'(x_j = 1/c_i) \) be denoted \( S_{ij} \). The set of ternary vectors \( S_i = (s_{i1}, s_{i2}, \ldots, s_{iL}) \), \( i = 1, 2, \ldots, K \) is called the reference for character classes \( c_i \). Note that, when \( S_{ij} = -1(1) \), recognition logic \( x_j \) matches (does not match) character class \( c_i \) most of the time, when \( s_{ij} = 0 \), \( x_j \) matches \( c_i \) rather randomly. To recognize an unknown character represented by a measurement vector \( X = (x_1, x_2, \ldots, x_L) \), its distance to the reference \( S_i \) of character class \( c_i \) is computed by:

\[ D_i = \sum_{j=1}^{L} d_{ij} \]

where \( d_{ij} = 1 \) if \( s_{ij} = -1 \), \( x_j = 0 \), or \( s_{ij} = 1 \), \( x_j = 1 \); \( d_{ij} = 0 \) otherwise. Note that a '0' in the reference component is treated as a 'don't care' condition. The unknown character is assigned to be a sample of class \( c_m \) if

\[ D_m = \text{Min} (D_1, D_2, D_3, \ldots, D_K) \]
A threshold may also be imposed on $D_{\min}$ to act as a rejection criterion.

A reduction in the number of references and thus in storage is achieved by using samples of all fonts to derive ternary vector. The authors report a poor performance on 9 different fonts, but an acceptable performance (1 percent reject, 0.1 percent substitution) on single fonts of fair to good quality documents. An algorithm along similar lines was developed by Stentiford (1985), which iteratively updates the set of features deemed useful in the training process. He argues that, in order a feature be useful, it must add materially to the information already extracted by others in the working set, and ideally, there should be no correlation between feature occurrences. Let a reference pattern $C_i$ give rise to an $N$-dimensional response vector $F_i$ in feature space where

$$F_i = f_{i1}, f_{i2}, \ldots, f_{iN} \quad i = 1, 2, \ldots, K$$

and where $f_{ij}$ is the response of the $j$th feature to the $i$th reference pattern.

Similarly, let the $j$th feature correspond to the vector response $f_j$ from the $K$ reference patterns $C_i$ where $f_j = f_{1j}, f_{2j}, \ldots, f_{Kj} \quad j = 1, 2, \ldots, N$

In the task of recognizing the $C_i$ in a noisy environment, the best discrimination is achieved when $F_i$ are all angularly separated from each other by
a maximum angle. Flores et al. [1960] have shown that, the largest angle \( \alpha \) such that the angular distance between any pair of reference vectors is at least \( \alpha \) and given by:

\[
\alpha = \cos^{-1}\left(\frac{-1}{K-1}\right) \text{ for } N \geq K - 1
\]

It is to be noted that \( \alpha \) is independent of \( N \) and that if the number of features is increased beyond \( K-1 \), the optimum solution will not improve. In this way the most economical and optimum solution occurs precisely when \( N=K-1 \) and any extension of the number of features must be accompanied by a similar increase in the number of reference patterns. In addition, as \( N \) or \( K \) increases, an orthogonal arrangement of the \( F_i \) becomes very close to the optimum. Consider a \( K \) class OCR problem in which each character \( C_K \) is represented as a two dimensional \( w \times h \) binary array.

\[
C_K = \mu_K(i,j) \quad i = 1,2,3,...,w \quad j=1,2,3,...,h
\]

\[
\mu_K(i,j) = 0 \text{ or } 1
\]

An operator or feature \( \sigma_j \) is defined as a set of integer triples [Stentiford, 1985]:

\[
\sigma_j = (x_{ij}, y_{ij}, q_{ij}) \quad i = 1,2,3,..u_j
\]

63
The operator $\sigma_j$ fits the pattern $c_K$ if there exists an offset $(X_i, Y_j)$ such that $\mu_K(X_{ij} + X_j, Y_{ij} + Y_j) = q_{ij}$ for $i = 1, 2, 3, ..., u_j$. Let $f_{ij} = +1$ if the $j$th operator fits the reference pattern from the $i$th class, and $f_{ij} = -1$ otherwise. A measure of an operator's performance is required, which when increased, guarantees an improvement in the separation or the orthogonality of a set of reference pattern responses. It can be shown [Stentiford, 1985] that, given $[f_{ij}]$ is a square matrix $(K = N)$, then a necessary and sufficient condition that the row vectors $f_i$ are mutually orthogonal is that the column vectors $f_j$ are mutually orthogonal. This means that any tendency for the $f_j$ to become mutually orthogonal necessarily implies that the $f_i$ will also.

Measures of angle between the operator response vectors $f_j$, therefore, have a direct bearing on the separation in feature space of the reference pattern response vectors $f_i$. Indeed

$$G = \sum_{i,j}(f_i f_j)^2 = \sum_{i,j}(F_i F_j)^2$$

Hence, $\sum_{i,j} (f_i f_j)^2$ can be used as a measure of the total performance of the reference pattern responses when $K = N$.

Now, assuming $(f_i f_i)^2 = \text{constant}$ for all $i$
\[ G = \sum_{i,j} (f_i f_j)^2 \]
\[ = \sum_{i \neq a} \sum_{j} (f_i f_j)^2 + \sum_{j} (f_a f_j)^2 \]
\[ = \sum_{i \neq a} \sum_{j \neq a} (f_i f_j)^2 + 2 \sum_{j \neq a} (f_a f_j)^2 + (f_a f_a)^2 \]

Hence, for G to improve and decrease with \( f_a \), \( f_a \) must change so as to decrease with the value of \( M_a \) where

\[ M_a = \sum_{j \neq a} (f_a \cdot f_j)^2 \]

Suppose now, that the operators in a set each have response \( f_j \) and measure \( M_j \). Suppose that

\[ M_m = \max (M_j) , \ j = 1,2,\ldots,N \]

A candidate operator, \( a \), would replace the worst performing operator ( \( m \) th.) in the set if \( M_a = \sum_{j \neq a} (f_a f_j)^2 < M_m \). A succession of such replacements would ensure an increasingly orthogonal set of the \( f_j \)'s and, hence, also the \( F_j \)'s. The algorithm was reportedly trained on 30600 printed characters and tested on another 10200 characters. An error rate (mis-classification) of 1.01% was obtained with 316 operators and 319 reference characters.
2.5.3 Moments

Summing the distances of all elements of a character matrix from a reference point i.e. the centre of gravity is used as a feature. In general, the moments are defined by:

\[ m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q \cdot f(x, y) \]

where \( p, q = 0,1,2,\ldots, \infty \)

\( N, M \) : Dimensions of the image matrix.

\( f(x,y) \) : The light intensity (gray level) at a point \((x,y)\) in the image.

For binarized characters, the above relationship is simplified to:

\[ m_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} x^p y^q \cdot f(x, y) \]

and \( p+q = \) order of the moment, \( f(x, y) = +1(\text{or}0) \).

These 'raw' moments are information preserving; the original image can be acceptably reconstructed using a finite but sufficiently large set of moments computed from the image [Teague, 1980]. The central moments of an image can be computed using:

\[ \mu_{pq} = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (x - \bar{x})^p (y - \bar{y})^q \cdot f(x, y) \]
where \( \bar{x} = \frac{m_{10}}{m_{00}} \), \( \bar{y} = \frac{m_{01}}{m_{00}} \) are the coordinates of the centroid of the image.

The central moments are invariant to translation of an image. Another set of moments may be derived from the central moments which are also invariant to the scale of an image. Denoted by \( \eta_{pq} \), these normalized central moments are given by [Wintz et al., 1987]:

\[
\eta_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}
\]

where \( \gamma = \frac{p+q}{2} + 1 \), \( p + q = 2, 3, \ldots \).

But it was Hu [1962] who developed a set of seven moments, that are invariant to translation, scale change and rotation. These moments are usually called 'moment invariants' or 'geometric moments'. Spanjersberg [1974] used moments of up to the 5th. order, while Kwan et al. [1979] used moments of up to six to describe characters. Tucker, et al. [1974] have used up to 15 geometric moments as the distinguishing features of characters.

More recently, however, Cash et al. [1987] carried out an in depth study of using the best suitable moments for printed character recognition. They used eight central moments of order three or less for 6 different fonts. For final classification, three different similarity measures were used, namely the Euclidean distance, the cross correlation and the Mahanabolis distance as
The straight line distance between two points in N-dimensional space is known as the Euclidean distance. The Euclidean distance between two feature vectors is given by:

\[ D_E = \sqrt{\sum_{i=1}^{N} (F_{L_i} - F_{I_i})^2} \]

where \( F_{L_i} \) is the ith library feature, and \( F_{I_i} \) is the ith input feature. The class of the library feature vector producing the smallest Euclidean distance when compared with the input feature vector is assigned to the input character. One method of increasing the performance of similarity measures involves a statistical analysis of the training set features. Those features which are found to be more reliable than others are given more importance when making classifications. The idea behind this is to try to make the intra-class distance as small as possible. For the Euclidean distance measure, weighting factors are determined which cause the more reliable features to make a larger contribution to the distance between two feature vectors. The weighted Euclidean distance between two feature vectors is given by:
\[ D_E = \sqrt{\sum_{K=1}^{K=N} w_K (F_{LK} - F_{IK})^2} \]

where \( w_K \) is the \( K \)th weighting factor. The problem now is how to select the appropriate weighting factors. Intuitively, the feature that has a smaller variance is more reliable and should contribute more to the decision process; therefore, a reasonable approach would be to let \( w_K = \frac{1}{\bar{\sigma}_K} \). The denominator of the above equation, \( \bar{\sigma}_K \), is computed by calculating the standard deviation for the \( K \)th feature over \( m \) characters of the \( n \) classes and then finding the mean of the \( n \) standard deviation. This can be expressed mathematically as:

\[
\bar{\sigma}_K = \frac{1}{n} \sum_{j=1}^{n} \left( \frac{m \sum_{j=1}^{m} F_{ji}^2 - (\sum_{i=1}^{m} F_{ji})^2}{m(m-1)} \right)^{1/2}
\]

where \( n \) is the number of classes and \( m \) is the number of characters in each class. \( F_{ji} \) is the feature for the \( i \)th character of the \( j \)th class. Four other different types of weighting factors are also proposed and tested experimentally. Another similarity measure used in this scheme is the normalized correlation. When this measure is used, the class of library feature vector
producing the largest result when cross correlated with the input vector is assigned to the input character.

The normalized cross correlation for two vectors is computed using the following equation:

\[
R = \frac{\sum_{i=1}^{N} F_{L_i}F_{I_i}}{\sqrt{\sum_{i=1}^{N} F_{L_i}^2 \sum_{i=1}^{N} F_{I_i}^2}}
\]

This measure did not prove to be very robust in practice. To improve its performance, weighting factors were incorporated into the formulation, while increasing the library sets up to 20. As a result, a correct recognition of up to 99% was achieved. The last similarity measure considered by Cash, et.al. [1987] was the Mahanabolis distance between the input feature vector and the mean library feature vectors of a particular class. The weight used, is the reciprocal of the intra-class feature variance. The Mahanobolis distance is mathematically expressed as:

\[
D_H = \sum_{i=1}^{i=N} \frac{(F_{L_i} - F_{I_i})^2}{\sigma_{L_i}^2}
\]

This measure is particularly attractive for software implementation, because, the number of comparisons required is the same, regardless of how many library sets are used. Once the mean of each feature for the specified
number of library sets has been computed, only the mean feature vectors are used in the classification process. The reported experiments indicate better than 95% correct recognition rate when 20 library sets were used to compute the mean library set. It should also be noted that experiments were carried out using phototype ready documents of high quality and not ordinary printed documents.

2.5.4 Characteristic loci

The 'characteristic-loci' features were devised by Glucksman [1971] to classify multi-font alphabets. In this technique, a five digit code is generated from each point in the character matrix. One digit (bit) refers to the state of the pixel itself, i.e. black (1) or white (0). The other four digits, however, contain the count of the number of line crossings from a point to one of the main directions: up, down, left and right respectively. To reduce storage and computing time, the count is restricted to a maximum of 2 for two or more crossings in any one direction; thus forming a ternary code. For example, in fig.2.27, the code 1201 is generated for point B and its neighbouring points. Only codes associated with white points are used in constructing the feature vector and points outside the frame (rectangle enclosing the character outline) i.e. codes with 3 or 4 zeros are ignored. Knoll [1969]
Figure 2.27: Characteristic loci of '2'.
investigated the method further and reduced the dimension of the feature vector by defining 'significant' components experimentally. He found that 16 components or less were significant for the ten numeric symbols and less than 30 for the 26 alphabetic symbols. Two different recognition schemes, 'Exact Match' (EM) and 'Linear Discriminant Function' (LDF) were proposed and tested. The formulae performs exact matching between the reference feature vector and the candidate (feature vector), ignoring 'don't care' entries for bits which do not have high probabilities of being either above or below the corresponding pre-defined threshold; the latter, however, employs a weighted function for classification. It is defined by:

\[(w_j - w_i).X_u > 0 \text{ for all } i \neq j\]

where \(w_i\) : Weight vector for each symbol.

\(X_u\) : Threshold feature.

\(w_j\) : Weight vector for each reference symbol.

If the above relationship is satisfied, the unknown sample is classified in class \(j\). Spanjersberg's characteristic loci scheme [1974] makes use of 4 perpendicular directions for each white pixel in the frame, incorporating a weighting process to determine 'the optimal position' for 'separation planes'.
In a recent paper Suen [1982] proposed a more sophisticated approach, called 'multi-directional loci' features; in his view, the vector directions could be increased in multiples of 8, 16, 32, etc., thereby, providing finer directional loci. A higher performance is reported for '8 directional' loci technique as compared with '4 directional' loci. Characteristic loci techniques are relatively insensitive to noise, breaks or stroke variations and a parallel process to detect the features is relatively simple to realize.

2.5.5 Crossings and Distances

This method is a special case of 'characteristic loci', where, the features are measured from the number of times line segments are traversed by vectors in specified directions, or the distances of elements or segments from a given boundary such as the bounding box.

Calvert [1970] used horizontal crossings and Kwon, et.al. [1976] used both horizontal and vertical crossing counts for character recognition. Extra features have also been included in such schemes to improve recognition performance. Lewis [1962] included the regional density of black points in his scheme, whereas, Doyle [1980] added length measurements. Ni, et.al. [1980] used the scan lines as the crossing vectors and registered the location, the length and the number of times the crossing between the vectors and char-
acter strokes. Kwan, et.al. [1979] incorporated geometrical and topological measurements to enhance the recognition rate of his scheme. Unfortunately, there is a vague mention of accuracy/performance of the above schemes.
2.6 Introduction

When a document is scanned and digitized, the raw data carries a certain amount of noise, for example, a low resolution scanner will produce touching line segments and smeared images. The digitized image will contain much higher level of noise if the input document is not cleaned and contains smudgy characters. In order to eliminate unwanted noise which may cause severe distortion in the digital image and hence distorted features and poor recognition rates, a pre-processor is used to remove the noise and smooth the digitized characters.

The main operations of a pre-processor are as follows (fig. 2.28):

1- Skew Normalization: To orient characters in an upright position.

2- Filling / Smoothing: To eliminate small breaks, gaps, holes, bumps and isolated pixels.

3- Thinning: To reduce the character matrix to a skeleton, so that line ends, lengths, directions, etc. can be extracted easily.

4- Segmentation of joined characters: To separate characters that have been connected to one another due to low scanning resolution, narrow spac-
Fig. 2.28: Pre-processing techniques.
ing between two characters or use of 'ligatures' used in printing, prior to classification.

5- Detecting a base line: To detect a text base line in order to pre-classify characters with ascenders and descenders.

6- Size normalization: To normalize a character matrix to a pre-defined size.

In the following sections, different preprocessing techniques are discussed and their likely effects on a typical OCR system are assessed, while a detailed account of pre-processing techniques designed/implemented in the course of this work is presented in chapter 4.

2.7 Skew normalization

Automatic segmentation of printed documents into columns, lines of text and characters can not be achieved without detecting and correcting skew angles of scanned documents. A skew angle is defined as the orientation angle of lines of text in a document where several angles occur, the dominant angle of the majority of text lines are taken into account. Casey [1970] describes a skew angle normalization algorithm for individual characters. The idea is that, once a pattern is extracted and its bounding box determined, a skew normalization process based on moments of the pattern, is initiated to correct
any possible non-zero skew angles. The pattern moments consist of the three quantities $m_x$, $m_y$ and $m_{xy}$ often arranged in matrix form as follows:

\[
\begin{bmatrix}
m_x & m_{xy} \\
m_{xy} & m_y
\end{bmatrix}
\]

The elements $m_x$ and $m_y$ are the mean square x and y deviations about orthogonal x and y axes through the centroid of the pattern. They are defined by the expressions:

\[
m_x = S^{-1} \int_p x^2 \, dS
\]

and

\[
m_y = S^{-1} \int_p y^2 \, dS
\]

where $S$ is the total area of the pattern $p$. The product term $m_{xy}$ is defined by $m_{xy} = S^{-1} \int_p xy \, dS$. Let us assume a pattern $P$ is transformed by a linear mapping procedure having matrix $A$ into a new pattern $P'$. The transformation can be expressed in terms of new coordinates $u$ and $v$ that are linearly related to the original coordinates. Casey [1970] proved that the moment matrix $M' = AMA'$ where prime indicates the transposed matrix. He also showed that, if two patterns related by a linear transformation are mapped
into new patterns having diagonal moment matrices, the transformed patterns are identical except for rotation and reflection as depicted in fig.2.29.

In order to digitize a binary pattern in upright position, its transformation matrix is computed and the flying spot scanner is adjusted accordingly for a re-scanning process where the scanner travels along oblique parallel paths to register the scanned character in the upright position. Although Bakis et.al. [1964] had already used a similar approach to normalize a character height by driving $m_{xy}$ to zero, it was Casey who provided a rigorous mathematical analysis as well as including 'rotation' and 'reflection' sensitive properties. However, re-scanning each character for height and rotation normalization is not a viable proposition for a high throughput OCR system.

An intuitive solution, however, would be to rotate the entire document into an upright position before isolating lines of text and individual characters. Hashizume et.al. [1986] draw on an earlier work by Antoy [1985] to design a method of detecting the skew angle. They assume that characters are often closer to one another in the direction along text lines than on other directions. The nearest neighbour of each component is computed and each pair of neighbours is connected by a straight line segment. Suppose the relevant components are dots, and that dots A, B, C, D, E, are located as shown...
Fig. 2.29: Examples of skew normalization using moments [Casey, 1970].
in figure 2.30, that is:

\[
d(B, C) < d(D, E) < d(B, D) < d(A, B) \quad \text{and} \quad d(A, B) < d(A, D)
\]

\[
d(B, D) < \min \{d(C, E), d(C, D), d(B, E)\}
\]

where \(d(X, Y)\) is the distance between dot \(X\) and dot \(Y\). In this case, the arcs represented by line segments in fig.2.31 are used to describe the proximity tree. Figure 2.31 is a tree representation of the proximity tree and figure 2.32 displays a table representation, in which shorter arcs are stored first. The nearest neighbour of each dot can, therefore, be found by searching the table from the top entry to the one for a given dot while checking if there is a shorter arc which includes dot \(K\) where dot \(K\) is one of the end points of the arc. Figure 2.33 shows an example on which the desired steps were carried out. At the next stage, a histogram of the orientations of these line segments is computed. Consider the corresponding map of nearest component pairs in figure 2.33, which is depicted in fig.2.34.

A vector is defined in such a way that the starting point is on the left, so that the direction of the vector varies from -90 degrees to +90 degrees. The histogram of the line orientations are computed at 36 5-degree intervals. A smoothing operator is applied to the original direction histogram and the
Fig. 2.30: An example of arcs represented by line segments.

Fig. 2.31: Tree representation of the proximity
a) an example, b) Tree representation of proximity.
<table>
<thead>
<tr>
<th>Entries</th>
<th>Distance</th>
<th>Comp. 1</th>
<th>Comp. 2</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>d(B, C)</td>
<td>B</td>
<td>C</td>
</tr>
<tr>
<td>2</td>
<td>d(D, E)</td>
<td>D</td>
<td>E</td>
</tr>
<tr>
<td>3</td>
<td>d(B, D)</td>
<td>B</td>
<td>D</td>
</tr>
<tr>
<td>4</td>
<td>d(A, B)</td>
<td>A</td>
<td>B</td>
</tr>
</tbody>
</table>

Fig. 2.32: the table representation of the tree representation.
Fig. 2.33: An example used to test the algorithm.

Fig. 2.34: The corresponding map of nearest component pairs.
position which has maximum frequency in the smoothed histogram is chosen as the peak of the direction histogram (figs 2.35 and 2.36). Among the thirteen examples reported, the average error was 1.5 degrees and the worst 4.1 degrees. The main drawback of this method is its failure to cope with wide character spacing which occurs in sparsely populated tables. Postl [1986] proposes two 'skew angle' detection techniques. Let us denote a document and scanner coordinates by \((x,y)\) and \((u,v)\), respectively. Document and scan samples are related by:

\[
D(x, y, \alpha) = S(u(x, y, \alpha), v(x, y, \alpha))
\]

where

\[
u(x, y, \alpha) = x \cos \alpha + y \sin \alpha
\]

\[
v(x, y, \alpha) = y \cos \alpha - x \sin \alpha
\]

where \(\alpha\) is the skew angle relative to the scanning field.

A sequence of skew scans can be simulated at search angles \(\alpha_i (i = 0, 1, \ldots)\) within a pre-set range \(\alpha_{\text{min}} \leq \alpha_i \leq \alpha_{\text{max}}\), typically with \(\alpha_0 = 0\). For each search angle, an alignment premium \(P(\alpha)\) is calculated, thus generating \(P_i (i = 0, 1, \ldots)\) where the next search angle \(\alpha_{i+1} \leftarrow (\cup_{j=0}^{i-1} \alpha_j, p_j)\)
<table>
<thead>
<tr>
<th>deg.</th>
<th>freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>1</td>
</tr>
<tr>
<td>85</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>1</td>
</tr>
<tr>
<td>75</td>
<td>1</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 2.35: Direct histogram of the left block in fig. 2.33.

<table>
<thead>
<tr>
<th>deg.</th>
<th>freq.</th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>1</td>
</tr>
<tr>
<td>85</td>
<td>1</td>
</tr>
<tr>
<td>80</td>
<td>1</td>
</tr>
<tr>
<td>75</td>
<td>1</td>
</tr>
<tr>
<td>70</td>
<td>0</td>
</tr>
<tr>
<td>65</td>
<td>0</td>
</tr>
<tr>
<td>60</td>
<td>0</td>
</tr>
<tr>
<td>55</td>
<td>0</td>
</tr>
<tr>
<td>50</td>
<td>0</td>
</tr>
<tr>
<td>45</td>
<td>0</td>
</tr>
<tr>
<td>40</td>
<td>0</td>
</tr>
<tr>
<td>35</td>
<td>0</td>
</tr>
<tr>
<td>30</td>
<td>0</td>
</tr>
<tr>
<td>25</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>15</td>
<td>0</td>
</tr>
<tr>
<td>10</td>
<td>0</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Figure 2.36: Direction histogram of the right block in 2.33.
is found following some optimization strategy. The search premium is calculated by computing the integral density along every inclined line at the scan angle. Then, for each pair of neighbouring scan lines, the difference of their densities is computed. At the last step the sum of squares of these differences is computed and its maximum value reveals the dominant skew angle. The Postl's second method is based on applying the discrete 2D Fourier transform to the image plane where a half plane of the power spectrum coefficients $W(u,v)$ are computed as follows: For each 'simulated scan angle' in a given range, a scalar alignment measure is computed by integrating along a radius vector from the origin at the angle to the v axis (fig 2.39).

The measure represents the accumulation of energy in spatial frequencies sharing that orientation angle; its global maximum reveals the dominant skew angle. An accuracy of 0.01 radian is reported for the first method. A major advantage of both techniques is their ability to operate on multilevel images, prior to binarization of a document. Rastogi et al. [1986] describe a method using Hough transforms. The Hough transform technique detects the presence of parametrically representable group of points in an image, such as a straight line or a circle through a mapping to a parameter space. The parameter space is regarded as an n-dimensional matrix of accumulators,
Fig. 2.39: Document obliquely positioned in a scan field.

$s = \text{simulated scan line.}$
where \( n \) is the number of parameters for the shape of interest. Each significant point from the original set of points casts a vote in one as many slots in the accumulator matrix. Thus looking for a pattern of interest in the original data set corresponding to searching for peak values in the \( n \)-dimensional space. An scanned document can be considered to be a binary image having black pixels carrying information and white pixels or the background. Text lines are concentrated into regions which form the parallel horizontal stripes. A normal representation of the pattern is given by:

\[
x \cos \theta + y \sin \theta = \rho
\]

as shown in fig. 2.40. Clearly, in the \((\rho, \theta)\) parameter space, we have a loci of sinusoidal curves; \( M \) colinear points lying on a line \( x \cos \theta_j + y \sin \theta_j = \rho_j \) will yield \( M \) sinusoidal curves that intercept at \((\rho_i, \theta_j)\) in the parameter space. The range of angle \( \theta \) is \(+(-)90°\), measured with respect to the \( x \) axis. Thus with reference to fig. 2.41, a horizontal line has \( \theta = 0 \) with \( \rho \) being equal to the positive \( x \) intercept. Similarly, a vertical line has \( \theta = 90° \), with \( \rho \) being equal to the positive \( y \) intercept, or \( \theta = -90° \) with \( \rho \) being equal to the negative \( y \) intercept. An array of accumulators is set up by quantizing the values of \( \theta \) and \( \rho \). For a \( 512 \times 512 \) image, the possible range of \( \rho \) and \( \theta \) can
Fig. 2.41: Quantization of $\rho \theta$ plane.
Fig. 2.40: A normal representation of line.
be calculated by assuming the origin of the \((\rho, \theta)\) space to be at the centre of the 512 X 512 image and hence the maximum value of \(\rho\) is \(256\sqrt{2}\) and 0 to 180 degrees for \(\theta\). The \(\rho\) and \(\theta\) quantities thus vary between -362 to +362 corresponding to 0 to 180 degrees. The algorithm for matrix computation is as follows:

For each black point in row \(x\), column \(y\). For \(\theta = 0\) to 180 degrees.

\[
\begin{align*}
\rho &= x \cos \theta + y \sin \theta \\
\text{accumulator}[\rho, \theta] &= \text{accumulator}[\rho, \theta] + 1
\end{align*}
\]

end]

All black pixels in a given document image are taken into account to detect evidence of straight lines. The \(\theta\) values were incremented by 15 degrees to 180 degrees in the anti-clockwise direction, starting from 0. Rastogi observed a 'low-high-low' transition corresponding to the rise and fall in the number of pixels along the horizontal line which is due in part to the regular text lines at the top and left side of the document.

The counts increase from a low value to a high near the centre of char-
acters (both upper and lower case) and then return back to a low value corresponding to the white space between lines. If the entire document is skewed along a particular orientation, then that value of $\theta$, when scanned for all possible values of $\rho$ yields the maximum number of changes in accumulator counts.

Thus, the orientation can be defined by detecting the maximum number of these 'low - high - low' transitions. More recently, Baird [1987] has proposed a new skew angle detection algorithm. His approach assumes that the document has:

1) a dominant text line orientation.
2) characters have been roughly segmented.
3) loose bounds on font point sizes are known.
4) non-textual graphics can be ignored by some means or are overwhelmed by the number of characters; and finally, text lines must have been accurately printed.

The algorithm locates a character at the mid point of the bottom of its bounding box. At a given orientation angle $\theta$, the locations of characters (abstracted as points) are projected on to an accumulator line perpendicular to the projection direction. The accumulator line is partitioned into $m$ bins,
and bin size is set at 1/3 of the height of an 'x' in 6 point text. Let $c_i(\theta)$ denote the number of points projected into the i-th bin at angle $\theta$. An 'energy alignment measure' of function $\theta$ is computed as:

$$A(\theta) = \sum_{i=1}^{m} c_i^2(\theta)$$

This function displays a global maximum at the correct skew angle. Baird reports an accuracy of 2 minutes of arc. Detecting skew angles of scanned documents provides an OCR system with a means of avoiding computationally expensive processes such as dealing with rotated characters.
2.8 Filling / Smoothing

The purpose of the 'filling' process is to smooth binary patterns. It fills in holes in patterns by changing the pixel X from 0 (white) to 1 (black) if it has more than two black 4-neighbours. For example:

```
0 0 0
1 0 1
1 1 1
```

is modified using fill process to:

```
0 0 0
1 1 1
1 1 1
```

which must include their symmetries in the other directions. The 'smoothing' process, however, eliminates isolated black pixels and small bumps in binary patterns. For example:

```
0 0 0
0 1 0
1 1 1
```

is modified by the smoothing process to:

```
0 0 0
0 0 0
1 1 1
```
and their symmetries in other directions. Filling/smoothing is particularly useful when applied before commencing the thinning process as it will reduce the level of noise in thinned characters.

2.9 Thinning

Thinning is the process of reducing a binary image into a line skeleton of unit thickness. Other terms widely used to describe this process are 'skeletonization' and 'medial axis transformation'. Thinning techniques play a central role in a broad spectrum of applications in image processing ranging from printed circuit boards, counting asbestos fibers in air filters to automated inspection. The skeleton of a character can be defined as what remains of the character when the information about its thickness is removed; from its skeleton a topological description of the character can be derived. Cox et al. [1982] present an excellent exposition on the necessity of extracting line description of characters, as well as tracing its origin to the Renaissance. The concept of thinning, as applied to characters, was first developed by Sherman [1959]; since then, there have been strenuous efforts by many researchers to devise a much faster and more accurate thinning process. Figure 2.42 depicts letter 'n' and its corresponding skeleton. Representing a character by its skeleton reduces the data volume significantly as well as allowing an ef-
Fig. 2.42: Letter 'n' and its corresponding skeleton.
ficient and fast extraction of topological features. Some research workers [e.g. Smith 1987], believe that thinning provides a 'unification of character shapes by reducing the effects of varying type fonts' to some extent; in my view it is a clear understatement as thinning does not reduces the effects of varying type fonts, but, on the contrary, it makes them deeply exposed to an OCR system. From a theoretical point of view, there are different definitions (e.g. Montanari[1969], Rosenfeld [1966, 1975]). We find a definition by Dacis and Plummer [1981] based on Rosenfeld's connectivity theorem [1966] more realistic and plausible; it may be described as follows:

'A character's skeleton is formed from the centres of maximal discs placed within the character and the resulting skeleton must remain connected. By retaining the radii of the maximal discs, the skeleton can be used to reconstruct the original character.'

The thinning algorithms are classified to 'parallel algorithms' and 'sequential algorithms' according to the form of implementation. Assessing effects of a thinning process is merely 'objective'; that is, there is no mathematical measure to quantify thinning effects on the end results and the sole criterion is an individual's judgement of the skeleton's shape. Although there are no mathematically defined measures of thinning performance, two main
attributes must feature prominently in any thinning process:

a) The character’s skeleton must be only one pixel wide all over. Some thinning processes produce skeletons of more than one pixel wide in some parts of the skeleton.

b) The extracted skeleton should be as close as possible to the true shape of a character. For example, a properly thinned 'K' must not resemble a thinned version of 'X' as shown in fig.2.43.

In the following a brief exposition of two recently developed algorithms is presented; they make important assumptions to extract a skeleton more faithful to the original shape of a character. Major reviews of different thinning algorithms, however, may be found in Tamura [1978], Davis et.al. [1981], Hilditch [1983], Naccache, et.al. [1984], Smith [1987] and references there in.

2.9.1 connectivity

Rosenfeld [1966] defined connectivity in a binary pattern as '4 neighbour' and '8 neighbour'. Consider a 3 x 3 binary pattern below. The neighbours B, D, E and G are each called 4 neighbour of the pixel X as depicted in the pattern. The neighbours A, B, C, D, E, F, G and H, however, are each called an 8 neighbour of the pixel X.
Fig. 2.43: A typical skeleton of K that resembles that of X.
If A, C and G are set to '1', it is said that 'X' is a third degree junction/or node, because, three immediate neighbours of 'X' have been set to '1', as shown in the figure below.

\[
\begin{array}{cc}
1 & 1 \\
X \\
1 \\
\end{array}
\]
2.10 Chu’s Thinning algorithm.

Chu and Suen [1986] identify 3 major obstacles experienced by many thinning algorithms:

a) Maintaining connectedness.

b) Retaining end points.

c) Ensuring points are removed systematically.

They set out to smooth the original binary pattern by shifting a 3 x 3 window to detect bumps and holes which are removed subsequently; this is very similar to the filling and smoothing processes described in the previous sections. It is meant to produce a uniform contour for consistent stripping of black pixels. At the next stage, a 'labelling' process is initiated to mark all contour points; there are two types of contour points. The first type is the 'corner point' and the other, the 'trace point'. A black pixel X is a corner point if it has seven black neighbours, four of which are 4 neighbours.
For example:

```
0 1 1
1 X 1
1 1 1
```

A black pixel X is a trace point if it has three or less black 4-neighbours.

For example:

```
1 0 1
0 1 0
0 1 0
```

The resulting labelled pattern is then passed to the 'strip' process which removes contour point Y in two stages. In the first stage, trace points are traced out one by one and deleted right after, if they are neither end points nor break points. To trace a contour, a 3 x 3 window is moved from left to right, top to bottom of the binary picture to search for the first trace point X. The eight neighbours of X are then examined in a pre-defined 'start scanning
sequence' to look for the next trace point. The 'start scanning sequence' depends on the type of trace. There are four types of trace:

1. outer contour-clockwise,
2. outer contour-counter clockwise,
3. inner contour-clockwise,

and (4) inner contour counter-clockwise traces.

In the 'stripping' phase, clockwise is assumed for an outer counter trace, counter clockwise for an inner contour trace at odd iterations and counter clockwise for an outer contour trace and clockwise for an inner contour trace at even iterations. The purpose of alternate inner and outer contour tracing and stripping is to balance the stripping of points from both the inside and outside contours of those binary patterns which have closed loop(s) like the letter 'e' as shown in fig.2.46. In the second stage, corner points are inspected one by one. Four types of corner points are defined as depicted in fig.2.44. A corner will be stripped if it is not a break-point or an end-point and it is less than or equal to 120 degrees. A corner point of type 1 is said to be less than or equal to 120 degrees if in its 5 x 5 window (see fig.2.45) both I and J are black or three of I, J, K, L are black. The angles for corner points of types 2, 3 and 4 are evaluated in a similar way. After all the corner points
Fig. 2.44: The four types of corner points.
Fig. 2.45: 5 X 5 windows used to evaluate angles for the four types of corner-points.
are processed, the resulting picture is checked to see if there are some more points to be stripped in the next iteration. If the resulting picture contains any point which is not a break point, an end point, or a cross point, then smooth and strip processes are repeated; otherwise the thin phase terminates.

The skeleton obtained from phase one is re-examined to detect parts which do not sit on the ideal medial line of the pattern. The 'adjust' phase tries to push the skeleton to the medial line. Chu and Suen cite two major reasons that cause the skeleton to be away from the medial line:

1) Most of the contour points deleted during the smoothing process lie on the outer contour, and

2) If the input pattern has an uneven line thickness, like the letter 'e' shown in fig 2.46, some point joining the thin line and thick line becomes a break point during the thinning process, which in turn causes the skeleton to lean inward as illustrated in fig 2.47.

The 'adjust' phase applies a distance function to calculate a value for each black pixel. The location of a point on the skeleton is adjusted by 'propagate' and 'move' processes if it satisfies the following conditions:

In the propagate step, it is established how close to the medial line a pixel is; it assigns an integer value of the distance function to each black pixel in
Fig. 2.46: Letter 'e' with uneven line thickness.

Fig. 2.47: The corresponding skeleton of 'e'
the binary pattern. A pixel X with a distance function value higher than its neighbours means that it is closer to the medial line. A black pixel is originally assigned a value of 1, its windows (3x3, 5x5, 7x7, 9x9, etc.) are then examined one by one, from small to large, to see if they are enlargeable.

A window is enlargeable if all the pixels are black. The distance function value is incremented by 1 whenever a window is found enlargeable. The 'move' step seeks at every point on the skeleton; the left and right neighbours of all skeleton points are examined first to determine the adjustments. The resulting binary pattern and the matrix of distance function are then rotated though an angle of 90 and the same logic is applied to the skeleton once more to account for top and bottom adjustment. Fig.2.48 shows an example of a thinned binary pattern.

Chu's thinning algorithm is fast and reliable; it produces skeletons with a lower level of noise.
Fig. 2.48: Examples of thinned binary patterns.
2.11 Sinha's thinning method

Sinha [1987], however, makes one more important assumption which was implied in the Chu's algorithm and that is the relation between a character's thickness and its 'skeleton'. Although, his algorithm assumes that characters are of uniform thickness, it is clear that further research would be able to extend it to cope with varying thickness in a character's body. The algorithm consists of the following steps executed in sequence:

a) Find the character boundary. A boundary pixel is defined as one that does not have all the four neighbours in the principle directions. Figure 2.50 shows the boundary of fig.2.49.

b) Label the boundary. Four types of labels (namely, H, V, R and L) are used to label the boundary. They exhibit the property of being part of a horizontal, vertical, right going diagonal or left going diagonal line segment respectively at a local level. The local spread in case of H and V labels has been chosen to be three pixels; and that in the case of R and L labels, as two pixels. Each boundary pixel is assigned labels; a pixel may have multiple labels if it satisfies more than one local property. Thus, for example, a pixel is assigned an H label if it forms part of at least 3 pixels in the horizontal direction in a sequence and may also be assigned an L label if it also forms
Fig. 2.49: The character to be thinned.

Fig. 2.50: Boundary of the character in fig.4.24.
part of a left going diagonal. Figure 2.51 shows labeled parts of the boundary of the example pattern.

c) Estimate character width. A width histogram is obtained by row scan on the H, R, and L labels; and by column scan on the V label. While scanning, only those widths are considered that have the same labels on the two sides of the character boundary. The maximum of the width histogram gives an estimate of the character width. A conservative estimate is used by adding 10% to it.

d) Obtain core skeletal segments. Scanning is performed in a direction that is perpendicular to the line segment denoted by the label. For each boundary label encountered, a search is made for the same kind of label on the other side of the boundary in the direction of the scan. If the search is successful and the distance between the pixels in question is found to be less than the estimated character width and is more than unity, then the mid point of the two boundary pixels is marked as a core skeletal pixel. The core skeletal pixel is assigned the same label as that at the boundary. Some cleaning operation is performed on the skeletal pixels that are not in conformity with the nature of the labels. Steps e - f deal with joining any disconnected limbs as well as removing jaggedness in the resulting skeleton.
Fig. 2.51: Labelled parts of the boundary.
It should be pointed out that all operations are carried out in parallel, which makes it attractive from 'computation' point of view. Figure 2.52 shows an example of a thinned character, as it can be seen, it is a highly accurate skeleton of the character in question. As the author himself acknowledges, the main drawback of the scheme is its inability to deal with non-uniform thickness in characters. The robustness of thinning processes in terms of tackling non-uniform thickness in binary patterns, and noise modulated boundaries has a profound effect on any recognition process. In the absence of a good thinning algorithm, the task of primitive description of characters at low level becomes highly inconsistent.

2.12 Segmentation of joined characters.

Once a pattern is isolated from a document, there is a possibility of encountering two connected characters posing as one (fig.2.53) in the recognition process; indeed it is not unusual to have a scanned document with up to 10% touching characters (Nagy, 1982). To increase an OCR system's reliability, it is reasonable to identify such cases and segment them prior to recognition. Most commercial OCRs in the market, include a special model for ligatures common in printing industry, like 'ff', which is meant to be two connected 'f's. However, in cases where characters have been connected due
Fig. 2.52: An example of a thinned character.
Fig. 2.53: An example of connected characters.

Fig. 2.54: Stroke count of several characters.
to low resolution scanning or poor quality input, it has to undergo a segmentation process prior to recognition. Should the segmentation process fail, the characters are marked to be dealt with by the post-processor. The earliest paper on segmenting joined characters is due to Hoffman, et al. (1971). They propose two separate algorithms based on the number of vertical black runs in a character as well as some form of feature extraction. The first algorithm computes the number of vertical black runs of the character, alternatively termed as the number of its horizontal strokes by the authors. Examples of stroke-count sequences for several characters are depicted in fig.2.54. The characters I, S, T and A produce patterns in their stroke count sequences that are symmetric about the centre line of the character, while characters like L and 7 have unsymmetric patterns. The character T, for example shows the pattern (1121211), which is symmetric about the centre vertical stroke. Stroke counting was applied to the entire alphabet of upper case alphanumeric characters, clustering them into a few classes. Serif characters like M, N, T, U, W and Y that are symmetric about the centre line exhibit, in contracted form, a (12121) pattern, while D, V and 7 exhibit a (121) pattern. In all, 11 classes or sub-classes were identified during experiments on serif characters.
Sectioning (partitioning) decisions are made by finding the closest match between the exhibited stroke-count pattern and one of a set of pre-stored reference stroke-count patterns. An important disadvantage of the method as pointed out by the authors themselves, is that it is not capable of coping with characters of variable width. As an added refinement to the algorithm, some statistical relationship between the number of black pixels and the character's shape is utilised to improve the reliability of the algorithm. The experiments reported in the paper show a correct segmentation of 93.3%. Casey and Nagy (1982), envisage a combination of recursive segmentation and character classification. The pattern array to be recognized is viewed by the character recognizer through a window whose width and location are dictated by a supervisory routine. The window is initially set at the full width of the patterns so that if the array contains a single character, the recognizer can identify it in one step. If the pattern is rejected, however, indicating that the full array does not belong to the alphabet, then the viewing window is narrowed from the right hand side and the recognizer is activated for the truncated array. Gradual closing of the window followed by recognition is repeated until either the truncated array is successfully recognized, or the window becomes so narrow that the search is abandoned. In the former case, the supervisor
records the ID and the truncation point. The window is then reset with its left-hand margin at the truncation point, and its right hand margin is once more set to the end of the input array. The combined windowing and recognition is repeated on this reduced array. The segmentation terminates successfully if the residual array after a positive recognition is either null, or narrower than any character in the alphabet. In the case where no classification can be found before the window closes completely, it is hypothesized that the failure is either due to the initial input or to the truncated version. In the first instance, the entire array is rejected. In the second instance, the supervisor looks at the previous successful segmentations and readjusts the truncation boundaries accordingly. Out of 106 connected characters used in the experiment, all but five of them were correctly recognized. Figures 2.55a and 2.55b show a block diagram of the scheme as well as some examples of successful segmentation.

Lee and Holt (1988) refine Casey's method of segmentation by devising a partitioning strategy which can make an educated guess of possible points to break the pattern prior to recognition. They use the vertical black runs of a character to decide where it should be partitioned; it bears remarkable resemblance to Hoffman's method, but is more efficient in its formulation.
a) Combined segmentation/classification method (Casey, et al., 1982).

b) Successful segmentation on a multi-font document

Fig. 2.55: Block diagram of segmentation/classification scheme and examples of successful segmentations.
and simpler to implement. Candidate 'partitioning' positions are selected as follows:

1- Each column of pixels in the pattern is assigned a value in the following manner: If the black pixels in the column are grouped into a single run, the column value is equal to the number of black pixels in the column; if there is more than one run of black pixels in the column, the column value is one more than the height in pixels of the object.

2- The assigned column values are then searched for local minima. If two or more consecutive columns have the same value (i.e. local minimum), only the right most of these columns is treated as a local minimum.

3- A candidate partition is inserted immediately to the left of each local minimum, unless:

a) The column value is greater than one third of the pattern height, and the column immediately to the right contains more than one run of black pixels, in that case the partition is inserted immediately to the right of the local minimum.

b) The partition is sufficiently close to either end of the original pattern to introduce a segment narrower than the prescribed minimum symbol width.

Figure 2.56 shows typical candidate partitions in a pattern of three con-
Fig. 2.56: Typical candidate partitions in a pattern of three connected characters.

Vertical projection V.

Break point criterion function.

Fig. 2.57: Kahan's Character segmentation technique.
Two different strategies (named A and B) have been adopted in the implementation of the scheme. In strategy A, the segment to the left of a partition is tested for a match with the reference library (of patterns). If a match is found, the character is coded, and the segment to the right of the partition is then treated as a new pattern. The main drawback of the strategy, as pointed out by the authors is that correct segmentation is only achieved when the characters in a connected group can be matched sequentially from the left.

The second strategy, namely strategy B overcomes this problem by accepting a segmentation if any segment of the group can be matched with one of the patterns in the reference library. First, the horizontal limits of all possible segments indicated by candidate partitions are tabulated. These segments are then stored, so that

(a) those at either end of the pattern are tested first, and

(b) among each category, those whose width are closest to the average symbol width are tested earlier.

Since the scheme had been designed in the context of a document coding scheme, the experimental results have been tabulated according to the
compression ratio achieved rather than focusing on the number of successful segmentation of characters. All segmentation schemes considered so far have been designed in the context of a pattern matching recognizer. The only character segmentation technique which has been designed in the context of a structural analysis/recognition system is due to Kahan, et al. [1987]. Unlike 'pattern matching' which can be computed with high speed, structural description and recognition of characters are complex and time consuming compared to simpler schemes like pattern matching. The proposed character segmentation method is, therefore, applied only once to break the connected characters. The segmentation scheme is based on the assumption that the vertical projection \( V \) of a character's black run takes a sharp dip when characters are joined at serifs; Although, serifs and double-o joins always produce low values of the vertical projection, the converse is not true as the examples of fig. 2.57 show. Many other characters such as 'H' also yield low values of \( V \). A major difference between joins and thin lines, however, is the quick change in \( V \) in the neighbourhood of a join. It always corresponds to a sharp minimum of \( V \), and therefore, the maximum in the second difference, \( V(x - 1) - 2 \times V(x) + V(x + 1) \) must be taken into account. A ratio of the second difference to the value of the projection is also found to be a better
measure (fig.2.57). Unfortunately, there is no mention of the scheme's performance in the paper. Designing a robust character segmentation scheme is still a challenging problem, particularly when they are to be incorporated in a multifont/omnifont character recognition system.
Chapter 3

Document analysis and Field segmentation

(DAFS)
Document analysis and field segmentation (DAFS).

3.1 Introduction

Document analysis and field segmentation is one of the most essential parts of office automation processing. A typical digitised document typically, consists of blocks of text, line drawings and halftone images; In order to store a digitised document efficiently, it is compelling to automatically analyse and separate the text and non text (i.e. graphics, line drawings and halftone images) allowing suitable coding strategies for different portions of a document to be adopted.

The main components of a document analysis and field segmentation (DAFS) as an integral part of an automated office environment are depicted in fig.3.1. At the first phase, documents are optically scanned, digitised and subsequently binarised. Phase two is comprised of block segmentation and labelling which is the heart of a DAFS system. The result of phase two is a set of regions labeled as text, graphics, line drawing and halftone images. At the next phase, labeled regions are (and possibly in parallel) processed separately according to the nature of their contents.

Two main philosophies, namely 'top-down' and 'bottom-up' approaches,
Figure 3.1: A typical Document Analysis and Field segmentation (DAFS) system as a part of a document processing environment.
have always been the undercurrent of DAFS systems proposed to-date. Top-
down, or knowledge based, techniques proceed with an expectation of the
nature of the document; the document is first divided into major regions
which are further divided into sub regions, etc. Bottom up, or data driven
methods progressively refine the data by layered grouping operations.

Although no DAFS system takes a pure approach, most can be identified
as being aligned with one philosophy. In the following, the most significant
contributions to document analysis and field segmentation are reviewed and
a new document classification algorithm is described.

3.2 Document segmentation in retrospect

One of the earliest papers on text discrimination is due to Johnston[1974].
She argues that printed text observed from a distance, no matter what the
language, appears to the eye in the form of horizontal stripes. Thus, a
program is designed to separate configurations which are stripe like, from
those which are not. Her scheme assumes some unrealistic constraints such
as prior knowledge of the font size and the fact that non text portions of the
document must be at least two character widths away from the text.

The algorithm propagates the black points out horizontally by an amount
equal to twice the character width and then back in by three character widths.
These steps convert the text area to irregular horizontal solid stripes, two character widths shorter than the lines of original text. In the next step white points (pixels) in the background are propagated vertically by one half the character height. At this point, the white points which formed the spacing between the lines of text will fuse together to form a solid white rectangle. At step 3, white points are propagated back vertically by the same amount as in step 2, and out horizontally by one character width; this effectively changes the dimensions of the white rectangle so that it matches the area corresponding to the text. The plane resulting from steps 1-3 is now inverted and used as a mask to lift up text from the original document.

Photographs (mostly dark) will have fused together and will disappear. On the second pass, using the output from the first pass as input, the data is processed similarly except that black points are the object of propagation rather than the white. Although Johnston's scheme is not robust, it will be shown later that the idea of pixel propagation can be used to design a much more reliable document segmentation scheme. Scherle, et al. [1980] proposed a frequency domain description of printed text. A document is optically transformed to the spatial frequency domain; fig. 3.2 shows a portion of a typed document and its spatial Fourier spectrum. Within the spectrum, the
vom Menschen auf
ten des Menschen
Arbeitsprozeß ist
1, vom Menschen zu
Mensch kann z.B.
und seiner Kenntn

Figure 3.2: A portion of a typed document and its spatial Fourier transform.

road, with Mam
psi still earned accl
orning we struggle
king lot like a reneg
team. Only a few mi
ugh, we were tache
ew day. Passing in

Figure 3.3: Spatial and spatial frequency domain of printed text.
energy distribution along vertical lines relates to the sequence of the text lines while the energy distribution along the horizontal direction corresponds to distance between single characters. Figure 3.3 displays the spatial frequency spectrum of printed text; as it can be seen, the difference between the text lines is constant, but the sequence of characters varies within certain limits. Scherle, et.al. believe that the optical implementation of two dimensional Fourier transforms is time consuming and not practical in the foreseeable future. The first statistical discriminator is based on testing the background light intensities. Since the brightest grey levels occur in general most frequently in the background text, the percentage of bright levels within a window can be tested. The brightest level $i_{\text{max}}$ of the whole image is calculated and the number of samples between $i_{\text{max}}$ and a threshold $S = 0.8i_{\text{max}}$ is evaluated. The formula

$$K = \frac{\sum_{i=0}^{i_{\text{max}}} h(i)}{N}$$

yields the desired percentage of bright levels within the window, where $h(i)$ is the value of the histogram at grey level $i$ and $N$ is the total number of samples within the window. The $K$ values of all windows yield a feature
field. To obtain a decision,

the feature field is compared with a threshold $t$, where by values of $K$ higher than $t$ are classified as text and those smaller as picture. For $t$ an empirical value of 0.4 was derived; an example of this algorithm is shown in fig.3.4. Scherle et. al.[1980] also suggested a second method makes use of 3rd. and 4th. statistical moments to model the shape of text histogram. To make the parameters independent of the absolute brightness as well as the dynamic range, a normalisation of the moments to the mean grey level $\mu$ and to the variance $\sigma^2$ is needed. These normalised moments are referred to as skewness ($S$) and curtosis ($C$):

$$S = \frac{\sum (i - \mu)^3 h(i)}{\sigma^3}$$

$$C = \frac{\sum (i - \mu)^4 h(i)}{\sigma^4}$$

$h(i)$ is the value of the histogram at grey level $i$, because of the typical shape of $h(i)$ for text, $S$ will always be negative. An experimental evaluation of $S$ and $C$ for many examples of text is displayed in fig.3.5, which shows a typical cluster along a quadratic curve.
Figure 3.4: An example of Scherle's algorithm.

Figure 3.5: Experimental values of $S$ and $C$ for text.
Further experimental tests suggested an analytical constraint for the classification of text where

\[ \hat{C} = a_2 S^2 + a_1 S + a_0 \]

Further experimental tests suggested an analytical constraint for the classification of text where

\[ |C - \hat{C}| < b_1 \hat{C} \quad \text{along the C axis} \]

\[ S < b_2 \quad \text{along the S axis} \]

while \( b_1 \) and \( b_2 \) are constant. Figure 3.6 shows the mask and the results obtained by applying this algorithm. As it is evident, the block segmentation is not accurate. To remove the inaccuracy, the authors propose the use of the 'blow/shrink' algorithm due to Johnston [1974], described earlier. The authors admit that the scheme is not robust particularly for pictures that yield, for large areas, histograms similar to that of text; nevertheless, they claim this happens very seldom. Abele et al. [1981] which included Scherle and Wahl, another member of his team, recognised the fact that printed regions should be formed in blocks of segmented areas, separated from the background before applying any statistical discriminators. They further developed the 'shrink and blow' algorithm to extract a more accurate block
Figure 3.6: The mask and the results obtained by Scherle's 2nd. algorithm.
segmentation of printed areas as follows. The binary data is processed line by line and then column by column.

At the first stage, the length of white runs is limited to a lower bound; whenever white runs occur which do not reach this bound, they are suppressed i.e. are changed to black runs. At the second stage a similar operation is also carried out on the vertical white run lengths. The resulting data is combined (logical OR) with the data obtained at the first stage, which results in a more reliable segmentation of the document into background and printed blocks. It is obvious that, for example, white runs which correspond with the space between two neighboured text lines are suppressed due to the lower bound of 'white runs', whereas black regions of stripe shape may be a good early indication of text. A typical example of the algorithm is displayed in fig.3.7. In order to classify each block of data, simple statistical features were evaluated. The relative frequency of 'white' pixels within the region \( p_w \) and the relative frequency of white or black pixels in the horizontal direction \( p_{\text{hw}} \) were used to discriminate between text and non-text areas and the subsequent preliminary results showed a relatively reliable classification scheme.

Wahl later joined a 'Document Analysis' team, lead by K.Y. Wong at
Figure 3.7: A typical example of Abele's algorithm.
'a' and 'c' show the horizontal white runlength suppression.
'b' and 'd' display the vertical white runlength suppression,
while 'e' and 'f' show the result of logical 'OR' operation.
IBM. The result of this cooperation appeared in 1982. Wong et al. [1982] describe by far the most sophisticated and extensively tested document segmentation schemes. Drawing on his previous experience, Wahl refined his method of block segmentation more accurately. The algorithm is called 'Run length Smoothing Algorithm' or RLSA for short. The basic RLSA is applied to a binary sequence in which white pixels are represented by 0 s and black pixels by 1 s. The algorithm transforms a binary sequence x into an output sequence y according to the following rules:

1. '0' s in x are changed to 1 s in y if the number of adjacent 0 s is less than or equal to a pre-defined limit c.
2. '1' s in x are unchanged in y.

For example, with c=4, the sequence x is mapped into y as follows:

\[ x:0001000001010001000000011000 \]
\[ y:11110000011111111100000001111 \]

When applied to pattern arrays, the RLSA has the effect of linking together neighbouring black areas that are separated by less than c pixels. With an appropriate choice of c, the linked areas will be regions of a common data type. The degree of linkage depends on c, the distribution of white and black pixels in the document, and the scanning resolution. The RLSA
Figure 3.8: A typical document to segment.
is applied row by row as well as column by column, yielding two distinct bit maps. Because spacings of document components tend to differ horizontally and vertically, different values of \( c \) are used for row (e.g. \( c_n = 300 \)) and column (e.g. \( c_v = 500 \)) processing. Figures 3.9 and 3.10 show the results of applying the RLSA in the horizontal and in the vertical directions of fig.3.8.

The two bit maps are then combined in a logical AND operation. Additional horizontal smoothing using the RLSA (\( c_s = 30 \)) produces the final result illustrated in fig.3.11. Clearly, this technique does separate lines of text (in form of black stripes) accurately, particularly when they are well spaced. The next step in the scheme is to identify the nature of each segmented block; the following measurements are, thus, taken:

a) Total number of black pixels in a segmented block (BC).

b) Minimum x-y coordinates of a block and its x,y lengths (\( x_{\text{min}}, \Delta x \), \( y_{\text{min}}, \Delta y \)).

c) Total number of black pixels in original data (TC).

These measurements are subsequently used to derive the following features:

1- The height of each block segment: \( H = \Delta y \).
Figure 3.9: Applying RLSA in horizontal direction.

Figure 3.10: Applying RLSA in vertical direction.
2- The eccentricity of the rectangle surrounding the block, \( E = \frac{\Delta x}{\Delta y} \).

3- The ratio of the number of black pixels to the area of the surrounding rectangle: \( S = \frac{BC}{\Delta x \Delta y} \). If \( S \) is close to one, the block segment is approximately rectangular.

4- The mean horizontal length of the black runs of the original data from each block: \( R_m = DC/TC \), where \( DC \) is the total number of black pixels in original data from the block.

These features are used to classify the block. Since text is considered to be the pre-dominating data type in office documents, and text lines are basically textured stripes of approximately a constant height \( H \) and mean length of black runs \( R_m \), text blocks tend to cluster with respect to these features. This is illustrated by plotting block segments in the R-H plane, as shown in figure 3.12. Each table entry is equal to the number of block segments in the corresponding range of R & H. Thus, such a plot can be considered as a two dimensional histogram. For instance, the text lines of the document shown in fig.3.8 form a clustered population within the ranges \( 20 \leq H \leq 35 \) and \( 2 \leq R \leq 8 \). The two solid black lines in the lower right part of the original document have high R and low H values in the R-H plane,
2. The eccentricity of the rectangle surrounding the block, \( E = \frac{\Delta x}{\Delta y} \).

3. The ratio of the number of black pixels to the area of the surrounding rectangle: \( S = \frac{BC}{\Delta x \Delta y} \). If \( S \) is close to one, the block segment is approximately rectangular.

4. The mean horizontal length of the black runs of the original data from each block: \( R_m = DC/TC \), where \( DC \) is the total number of black pixels in original data from the block.

These features are used to classify the block. Since text is considered to be the pre-dominating data type in office documents, and text lines are basically textured stripes of approximately a constant height \( H \) and mean length of black runs \( R_m \), text blocks tend to cluster with respect to these features. This is illustrated by plotting block segments in the \( R-H \) plane, as shown in figure 3.12. Each table entry is equal to the number of block segments in the corresponding range of \( R \) & \( H \). Thus, such a plot can be considered as a two dimensional histogram. For instance, the text lines of the document shown in fig.3.8 form a clustered population within the ranges \( 20 \leq H \leq 35 \) and \( 2 \leq R \leq 8 \). The two solid black lines in the lower right part of the original document have high \( R \) and low \( H \) values in the \( R-H \) plane,
whereas graphic and halftone images have high values of H. Note that the scale used in R-H plane (fig.3.12) is highly nonlinear. The mean value of block height $H_m$ and the block mean black pixel run length $R_m$ for the text cluster may vary for different types of documents, depending on character size and font. Furthermore, the text cluster's standard deviations $\sigma(H_m)$ may also vary depending on whether a document is in a single font or multiple fonts and character sizes. To permit self adjustment of the decision boundaries for text discrimination, estimates are calculated for the mean values $H_m$ and $R_m$ of blocks from a tightly defined text region of the R-H plane. Finally, a variable, linear, separable classification scheme assigns the following four classes to the blocks:

Class 1 Text:

$$R < C_{21}R_m \& \ H < C_{22}H_m$$

Class 2 Horizontal solid black lines:

$$R > C_{21}R_m \& \ H < C_{22}H_m$$

Class 3 Graphic and halftone images:

$$E > 1/C_{23} \& \ H > C_{22}H_m$$

116
Figure 3.11: Smoothing using the RLSA.

Figure 3.12: The R-H plane, Wong[1982].
Empirical values have been assigned to the parameters based on several training documents. With $C_{21} = 3$, $C_{22} = 3$ and $C_{23} = 5$, the scheme was tested on a number of documents with satisfactory performance. Wong et al. [1982] acknowledge certain limitations of their block segmentation and text discrimination scheme due to small line-to-line spacing found in some documents; they also acknowledge its relative limitation due to font size implicitly. They fail to mention, however, the situation where there are no completely white scan lines between text lines (i.e. interlaced text lines). In this case all text lines are black stripes of rectangular shape joined to one another, forming a big black patch as would be the case should we apply it to the piece of text in fig.3.13. It is obvious that no rectangular black blocks could be detected. In my view, perhaps a more efficient way is to detect a text base line (normally found in text lines) in a block or blocks which cannot be confidently identified; should the scheme fail to detect a base line, it is most probably a non-text piece of data.
These text specimens demonstrate character spacing in three versions known as tracks, the examples being set solid. Typefaces are set track two by us unless otherwise specified.

Figure 3.13: An example which fails the segmentation algorithm.
Almost a year later, Okamoto, et al. [1983] reported a similar development. Although the mathematical representation is totally different, the segmentation process is similar. The underpinning assumptions of the scheme are that:

a) A text is composed of character lines and vertical intervals between lines which are almost constant.

b) Intervals between words are also nearly constant.

c) Words are composed of letters in which letters are printed at regular intervals.

The input document is first pre-processed to remove noise, it then proceeds to block segment the document's contents. After completing 'block segmentation', the scheme attempts to label the blocks by collecting their corresponding run length statistics. The statistical measurements and features used to identify text lines are rudimentary compared with Wong’s segmentation and text discrimination scheme. It also concentrates only on discriminating text and graphics; no attempt is made to separate halftone images, vertical and horizontal solid lines as reported in Wong’s proposed scheme. Masuda et al. [1985] make use of horizontal and vertical projection profiles to identify blocks of text.
Projection profiles of text in horizontal and vertical directions provide regular patterns which are distinct from non-text data. Slant projections, however, give a poor indication of a document page as illustrated in fig.3.14. As it can be seen steep variations in vertical and horizontal projections are prominent while in slant projections the variation is much smoother. Documents are often tilted by a few degrees when placed on a digitizer which, in turn, has a significant bearing on a document's projection profile as well as other document processing stages. The segmentation scheme checks the projection profiles successively for steep changes every time the document image is rotated by a few degrees; repeating this process will detect the angle which yields the steepest variation and hence text. Zen et.al.[1985] take a similar approach but assume that text is hand written and normally confined within line drawings. Figure 3.15 shows an example cited by the authors, where the corresponding projection profiles are also included. It is clear that long spikes refer to line drawings whilst the smaller ones represent text. The vertical and horizontal projection profiles are used to divide the pattern in vertical and horizontal directions. The resulting boxes of data are examined to determine their corresponding nature. This task is carried out by computing a feature value named $V$ from the variance of horizontal and vertical
Figure 3.14: Masuda's projection profile [1985].
projections as described by:

\[ V = | \log \sigma_v - \log \sigma_h | \]

Where

\[ \sigma_v = \frac{1}{N} \sum_{Y=1}^{N} (p_v(y) - m_v)^2 \]

\[ \sigma_h = \frac{1}{M} \sum_{X=1}^{M} (p_h(x) - m_h)^2 \]

\[ m_v = \frac{1}{N} \sum_{Y=1}^{N} p_v(y) \]

\[ m_h = \frac{1}{M} \sum_{X=1}^{M} p_h(x) \]

and \( p_v \): vertical projection, \( p_h \): horizontal projection.

The box which corresponds to character(s) has relatively small \( V \), while a box which contains non-text has a greater value of \( V \). This particular method has been tried on handwritten flow charts, circle graphs and bar graphs; the authors have reported a satisfactory performance. Although profile projections provide some vital information regarding a document's contents, they can not be used as a vehicle for a robust segmentation. Higashino et.al.[1986] propose a formal document description language which can be conveniently used to describe a document's layout. This approach makes it possible to
write layout models of documents and provides a mechanism that analyses the document image in a top down manner. The assumption is that most documents, particularly printed matter, have their own layout rules. Within each class of documents, these rules are common, creating nearly fixed forms. Thus, by incorporating models of a layout as document knowledge, regions that contain bibliographic items, such as the title, authors and other important items, can be identified and extracted from the image of document automatically.

Recently, Nagy et.al.[1984,1986] have proposed a hierarchical 'bottom-up' approach to document analysis. It is assumed that an OCR system has already performed the character recognition task and what remains is to derive the document's structure. The main concept at the heart of the scheme is the X-Y hierarchical representation. The structural elements of a page are defined as: Columns, Paragraphs, Titles, Figures, Lines of text. The data structure, namely X-Y trees, has nodes which correspond to a rectangle (block of data). The successors of a node correspond either to a set of rectangles obtained by horizontal partitions (X-cuts) of the parent rectangle, or to a set of rectangles obtained by vertical partitions (Y-cuts). Horizontal and vertical sub-divisions (X-cut sets and Y-cut sets) alternate
Figure 3.15: An example with the relevant projection profiles.
strictly, level by level.

The first sub division is set arbitrarily either to horizontal or vertical. The root of the X-Y tree is the rectangle corresponding to the entire page; the leaves form a tesselation of the page. To obtain the X-Y tree decomposition of a page as a set of nested rectangles, at each step of the recursive process of page decomposition, a rectangle is subdivided into smaller rectangles by making cuts which are pre-determined along the horizontal or vertical direction. This process is aided by examining the projection profiles in both directions. A set of rules to label blocks of data are also incorporated to facilitate the task of document analysis and description. However, the X-Y tree approach described above has one major drawback; it fails with skewed lines. In the case of noisy documents, the cuts may be erroneously placed. Once cuts are placed, they are impossible to recover since cut-building is a generative process. Wilcox et al. [1988] follow similar concepts but a different hierarchical description. The document is first described in terms of a list of marks (connected black regions). Individual marks are classified as text or graphics based on their size, complexity and classification of neighbouring marks. The information of text lines is based on their (horizontal) adjacency of individual marks which have been classified as text. Thus, with the
extracted information on the distribution of character widths, an inference is
drawn about maximal inter-character spacing which in turn is used to group
characters into words.

Text blocks are assumed to be formed of text lines. They must display
periodicity in the line spacing. The auto-correlation of the absolute position
of the base of each mark is used to estimate the vertical periodicity. Figure
3.16 shows the hierarchical representation of a document according to Wilcox
et.al.[1988]. The tree is built from bottom up, with the segmenter extracting
the text lines, text blocks and graphics. Columns of text are built by merging
text blocks of approximately equal width, that are vertically adjacent and
that have no intervening graphics.

Text on graphics is considered to be the union of all text blocks or text
lines that intersect a given graphics block. Kubota et.al.[1984] report a
'bottom-up' rule based scheme which is based on production systems and
blackboard communication. A production system is organised with different
levels of rules. A typical control flow can be summed up as follows: the im-
age is segmented and data about various regions are obtained. The acquired
data includes intrinsic properties (shape, size, etc) as well as spatial relation-
ship between regions. Knowledge rules pertain to intrinsic properties and
Figure 3.16: Hierarchical representation of a document page.
spatial relationships. Control rules decide what knowledge rules are to be executed and in what order, and thus act as focus of attention mechanisms to guide the search. Strategy rules supervise the entire search and classification process, and determine whether a consistent interpretation of the image has been obtained.

If the data from the initial segmentation is insufficient, then the system invokes further processing to obtain more data. Although, using production rules and blackboard communication is not always a trivial design in natural scene interpretation, it has some advantages when applied to processing highly structured documents. Kubota's scheme is influenced by this concept. More than 100 rules are used to extract connected components in a document. The blackboard concept embedded in the system can be conveniently illustrated in the form of a diagram as depicted in fig.3.17. The blackboard is used as read/write data area in both the production system and program modules and is made up of three planes of identical sized arrays. Data, indicators and program module names and arguments are respectively stored in these planes. Data concerning position, size, likelihood and attributes for connected components are stored in the first data plane. The indicator plane indicates whether or not the value at the same position in the first plane
Figure 3.17: Mechanism for driving program modules; (Kubota's Blackboard approach, 1984).
has already been collected. When the production system requires some data for matching, the indicator is checked first; if the desired data has already been collected, the values in the data plane are transferred to the production system.

In the case where the data has not yet been collected, the program module stored in the third plane, is activated as its function is carried out. The pure bottom-up approach is generally more reliable than the top-down approach, but it is time consuming. It requires the detection of the various connected components, and then a successive agglomeration into bigger blocks. With a high resolution imaging necessary in such applications, connected component analysis is very slow. While bottom-up techniques require extensive usage of memory resources, top-down techniques are fast but error prone. A hybrid scheme is thus needed to combine both methods in the most efficient manner. Designing robust hybrid schemes is still an open research problem; but it is clear that the new trend of research in this field will emerge as a compromise between the two differing approaches.

All document analysis/segmentation schemes reviewed so far, do not make the simple assumption that classifying scanned documents to those with entirely text contents, 'entirely non-text 'contents, and a mixture of
both, will save considerable processing power as well as improving the overall speed of the system. In the next section a new classification algorithm based on Fuzzy theory is described.
3.3 Basis of the proposed method

Texts of different fonts are well structured images; they display some invariant statistical features which are useful in building up some criteria to distinguish text from non-text. Identifying these features and comparing them with a mathematical model form the basis of our approach; it is similar to the concept of 'model reference control' [Harris, et.al. 1981] where physically measured parameters are constantly compared with a pre-defined model in order to predict or update a control strategy. However, in our case, there is no accurate mathematical representation to model 'text', which will assist us to produce a robust deterministic approach. The only available (statistical) model [Kunt, 1979] is based on the white run length distribution (WRD) and presents two major deficiencies for our purpose:

1- Most parameters are to be computed off line i.e. they can not be utilized in an automatic environment.

2- It only represents one single font at a time, where as a document may consist of several different fonts. It also carries an error of 15% at best, because, the input data is not as 'exact' as the mathematical model.
tends to display; therefore some aspects of the input data will always escape precise mathematical representation and usually there is an inexactness in the empirical observations that can, more suitably be represented in Fuzzy sets.

For a typical sample text, the WRD (fig. 3.18) has two distinct peaks; the first peak results from frequently occurring internal white runs of letters, while the 2nd peak represents white run lengths common to the gaps between letters. Clearly, $P_1 > P_2$ in magnitude. Although, the WRD curve for different fonts may vary in magnitude but its topological shape remains invariant, that is, it displays 2 distinct peaks peculiar to text white run length distribution. This observation leads us to the question of how to model the WRD curve as well as proving that non-text documents do not show similar characteristics to those of text.

The answer to the first question is to be found in Fuzzy theory, as will be explained later but the answer to the second question is unfortunately negative; there are non-text images that display indistinguishable WRD curves as displayed by text documents. Fig. 3.19a shows a part of the Exeter map, the run length distribution curve has the same characteristics of a typical text document’s WRD (fig. 3.19b). But there is however, one feature that
Figure 3.18: WRD curve of a text document [Kunt, 1979].
non-text imagery can not have and that is a strongly well structured correlation among the black pixels in scan lines. The 'correlation' function may be formulated as follows.

The 'cross-correlation' of two continuous function is defined by $f(x)$ and $g(x)$, denoted by $f(x) \circ g(x) = \int_{-\infty}^{\infty} f^*(\alpha)g(x + \alpha)d\alpha$ where $^*$ is the complex conjugate. The discrete equivalent of the above function is defined as:

$$f_e(x) \circ g_e(x) = \sum_{\tau=0}^{T-1} f_\tau^*(\tau)g_e(x + \tau)$$

for $x = 0, 1, 2, \cdots, T-1$, where $f_e(x)$ and $g_e(x)$ are assumed to be periodic. If $g_e(x + \tau)$ is replaced by $f_e(x + \tau)$ in the above equation, auto-correlation of the discrete function is defined by:

$$f_e(x) \circ f_e(x) = \sum_{\tau=0}^{T-1} f_\tau^*(\tau)f_e(x + \tau)$$

for $x = 0, 1, 2, \cdots, T - 1$. It is a well established technique to compute correlation functions using Fourier Transform. To avoid aliasing, a $2T$ point DFT(Discrete Fourier Transform) in which the $T$ point data sequence is padded with $N$ zero valued samples. In our implementation, an algorithm due to Ahmed et.al[1975] was utilised to compute the auto-correlation function of the horizontal profile of black pixels per scan line. In this context, $T$ represents the number of scan lines.
Figure 3.19a: A portion of 'EXETER' map.
Figure 3.19b: White Run Length (WRD) of the 'EXETER' map.
A typical text document is depicted in fig. 3.20a with the corresponding auto-correlation of its scan lines’ projection profile in fig. 3.20b; it is obvious that a regular, well-structured pattern of peaks and troughs has appeared, while the auto-correlation of the Exeter map does not show a similar pattern (fig. 3.20c). Documents with interlaced text lines of varying fonts, still display the repetitive, well-structured 'peak and trough' pattern in their auto-correlation curves (see figs. 3.21a and 3.21b).

Let us assume that a document is made of text at the beginning and non-text imagery such as graphics and halftone images are gradually added to the document, thereby diffusing the well-structured features of an entirely text document. Quantifying the extent to which these features have been affected will provide us with some clues of 'non-text' presence. What we propose is effectively an extra stage of processing in the general system of fig. 3.1, located between the 'binary image' and the 'block segmentation' parts, which decides coarsely on the nature of the binary image as:

- Entirely text.
- Entirely non-text (e.g. graphics and halftone images).
- A mixture of both text and non-text.

The resulting decision allows the overall strategy of the system to avoid
manipulation. Compared with a
tentation is more suitable for
ultiplexing, mass data storage
Furthermore, digital signals c
equipment resulting from t
ntegration (VLSI) technology.
tization of an image typically
ntization and coding. In the

dustry.
Figure 3.20b: Auto-correlation of the vertical projection profile of fig.3.20a.
Figure 3.20c: Auto correlation of the 'EXETER' map.
systems. Moreover, playing an increas-
approach to design
ned VLSI systems
these considerations,
ted to high-density
duction in both the
vices [27]. This has
methodology that
sation of commun-
system realisation
em is composed of
rform computation
nts at their inputs,
gnal events at-their
l steps performed is
elements are inter-

various sorts of d (these include Lo
Aspinall et al. [6])
reported here ove
arily to the use
cification, rather t
used in earlier wo
that they allow p
fashion. The appli
the detection of
global or compl
control primitives
incorporated. How
duced into von Ne
be explicitly spec
responsibility to s

Figure 3.21a: A typical document with interlaced lines.
Figure 3.21b: the corresponding auto-correlation of fig.3.21a.
an unnecessary detailed segmentation involving heavy computation. For example, some documents with only a few lines of text are not worth detailed segmentation as the overall performance of a non-text coding algorithm will not be seriously compromised. Figure 3.22 suggests how the proposed classification can be integrated in the general system of figure 3.1.

3.4 Fuzzy sets, Fuzzy reasoning.

A common concept in most of the applications of Fuzzy sets theory is that of a 'Fuzzy restriction', meaning a fuzzy relation which acts as an elastic constraint on the values that may be assigned to a variable. Such restrictions appear to play an important role in human cognition, especially in situations involving concept formation, pattern recognition and decision making in fuzzy or uncertain environments. A more specific aim of the calculus of fuzzy restrictions is to furnish a conceptual basis for fuzzy logic and what might be called approximate reasoning which is neither very exact nor very inexact. Such reasoning plays a basic role in human decision making, because it provides a way of dealing with problems which are too complex for precise solution [Zadeh, 1968, 1972].

3.4.1 Definitions

Let $X$ be a space of points (objects), with a generic element of $X$ denoted
Figure 3.22: Block diagram of a document classifier/segmentation scheme.
by $x$. Thus, $X = \{x\}$. A fuzzy set (class) $A$ in $X$ is characterized by a membership (characteristic) function $f_A(x)$ at $x$ representing the 'grade of membership' of $x$ in $A$. Thus, nearer the value of $f_A(x)$ to unity, the higher the grade of membership of $x$ in $A$. When $A$ is a set in the conventional set theory, its membership function can take on only two values 0 and 1, with $f_A(x) = 1$ or 0 accordingly as $x$ does or does not belong to $A$. Thus, in this case $f_A(x)$ reduces to the familiar characteristic function of a set $A$ [Zadeh, 1968, 1972].

Full membership grade is designated as 1 and grade 0 implies nonmembership. The support of $A$ is defined as the set of all those elements in $X$ having a membership greater than zero. A cross-over point in the sub-set $A$ is the element $y$ whose grade of membership is 0.5. A fuzzy singleton is a fuzzy set whose support is a single element. The membership function of a fuzzy set can be expressed in terms of a standard function whose parameters approximately fit a specified membership function. Fig.3.23a and fig.3.23b depict standard functions $S$ and $\Pi$ commonly in use which can be represented mathematically by:
Figure 3.23: a) S Fuzzy membership function.

b) \(\pi\) Fuzzy membership function.
\[ S(y; \alpha, \beta, \gamma) = \begin{cases} 
0 & \text{if } y \leq \alpha \\
2((y - \alpha)/(\gamma - \alpha))^2 & \text{if } \alpha \leq y \leq \beta \\
1 - 2((y - \gamma)/(\gamma - \alpha))^2 & \text{if } \beta \leq y \leq \gamma \\
1 & \text{if } y \geq \gamma 
\end{cases} \]

where \( \beta = (\alpha + \gamma)/2 \) is the cross over point.

\[ \Pi(y; \alpha, \beta, \gamma) = \begin{cases} 
S(y; \alpha - \beta, \gamma - \beta/2, \gamma) & \text{if } y \leq \gamma \\
1 - S(y; \gamma, \gamma + \beta/2, \gamma + \beta) & \text{if } y \geq \gamma 
\end{cases} \]

where \( \beta \) is the separation between the two cross over points.

3.4.2 Operations on Fuzzy Sets

If \( F \) and \( G \) are fuzzy subsets of \( X \), their union \( F \cup G \); intersection \( F \cap G \); bounded sum \( F \oplus G \) and unbounded difference \( F \ominus G \) are fuzzy subsets of \( X \) defined by:

\[
F \cup G = \int_x \mu_F(x) \lor \mu_G(x) | x \\
F \cap G = \int_x \mu_F(x) \land \mu_G(x) | x \\
F \oplus G = \int_x 1 \land (\mu_F(x) + \mu_G(x)) | x \\
F \ominus G = \int_x 0 \lor (\mu_F(x) - \mu_G(x)) | x
\]

where \( \lor \) and \( \land \) denote maximum and minimum respectively, \( 0 \) means zero and the integral \( \int_x \) denotes the union of fuzzy singletons \( \mu_F(x)/x \) over the
universe discourse of X.

3.5 Details of the scheme and Results

To model the general shape of text’s white run length distribution (WRD), the following measures are defined using fuzzy theory:

- $C_{f_1}$: Run length of peak 2 / Run length of peak 1.
- $C_{f_2}$: Area of white run length distribution to the first peak.
- $C_{f_3}$: Area of white run length distribution up to the second peak.
- $C_{f_4}$: Magnitude of peak 1 / Magnitude of peak 2.
- $C_{f_5}$: Total number of black pixels / total number of white pixels.

To devise a measure of correlation, the ratio of the magnitudes of trough (local minima) and adjacent peaks (of the auto correlation function) are computed, together with their average ratio; thus:

- $C_{f_6}$: Average ratio of magnitude of troughs / adjacent peaks in auto-correlation function.

Samples of entirely textual contents were used to define full membership grades as well as processing documents with varying proportion of text (e.g. 3%, 10%, 50%, 90%) to define the lower and upper ends of all membership functions. Although, all coefficients represent specific features of text documents, none of them could be used to characterize a text document on its
own. The following fuzzy rule was thus devised:

\[
PPTD = \mu_{c_{f_1}}(x) \land \mu_{c_{f_2}}(x) \land \mu_{c_{f_3}}(x) \land \mu_{c_{f_4}}(x) \land \mu_{c_{f_5}}(x)
\]

where 'PPTD' stands for 'Possible Proportion of Text in a Document' and \( \land \) denotes intersection of fuzzy sets. The 'PPTD' relationship indicates the degree to which all the fuzzy constraints are valid for a digitized input document. In fuzzy sets' theory, it is the numerical minimum of the coefficients extracted from the membership grades. To illustrate the point, consider the intersecting sets in fig. 3.23c, in which, points A and B are considered to be within the region, common to all three sets. Conventional sets' theory is only concerned with the fact that, either A and B are inside or outside the constrained region; no matter how far, for example A is inside the intersection region. Fuzzy sets theory, however, distinguishes between points A and B on the basis of how much they are within the constraints and the membership grades reflect that premise.

The algorithm was implemented on an ICL Perq 1 computer. Computing
Fig. 3.23c Intersection of three different sets.
the auto correlation function for an A4 document via discrete Fourier transform (DFFT) and inverse Discrete Fourier transform (IDFFT) takes about 9 seconds on Perq 1 which can be amply reduced using more efficient algorithms. Hardware implemented DFFT/IDFFT takes far less than a second to execute.

Fuzzy decision making takes a split of a second to compute and its execution time could not be measured accurately on the computer. One potential advantage of the algorithm is its simple implementation. All the necessary input data can be acquired while the scanning is in progress; the collected data could also be used in the later stages of a document processing system.

For example, the collected information concerning the presence of various white/black run lengths in the image can be used to guess whether the non-text image is a halftone image or a drawing. Another useful feature is the detection of textual base lines using the projection profiles of black pixels. The algorithm was trained on three different documents containing various fonts. It was then tested on 5 other documents not used in the training and the results are shown in figures 3.24, 3.25, 3.26, 3.27, 3.28. The training documents are also depicted in figs. 3.29, 3.30, 3.31.
Subsidiaries are wholly-owned unless otherwise indicated.

- Computers Limited is the only diary of ICL Plc. The shares in other companies shown are held in International Computers Limited, which are held by ICL Plc.

- Overseas companies are held in the country where the business is situated except for ICL Standard Computer Komponenten, which is incorporated in West Germany.

- Subsidiaries, related companies and plants which do not materially affect the Group or the results of the Group.

**UK subsidiary**

- International (Europe) Limited
- Baric Comput
- International (Asia)

**Overseas subsidiaries**

- International (Africa)
- ICL International (Japan)
- ICL (Belgium)
- ICL Computer International (USA)
- ICL Finland

Final decision: 1.0

Figure 3.24: Test document 1.
gas outlets, and sign and layout to back or fire as a safety. Although we chosen these and a few others from the ceiling, colleges continue fixed benches in mounted at the end.

Students cannot be technical knowledge apparatus used generally, horticultural, as must be safe. In charge of the makes not the precautions must reflect experience and trainir who use the laboratorie up through the scale, i precautions rely more.

Final decision: 0.8099

Figure 3.25: Test document 2.
of the proprietary protecting strips should be used to minimise the risk of danger to the cable and to reduce the tripping hazard.

Multiple-socket-outlet installations are needed for computer equipment, particularly when peripherals such as VDUs and printers are used, as these often need their own 240 V supply. Again the keynote is planning. Adaptors are not recommended for use in schools and colleges.

In colleges, the same considerations apply as in schools, although they may differ in degree. Some types of damage to electrical equipment may be less, but students may be more inclined to bring their own equipment and use it on college premises. Some colleges manage to have strict rules about this, others fight a losing battle and meet only resentment when attempting to inspect students' property.

Laboratories and practical areas pose special problems for a number of reasons. In such areas students are often moving about and there is also usually a requirement for equipment to be moved. The provision of a large number of socket outlets is usual, but they are often located near water and gas outlets, and so care is needed in design and layout to minimise the risk of shock or fire as a result of such close proximity. Although some local authorities have chosen these

Final decision: 0.3784

Figure 3.26: Test document 3.
playing an increased approach to design VLSI systems these considerations.

led to high-density diminution in both the devices [27]. This has methodology that

of communit-system realisation

is composed of

form computation

its at their inputs, final events at their steps performed is elements are inter-

various sorts of (these include Lo Aspinall et al. [6]) reported here over

arily to the use that they allow p

fashion. The appli the detection of global or complex control primitives incorporated. How

duced into von Ne cobegin-coend blo be explicitly spec respon

ibility to s

Final decision: 1.0

Figure 3.27: Test document 4.
Final decision: 0

Figure 3.28: Test document 5.
nipulation. Compared with an anal-
tation is more suitable for low bi-
tiplexing, mass data storage and o-
thermore, digital signals can be-
et equipment resulting from the rap-
egration (VLSI) technology.
Zation of an image typically invol-
ization and coding. In the scanni

Figure 3.29: Training document 1.
aliquot for each release temperature (2-K intervals between 14 and 44 K). Again the helium and neon components of the standard were drawn through the HTCT onto the LTCT at 10 K for 5 min. The LTCT was warmed to the release temperature and the gas phase allowed to reach equilibrium with the mass spectrometer before \(^{3}\text{He}\) and \(^{4}\text{He}\) measurement. The LTCT was then warmed to 40 K and the component which remained on the trap was expanded into the mass spectrometer for measurement. The results, corrected for volume partitioning between the trap and mass spectrometer are plotted in Fig. 4. The release temperatures are average values for the trap during inlet. The ranges during inlet were 0.1 K below 20 K, 1.6 K between 20 and 30 K, and 0.2 K above 30 K. The value at 44 K was not plotted because of near maximum suppression (approximately 0.7%) due to the presence of neon. The ratio of the \(^{3}\text{He}\) released to \(^{4}\text{He}\) released is plotted in Fig. 5. The error bars are representative of the error in the \(^{3}\text{He}\) integrations of the \(^{4}\text{He}\) measurement. Above 30 K there is no observable isotopic fractionation with measurement error.

One design objective was to thermally anchor the charcoal to the chamber walls such that each grain tracked the expander head temperature with minimal thermal lag. This is dramatically demonstrated by the \(^{3}\text{He}\) and \(^{4}\text{He}\) profiles at 26.6 K during the expander temperature range was 1.6 K; see Inlet profile A, Fig. 6. When the APD-H is

![Figure 3.30: Training document 2.](image-url)
Chapter 4:

Pre-processing techniques implemented/designed
4.1 Introduction

As it was explained in chapter 2, pre-processing techniques play a significant role in refining the captured image prior to initiating the recognition process; in this chapter, pre-processing techniques utilised in the newly proposed OCR system, are described while the actual recognition systems are explained in the next two chapters.

4.2 Detection of loops/holes in a character's body

Detecting holes in a character's body reduces the number of choices the recognition process is to consider. It will considerably increase the recognition speed and significantly enhances the correct recognition rate. The easiest way to detect hole/holes in a character image is by counting the white pixels bounded between black runs within the bounding box bordering the perimeter of the character as shown in fig. 4.1. The total number of pixels \( x \), regardless of being black or white can be computed by counting the pixels within the frame. It is also possible to count the number of white pixels bounded between edges of the frame and the black pixels inside the frame \( w_2 \) and also the total number of black pixels within the bounding frame (}

137
Fig. 4.1: Detecting hole(s) in a character's body using black and white pixels.
or \( w_1 \); that is \( x - (b_1 + w_2) = w_1 \). If \( w_1 \) is non-zero it may be concluded that, there is at least one white hole in the character's body.

A better way, however, to detect holes is by tracing closed loops on a character's body. The algorithm operates as follows:

Assume a scanned image as depicted in fig. 4.2.

1- Scan the frame from top to bottom and from left to right, to find the first black pixel which has at least one white neighbour (i.e. \( p_1 \)) and commence tracing the outer boundary (while marking the visited pixels) until \( p_1 \) is reached again.

2- Find the next unvisited black pixel which has at least one white neighbour by scanning the frame from \((p_1 + 1)\) th. scan line from top to bottom and from left to right, and initiate the boundary detection until the loop is closed. Repeat this process until no other starting points can be found in the bounding box.

Since the first detected boundary is always the outer perimeter, \( T - 1 \) represents the total number of holes, where \( T \) is the total number of loops detected.
Fig. 4.2: Detecting hole(s) using contour tracing.
4.3 Detecting base lines

In a given standard document, each character has a specific vertical position relative to an imaginary horizontal line called base line. For example the base line for 'X' is the bottom horizontal line on which the character stands (fig. 4.3). However, some characters, such as the lower case 'j' have a descender below the base line. Jih [1981] patented a scheme which identifies base lines in a text document. In order to estimate the base line a series of horizontal density histograms [Casey, 1983] representing consecutive scans of the same length across the scanned line of characters is segmented; each frame contains a plot of the number of black picture elements received by the scanner as a function of character or line height. The peak in the lower portion of the histogram is used as the predicted location of the base line. By averaging the histograms for the entire line, a base line for the complete text line can be predicted. Jih defines the skew as the difference between base lines of consecutive frames on the same scan line. Although, the algorithm allows for possible skew inclination in a scanned document, it could be avoided by adjusting the entire document to compensate for any detected skew angle. It is also possible to estimate an upper case limit for character heights using the histogram of text lines; that will allow an OCR system to
Fig. 4.3: A character base line.
group characters with ascenders in one class. In general, base line identification makes the recognition task more reliable by providing distinct features (at a pre-processing stage) to extract characters from a detected text-line.

4.4 Size Normalization

Normalizing characters to a pre-determined size is the most common stage in all multifont recognizers. Size normalization reduces noise due to varying font sizes. Gudesen [1976] reports a size normalization technique which has been tested extensively. The actual character width \( \Delta w \) and height \( \Delta h \), (fig.4.4) measured by the minimum bounding box serve as the basic parameters for the normalization process shown in fig.4.5. Assuming a character width \( \Delta w \) of 5 points and a maximum grid width of 9 points, a binary vector of 5x9 elements is first composed. Starting from the left groups of 5 components are picked from this vector and summed. A thresholding procedure is next. If the sum exceeds 5 x s, where s is a threshold parameter, the result is 1, otherwise the result is 0. Consequently, a vector of 9 components is obtained which is the expansion of the row originally containing 5 grid points. This operation is sequentially carried out for all rows of the sub grid and then for the columns. At the end of the procedure a character is obtained touching all limiting lines of the original grid as demonstrated schematically in fig.4.6.
Fig. 4.4: The minimum bounding box of a character.

Fig. 4.5: Normalization algorithm.
Fig. 4.6: The result of size normalization.
which shows normalisation of two characters of different sizes. Threshold parameter $s$ ($0 < s < 1$) plays an important role in the normalization procedure. Stroke width, i.e. the mean number of black points in the grid widely depends on the setting of $s$. Gudesen derived an empirical value of 0.3 as the optimum value for $s$.

4.5 Thinning

The thinning algorithm used in our work is due to Holt, et al[1986]. The algorithm can be described as follows:

1- The original scanned image data is first filtered to remove any possible distortion which may have been caused as a result of minor deficiencies in scanning and thresholding processes. The noise-filtering in the proposed scheme uses the sets of masks shown in Figs 4.7a and 4.7b. The 3x3 masks eliminate single-pel dots and holes and irregularities in horizontal and vertical edges, and the 5x5 masks eliminate irregularities in diagonal edges. The masks in Fig 4.7a are applied first, changing selected black pels to white. Next, the masks in Fig. 4.7b are applied to the resulting image, changing selected white pels to black. Application of the masks in this order, filter the noise without introducing any unwanted distortion.

2- Edge pels of the black objects are removed by the iterative application
Fig. 4.7: Masks used for pre and post processing in the proposed thinning scheme.  a) Noise filtering masks (central pel is changed to white).  b) Noise filtering masks (central pel is changed to black).  c) Thinning masks.
of a set of 3x3 masks, subject to certain constraints. The algorithm consists
of four subcycles, which remove separately the left, right, top and bottom
edges of black objects. The masks used in the left-edge subcycle are shown
in Fig. 4.7c. If a 3x3 neighbourhood matches the edge-detection mask, the
central black pel is changed to white provided the neighbourhood does not
match any of the three constraining masks. The masks used in the other
three sub-cycles are rotations of the masks in Fig. 4.7c by multiples of 90
degrees. The constraining masks are included to preserve connectivity in
black objects and to prevent the erosion of line ends. The final thinning
cycle is followed by a further smoothing, again using the 3x3 masks of Fig.
4.7a. The purpose of this smoothing is to remove occasional notches from
otherwise skeletal lines.
Chapter 5

Hidden Markov Modelling of shape.
Hidden Markov Modelling of shape

5.1 Introduction

It is generally agreed that recognition of digitized characters depends on the system's ability to cope with noise in form of spurious edges and varying fonts as well as tilted characters. Although, a few OCR systems cater for different fonts, they do not, however, perform robustly in presence of frequent change of fonts in a document. The use of hidden Markov models for speech recognition was proposed by Baker[1975]. The theory on which his work rests is due to Baum, et.al. [1966,1967,1968,1970]; it first appeared in late 60s and early seventies and was explored more recently by Levinson, et al.[1983]. In the subsequent sections the hidden Markov theory is explained and re-interpreted to describe shape of 2 dimensional binary patterns; Levinson's mathematical notation is used throughout.

5.2 Theory of hidden Markov modelling

A probabilistic function of a hidden Markov chain is a stochastic process generated by two inter-related mechanisms, an underlying Markov chain having a finite number of states and a set of random functions, one of which is associated with each state. At discrete instants of time, the process is

143
assumed to be in some state and an observation is generated by the random function corresponding to the current state. The underlying Markov chain then changes states according to its transition probability matrix. Output of the random functions associated with each state can only be observed and the states of the underlying Markov chain can not be seen directly, hence the term hidden Markov model.

In principle, the underlying Markov chain may be of any order and the outputs from its states may be multivariate random processes having some continuous joint probability density function. We shall, however, restrict ourselves to consideration of Markov chains of order one i.e. those for which the probability of transition to any state depends only upon that state and its predecessor. We shall also limit the discussion to processes whose observations are drawn from a discrete finite alphabet according to discrete probability distribution functions associated with the states.

It is quite natural to think of a binary pattern (or in our case printed characters) as being generated by such a process. A binary pattern can be described by its contour using chain code as depicted in figure 5.1.

We can imagine the chain codes as being one of a finite number of states; in each state a symbol is produced that has one of a finite number of codes
Figure 5.1: Contour of a binary pattern can be described by chain codes.
depending, of course, on the state. Thus, the chain code at any instance of
time is determined solely by the current state of the model while the vari-
ation of the code composition of the pattern is governed predominantly by
the probabilistic state transition law of the underlying Markov chain. Hid-
den Markov theory can be formulated as follows: Let us assume that the
underlying Markov chain has \( N \) states \( q_1, q_2, \ldots, q_N \) and the observations are
drawn from an alphabet \( V \) of \( M \) sample codes \( v_1, v_2, \ldots, v_M \) The underly-
ing Markov chain can then be specified in terms of an initial state distri-
bution vector \( \pi' = (\pi_1, \pi_2, \ldots, \pi_N) \) and a state transition matrix \( A = [a_{ij}] \)
\( 1 \leq i, j \leq N \). Here \( \pi_i \) is the probability of \( q_i \) at some arbitrary time, \( t = 0 \)
and \( a_{ij} \) is the probability of transiting to state \( q_j \) given current state, \( q_i \), that
is \( a_{ij} = \text{prob}(q_j \text{ at } t + 1/q_i \text{ at } t) \) The random processes associated with the
states can be collectively represented by another stochastic matrix \( B = [b_{jk}] \)
in which for \( 1 \leq j \leq N \) and \( 1 \leq k \leq M \), \( b_{jk} \) is the probability of observing
symbol \( v_k \) given current state \( q_j \) which is denoted as \( b_{jk} = \text{prob}(v_k \text{ at } t/q_j \text{ at } t) \). Thus a hidden Markov model, \( M \), is identified with the parameter set
(\( \pi, A, B \)). To use hidden Markov models to perform recognition (of shape or
speech), two specific problems must be solved: observation sequence proba-
bility estimation, which will be used for classification; and model parameter
estimation, which will serve as a procedure for training models for each pattern. Both problems proceed from a sequence, $O$, of observations $o_1, o_2, ..., o_t$ where each $o_t$ for $1 \leq t \leq T$ is some $v_k \in V$. The classification problem can now be stated: We wish to recognize patterns known to be selected from some set, $W$, of patterns $w_1, w_2, ..., w_v$. We are given an observation sequence, $O$, derived from the set of some unknown patterns $w_i \in W$ and a set of $V$ models $M_1, M_2, ..., M_v$.

We must, therefore, compute $p_i = \text{prob}(O/M_i)$ for $1 \leq i \leq v$, and then classify the unknown pattern as $w_i$ if $p_i \geq p_j$ for $1 \leq j \leq v$. The training problem is simply that of determining the models $M_i = (\pi_i, A_i, B_i)$ for $1 \leq i \leq v$ given training sequences $O^1, O^2, ..., O^v$ where $O^i$ is known to have been derived from a pattern $w_i$ for $1 \leq i \leq v$. One could, in principle, compute $\text{prob}(O/M)$ by computing the joint probability $\text{prob}(O, S/M)$ for each state sequence, $S$ of length $T$, and summing over all the sequences.

\[
\text{prob}(O/M) = \sum_{i_1, i_2, ..., i_T} \Pi_{i_t} b_{i_1}(o_1)a_i b_{i_2}(o_2)...a_{i_{T-1}}b_{i_T}(o_T)  
\]

where $b_{i_T}(o_T)$ means $b_{i_Tk}$ if $o_T \equiv v_k$ and $I$ indicates the underlying state sequence $i_1, ..., i_T$. The interpretation of the computation in the above equa-
tion is the following. Initially (at time $t=1$), we are in state $i_1$ with probability $\pi_{i_1}$, and generate the symbol $o_1$ with probability $b_{i_1}(o_1)$. We then make a transition to state $i_2$ with probability $a_{i_1}$ and generate symbol $o_2$ with probability $b_{i_2}(o_2)$. This process continues until we make the last transition from state $i_{T-1}$ to state $i_T$ with probability $a_{i_{T-1}}$ and generate symbol $o_T$ with probability $b_{i_T}(o_T)$.

Calculation of $P(O/M)$ according to its direct definition involves on the order of $2T-1$ calculations; since at every time $t=1,2,...,T$, there are $N^T$ possible states to go through and for each summand about $2T$ calculations are required. To be precise, $(2T-1)N^T$ multiplications and $N^T-1$ additions are needed. Clearly, it is computationally intractable. Fortunately, however, there is an efficient method for computing $P(O/M)$ due to Baum et.al.[1970].

Let us define the function $\alpha_t(i)$ for $1 \leq t \leq T$ as $\text{Prob}(o_1o_2\ldots o_t, i_t = q_t/M)$.

\[
\alpha_t(i) = \text{Prob}(o_1o_2\ldots o_t, i_t = q_t/M)
\]  

(5.2)

i.e. the probability of the partial observation sequence (until time $t$) and state $q_t$ at time $t$, given the model $M$. According to the definition $\alpha_t(i) = \pi_i b_t(o_t)$, where, $b_t(o_t)$ is understood to mean $b_{ik}$ iff $o_t \equiv v_k$, then we have the
following recursive relationship for the "forward probabilities":

\[ \alpha_{t+1}(j) = \left( \sum_{i=1}^{N} \alpha_{t}(i) a_{ij} b_{j}(o_{t+1}) \right), 1 \leq t \leq T - 1 \quad (5.3) \]

which means the state \( q_j \) is reached at time \( t+1 \) from \( N \) possible states \( q_i \) \( i = 1, 2, \ldots, N \) at time \( t \). \( \alpha_{t}(i) a_{ij} \) is the probability of the joint event that \( o_1, o_2, \ldots, o_t \) are observed and state \( q_j \) is reached at time \( t+1 \) via state \( q_i \) at time \( t \). Summing this product overall the \( N \) possible states \( q_i, i=1,2,\ldots,N \) at time \( t \) results in the probability of \( q_j \) at time \( t+1 \) with all the accompanying previous observations; once this is done and \( q_j \) is known, it is easy to see that \( \alpha_{t+1}(j) \) is obtained by augmenting multiplicatively the summand quantity with the probability \( b_{j}(o_{t+1}) \). Similarly, we define another function \( \beta_{t}(j) = \text{Prob}(o_{t+1}o_{t+2}\ldots o_{T}/q_j \text{ at } t \text{ and } M) \). We set \( \beta_T(j) = 1 \ \forall \ j \) and then use the backward recursion

\[ \beta_{t}(i) = \sum_{j=1}^{N} a_{ij} b_{j}(o_{t+1}) \beta_{t+1}(j), T - 1 \geq t \geq 1 \quad (5.4) \]

to compute the "backward probabilities". The above formulation of "backward recursion" shows that in order to have been in state \( q_i \) at time \( t \) and to account for the rest of the observation sequence, you had to make
a transition of every one of the N possible states at time t+1; account for the observation symbol \( \alpha_{t+1} \) in that state, and then account for the rest of the observation sequence. If we examine the computation involved in the calculation of \( \alpha_t(j), 1 \leq t \leq T, 1 \leq j \leq N \) as well as \( \beta_t(i) \) for \( 1 \leq t \leq T, 1 \leq i \leq N \), we see that they require \( N(N+1)(T-1)+N \) multiplications and \( N(N-1)(T-1) \) additions (or about \( N^2T \) calculations) in each case. The two functions can be used to compute \( P \) according to

\[
P = \text{Prob}(O/M) = \sum_{i=1}^{N} \sum_{j=1}^{N} \alpha_t(i)a_{ij}b_j(o_{t+1})\beta_{t+1}(j)
\]

for any \( t \) such that \( 1 \leq t \leq T - 1 \). Setting \( t = T - 1 \) in the above equation gives

\[
P = \sum_{i=1}^{N} \alpha_T(i)
\]

(5.5)

where \( P = \text{Prob}(O/M) \); so that Prob(O/M) can be computed from the forward probabilities alone. A similar formula for \( P \) can be obtained from setting the backward probabilities by setting \( t = 1 \). The problem of training a model, unfortunately, does not have such a simple solution; in fact given any finite observation sequence as training data, we can not optimally train the model. We can, however, choose \( \pi, A \) and \( B \) such that \( \text{prob}(O/M) \) is locally maximized. An asymptotic analysis of the training problem has
been discussed by Baum et.al.[1966]. We can use the forward and backward probabilities to formulate a solution to the problem of training by parameter estimation. Given some estimates of the parameter values we can compute, for example, that the expected number of transitions \( \gamma_{ij} \), from \( q_i \) to \( q_j \), conditioned on the observation sequence is:

\[
\gamma_{ij} = \frac{1}{P} \sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j) \quad (5.6)
\]

Then, the expected number of transitions, \( \gamma_i \) out of \( q_i \), given \( O \) is:

\[
\gamma_i = \sum_{j=1}^{N} \gamma_{ij} = \frac{1}{P} \sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i) \quad (5.7)
\]

The last step of which is based on eqn.5.4. The ratio \( \frac{\gamma_{ij}}{\gamma_i} \) is then an estimate of the probabilities of state \( q_j \), given that the previous state was \( q_i \). This ratio may be taken as a new estimate, \( \tilde{a}_{ij} \) of \( a_{ij} \) that is:

\[
\tilde{a}_{ij} = \frac{\gamma_{ij}}{\gamma_i} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(o_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)} \quad (5.8)
\]

Similarly, a new estimate of \( b_{jk} \) as the frequency of occurrence of \( v_k \) in \( q_j \) relative to the frequency of occurrence of any symbol in state \( q_j \) can be made. It is stated in terms of the forward and backward probabilities:
Finally, new values of the initial state probabilities may be obtained from:

\[
\tilde{\pi}_i = \frac{1}{P} \alpha_1(i) \beta_1(i)
\]  

(5.10)

It can be proved [Levinson et.al.,1983] that the re-estimates are guaranteed to increase P, except at a critical point.

5.3 Considerations for implementation

Formulation of hidden Markov modelling technique may give the false impression that direct translation of the relevant formulae in to computer programs will produce immediate results; unfortunately, the crude implementation will not succeed for two main reasons: First, methods of solutions presented for training and classification require evaluation of \(\alpha_t(i)\) and \(\beta_t(i)\) for \(1 \leq t \leq T\) and \(1 \leq i \leq N\). From the recursive formulae for these quantities, eqns. (5.3) and (5.4), it is clear that as \(T \rightarrow \infty\), \(\alpha_T(i) \rightarrow 0\) and \(\beta_1(i) \rightarrow 0\) in exponential fashion. In practice, the number of observations necessary to adequately train a model and /or compute its probability will result in underflow on any computer if equations (5.3) and (5.4) are evaluated.
directly. Levinson et al. [1983] developed a method for scaling these computations that not only solves the underflow problem but also greatly simplifies several other calculations. The second problem, however, is more serious, Baum and Petri [1970] have shown that maximum likelihood estimates of the parameters of a hidden Markov process are consistent estimates (converges to the true values as $T \to \infty$) of the parameters. The practical implication of the theorem is that in training, one should use as many observations as possible which, as it was noted, make scaling necessary. In reality, of course, the observation sequence will be finite; then the following situation can arise. Assume that a given training sequence of length $T$ results in $b_{jk}=0$. To compute the probability that a new observation sequence was generated by our model, it is possible that $\alpha_{t-1}(i)a_{ij}$ is non-zero for one value of $j$ and that $O_t = v_k$, whence $\alpha_t(j) = 0$ and probability of the observation becomes zero.

5.4 Scaling

Levinson's scaling technique is based on multiplying $\alpha_t(i)$ by some scaling coefficient independent of $i$ so that it remains within the dynamic range of the computer for $1 \leq t \leq T$. A similar operation on $\beta_t(i)$ is proposed and then, at the end of the computation, the total effect of the scaling is removed. The procedure can be illustrated for equation 5.8, the re-estimation formula for

152
state transition probabilities. Let $\alpha_t(i)$ be computed according to equation 5.8 and then be multiplied by a scaling coefficient where say

$$c_t = \left( \sum_{i=1}^{N} \alpha_t(i) \right)^{-1}$$

so that $\sum_{i=1}^{N} c_t \alpha_t(i) = 1$ for $1 \leq t \leq T$, as $\beta_t(i)$ is computed from equation (5.4), we perform product $c_t \beta_t(i)$ for $T \geq t \geq 1$ and $1 \leq i \leq N$. In terms of the scaled forward and backward probabilities, the right hand side of equation 5.8 becomes:

$$\tilde{a}_{ij} = \frac{\sum_{r=1}^{t-1} \prod_{r=1}^{t-1} c_r \alpha_t(i) a_{ij} b_j(o_t+1) \beta_{t+1}(j) \prod_{r=t+1}^{T-1} c_r}{\sum_{t=1}^{T-1} \sum_{i=1}^{N} \prod_{r=1}^{t-1} c_r \alpha_r(i) a_{di} b_d(o_t+1) \beta_{t+1}(j) \prod_{r=t+1}^{T-1} c_r}$$  (5.12)

which can be simplified as:

$$\tilde{a}_{ij} = \frac{\sum_{r=1}^{t-1} C_t \alpha_t(i) a_{ij} b_j(o_t+1) \beta_{t+1}(j) D_{t+1}}{\sum_{i=1}^{T-1} C_t \alpha_t(i) \beta_t(i) D_t}$$  (5.13)

where

$$C_t = \prod_{r=1}^{t} c_r$$  (5.13a)

$$D_t = \prod_{r=t}^{T} c_r$$

and

153
Expanding equation 5.13, we have:

\[
\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} (c_1 c_2 \ldots c_t) \alpha_t(i) a_{ij} b_j(\alpha_{t+1}) \beta_{t+1}(j) (c_{t+1} \ldots c_{T-1})}{\sum_{t=1}^{T-1} (c_1 c_2 \ldots c_t) \alpha_t(i) \beta_t(i) (c_{t+1} \ldots c_{T-1})} \tag{5.14}
\]

which could in turn be simplified by cancelling out equivalent terms top and bottom of the formula, thus:

\[
\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(\alpha_{t+1}) \beta_{t+1}(j)}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i)} \tag{5.15}
\]

The above formula was derived by Levinson[1983] as:

\[
\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} C_t \alpha_t(i) a_{ij} b_j(\alpha_{t+1}) \beta_{t+1}(j) D_{t+1}}{\sum_{t=1}^{T-1} \sum_{l=1}^{T-1} C_t \alpha_t(i) a_{il} b_l(\alpha_{t+1}) \beta_{t+1}(l) D_{t+1}} \tag{5.16}
\]

It was not, however, descaled to remove the effect of scaling. To descale equation 5.15, a reversing process is performed to remove the scaling factors incorporated in the calculations, thus:

\[
\tilde{a}_{ij} = \frac{\sum_{t=1}^{T-1} \alpha_t(i) a_{ij} b_j(\alpha_{t+1}) \beta_{t+1}(j) c_t}{\sum_{t=1}^{T-1} \alpha_t(i) \beta_t(i) c_t} \tag{5.17}
\]
where $\alpha_i(i)$ and $\beta_i(i)$ have already been scaled. Similarly, for equations 5.9 and 5.10 respectively, we have:

\[
\bar{b}_{jk} = \frac{\sum_{i\geq j} \alpha_i(i) \beta_i(j)/c_t}{\sum_{i=1}^{T} \alpha_i(i) \beta_i(j)/c_t}
\]  

(5.18)

and

\[
\bar{\pi}_t = \frac{1}{\bar{p}} \frac{\alpha_1(i) \beta_1(i)}{c_1} \frac{\prod_{t=1}^{T} c_t}{c_1} = \frac{\alpha_1(i) \beta_1(i)}{c_1}
\]  

(5.19)

Although, the above scaling technique leaves the re-estimation formulae invariant, they cannot still compute $P(O/M)$. However, $\log p(O/M)$ may be recovered from the scale factors as follows. Assume that $c_t$ is computed according to equation 5.11 for $t = 1, 2, ..., T$. Then

\[
C_T \sum_{i=1}^{N} \alpha_T(i) = 1
\]  

(5.20)

and from equation 5.20 it is obvious that $C_T = \frac{1}{\bar{p}}$. Thus, from equation 5.13.a, we have

\[
\prod_{t=1}^{T} c_t = \frac{1}{\bar{p}}
\]  

(5.21)

To prevent underflow equation 5.21 can be expressed by:
5.5 Finite training sets

As it was noted earlier, the effect of 'finite training-set' size is the observation sequences generated by a putative model will have zero probability conditioned on model parameters. Since the cause of the difficulty is the assignment of zero to some parameters, usually one or more symbol probabilities, it is reasonable to try to solve the problem by constraining the parameters to be positive, i.e. maximizing \( P \) subject to the new constraints

\[
a_{ij} \geq \epsilon \geq 0, \quad b_{jk} \geq \epsilon \geq 0
\]

where \( \epsilon \) is a smaller number greater than zero. Levinson et.al. [1983] suggest a method to incorporate the above constraints in the Baum-Welch formulae; they also offer a mathematical proof which is skipped here for brevity. The modified Baum-Welch algorithm is as follows:

Suppose we wish to constrain \( b_{jk} \geq \epsilon \) for \( 1 \leq j \leq N \) and \( 1 \leq k \leq M \). We first formulate \( B \) using the re-estimation formulae. Assume that some set of the parameters in the \( j \) th. row of \( B \) violates the constraint so that \( b_{jk_i} < \epsilon \) for \( 1 \leq i \leq l \). Then set \( b_{jk_i} = \epsilon \) for \( 1 \leq i \leq l \) and re-adjust the remaining parameters so that:

\[
\log P = - \sum_{t=1}^{T} \log c_t \tag{5.22}
\]
\[ b_{jk} = (1 - l)e \frac{b_{jk}}{\sum_{i=1}^{N-1} b_{ji}}, \forall k \in [k, 1 \leq i \leq l] \] (5.23)

After performing the operation of equation 5.23 for each row of B, the resulting \( B \) is the optimal update with respect to the desired constraints. The method can be extended to include the state transition matrix if so desired; equation 5.23 may be applied at each iteration of the re-estimation formulae, or once as a post processing stage after the Baum-Welch algorithm has converged.

5.6 Types of hidden Markov models and training process

Experiments carried out using hidden Markov modelling techniques rely on specific models which are application oriented. For example hidden Markov models widely used in isolated word speech recognition, consider a special class of absorbing Markov chains that leads to what is termed as 'left to right' models. Although this particular model is not suitable for describing shape; a brief description of this type of models will prove useful in later sections.

Left to right hidden Markov models have the following properties:

i) The first observation is produced while the Markov chain is in a distinguished state called the starting state, designated \( q_1 \).
ii) The last observation sequence is generated while the Markov chain is in a distinguished state called the final or absorbing state designated $q_N$.

iii) Once the Markov chain leaves a state, the state can not be re-visited at later time.

The simplest form of a 'left to right' model is shown in fig.5.2 from which the origin of the term left to right becomes clear. The three conditions mentioned above can be satisfied as follows: Condition (i) will be satisfied if we set $\pi = (1,0,...,0)$ and do not re-estimate it. Condition (ii) can be imposed by setting

$$\beta_T(j) = \begin{cases} 
1 & \text{if } j = N \\
0 & \text{if } j < N \\
0 & \text{if } j > N
\end{cases}$$

Condition (iii) can be guaranteed in the Baum-Welch algorithm by initially setting $a_{ij} = 0$ for $j \neq 0$ (and in fact for any other combination of indices that specify transitions to be disallowed). Describing contour of a binary pattern using Freeman codes, however, can not be mathematically modelled using 'left-to-right' models as states visited once may be re-visited again. A 'constraint free' model had therefore, to be used; a model whose states can be visited without any constraint as depicted in fig.5.3, for a three state
Figure 5.2: The simplest form of left-to-right model.
model. The published literature available on the hidden Markov modelling does only deal with constrained (i.e. left-to-right) models and no other forms of models have been tested. To modify the training procedure, let us denote $O = [O^1, O^2...O^K]$, the set of observation sequences, where $O^k_1, O^k_2...O^k_{T_k}$ is the $k$ th. sequence; the observation sequences are treated as independent of the model $M$ is adjusted to maximize

$$P = \prod_{k=1}^{K} \text{Prob}(O^k \mid M)$$

$$= \prod_{k=1}^{K} P_k$$

Since the Baum-Welch algorithm computes the frequency of occurrence of various events, these frequencies of occurrence in each sequence is computed separately and then added together. Thus the new re-estimation formulae may be expressed as:

$$\bar{a}_{ij} = \frac{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_j(O_{t+1}) \beta_{t+1}^k(j)}{\sum_{k=1}^{K} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_t^k(i)} \quad (5.25)$$

and

$$\bar{b}_{ij} = \frac{\sum_{k=1}^{K} \sum_{t\geq O_t(k)=v_j} \alpha_t^k(i) \beta_t^k(j)}{\sum_{k=1}^{K} \sum_{t=1}^{T_k} \alpha_t^k(i) \beta_t^k(i)} \quad (5.26)$$
5.7 Character recognition using First order Markov models.

In the previous sections, the general form of hidden Markov modelling was formulated, in this section, however, hidden Markov modelling is precisely defined in the context of character description/classification. As it was mentioned earlier, the sequence of direction codes obtained from following the contour of a character is considered to be the output of a stochastic first order Markov process which consists of two inter-related mechanisms, an underlying Markov chain of eight states $q_1, q_2, ..., q_8$ and a set of eight random functions one of which is associated to each state. The eight states of the process are the direction codes $D_0, D_1, D_2, ..., D_7$ of fig. 5.4, while the set $V = (V_1, V_2, ..., V_m)$ from which the observations are drawn, also consists of the eight direction codes i.e. $V = (D_0, D_1, ..., D_7)$. The hidden Markov model is thus defined in terms of three parameters $\pi, A$ and $B$; $\pi$ is the initial distribution vector $\pi = (\pi_1, \pi_2, ..., \pi_8)$ where $\pi_1 = \text{Prob}(q_1 \text{ at } t = 0)$. $A$ is an $8 \times 8$ transition matrix $A = [a_{ij}]$ $1 \leq i, j \leq 8$ where $a_{ij} = \text{Prob}(q_j \text{ at } t + 1/q_i \text{ at } t)$ and models the probability of change of direction along the contour of a character. $B$ is a stochastic $8 \times 8$ matrix $B = [b_{jk}]$, $1 \leq j \leq 8$ and $0 \leq k \leq 7$ whose element $b_{jk}$ provides the property of observing the kth. direction $D_k$ at the current state $q_j$ i.e. $b_{jk} = \text{Prob}(D_k \text{ at } t/q_j \text{ at } t)$. Now, in order
to recognize R different characters, the parameters \((\pi, A, B)\) of R different Markov models are first estimated using a training process that operates on many observation sequences \(O_1^d, O_2^d, \ldots, O_n^d\) of contour direction codes, obtained for each of the R characters, \(1 \leq d \leq R\). Thus the kth. observation sequence of the dth. character will be \(O_k^d = [o_1, o_2, \ldots, o_t, \ldots, o_{T_k}]\), where each \(o_r \in V\). Once the R models \(M_d\) \(1 \leq d \leq R\) are defined, an input sequence of direction codes \(O_{in}\) of length T is classified to one of the R characters by first computing \(P_d = \text{Prob}(O_{in}/M_d)\) for \(1 \leq d \leq R\) and then assigning the input pattern to the character for which \(P_d\) is maximum; \(P_d\) is calculated as in eqn 5.5:

\[
P_d = \sum_{i=1}^{8} \alpha(i) \tag{5.27}
\]

Using the forward probabilities \(\alpha(i)\) for \(1 \leq i \leq 8\) and \(1 \leq t \leq T\) where \(\alpha_t(i) = \text{Prob}(o_1, o_2, \ldots, o_t \text{ and } q_t \text{ at } t/M_d)\) and it is computed recursively from:

\[
\alpha_1(i) = \pi_i \cdot b_i(o_1) \tag{5.28}
\]
Figure 5.3: A three state constraint free model.

\[ \begin{align*}
D_0 &= 0 \\
D_1 &= 1 \\
D_2 &= 2 \\
D_3 &= 3 \\
D_4 &= 4 \\
D_5 &= 5 \\
D_6 &= 6 \\
D_7 &= 7 
\end{align*} \]

Figure 5.4: The Freeman (chain) codes used as states to describe a character's contour.
\[ \alpha_{t+1}(j) = \sum_{i=1}^{8} \alpha_t(i)a_{ij}b_j(o_{t+1}) \quad 1 \leq j \leq 8 \] (5.29)

\[ 1 \leq t \leq T - 1 \]

where \( b_j(o_t) = b_{jk} \) iff \( o_t \equiv D_k \).

### 5.8 Model estimation

Given a set of sequences \( O^1_d, O^2_d, ..., O^n_d \) of different 'versions/fonts' of the dth character, the parameters \( (\pi^d_d, A^d_d, B^d_d) \) of the Markov model are estimated recursively, so that the probability \( P = \Pi_{i=1}^{n} p_i \) is maximized, where \( p_i = \text{Prob}(O^i_d/M_d) \). The training process for the estimation of the dth model can be described as follows.

**Step 1:** The optimization process commences with an initial set \( (\pi^d_{in}, A^d_{in}, B^d_{in}) \); \( \pi^d_{in} \) is set to \( (1, 0, 0, ..., 0) \) since the contour of a character is always traced clockwise starting from the upper left-hand corner of a window whose sides are tangent to the character. The initial estimate for \( A_{in} \) is given in table 1; it was defined heuristically giving the highest values \( a_{ji} \) for \( j = i \) and the lowest values for \( |i-j|=4 \) (a reversal of direction is extremely unlikely). \( B_{in} \) is initially assumed to be equal \( A_{in} \).
Step 2: The forward probabilities $\alpha_t^k(i)$ where $1 \leq i \leq 8$ and $1 \leq t \leq T_k$ are estimated using eqns. (5.28) and (5.29) for each $O_j^k$ sequence $1 \leq k \leq n$. The backward probabilities $\beta_t^k(i), 1 \leq i \leq 8$ where $\beta_t^k(i) = \operatorname{Prob}(o_{t+1}^k, o_{t+2}^k, \ldots, o_{T_k}^k / q_i$ at $t$ and $M_d)$ are also estimated recursively for each $O_j^k$ according to:

\[
\beta_t^k(i) = 1 \quad 1 \leq i \leq 8 \tag{5.30}
\]

\[
\beta_t^k(i) = \sum_{j=1}^{8} a_{ij} b_{j}(o_{t+1}^k) \beta_{t+1}^k(j), \quad 1 \leq i \leq 8 \tag{5.31}
\]

\[ T_k - 1 \geq t \geq 1 \]

Step 3: The $A_d^d, B_d^d$ and $\pi^d$ elements of the model are re-estimated according to eqns. (5.25) and (5.26).

\[
a_{ij} = \frac{\sum_{k=1}^{n} \sum_{t=1}^{T_k-1} \alpha_t^k(i) a_{ij} b_{j}(o_{t+1}^k) \beta_{t+1}^k(j)}{\sum_{k=1}^{n} \sum_{t=1}^{T_k-1} \alpha_t^k(i) \beta_{t}^k(i)} \quad 1 \leq i, j \leq 8 \tag{5.32}
\]

\[
b_{ij} = \frac{\sum_{k=1}^{n} \sum_{t=3}^{T_k} \alpha_t^k(i) \beta_{t}^k(i)}{\sum_{k=1}^{n} \sum_{t=1}^{T_k} \alpha_t^k(i) \beta_{t}^k(i)} \quad 1 \leq i, j \leq 8 \tag{5.33}
\]

Although the initial state distribution vector $\pi$ does not need re-estimation in this particular case, its re-estimation form is derived (for future use in the
next chapter) as:

\[ \pi_i = \frac{1}{n} \sum_{j=1}^{n} \frac{1}{P_j} \alpha_j^i(i) \beta_j^i(i) \]  

(5.34)

where \( P_j = \text{Prob}(O_j^i/M_d) = \sum_{i=1}^{8} \alpha_j^i(i) \). The model estimation returns to step 2 iff the ratio \( P^{k+1}/P^k \) of probabilities obtained at \( k+1 \) and \( k \) iterations is larger than a threshold \( \epsilon \) whose value is very close to unity, where \( P^k = \Pi_{i=1}^{n} \text{Prob}(O_i^j/M_j) \). As it was explained in section 5.3, the implementation of both the classification and the model estimation algorithms is likely to yield arithmetic underflow. Thus at each stage of the forward/backward probability estimation process i.e. for each \( t \) in eqn. 5.29 and 5.31 the values of \( \alpha_t(i) \) and \( \beta_t(i) \) are normalized with \( c_t = [\sum_{i=1}^{8} \alpha_t(i)] \) as prescribed in section 5.3.

It is interesting to note that B matrix can be considered as the 'noise' matrix since it does effectively record the permitted amount of variation of the character's shape according to the samples processed. Shape variation of characters are mainly due to quantization noise as well as varying fonts.
5.8 Experimental results and discussion

The proposed stochastic modelling approach for character recognition has been simulated on ICL PERQ1 system. The training data base, used in our experiments for the estimation of the models of the 26 lower case characters was formed from different fonts (of similar topology e.g. fonts with serifs) per character and four samples per font giving a total of 12 samples per character. Figures 5.5, 5.6 and 5.7 provide examples of the three fonts used in the data base. The characters were scanned at 200 pels per inch (ppi) and since they are type written materials, they provide a wealth of noise modulated characters due to low resolution of the scanner. When the R reference models \( (\pi^d, A^d, B^d) \) are estimated, the B matrices are averaged to produce a \( B_e \) matrix that is 'common' to all the R models. In addition, the \( B_e \) matrix forms the initial condition \( B_{in}^d = B_e \) for an initial set \( (\pi_{in}^d, A_{in}^d, B_{in}^d) \), \( 1 \leq d \leq R \) and \( \pi^d, A \) are re-estimated once again using steps 2 and 3 of the previous section.

In order to reduce the number of Markov models tested for maximum \( P_d \) (during the classification process of an input character), a simple deterministic pre-classifier (e.g. Jih, 1983) was also employed that divides the 26 characters into six classes. The first five classes, specified as class 1=(j,y),
Table 5.1: The initial estimate for $A_{in}$

\[
\begin{bmatrix}
0.6000 & 0.1750 & 0.0100 & 0.0100 & 0.0100 & 0.0100 & 0.1750 \\
0.1750 & 0.6000 & 0.1750 & 0.0100 & 0.0100 & 0.0100 & 0.0100 \\
0.0100 & 0.1750 & 0.6000 & 0.1750 & 0.0100 & 0.0100 & 0.0100 \\
0.0100 & 0.0100 & 0.1750 & 0.600 & 0.1750 & 0.0100 & 0.0100 \\
0.0100 & 0.0100 & 0.1750 & 0.600 & 0.1750 & 0.0100 & 0.0100 \\
0.0100 & 0.0100 & 0.0100 & 0.1750 & 0.6000 & 0.1750 & 0.0100 \\
0.1750 & 0.0100 & 0.0100 & 0.0100 & 0.0100 & 0.0100 & 0.1750 & 0.6000
\end{bmatrix}
\]
Scanning is the are converted to single channel.

Figure 5.5: Font 1, used in the training.

A number of document resolution and st screen. Images are

Figure 5.6: Font 2, used in the training.

Recent years being converted to analysis and mani

Figure 5.7: Font 3, used in the training.
class 2 = (f, h, k, l, t), class 3 = (a, e, o), class 4 = (p, q, g), class 5 = (b, d) are determined according to a character having a descender, an ascender, a loop, a loop and a descender, a loop and an ascender respectively; class 6 contains all the remaining characters. When using as input, a set of 250 characters (which are different from those employed in the training) including characters of a font not used in the training and shown in fig.5.8, the system performed reasonably well with a recognition accuracy of 96%.

Some examples of cases where the proposed scheme failed to identify the input characters are depicted in figs. 5.9a, 5.9b, 5.9c and can be attributed to the small number of characters used in the training process. Figure 5.9c shows the interesting case where a 't' is classified as 'f', since the contour sequence direction codes for the two characters are similar, although the starting points when tracing the contour are different. The only remedy to overcome this problem is using adequately large database for training which will allow the introduction of separate matrix B per model and thereby an improved performance. Another modification is to size normalize the character before activating the recognition process which will relieve the recognizer of any size variation. It should also be noted that in all cases of failure the correct choice was always in the first two top choices; it is therefore possi-
ts an image into a numerical representation suitable for input into a
Amongst the most commonly used input devices are microdensitome-
canner, image dissectors, and TV camera digitizers. The first two
st the image to be digitized be in the form of a transparency (e.g. film
tograph. Image dissectors and TV cameras can accept images
manner, but they have the additional advantage of being able to digi-
that have sufficient light intensity.

optical fibres, laser diode
detector arrays are improvin
tics provides high bandwidth

Fig. 5.8: Sample characters used for testing.
Figure 5.9: Examples of failed recognition; a) m mistaken for 'n',
b) 9 mistaken for q, c) t mistaken for f.
Figure 5.10: Examples of character fonts, used in training and testing.

(continued)
Figure 5.10: Examples of character fonts, used in training and testing.

(continued)
A digitizer converts an image into a numerical representation suitable for input into a digital computer. Amongst the most commonly used input devices are microdensitometers, flying spot scanner, image dissectors, and TV camera digitizers. The first two devices require that the image to be digitized be in the form of a transparency (e.g. film negative) or photograph. Image dissectors and TV cameras can accept images recorded in this manner, but they have the additional advantage of being able to digitize natural images that have sufficient light intensity.

A digitizer converts an image into a numerical representation suitable for input into a digital computer. Amongst the most commonly used input devices are microdensitometers, flying spot scanner, image dissectors, and TV camera digitizers. The first two devices require that the image to be digitized be in the form of a transparency (e.g. film negative) or photograph. Image dissectors and TV cameras can accept images recorded in this manner, but they have the additional advantage of being able to digitize natural images that have sufficient light intensity.

A digitizer converts an image into a numerical representation suitable for input into a digital computer. Amongst the most commonly used input devices are microdensitometers, flying spot scanner, image dissectors, and TV camera digitizers. The first two devices require that the image to be digitized be in the form of a transparency (e.g. film negative) or photograph. Image dissectors and TV cameras can accept images recorded in this manner, but they have the additional advantage of being able to digitize natural images that have sufficient light intensity.

Figure 5.10: Examples of character fonts, used in training and testing.

(continued)
A digitizer converts an image into a numerical representation suitable for input into a digital computer. Amongst the most commonly used input devices are microdensitometers, flying spot scanner, image dissectors, and TV camera digitizers. The first two devices require that the image to be digitized be in the form of a transparency (e.g. film negative) or photograph. Image dissectors and TV cameras can accept images recorded in this manner, but they have the additional advantage of being able to digitize natural images that have sufficient light intensity.

A digitizer converts an image into a numerical representation suitable for input into a digital computer. Amongst the most commonly used input devices are microdensitometers, flying spot scanner, image dissectors, and TV camera digitizers. The first two devices require that the image to be digitized be in the form of a transparency (e.g. film negative) or photograph. Image dissectors and TV cameras can accept images recorded in this manner, but they have the additional advantage of being able to digitize natural images that have sufficient light intensity.

A digitizer converts an image into a numerical representation suitable for input into a digital computer. Amongst the most commonly used input devices are microdensitometers, flying spot scanner, image dissectors, and TV camera digitizers. The first two devices require that the image to be digitized be in the form of a transparency (e.g. film negative) or photograph. Image dissectors and TV cameras can accept images recorded in this manner, but they have the additional advantage of being able to digitize natural images that have sufficient light intensity.

A digitizer converts an image into a numerical representation suitable for input into a digital computer. Amongst the most commonly used input devices are microdensitometers, flying spot scanner, image dissectors, and TV camera digitizers. The first two devices require that the image to be digitized be in the form of a transparency (e.g. film negative) or photograph. Image dissectors and TV cameras can accept images recorded in this manner, but they have the additional advantage of being able to digitize natural images that have sufficient light intensity.

A digitizer converts an image into a numerical representation suitable for input into a digital computer. Amongst the most commonly used input devices are microdensitometers, flying spot scanner, image dissectors, and TV camera digitizers. The first two devices require that the image to be digitized be in the form of a transparency (e.g. film negative) or photograph. Image dissectors and TV cameras can accept images recorded in this manner, but they have the additional advantage of being able to digitize natural images that have sufficient light intensity.

Figure 5.10: Examples of character fonts, used in training and testing.

(continued)
ble to incorporate some kind of contextual post processing analysis to correct the misrecognized characters according to the frequency of letters, words and even English language grammar.

In a separate experiment 4 different fonts of size 10, 12, 14, 16, 18 and 24 were chosen (fig. 5.10). A training set of 10 samples from each font and size were fed with a size normalization algorithm due to Gudesen[1980] before being used in the hidden Markov model training process. In the recognition process the incoming patterns were first size normalized and then passed to a pre-classifier (see chapter four ) to classify them to 9 different categories (fig.5.11). A data base of 1000 characters were chosen at random to test the recognizer; consequently a recognition rate of 98.8 % was achieved. As usual characters 'm' and 'n' as well as 'f' and 't' were occasionally misrecognized.
5.9 Rotation invariant property of hidden Markov models

The hidden Markov models in this context (i.e. using chain codes) are rotation invariant by multiples of 90 degrees, that is to say if a character (or a pattern) is rotated by 90 degrees or multiples of 90 degrees, it can be recognized from the original model. To illustrate the point consider the chain codes and their corresponding rotated codes in figure 5.12 as well as character 'a' in upright position and in rotated position. As it can be seen, once the pattern is rotated clockwise and by 90 degrees, the chain code '0' is changed to '2', '1' to '3' and so on. Elements of A and B matrices can therefore be conveniently re-interpreted; for example the previous $a_{00}$ is equal to (the new) $a_{22}$, $a_{77}$ is equal to $a_{11}$ and so on. If a rotation angle of less than 90 degrees is considered, the corresponding model can not be reliably derived from the model in the upright position. The problem can be explained as follows: Every pixel has eight neighbours which are not equidistant from the pixel itself; pixel 'A' in figure 5.15 is a typical example. If the distance between 'A' and 'C' is one, the distance between 'A' and 'B' will be $\sqrt{2}$, that is longer distances are replaced with shorter distances once a 45 degree (i.e. less than 90 degrees) rotation been performed and the resulting Freeman code does not produce a closed contour. Thus, recognition can not be carried out reliably.
Figure 5.11: Classifying characters prior to the recognition process.
An experiment to recognize rotated characters as well as chromosomes was carried out to test the hypothesis. Figures 5.13 and 5.14 depict examples of the training and test patterns. As it will be shown in the next chapter, the H.M.M's rotation invariant property will considerably ease the training process in a more advanced character recognition scheme.

Note on publication

A paper based on the work described in this chapter has been presented at a conference in France; the details are as follows:

Figure 5.12: Character 'a' described by chain codes in upright and rotated position.
Figure 5.13: Examples of training patterns.

Figure 5.14: Examples of test patterns.
Fig. 5.15: Grid points of a digitized image.
Chapter 6

Character recognition

using

Hidden Markov models and Structural analysis.
6.1 Introduction

As it was explained in chapter two, structural description/recognition of characters have received much attention in the hope of designing a robust character recogniser. In this context, once a character is thinned its primitives are extracted according to a pre-set rule and each primitive is recognized using a low level recognition mechanism. The next stage is to derive structural attributes of these primitives with respect to one another. The final decision is made by comparing attributes of the recognised primitives with a set of pre-stored high level descriptions. A typical character recogniser can be described in form of a block diagram as depicted in fig.6.1.

It is essentially a hierarchical multilevel system where the lower levels are the symbolic tokens extracted from the sensory data and the higher levels are inferred concepts of characters as well as model hypothesis. The schema contain interpretation strategies that are responsible for matching the expected structures in characters to the hypothesised models. It is interesting that different researchers propose different primitives to be extracted from
Figure 6.1: Block diagram of a structural recogniser
a character's body, for example Lam, et.al.[1988] approximate a symbol's skeleton by a sequence of line segments which in turn are grouped together to detect convex primitives. These convex primitives are classified as 'EAST', 'NORTH', 'WEST', 'SOUTH' with respect to the centre of the bounding box, as shown in fig.6.2.

Kahan, et.al.[1987] however, extract strokes (straight lines) which are described using their geometric orientation by measuring the angle of incidence of the stroke with the x-axis as well as measuring the stroke's length. Their corresponding structural positions are described with respect to the centre of the box bounding the character.

In our scheme, junction points (or nodes of 3rd. degree) play an important role in structural description of characters. At the low level of processing, that is converting the sensory data to some meaningful symbolic token, hidden Markov modelling (i.e. a stochastic process) is used while the structural description of character shapes takes a deterministic form.

6.2 The scheme's concepts.

The character is first boxed and thinned using a thinning algorithm due to Holt et al. [1986]. The start of the "parsing-point" is detected by scanning the frame from the left most pixel at the top, to the rightmost one and from
Figure 6.2: Classification of polygons (Lam et al., 1988).
top to bottom. The first pixel with only one black neighbour is identified as the starting point which is shown in fig.6.3

At the next step the neighbouring black pixels are examined to see if any of them have:

i) Two black neighbours, i.e. indicating continuation of the primitive trace; in this case the corresponding Freeman code is registered.

ii) Three (or more) neighbours i.e. node(junction) where a primitive ends and another one starts.

Once a junction is detected, the traced primitive is recognised using "hidden Markov modelling" technique. The junction is then issued with a "history tag" of the traced primitive which includes:

a) Coordinates of the starting point.

b) Coordinates of the end point.

c) Coordinates of the traced black pixel.

d) Freeman code of the traced primitive.

e) Label of the third degree node it 'started from'.

f) Structural attribute of the traced primitive with respect to the 3rd.degree nodes that it 'ended up' with; the structural attribute of the primitive with respect to the node it started from is also recorded in the corresponding
Figure 6.3: The starting 'parsing' point is marked by 'x'.

Figure 6.4: The imaginary axes drawn at each 3rd. degree node.
node's data record.

The structural attribute of the traced primitive is derived by drawing an imaginary orthogonal axis at the junction (node) and examining the 'starting' and 'end' points of the traced primitive to see if they are "above", "below", "left", or "right" of the node as illustrated in Fig. 6.4.

Should the process fail to reach a decision, coordinates of all pixels traced are examined one by one to determine where and what proportion of the primitive is located with respect to the imaginary axis. If more than seventy five percent of the pixels fall in one portion of the axis, the corresponding structural attribute is adopted and recorded, otherwise, it is marked as "undefined".

6.3 Recognition process

Once a candidate pattern is parsed and described in the manner outlined above, the reference models stored in the database attempt to identify the described primitives which they associate with each symbol, from the described character.

If a reference primitive and its structural attribute match one of the candidate's primitives and its corresponding structural attribute, a score is added to the score board registering the total score for each reference model
with respect to the candidate character. Should a reference primitive find a match in the candidate's description but no corresponding structural parity, then half the score is added to the score board, otherwise, no score is added to the score board. In the event of finding a complete 'primitive' and 'structural' match of a model, the score registering is abandoned and the corresponding reference character is identified as the recognised symbol. When no complete 'model' match is encountered, the highest score registered on the score board points to the recognised symbol, in which case it is marked for further post processing at the next stage.

To speed up the recognition process a number of steps can be taken as follows:

1)If unique cues such as \( \mathcal{O} \), \( \mathcal{L} \), and \( \mathcal{U} \) are recognised, the candidate character can immediately be identified as 'a', 'l' or 't' respectively.

2)Ascenders, descenders and loops in the character set may be utilised to speed up the recognition process considerably.
6.4 The Algorithm and it's implementation

The algorithm consists of the following processes:

1. The encased, thinned character is scanned from left to right and from top to bottom to identify nodes of degree one and three, and registering their respective coordinates. Figure 6.5 is a typical example where '3' indicates node of degree three and '1' node of degree one. In some cases where a noisy pattern has undergone thinning, a cluster of 3rd. degree nodes are identified. Clearly a choice has to be made to select the right junction as the accepted node; fig.6.6 is an example of such cases.

   To select the right node, the 3rd. degree node with the highest number of neighbouring pixels which are also 3rd. degree nodes are considered to be the correct junction. For instance a cluster of 3rd. degree nodes portrayed in fig.6.6 is re-interpreted as shown in fig.6.7. If no 3rd. degree nodes are detected, the algorithm will extract its outer perimeter's chain code.

2. Set a 'Boolean pad', size of the rectangle encasing the character to true. Start parsing from the 'starting point' already marked in the last stage; continue parsing and setting the corresponding elements in the Boolean pad to false as the primitive's chain code is registered. When a 3rd. degree node is encountered, identify the parsed primitive using hidden Markov models.
Figure 6.5: '3' indicates a nodes of degree 3 and '1' of degree 1...
Figure 6.6: A cluster of 3rd. degree nodes present at a junction...
and register its structural attribute with respect to the node arrived at as well as the 3rd degree node it started from. In the event that either of the nodes is a first degree node, only the third degree node is taken into account. Continue parsing by picking a branch emanating from the node. If all branches have already been visited, look at other nodes in order of their coordinates and pick up the first which has unvisited branches.

This process is continued until all nodes and branches are accounted for. One of the main problems in the parsing process is the occasional presence of 'blind nodes', that is nodes whose emanating branches share the same starting point as shown in fig.6.7. To remedy the situation, 8 neighbours of each node is examined by comparing their corresponding coordinates with those of the primitive's starting points one by one. If there is a match, then the 8 neighbours of the starting points are checked in case they have more than 2 neighbouring black pixels indicating another primitive sharing the same starting point. Should that be detected then the blind node is marked by the algorithm to allow the parser segment and recognise the unvisited primitive. To cite an example, consider the character displayed in fig.6.7 where each detected primitive is defined by its 'starting' and 'end' coordinates. A primitive's 'starting' coordinates are defined as the first black pixel which
Figure 6.7: The connected 3rd. degree node detected.
neighbours a node and is also on the corresponding primitive. In Fig. 6.8, the primitive joining nodes 2 and 3 has (18,17) as its starting coordinates and (11,20) as its 'end' coordinates, thus avoiding any confusion between the 'starting' coordinates of primitives and those of a 3rd. degree node. In Fig. 6.7 however, the node has four branches, each pair of which share the same 'starting' coordinates i.e. a blind node.

6.5 Results and Discussion

The algorithm is charted in figs. 6.9a, 6.9b, 6.9c and was simulated on an ICL PERQ1 system. All documents available had to be scanned at 200 pels per inch (ppi) resolution which is inadequate for fonts of size 11 points and below, but it produced a wealth of noisy characters. In our experiments photocopies of text documents of 10, 11, 12 and 14 point sizes of different fonts were selected; some of them had been produced using a laser printer. High level of noise due to the printing quality, scanning resolution and the character's size resulted in multiple models for each primitive. For example primitive \( \sqrt{} \) had to be represented by three different models due to severe geometric distortion of the primitive in different characters.

Since Markov models are direction sensitive, for each primitive at least two different Markov models had to be computed. The primitive cited above
Figure 6.8: 'X' indicates possible starting points of the primitive in two different characters.
is a case in point; the parser can either start from top to bottom or from bottom to the top, depending where the 'starting' point is located as depicted in Fig.6.8; models for such primitives can be obtained by rotating their corresponding original model by 90 degrees or a multiple of 90 degrees in the manner described in chapter two. In all less than 100 primitives were modelled which also included multiple models. The 'structural description' training process was carried out with minimum hand tuning.

A database of 1000 characters of different fonts and sizes (10, 12, 14,16, 18 points) were prepared to test the system. Samples of the database have already been depicted in chapter 5 (fig.5.10). Two different structural descriptions of a character were put to test:

a) the structural description of the candidate character is derived as the parser segments and describes the character. It does not refer to the 3rd. degree nodes explicitly. For example character 'a' in Fig.6.3 is described by: 1a,2b,3b,4r where 'a' means above,'b' below, 'l' left and 'r' right. Although this structural description was obtained with respect to the nodes encountered, there is no mention of the nodes themselves.

b) Structural description of the candidate character with explicit reference with the 3rd. degree nodes of the skeleton. The above example is described
by:

\[ n1:1a,2b,3b \]
\[ n2:2a,3a,4b \]

where \( n \) stands for node. It clearly expresses the relationship of different primitives with respect to the nodes. The following character set of the database (illustrated in chapter five), was used in our experiments:

\[ \text{ABCDEFGHIJKLMNOPQRSTUVWXYZ} \]
\[ \text{abcdefghijklmnopqrstuvwxyz} \]

Since the variations in undistorted printed characters are not violent the first structural recogniser performed as well as the second one, but as distorted characters were added to the test data, it consistently failed to cope with the unprecedented structural variations. It was therefore decided that the second scheme is more superior and was tested with the entire database. To speed up the recognition process, the 'pre-processors' described in chapter four, were also used to group the incoming characters according to their loops, descenders and ascenders. Out of 1000 characters only 10 severely distorted characters were not recognised. A limited number (50) of characters of size 24 were also tested; the recogniser could not recognise them all, in fact, only 40 of them were correctly recognised and the rest rejected. The
recogniser’s failure is clearly due to the size variation.

From the theoretical point of view, hidden Markov models can not cope with unlimited size variation for the simple reason that all probabilities collected during the training process do not reflect unlimited change in shape. Consequently, a size normalisation algorithm due to Gudesen was added to the scheme to reduce the character size to that of a 12 point size character, the resulting size normalised characters were then presented to the recogniser which reconised them correctly. Another interesting problem which surfaced in the experiments was the presence of serifs (usually small extension of limbs); since they are treated as a primitive in the recognition process and their absence has no bearing on the recogniser’s performance it is desirable to remove them somehow. One possible way to overcome this problem is to impose a threshold on how much of a segmented limb must be removed to rid the system of any possible serifs; but varying sizes of character limbs fail this remedy. Perhaps the only robust way of dealing with serifs is to ignore them when building the actual reference models; in this way the recognition process only picks up the elements present in the reference model leaving out all the segmented serifs. Unfortunately there is a price to be paid and that is imposing more computational load on the parser.
Figure 6.9.a: Flow chart of the structural character recognition scheme.
Fig. 6.9 b: Flow chart of the structural character recognition scheme (continued)
From fig. 6.9b

score := 0
Get the candidate character

Get the structural model for a reference character

Detect the reference primitive and its structural attribute in the candidate model.

Complete primitive/structural match?

Yes
Score = Score + weight

No
Score = Score + 0

Are all elements of the model accounted for?

No

Yes

Is there an exact and complete match?

No

Any more reference models?

Yes

Choose the highest score.

No

The Character is recognised.

To fig. 6.9a

Fig. 6.9c: Flow chart of the recognition process.

To fig. 6.9a
From fig. 6.9.a

Select the H.M models of patterns with no 3rd. degree nodes

Carry out the H.M.M Recognition

Output the recognised character

go to Fig. 6.9.a

Figure 6.9.d: Flow chart of the structural Character recognition system
Chapter 7

Suggested further work.
Suggested further work

7.1 Introduction

As the OCR field moves forward, more and more research efforts have been directed towards the recognition of less constrained and totally unconstrained characters, either handwritten (e.g. symbols on cheques, maps, engineering drawings, etc.) or machine printed. Supporting this trend are active research and development projects in search of better and more cost effective scanning, pre-processing techniques, feature extraction and classification methods. It now appears that, the most promising approach is to divide recognition processes into three broad stages:

1) The pre-recognition stage which is made of pre-processing and document analysis processes reduce the number of possible choices and speed up recognition.

2) The discriminant stage which focuses on the detection of detailed features to discriminate confusing groups (or pairs) of characters.

3) The final stage which makes use of linguistic, contextual or statistical information, leading to the correct identity of characters, symbols, words and so forth.
7.2 Extension to the present work

As an extension to the present work, further research could be carried out to formulate second order hidden Markov models to represent a character's shape; naturally, deriving such models will be more complicated. Particular care should be taken when formulating the 'forward' and 'backward' probabilities.

Another possibility is to use the H.M.M technique in continuous mode, that is to derive representative Probability Density Function (p.d.f) distributions for the features (states) considered, rather than using discrete Markov models.

Finally, since there are commercial OCR machines available in the market, it is useful to design a methodology to assess their performance. Testing OCR systems is usually done by feeding a large data base of scanned documents, which is clearly time consuming, especially if a cross comparison is to be made. Developing statistical methods to assess OCR machines can speed up the assessment work and provide more insight into an OCR system's behaviour. It will simply allow you to assess an OCR product much smaller test data set. For example, sensitivity to noise, font, size, angle of inclination are four prominent factors to be taken into account. A methodology may
therefore be proposed to determine the sensitivity of character recognizers to
various user defined variables such as noise, font size, etc. Generally, there is
a substantial difference between types and structures of character recognizer
tests in the laboratory and those carried out in the field trial. In laboratory a
pre-recorded data base is used to train and test a character recognizer. The
data base, however, may not be representative of the conditions or the words
encountered in the expected environment, and there has been no unified
approach to produce a database which characterize, in some fashion, the
variability between and inside different input documents. Such limitations
are severe and can be summarised as follows:

- The database may not have been recorded in conditions similar to those
  in which OCR has to work.

- The database is not structured in any way.

- The database is unlikely to have been constructed to reflect the vari-
  ability in font, size, noise and angle of inclination.

Laboratory experiments can always be controlled so that, the working
environment is always the same. It is generally assumed that to test an
OCR system well, a very large database is necessary. This is indeed the case
for a randomly constructed database, and from statistics, one can produce a
table listing the number of testing characters necessary to have a particular confidence in a recognizer's performance. These numbers are very large, and commercial OCR products claiming better than 99% correct recognition, a figure which indicates a minimum of 40,000 test characters have passed through the OCR to achieve 95% confidence level. Since the database has no structure, detailed analysis can not be undertaken. For example, it would be impossible to determine whether the OCR performed more reliably with frequent change of font and size within input documents or whether the OCR system can cope with noisy and distorted input. A mathematical methodology based upon the analysis of variance approach to experimental design can be devised to produce a statistical account of the factors considered. Once a database is constructed so that, some measure of font (or size, noise, etc) variability is known, then, the methodology will be able to predict the sensitivity of the character recognizer to font, size, noise variability.
Bibliography


Amin [1984]: "Recognition of handwritten Arabic words and sentences", Proc. 7th. int. joint Conf. on Pattern Recognition, pp. 1055-1057.


Bledsoe, W.W. and Bisson, C.L. [1962]: 'Improved memory matrices for the


Casey, R.G., et.al. [1982]: "Recursive segmentation and classification of composite character patterns", IEEE Conf. on Computer Vision.


Deutsch, E.S. (1969): 'On some preprocessing techniques for character recognition', Symposium on computer processing in communication, Polytechnic Institute of Brooklyn, April 8-10.


Fujimoto, Y. et.al. [1976]: 'Recognition of handprinted characters by nonlinear elastic matching.', proc. 3rd. int. joint conf. on pattern recognition.


Gonzalez, R. et.al.[1987]: "Introduction to digital image processing", Addison Wesley.


Higashino, J., et.al. [1986]: 'A knowledge-based segmentation method for document understanding.'.


Holt, M.J.J., et.al. [1986]: "Coding scheme for scanned images of handwriting and graphics", IEEE Int. Conf. on Communication, Toronto.


Ikeda, et.al. [1978]: Proc. 4th. international joint conference on Pattern Recognition, pp. 813-815.


Kubota, K. et.al. [1984]; "Document understanding system", Proc. 7th. int. conf. on pattern recognition.


Kwon, S.K., et.al. [1976]: “Recognition experiments with handprinted numerals”, in Proc. joint workshop on pattern recognition and artificial intelligence, pp. 74-83, June.


Lee, K.Y., et.al. [1988]: "Recursive segmentation of binary character patterns for improved data compression in digital facsimile".


Mantas, J. [1985]: 'A survey of character recognition methodologies.', MELECON'85, Vol.II.


Morill, T. et.al. [1963]: 'Cyclops-1: a second generation recognition system', in proc. Fall joint computer conf., pp. 27-33, Nov.


Nagy, G et.al.[1984]: 'Hierarchical representation of optically scanned documents.', Conf. on pattern recognition, Montreal, July 30 - 2 August.


Nakano, Y, et.al. [1986]: 'A document understanding system incorporating with character recognition.', ICCASP conf.


Postl, W [1986]: "Detection of linear oblique structures and skew scan in digitized documents", IEEE conf. on computer vision.


Rastogi, A., et.al. [1986]: "Recognizing textual blocks in document images using the Hough transform", internal report, Jan 1986, Dept. of computer science, University of Buffalo(SUNY), U.S.A.


Rosenfeld, A. et.al. [1966]: 'Sequential operations in digital picture processing', Journal of the Association for Computing Machinery, Vol. 13, No. 4.


Siy, P., et.al. [1974]: "Fuzzy logic for handwritten numeral character recognition", IEEE Trans. on system, man and cybernetics, November.


Srihari, S.N. [1986]: et.al. 'Document image analysis.', University of Buffalo (SUNY), Computer Science Dept; internal report 86-01, Jan.


Watt, A.H. et.al. [1971]: 'Recognition of hand printed numerals reduced to graph representable form', proc. 2nd. int. joint conf. on artificial intelligence, pp. 322-332, Sept.

Weaver, J.A. [1972]: 'Reading Machines', Mills and Boon.


Wong, K.Y., et.al. [1982]: "Document analysis system", IBM journal of research and development, No. 6, November.


Zen, H., et.al. [1985]: "Extraction of the fair document from mixed mode manuscript", IEEE conf. on pattern recognition.