# ON-LINE HANDWRITTEN DOCUMENT UNDERSTANDING

By

Anoop M. Namboodiri

AN ABSTRACT OF A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

2004

Professor Anil K. Jain

# Abstract

# On-line Handwritten Document Understanding

By

Anoop M. Namboodiri

This thesis develops a mathematical basis for understanding the structure of on-line handwritten documents and explores some of the related problems in detail. With the increase in popularity of portable computing devices, such as PDAs and handheld computers, non-keyboard based methods for data entry are receiving more attention in the research communities and commercial sector. The most promising options are pen-based and speech-based inputs. Pen-based input devices generate handwritten documents which have on-line or dynamic (temporal) information encoded in them. Digitizing devices like SmartBoards and computing platforms, which use pen-based input such as the IBM Thinkpad TransNote and Tablet PCs, create on-line documents. As these devices become available and affordable, large volumes of digital handwritten data will be generated and the problem of archiving and retrieving such data becomes an important concern. This thesis addresses the problem of on-line document understanding, which is an essential component of an effective retrieval algorithm.

This thesis describes a mathematical model for representation of on-line handwritten data based on the mechanics of the writing process. The representation is used for extracting various properties of the handwritten data, which forms the basis for understanding the higher level structure of a document. A principled approach to the problem of document segmentation is presented using a parsing technique based on stochastic context free grammar (SCFG). The parser generates a segmentation of the handwritten page, which is optimal with respect to a proposed compactness criterion. Additional properties of a region are identified based on the nature and layout of handwritten strokes in the region. An algorithm to identify ruled and unruled tables is presented. The thesis also proposes a set of features for identification of scripts within the text regions and develops an algorithm for script classification.

The thesis presents a matching algorithm, which computes a similarity measure between the query and handwriting samples in a database. The query and database samples could be either words or sketches, which are automatically identified by the document segmentation algorithm. We also address the related problem of document security, where on-line handwritten signatures are used to watermark on-line and off-line documents. The use of a biometric trait, such as signature, for watermarking allows the recipient of a document to verify the identity of the author in addition to document integrity.

# ON-LINE HANDWRITTEN DOCUMENT UNDERSTANDING

By

Anoop M. Namboodiri

A DISSERTATION

Submitted to
Michigan State University
in partial fulfillment of the requirements
for the degree of

DOCTOR OF PHILOSOPHY

Department of Computer Science

2004

## Abstract

## On-line Handwritten Document Understanding

By

Anoop M. Namboodiri

This thesis develops a mathematical basis for understanding the structure of on-line handwritten documents and explores some of the related problems in detail. With the increase in popularity of portable computing devices, such as PDAs and handheld computers, non-keyboard based methods for data entry are receiving more attention in the research communities and commercial sector. The most promising options are pen-based and speech-based inputs. Pen-based input devices generate handwritten documents which have on-line or dynamic (temporal) information encoded in them. Digitizing devices like SmartBoards and computing platforms, which use pen-based input such as the IBM Thinkpad TransNote and Tablet PCs, create on-line documents. As these devices become available and affordable, large volumes of digital handwritten data will be generated and the problem of archiving and retrieving such data becomes an important concern. This thesis addresses the problem of on-line document understanding, which is an essential component of an effective retrieval algorithm.

This thesis describes a mathematical model for representation of on-line handwritten data based on the mechanics of the writing process. The representation is used for extracting various properties of the handwritten data, which forms the basis for understanding the higher level structure of a document. A principled approach to the problem of document segmentation is presented using a parsing technique based on stochastic context free grammar (SCFG). The parser generates a segmentation of the handwritten page, which is optimal with respect to a proposed compactness criterion. Additional properties of a region are identified based on the nature and layout of handwritten strokes in the region. An algorithm to identify ruled and unruled tables is presented. The thesis also proposes a set of features for identification of scripts within the text regions and develops an algorithm for script classification.

The thesis presents a matching algorithm, which computes a similarity measure between the query and handwriting samples in a database. The query and database samples could be either words or sketches, which are automatically identified by the document segmentation algorithm. We also address the related problem of document security, where on-line handwritten signatures are used to watermark on-line and off-line documents. The use of a biometric trait, such as signature, for watermarking allows the recipient of a document to verify the identity of the author in addition to document integrity.

To My Family

# Acknowledgments

I would like to express my sincere thanks to Dr. Anil Jain for his constant support, inspiration and guidance in both academic and personal life. I am extremely grateful to him for being an excellent advisor and a wonderful teacher. His attention to detail and completeness of every aspect of the work, including research, presentation and technical writing has helped me tremendously to improve my skills. I would also like to thank Dr. George Stockman and Dr. Sarat Dass for their help through discussions and suggestions. I am indebted to Dr. John Handley for his extremely thoughtful suggestions and guidance for my thesis and for making the months I spent at Xerox Wilson Research Center, both enjoyable and productive. I would also like to express my gratitude to Dr. Jayashree Subrahmonia of IBM T.J. Watson research center for her help in defining and developing the thesis topic and for providing me with both technical and financial help, which made this work possible.

I would like to thank all my colleagues in the PRIP lab and the research community, especially Dr. Paul Albee, David Cherba, Dr. Scott Connell, Dr. Michael Farmer, Friederike Griess, Dr. C.V. Jawahar, Martin Law, Xiaoguang Lu, Erin McGarrity, Silviu Minut, Dr. Salil Prabhakar, Dr. Arun Ross, and Shailesh Saini for the useful discussions and making the lab a wonderful place to work in. I am really grateful to Linda Moore, Starr Portice, Cathy Davison, and Debbie Kruch for their administrative assistance and support. Finally, I would like to thank my parents, sister and my wonderful wife Preetha for their love and support.

## Table of Contents

# List of Tables

# LIST OF FIGURES

# Chapter 1

# Introduction

The number of documents that are available in the digital mode has been increasing exponentially since the inception of the Internet. According to the Internet Software Consortium, as of January 2004, there are more than 230 million registered web sites and about 60 million were added last year [50] (see Figure 1.1). The number of indexed web pages exceeds 4.3 billion as of July 2004. This exponential increase in the number of documents puts a heavy demand on reliable search mechanisms without which the increased availability of documents is an obstacle in locating the desired information. To add to the problem, the proportion of non-text (image, audio and video) documents,which remain hidden from the conventional text-based search mechanisms, on the web is also increasing. These documents are primarily generated from scanners and other imaging devices such as digital cameras.

Even with the popularity of the Internet, digital documents have not completely replaced paper documents within offices (*The Myth of Paperless Office* [133]). This is primarily due to the many desirable qualities of paper such as ease of navigation and annotation and its non-dependence on any support infrastructure, such as computers. This leads to the coexistence and conversion between documents in the paper and digital forms. In addition to office documents, vast amounts of books and historic documents are now be-

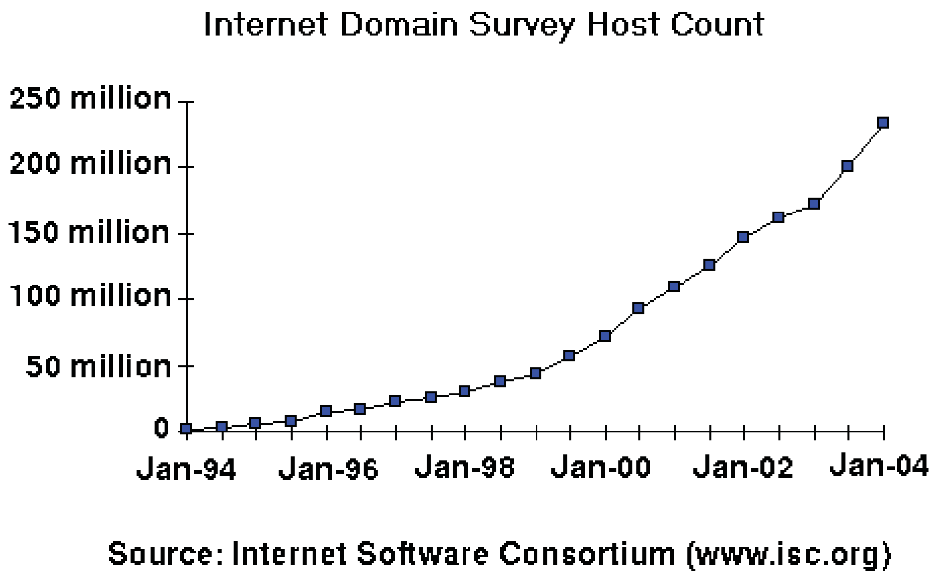


Figure 1.1: Increase in the number of hosts on the Internet.

ing converted from the paper to digital form [149]. These include printed books [151], handwritten manuscripts [98], and other sources [13].

## 1.1 Online Documents

A class of documents that is gaining popularity in recent times is on-line handwritten documents. Handwritten data, which is captured (digitized) at the time of writing, encodes the dynamic information in the strokes [1] and is referred to as on-line handwriting. The development of devices that can capture on-line handwriting spans a few decades (see Figure 1.1). However, in recent times, pen-based input devices (e.g., PDAs [79], electronic whiteboards [139] and devices like IBM Thinkpad Transnote ® [48] and Tablet PCs [92]) have become very popular, resulting in the creation of large amounts of digitized handwritten data [1].

In addition to being a medium for data input, the pen also serves as an excellent interface

[1]A stroke is defined as the locus of the tip of the pen from pen-down to the next pen-up position.

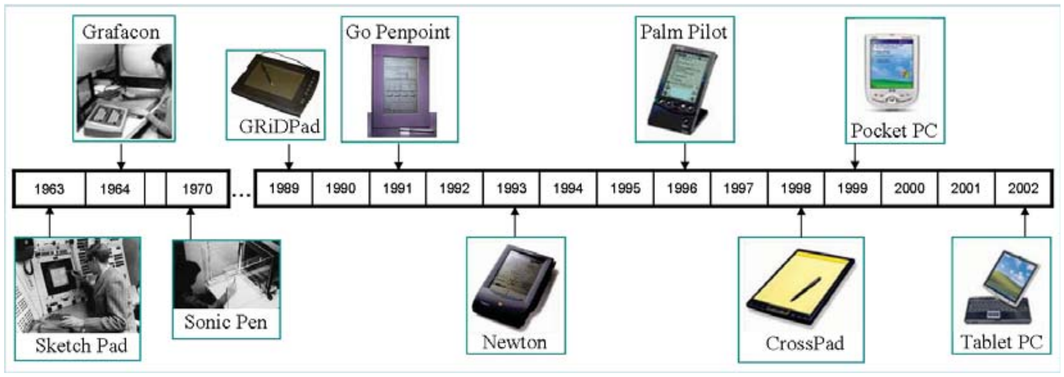


Figure 1.2: Evolution of the pen: Salient events in the development of pen-based input [62, 32].

for human-computer interaction (HCI). Compared to other non-conventional modalities of input such as speech and gaze, pen-based input has a number of advantages. Table 1.1 provides a list of advantages and disadvantages of the popular input modalities.

Figure 1.3 shows some of the popular devices that can capture on-line handwritten data. The total sales of the Tablet PCs, for the year 2003 was around $1.0\%$ of the notebook market [87]. However, there are several critical issues in the hardware technology as well as interaction algorithms that need to be improved before pen-based input becomes widely accepted. As the number of on-line handwritten documents that are stored digitally increases, one of the main problems that needs to be addressed is the retrieval of a specific document of interest from a database. Although retrieval systems for this class of documents can be similar to (off-line) document image retrieval systems, such as those described in [26] and [66], the temporal information encoded in on-line documents can help us to develop better matching algorithms.

Figure 1.4 shows examples of printed, off-line handwritten and on-line handwritten documents. Note that the internal representation of the on-line document (shown in Figure 1.4(d)) is not a two-dimensional image, but a (temporal) sequence of $(x, y)$ coordinates. A detailed discussion of on-line documents, their representation and processing can be found in Chapter 3.

There are some important aspects of on-line documents that enable us to process them in

Table 1.1: Comparison of input modalities: Advantages and disadvantages of popular modalities for human computer interaction.

| Modality | Advantages | Disadvantages |
|---|---|---|
| Speech | Hands- and eyes-free interaction | Difficult to use in noisy environments |
| | Useful for physically handicapped | Difficult to be used for HCI |
| | Can be used for annotation | Linear input |
| Gaze | Least effort from user | Limited use for data input |
| | Can be used by severely handicapped | Least powerful of the three modalities |
| | Language independent interaction | |
| Pen | Usable in noisy environments | Needs visual attention of the user |
| | Doubles as a pointing device | Slower than speech for text input |
| | Most accurate of the three for HCI | |
| | Language independent interaction | |
| | Least processor requirement | |
| | Can be used for sketching | |
| | Can be used for annotation | |

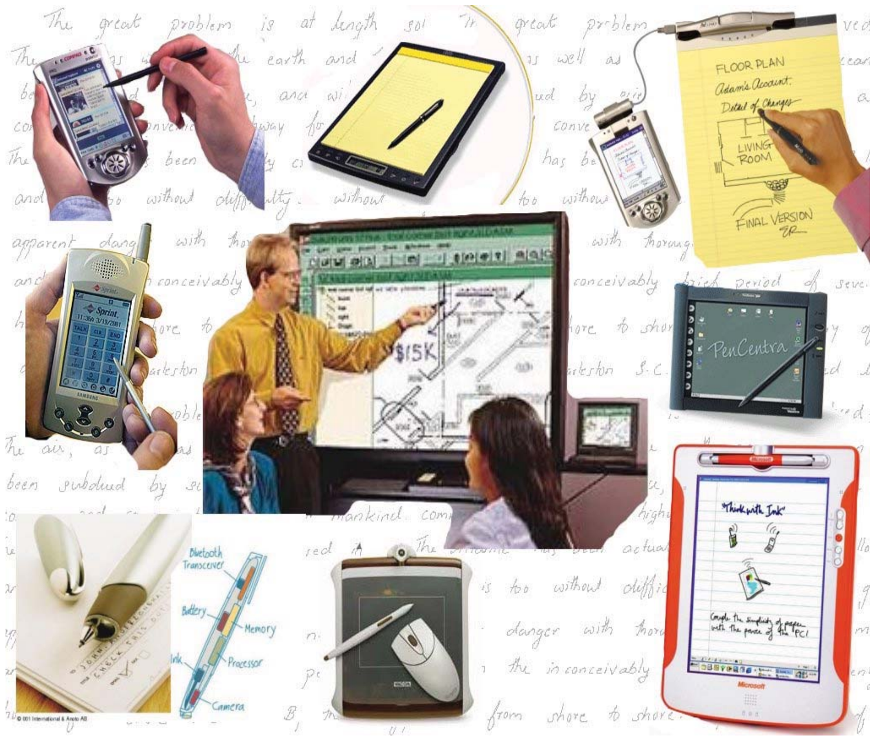
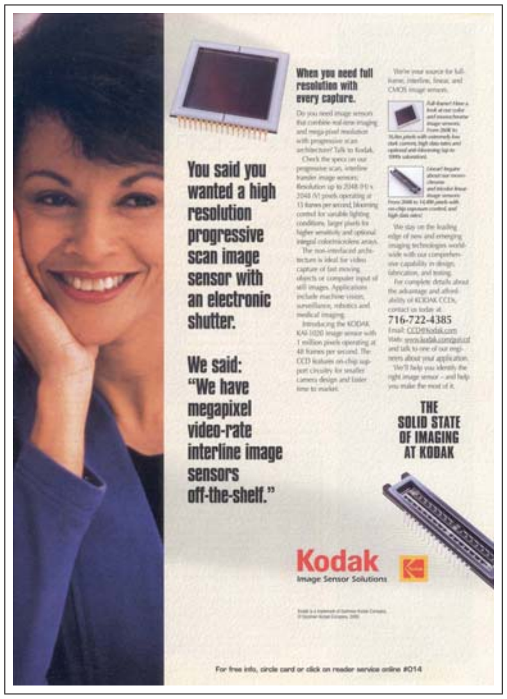


Figure 1.3: Devices that accept on-line handwritten data: From the top left, Pocket PC, CrossPad, Ink Link, Cell Phone, Smart Board, Tablet with display, Anoto pen, Wacom Tablet, Tablet PC.



When you need full resolution with every capture.

Do you need image sensors that combine real-time imaging and mega-pixel resolution with progressive scan architecture? Talk to Kodak.

Check the specs on our progressive scan, interline transfer image sensors: Resolution up to 2048 (H) x 2048 (V) pixels operating at 15 frames per second, blooming control for variable lighting conditions, larger pixels for higher sensitivity and optional integral color/microlens arrays.

The non-interlaced architecture is ideal for video capture of fast moving objects or computer input of still images. Applications include machine vision, surveillance, robotics and medical imaging.

Introducing the KODAK KAI-1020 image sensor with 1 million pixels operating at 48 frames per second. The CCD features on-chip support circuitry for smaller camera design and faster time to market.

We're your source for full-frame, interline, linear, and CMOS image sensors.

[illegible]

[illegible]

We stay on the leading edge of new and emerging imaging technologies worldwide with our comprehensive capability in design, fabrication, and testing.

For complete details about the advantage and affordability of KODAK CCDs, contact us today at:

716-722-4385

Email: CCD@kodak.com

Visit: www.kodak.com/go/ccd and talk to one of our engineers about your application.

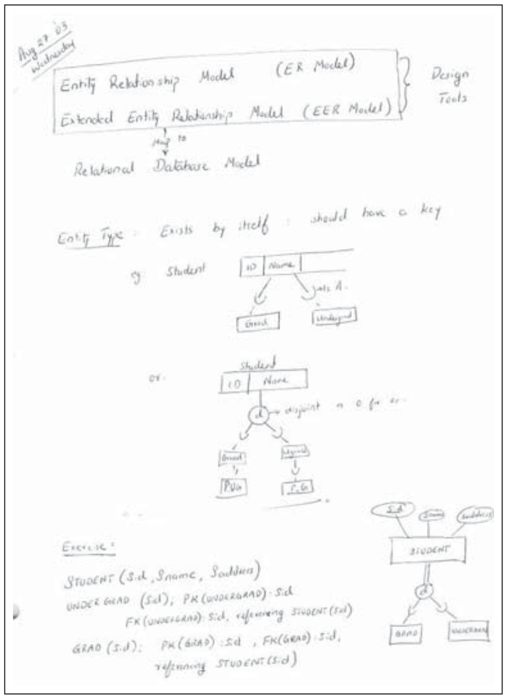
We'll help you identify the right image sensor – and help you make the most of it.

[illegible]

For free info, circle card or click on reader service online #014

You said you wanted a high resolution progressive scan image sensor with an electronic shutter.
We said: "We have megapixel video-rate interline image sensors off-the-shelf."
THE SOLID STATE OF IMAGING AT KODAK
Kodak
Image Sensor Solutions

(a)



Aug 27 '03 Wednesday

Entity Relationship Model (ER Model)

Extended Entity Relationship Model (EER Model)

Design Tools

Map to

Relational Database Model

Entity Type: Exists by itself : should have a key

eg Student

or

disjoint

Exercise:

STUDENT (S.id, Sname, Saddress)

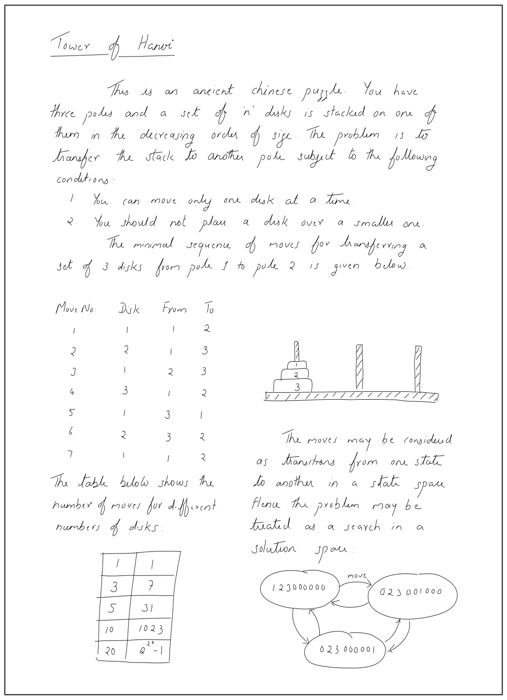
UNDERGRAD (S.id); PK (UNDERGRAD): S.id

FK (UNDERGRAD): S.id, referencing STUDENT (S.id)

GRAD (S.id): PK (GRAD): S.id, FK (GRAD): S.id,

referencing STUDENT (S.id)

(b)



Tower of Hanoi

This is an ancient chinese puzzle. You have three poles and a set of 'n' disks is stacked on one of them in the decreasing order of size. The problem is to transfer the stack to another pole subject to the following conditions:

1. You can move only one disk at a time.
2. You should not place a disk over a smaller one.

The minimal sequence of moves for transferring a set of 3 disks from pole 1 to pole 2 is given below.

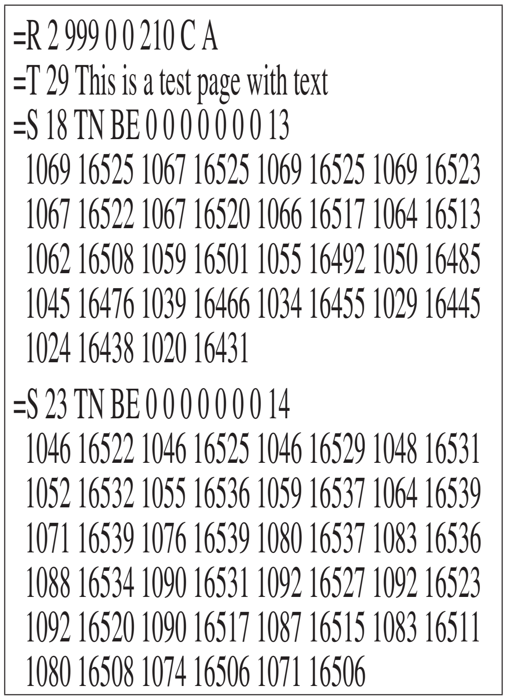
| Move No. | Disk | From | To |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 2 | 2 | 1 | 3 |
| 3 | 1 | 2 | 3 |
| 4 | 3 | 1 | 2 |
| 5 | 1 | 3 | 1 |
| 6 | 2 | 3 | 2 |
| 7 | 1 | 1 | 2 |

The table below shows the number of moves for different numbers of disks:

| | |
|---|---|
| 1 | 1 |
| 3 | 7 |
| 5 | 31 |
| 10 | 1023 |
| 20 | $2^{20}-1$ |

The moves may be considered as transitions from one state to another in a state space. Hence the problem may be treated as a search in a solution space:

(c)

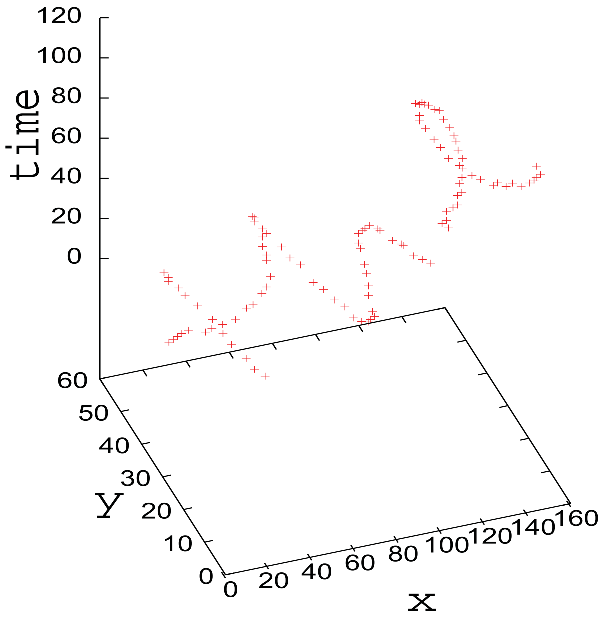


```
=R 2 999 0 0 210 C A
=T 29 This is a test page with text
=S 18 TN BE 0 0 0 0 0 0 0 13
 1069 16525 1067 16525 1069 16525 1069 16523
 1067 16522 1067 16520 1066 16517 1064 16513
 1062 16508 1059 16501 1055 16492 1050 16485
 1045 16476 1039 16466 1034 16455 1029 16445
 1024 16438 1020 16431
=S 23 TN BE 0 0 0 0 0 0 0 14
 1046 16522 1046 16525 1046 16529 1048 16531
 1052 16532 1055 16536 1059 16537 1064 16539
 1071 16539 1076 16539 1080 16537 1083 16536
 1088 16534 1090 16531 1092 16527 1092 16523
 1092 16520 1090 16517 1087 16515 1083 16511
 1080 16508 1074 16506 1071 16506
```

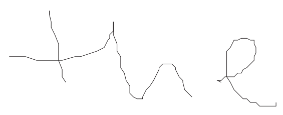
(d)

Figure 1.4: Examples of digital documents: (a) printed, (b) handwritten, (c) on-line, and (d) the internal representation of the on-line document, which is a sequence of $(x, y)$ co-ordinates.

a fundamentally different way than off-line documents. The most important characteristic of on-line documents is that they capture the temporal sequence of strokes while writing the document (see Figure 1.5). This allows us to analyze the individual strokes and use the additional temporal information for document understanding as well as text recognition.



(a)



(b)

Figure 1.5: Online handwriting: (a) strokes of the word ‘the’ that encode the writing sequence. The vertical axis shows the temporal sequence in which the strokes were drawn. (b) the same word without the temporal information.

In the case of on-line documents, segmentation of foreground from the background is a relatively simple task as the captured data, i.e. the $(x, y)$ coordinates of the locus of the stylus, defines the characters and any other point on the page belongs to the background. We use stroke properties as well as the spatial and temporal information of a collection of strokes to identify the properties of regions in an on-line document. Unfortunately, the temporal information also introduces additional variability in the handwritten characters, which creates large intra-class variations of strokes in each of the classes. Figure 1.6 shows two samples of the character ‘r’, with and without the temporal information. Even though the spatial representations in Figures 1.6(a) and 1.6(b) look similar, the temporal differences introduce large intra-class variability in the on-line data as shown in Figures 1.6(c) and 1.6(d).

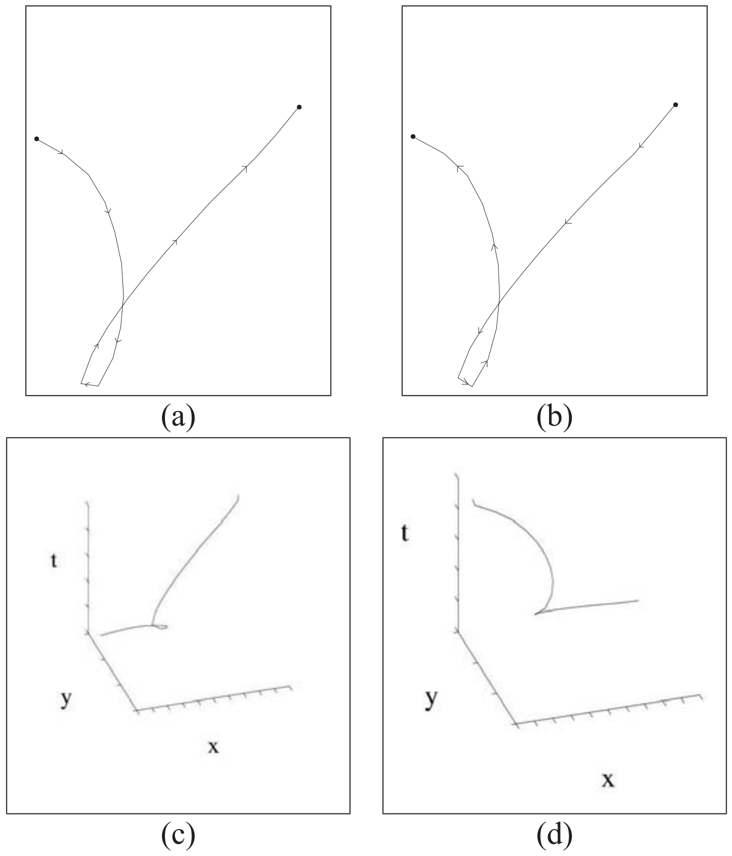


Figure 1.6: On-line handwriting variability. The character ‘r’ written in two different styles. The off-line representations in (a) and (b) look similar, but the temporal variations in (c) and (d) make them look very different. The writing direction is indicated using arrows in Figures (a) and (b). The vertical axes in (c) and (d) represent time.

## 1.2 On-line Data Capture

The technology used for the capture of on-line data is an important factor in determining the quality of the data that is available for processing. The technologies vary in the sampling resolution (both spatial and temporal), user feedback, active components of the system, durability, accuracy, cost, etc. Depending on the technology, the capturing devices can provide a variety of information such as the $(x, y)$ co-ordinates of the pen tip, a boolean value indicating contact with $x - y$ plane, pressure, angles with the $x - y$ plane $(\phi_x, \phi_y)$ or tilt, distance from $x - y$ plane (height), etc. This section describes some of the popular technologies for capture of on-line data.

- *Touch-sensitive Devices*: The most common types of touch sensitive devices are the 'analog resistive' and 'capacitive' devices. The analog resistive devices use a double layer overlay on the touch sensitive surface, where the lower layer is a resistive material and a voltage is applied to the top later. Pressure applied on the top layer with any pointed object (a stylus) causes contact between the layers. The point of contact is determined by the current measured at the four corners of the display. The capacitive devices use a single overlay with an electric mesh, and the touch of the finger or any object, which changes the capacitance, triggers computation of the position of contact based on the current flowing from the edges. Most of the PDAs and touch screens use one of these two technologies, since they are simple and inexpensive. Figure 1.7 shows schematic diagrams of tablets with resistive and capacitive touch sensors.

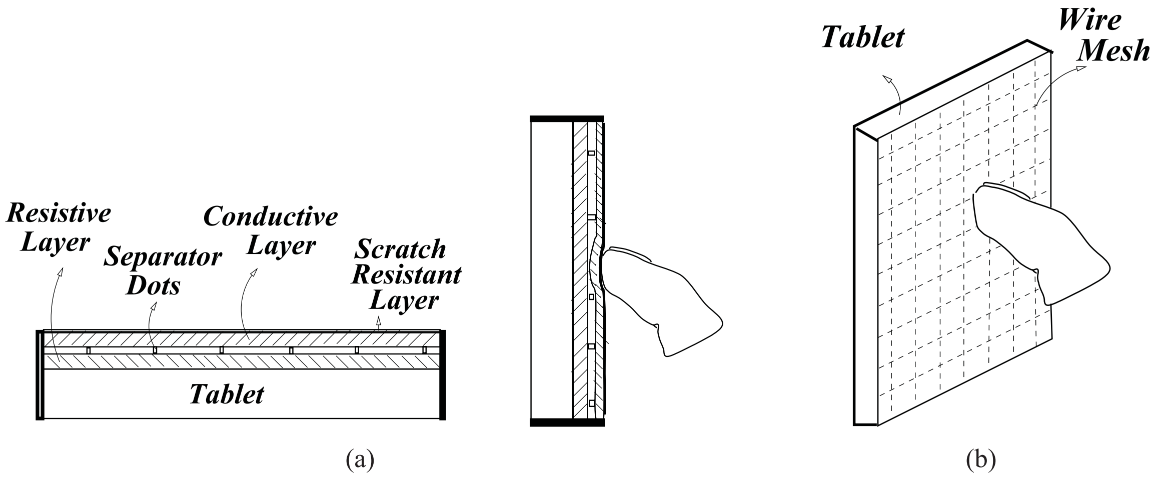


Figure 1.7: Touch Sensitive Devices: (a) resistive touch sensor and (b) capacitive touch sensor.

- *Magnetic Tracking*: Magnetic tracking devices are most commonly used in digitizing tablets such as Wacom® [155], CrossPad® [47], and Tablet PCs® [92]. They use an array of coils in the tablet, which creates a magnetic field pattern above the tablet. This couples a magnetic field into a passive circuit in the pen (a coil and a capacitor). The coils in the pad pick up the magnetic field from the pen, and their relative strength indicates the pen position. The CrossPad uses an active circuit in the pen, which generates a magnetic field, which is detected by the coils in the pad.

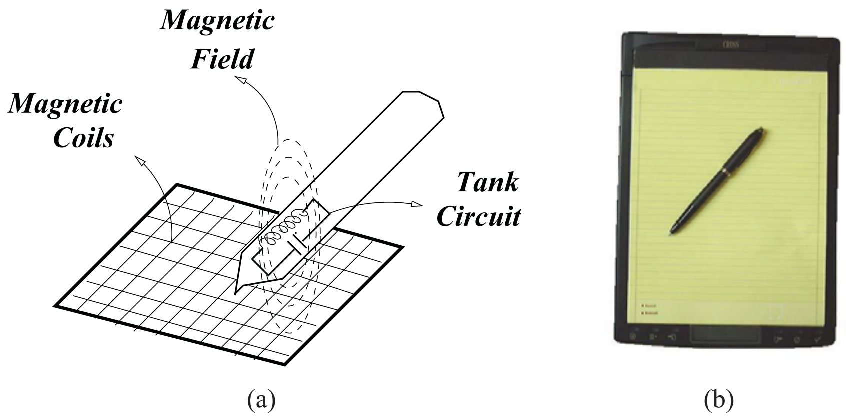
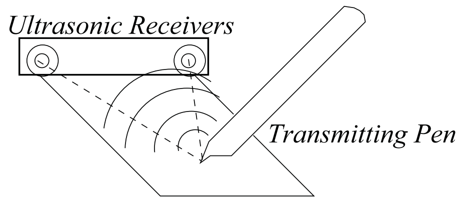


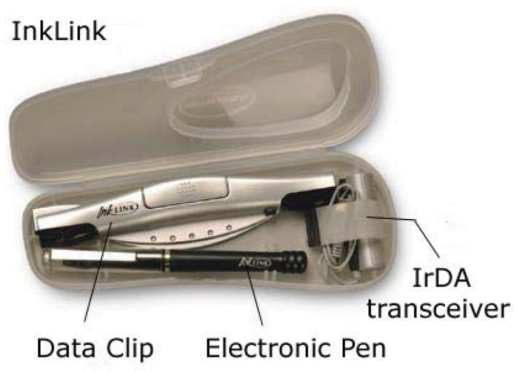
Figure 1.8: Magnetic Tracking: (a) schematic diagram of a magnetic position sensor and (b) the CrossPad® digitizing tablet.

- Ultrasonic Tracking: In this technology, a special pen transmits pulses of ultrasonic sound that are captured by two (or more) receivers located on the writing plane. The difference between the time of arrival of the sound pulses at individual sensors is used to compute the pen position. The technology does not limit the writing surface as it is quite reliable and accurate. The Casio E-Pen® [28] and the Mimio® [93] boards use ultrasonic tracking. The Seiko InkLink® [49] (see Figure 1.9 (b)) uses two receivers and an infrared transducer to compute the position of the pen.

- *Optical Tracking:* Devices such as Anoto® pen [3] and CompuPen® [109] use a vision-based technology to detect the position of the pen. The interface is of the pen and paper type, where the pen is fitted with a camera (see Figure 1.10(a)). The cam-



(a)



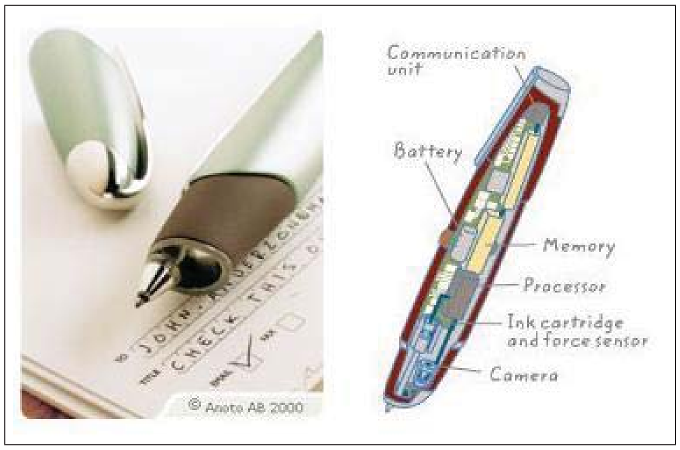
(b)

Figure 1.9: Ultrasonic Tracking: (a) schematic diagram of an ultrasonic position sensor and (b) the InkLink® sensor.

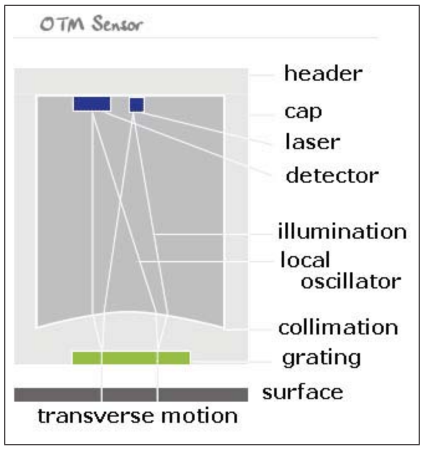
era observes the writing surface relative to the pen tip while writing to determine its motion. The Anoto pen uses a special paper for writing, on which a specific pattern of dots is printed. The pen uses the dot pattern, which it sees through the camera to determine the position of the pen tip on the paper. The pen collects approximately 50 sample points per second, which are stored in the pen itself. The data can be transferred to a computer using a connector or BlueTooth® wireless link. The technology has the advantage of using the natural pen and paper interface, and being compact. However, it requires a special printed paper for reliable determination of absolute position. The accelerometer pens, which use motion sensors inside the pen, try to avoid the requirement for special paper. However, they are not accurate enough for capturing handwritten data [153]. A similar technology, used by the VPen® uses interference between laser reflected from an optical grating inside the pen and the writing surface to determine the direction of motion [101] (see Figure 1.10 (b)).

## 1.3 A Document Understanding System

Scanning documents is the first step in generating document images. The document images by themselves do not lend readily to archival and search of their contents in a database. In



(a)



(b)

Figure 1.10: Optical Tracking: (a) the Anoto® pen and its schematic diagram and (b) the OTM® sensor.

some cases, the documents are tagged with meta-data (e.g., author, topic, language, date, content description, etc.) that can be used to index or query the document images. Certain documents such as books and manuscripts that contain text may also be converted to a searchable representation using Optical Character Recognition (OCR). A system, which automatically analyzes a document image and generates information for tagging or converting the document image, is called a Document Understanding System [2]. The process of document understanding includes several tasks such as (i) identifying the genre of the document, (ii) separation of background and foreground, (iii) segmentation of document images and classification of individual segments (regions), (iv) extraction of relevant regions from document images, especially for form processing, (v) identifying the script, language and font of text regions, (vi) understanding the layout and reading order of text, (vii) recognition of characters in text regions, (viii) processing tabular data, (ix) interpreting drawings and sketches, (x) generating a representation, which captures the document structure, etc. Figure 1.11 shows the schematic diagram of a generic document understanding system.

The problem of document understanding can be extended to handwritten document images and on-line documents as well. In this case, the document is less structured than the printed documents. In the field of on-line documents, most of the research has been geared

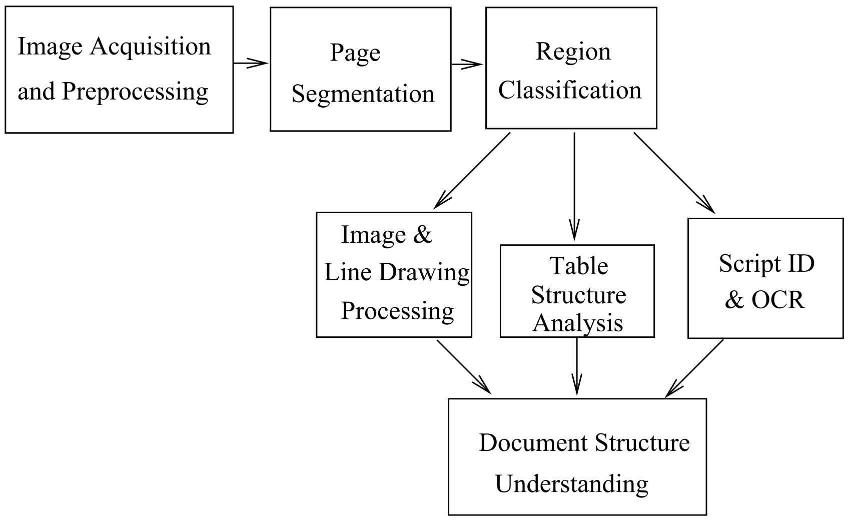


Image Acquisition and Preprocessing
Page Segmentation
Region Classification
Image & Line Drawing Processing
Table Structure Analysis
Script ID & OCR
Document Structure Understanding

Figure 1.11: A generic document understanding system.

towards developing robust recognizers for unconstrained handwriting. Specific problems in understanding document structure, such as recognition of on-line mathematical expressions have been attempted [15, 37]. In this thesis, we will be concentrating on the complementary problem of understanding the structure of on-line handwritten documents, without using a recognizer.

### 1.3.1 Document Layout

The layout of a document refers to the spatial arrangement of different sections of a document within the document image. This includes the number of columns of text, the text flow, placement of image regions and drawings, captions, titles, footnotes, structure of tables, etc.

Document understanding is a challenging problem even for printed documents due to the sheer variety of layouts possible for a document image. Hence most of the document understanding systems concentrate on specific classes of documents, such as document images from a specific set of journals or those from the yellow pages in a telephone book. Domain knowledge can be incorporated into the document understanding systems in such cases to improve the layout recognition performance. The common approaches to integrating domain knowledge into the document understanding process include modelling the document structure using heuristics, formal grammars, graph models, etc.

Examples of structure analysis and retrieval systems for document images can be found in [84], [12] and [73]. Research in the field of extraction of structure in on-line documents is limited. To the best of our knowledge, the work presented in this thesis is the first attempt to infer structure in on-line handwritten documents.

## 1.4 Document Retrieval

One of the important applications of developing a document understanding system is that one can query a document database to retrieve specific documents based on their content. The metadata, which is generated by the document understanding system, is used to match the users' query against the documents (see Figure 1.12). Note that a complete understanding of the document structure and its contents is not necessary to facilitate applications such as retrieval. One can query documents based on whatever information a document understanding system is able to extract.

Depending on the nature of queries, there are different types of retrieval systems possible. Keyword-based systems try to identify the relevant documents that deal with the topic described by the user's query. A different approach to querying is to specify a sample document to retrieve documents in the database that are similar to the one in the query. Such systems are popular for content-based image retrieval. The user can also produce a sketch as a query to retrieve documents with similar sketches. This is useful for retrieval of handwritten documents. One can also query a document based on its layout.

### 1.4.1 Text and Sketch-based Retrieval

The most common approach to retrieving documents, based on their contents, is to specify a set of keywords that are present in the document. In the case of on-line handwritten documents, the query can either be a handwritten or typed keyword. One could use a recognizer to convert all the text data in a handwritten document to ASCII text. However, the expressive power of handwritten data over typed text has led researchers to the conclusion that it is desirable to treat handwritten data (*digital ink*) as a primary data-type [91, 88, 4]. Another reason for requiring a recognition-free solution is the multitude of languages and symbols that an application using pen-based input needs to handle. Due to the inherent advantages of storing the handwritten data itself in documents as opposed to the recognized text, even recognition-based solutions store the recognized text as supplemental information to the handwritten data [127].

An alternate approach to retrieval of handwritten data is to use a search technique that can find a document in a database using a handwritten keyword as the query. The process of comparing handwritten or spoken words of the query to those in a document database, without explicit recognition, is referred to as 'word-spotting' [90]. The technique has also been used for recognition of on-line handwritten databases generated by a single writer [30]. This approach also enables the user to do retrieval based on hand-drawn pictures or diagrams as opposed to words. However, the matching of sketches and words poses different challenges in the case of on-line documents, since the temporal variations between similar sketches are larger compared to that of off-line handwritten words.

### 1.4.2 Structure-based Retrieval

The bottleneck in the word-based retrieval systems for document images and on-line documents is the recognition of text in the document to match the user's query. Retrieval systems try to deal with the ambiguity in recognition by using multiple interpretations of the text provided by the recognition algorithm. This usually results in poor precision rates in the retrieved documents. Additional constraints in the query can significantly improve the performance of the retrieval systems. The layout of a handwritten page provides such a constraint, which is both intuitive for the user to specify and easy to combine with the word-based retrieval algorithms. People generally tend to remember some of the attributes of the page structure that they have written or seen. Hence, a layout based retrieval system is primarily aimed to work with a small to medium sized database (hundreds of pages) of handwritten notes, typically written and queried by a single user. Such a system can also be employed in querying pages captured using electronic white-boards during seminars or lectures since the attendees would have seen the pages being written.

The user who queries the database describes a layout of the page from his memory and wants to retrieve the page based on the layout alone or in conjunction with some words in the document. Typical queries could be of the form: "*Retrieve the page that contains a large map (figure)*". Even in cases where the user has not seen the page to be retrieved before, such as while searching for documents on the web, layout or non-text content can be used in the queries. The user could specify the presence of a particular region to restrict the range of retrieved documents. For example, the user could pose a query such as: "*Retrieve pages on* **DNA structure** *with a* **sketch** *(figure)*".

## 1.5 Script Recognition

Handwritten documents can contain multiple languages that can possibly be in different scripts. For example, a document database could contain handwritten documents in English, Chinese and Arabic, which use three different scripts (Roman, Han and Arabic, respectively). Identifying the script of text regions can be useful for retrieval as the presence or absence of a particular script can be used to query the document. The user can specify the scripts that might be present in a particular document, which can greatly enhance the accuracy and speed of the retrieval system since the set of documents that need to be matched with the query become limited. In addition, the script recognizer can form a pre-processing step in text recognition, since the recognizers are usually developed for a specific script or language.

In addition to script, there is a variety of information that can be extracted from documents that can help in the document retrieval process. Examples of such information include presence of mathematical equations [14], music notations [122], specific classes of drawings such as maps [85], etc. However, such systems are developed with a specific document type in mind and hence are not dealt with in this thesis.

## 1.6 Security Issues in On-line Documents

Paper documents are commonly used for many applications where the document is legally binding. The authorship of a document is established in such cases by the signature of the author or a fingerprint impression on the document. The paper itself usually contains watermarks and prints which help in authenticating the document. The widespread use of digital documents in place of paper documents will require techniques to authenticate the documents. In addition, digital documents pose an additional challenge of verifying the integrity of documents after their creation, since the documents are easily modified on a computer. Techniques such as public key encryption-based digital signatures and digital watermarks have been used for this purpose in the case of document images.

One of the main approaches to authenticating digital documents is called (digital) watermarking. Watermarking techniques can be classified according to the various information hiding techniques and according to their use and properties (see Figure 1.13). The class of watermarking techniques suitable for document authentication fall under the category of fragile watermarking.

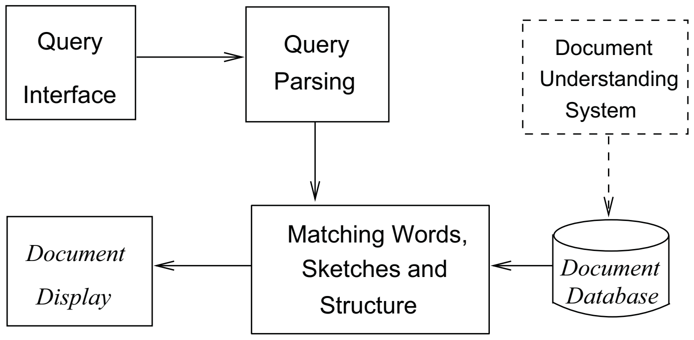

Query Interface
Query Parsing
Document Understanding System
Document Display
Matching Words, Sketches and Structure
Document Database

Figure 1.12: A document retrieval system. The database contains metadata generated by the document understanding system.

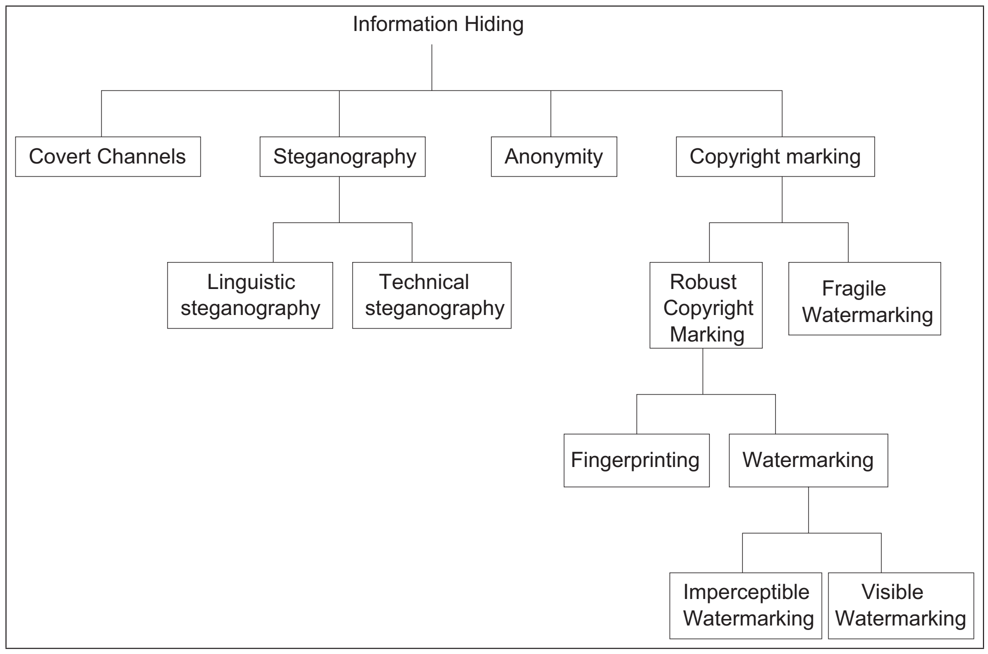


Figure 1.13: Classification of information hiding techniques by Petitcolas et al. [111].

## 1.7 Thesis Contribution and Outline

This thesis develops a mathematical basis for understanding the structure of on-line handwritten documents and explores some of the related problems in detail. Figure 1.14 gives an outline of the different problems that are dealt with in this thesis and their relationship to each other. A document retrieval system is developed, which demonstrates a practical application of the document understanding problem. A retrieval system that uses structure and content of the documents should be able to perform the following tasks.

1. Segment a document page into component regions.
2. Use the segmentation algorithm to infer the layout of the page.
3. Generate a representation of the layout in memory with multiple interpretations for efficient retrieval.
4. Match user's query with the layout and contents of the document.
5. Provide an interface to accept user's queries and display results.

In this thesis, we concentrate on the problem of document understanding. The aspects of efficiency in matching and representation is an independent problem, although related.

There are several contributions in this thesis, which help to develop a better understanding of the problem of on-line handwritten document understanding.

1. The thesis proposes a mathematical model for representation of on-line handwritten strokes. The model is completely based on the mechanics of the handwriting process and does not make any assumptions about the content or intent of the specific handwriting sample. This is different from most of the generational models that are used for recognition applications, which make specific assumptions about the script and style of writing. Moreover, the model is independent of the hardware that is used for

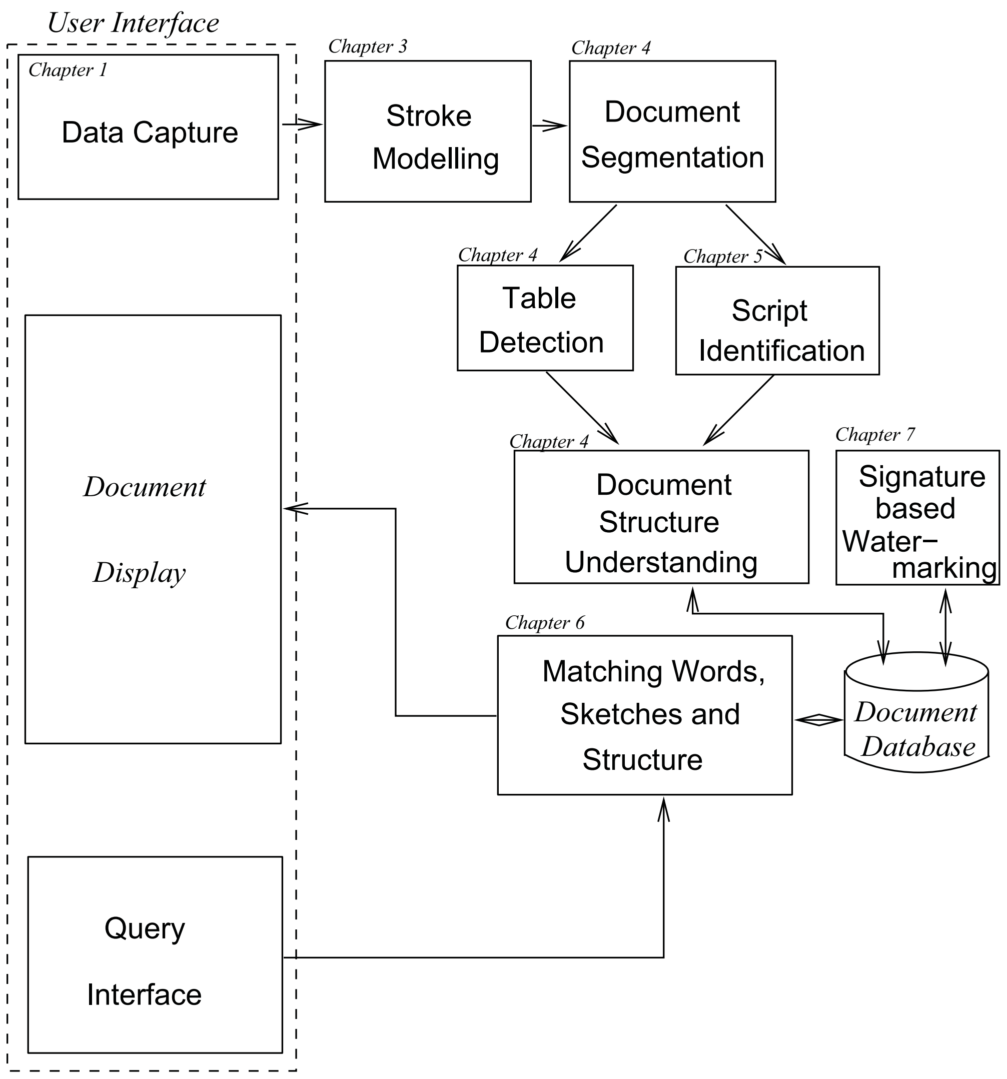


Figure 1.14: A schematic diagram of the proposed on-line document understanding and retrieval system. Note that the system combines a document understanding system (Figure 1.11) and a document retrieval system (Figure 1.12).

capturing the handwriting. Since the model makes no assumptions about the content of handwriting, it could be used in a variety of applications, such as recognition, script identification, person identification, handwritten data compression, etc.

2. Chapter 4 develops a principled approach to the problem of segmentation of on-line handwritten documents. The algorithm uses a stochastic context-free grammar based parsing technique to compute an optimal segmentation of a given handwritten page. The algorithm proceeds in a bottom-up manner by classifying the strokes and clustering them into regions. The region types are identified by the properties of strokes within them as well as their layout. The regions identified include words, text lines, ruled tables, and sketches (or figures).

3. A script recognition algorithm is presented in Chapter 5, which identifies six major scripts, Arabic, Cyrillic, Devnagari, Han, Hebrew and Roman, in on-line documents. The identification of scripts helps in developing a better understanding of the nature of a document. Scripts of individual words in a document can be detected by the algorithm, which extracts features based on the spatial and temporal properties of the strokes within each word.

4. A word-spotting-based retrieval scheme for personal on-line document databases is presented in Chapter 6. The algorithm computes a sequence of feature vectors at the sample points of the strokes of each word. An optimal alignment between the two sequences is determined using a dynamic time warping algorithm.

5. Chapter 6 also presents a sketch-based retrieval algorithm that accepts queries in the form of hand-drawn sketches. The algorithm, uses a line-based representation of hand-drawn sketches. The matching considers the spatial relationship between component lines in addition to their similarity. The sketch matching algorithm does not make any assumption about the nature of the sketch and hence is applicable to a wide variety of sketches.

6. An on-line signature-based watermarking technique that can establish the authorship and integrity of on-line documents is presented in Chapter 7. The system uses fragile watermarking to detect tampering of documents.

# Chapter 2

# Background in Document Understanding

In this chapter, we present an overview of the work done in the fields related to this thesis. The intent is to provide an overview of the state of the art in the corresponding areas. Hence, details of algorithms are not described, unless they are required to understand the work presented in this thesis.

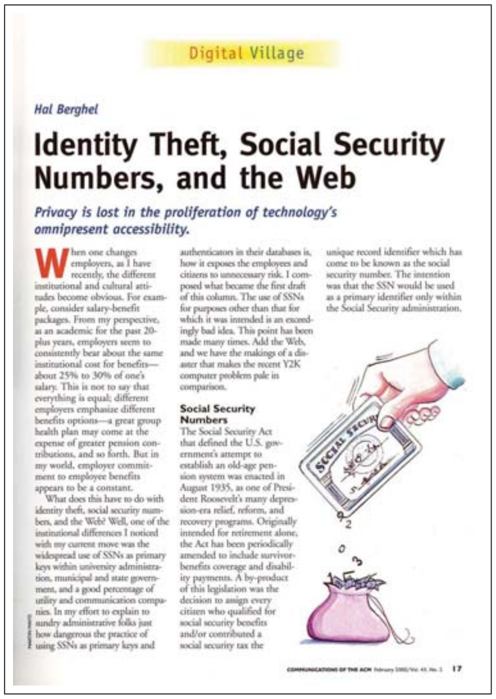
This chapter is organized as follows. Section 2.1 surveys the work that has been done in the field of document understanding. Work done on segmentation of documents is reviewed in Section 2.2. Sections 2.3 and 2.4 present work done in the field of indexing and retrieval of document images in general and describe the specific topic of on-line handwritten document retrieval. Section 2.5 presents a literature survey of the field of script recognition, and Section 2.6 gives a brief overview of the work related to watermarking of document images.

## 2.1 Document Understanding

Most of the research in document analysis has focused on off-line (scanned) documents [142]. Examples of this work include page decomposition [58], locating embedded text in color images [163], skew detection [162] and table identification [68, 45, 148, 41, 75]. With the introduction of devices like *IBM ThinkPad TransNote* © [143] and Electronic Whiteboards, it is now possible to store and process the entire on-line document. The temporal information, captured by the on-line documents can be used for both text recognition as well as segmentation of documents.

Recent advances in the processing of on-line handwritten data include algorithms for *(i)* segmentation of handwritten text into lines, words and sub-strokes [121, 82], *(ii)* character and word recognition [156, 116, 51], and *(iii)* indexing and retrieval of on-line handwritten documents.

## 2.2 Document Segmentation



Digital Village

*Hal Berghel*

**Identity Theft, Social Security Numbers, and the Web**

*Privacy is lost in the proliferation of technology's omnipresent accessibility.*

When one changes employers, as I have recently, the different institutional and cultural attitudes become obvious. For example, consider salary-benefit packages. From my perspective, as an academic for the past 20-plus years, employers seem to consistently bear about the same institutional cost for benefits—about 25% to 30% of one's salary. This is not to say that everything is equal; different employers emphasize different benefits options—a great group health plan may come at the expense of greater pension contributions, and so forth. But in my world, employer commitment to employee benefits appears to be a constant.

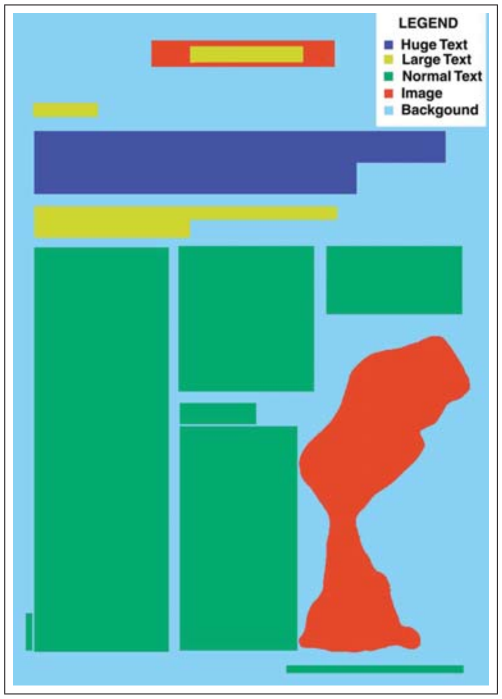
What does this have to do with identity theft, social security numbers, and the Web? Well, one of the institutional differences I noticed with my current move was the widespread use of SSNs as primary keys within university administration, municipal and state government, and a good percentage of utility and communication companies. In my effort to explain to sundry administrative folks just how dangerous the practice of using SSNs as primary keys and authenticators in their databases is, how it exposes the employees and citizens to unnecessary risk, I composed what became the first draft of this column. The use of SSNs for purposes other than that for which it was intended is an exceedingly bad idea. This point has been made many times. Add the Web, and we have the makings of a disaster that makes the recent Y2K computer problem pale in comparison.

**Social Security Numbers**

The Social Security Act that defined the U.S. government's attempt to establish an old-age pension system was enacted in August 1935, as one of President Roosevelt's many depression-era relief, reform, and recovery programs. Originally intended for retirement alone, the Act has been periodically amended to include survivor-benefits coverage and disability payments. A by-product of this legislation was the decision to assign every citizen who qualified for social security benefits and/or contributed a social security tax the unique record identifier which has come to be known as the social security number. The intention was that the SSN would be used as a primary identifier only within the Social Security administration.

COMMUNICATIONS OF THE ACM February 2000/Vol. 43, No. 2 17

(a)



(b)

Figure 2.1: Document Segmentation: (a) a document image and (b) the output of segmentation of the document in (a).

The problem of segmenting document images has received considerable research attention in the past. Figure 2.1 shows an example of a document image and the desired output of the segmentation algorithm. The desired output may vary depending on the application and could result in a coarser or finer segmentation of the document page. Various approaches for segmentation can be broadly classified into two categories, top-down and bottom-up. The top-down approaches start with the whole document page and divide it into blocks. The blocks may be further divided into smaller blocks if an initial block is believed to contain multiple regions. Wang et al. [157] used such an approach to segment newspaper images into component regions and Li and Gray [81] used wavelet coefficient distributions to perform top-down classification of complex document images. The bottom-up approaches start with individual pixels or a collection of neighboring pixels and group them together to form contiguous regions of the same type. Jain and Yu [58] used a top-down document model for performing a bottom-up clustering of pixels while Etemad et al. [31] used fuzzy decision rules for bottom-up clustering of pixels using a neural network. Cheng et al. [17] proposed a hybrid bottom-up, top-down approach where the algorithm adopts a fine-to-coarse-to-fine approach for segmenting a document image. A third approach is to use the white spaces available in document images to find the boundaries of text or image regions as proposed by Pavlidis [107]. Jain and Bhattacharjee [52] used Gabor filters for texture-based segmentation of text in document images. Connected component based methods have been used for both text segmentation in document images [104] and for generic image segmentation [158, 135]. Nagy [94] provides an excellent survey of the different document analysis approaches that have been attempted.

Grammar-based approaches have been used in many problems related to document understanding in the case of printed documents. Chou and Kopec [18, 74] described an attribute grammar-based model of off-line documents and its use in document recognition. Grammar-based approaches have also been used in the recognition of table structure. Rahgozar and Cooperman [119] used a graph-grammar model for recognition of table structures in printed documents. Zanibbi et al. [118] give a comprehensive review of the various table recognition models and algorithms. Description of pictures using grammars and their use in comparison of pictures were investigated by Shaw [134] and Clowes [19]. Such approaches could be extended to deal with on-line sketches.

The work in on-line document analysis till date is limited to segmentation of text lines [121, 10, 117] Artieres [6] described a text line and word segmentation system using probabilistic feature grammar, which uses a text model that is similar to that in this thesis.

## 2.3 Document Indexing and Retrieval

The area of content-based information retrieval has been an active research topic for almost two decades. The specific field of content-based image retrieval has been explored in great detail [26, 66, 84]. The notable commercial systems that perform content-based image retrieval include the QBIC system [34] from IBM and the VIR Image Engine [7] from Virage, Inc. Figure 2.2 shows the results of content based image retrieval by the *BlobWord* image search system by Carson et al. [11]. The query image is shown at the top left of Figure 2.2, in a box. The performance of content-based image retrieval systems leave a lot to be desired. Most of the image search engines on the Internet, such as the Google Image Search and Yahoo Picture Gallery, use the text associated with the images in HTML documents for image retrieval.

The literature in content-based image retrieval is rich and this chapter does not intend to describe work in this area. The reader may refer to the surveys by Smeulders et al. [140] and Veltkamp [152] for that purpose. Indexing and retrieval of document images is a specific topic in this area that has received special attention due to the large quantities of document images archived in databases and the need to query and retrieve specific document based on its contents. Doermann [26] presents an extensive survey of document image indexing and retrieval.

Approaches to on-line handwritten document retrieval can be divided into different

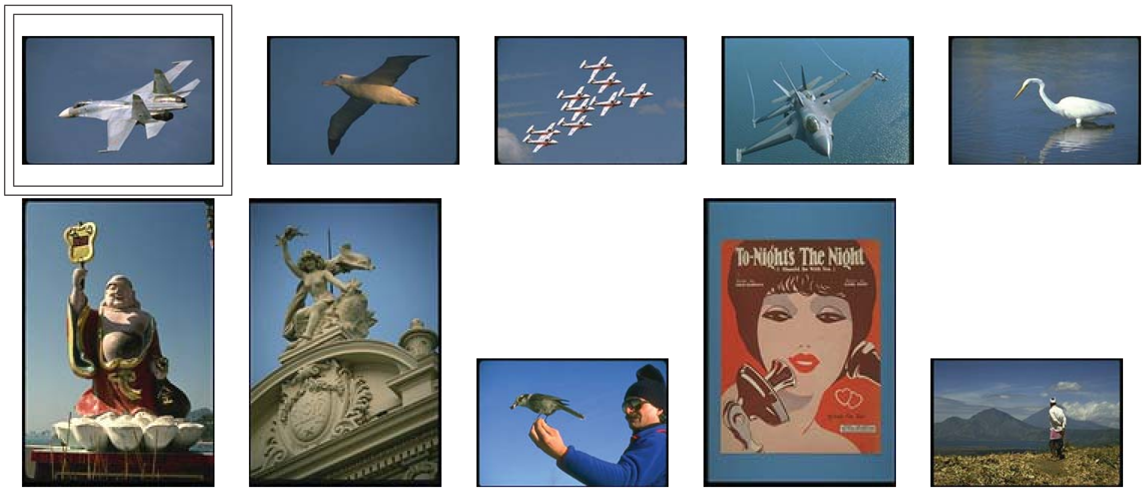


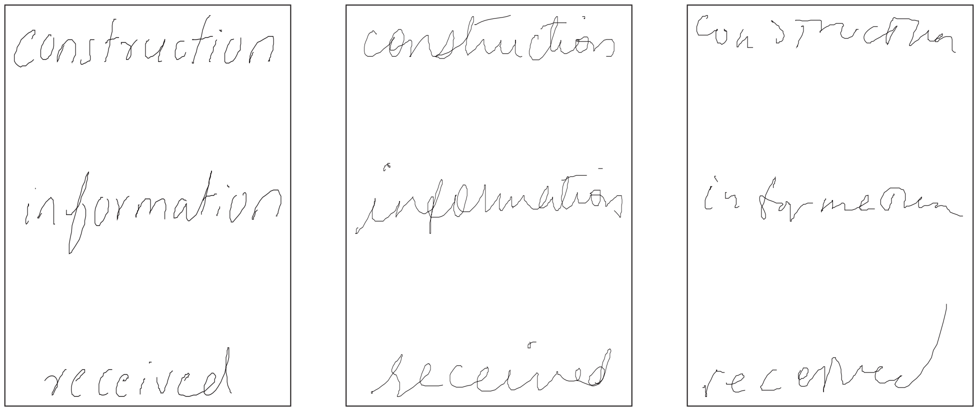
Figure 2.2: A query image (in the box) and the results of retrieval (top 9) by the BlobWorld Image retrieval system [11].

categories based on the representation of the document used in the database, the features that are extracted from the queries and the documents and the algorithms used for matching. Russell et al. [127] store the top $N$ candidates of the recognition output of each word along with the handwritten data in the document database. Lopresti and Tomkins [88] use a stroke representation, where the handwritten data (or digital ink) is divided into smaller segments at points of local minima of the $y$ coordinates (vertical axis). Kamel [65] uses a set of features extracted from the strokes to represent the documents in a database. Leung and Chen [80] represent the diagrams in each document using a feature vector, whose elements are confidence values corresponding to different shapes derived from a shape estimator. Jain and Namboodiri [54] represent the handwritten data as a set of points that are sampled from the ink trace. Singer and Tishby [137] modeled on-line handwriting as modulated cycloidal motions of the pen tip to do retrieval and recognition. Rath and Manmatha [120] store the segmented images of words in an off-line handwritten document database, which is used during querying to match the keywords supplied by the user.

A second classification of retrieval algorithms is based on the features extracted from the data and the matching algorithm used in retrieval. We note that the data representations can belong to two different classes. Fixed length representations such as global feature vectors lead to simple matching techniques (e.g., Euclidean distance between points in a parameter space), where as variable length representations such as point sequences or sequence of feature vectors from stroke sequences require a soft distance measure (e.g., elastic string matching). Russell et al. [127] use the dot product of the confidence values of the top $N$ results of recognition of two handwritten words to arrive at a distance measure. Leung and Chen [80] use the distance measure between features computed for each shape-type, weighted by their confidence values. Kamel [65] uses a voting method to find the distance between strings, where the presence of a stroke type from the query, represented as a feature vector, in a word in the database, results in a vote for the word. Lopresti and Tomkins [88] use Dynamic Time Warping (DTW) to match the feature vector sequences derived from the sample points and strokes. Manmatha et al. [90] used the edit distance measure to find the distance between two word images. The matching techniques used for on-line handwritten data in [88, 127, 65, 80] are described later.

Several approaches to word-spotting in documents have been reported in the literature. However, these studies are restricted to off-line handwritten or printed documents and audio documents [35] for the most part. Kuo and Agazzi [77] used a Pseudo 2-D HMM model to spot keywords in poorly printed documents. They achieved $96\%$ accuracy on a synthetically generated dataset with words of different font sizes. Curtins [24] employs multi-font character templates to match individual characters to spot keywords in noisy printed text and attains a recall rate of $90\%$ and precision rate of $97\%$ on a set of $380$ document images. O'Neill et al. [100] reported $90\%$ recall and more than $95\%$ precision in spotting printed words using moment features of the word pixels. The test set contained two words in $13$ font sizes with $20\%$ salt-and-pepper noise.

In the case of handwritten documents, word-spotting systems attain considerably lower recall rates due to the intra-class variability in handwritten words (see Figure 2.3). Manmatha et al. [90] used an Euclidean distance map-based algorithm to index handwritten text using word image templates. Kolcz et al. [72] used the profile of the words to match handwritten words. Their method achieves a recall rate of $45\%$ with no false alarms. Singer and Tishby [137] modelled on-line handwriting as modulated cycloidal motions of the pen tip to do recognition. The authors reported successful spotting of parts of a word in a small database of $30$ words using dynamic time warping. Lopresti and Tomkins [88] used an edit distance measure to match a sequence of feature vectors extracted from segments of strokes in on-line handwritten data. The method achieves a recall rate of $95\%$ with a precision of $4\% to 7\%$ on databases containing $2,000$ and $4,000$ words, respectively, by two writers.



(a) Writer 1 (b) Writer 2 (c) Writer 3

Figure 2.3: Variability in writing styles of three different writers for the same set of words.

Matching of hand-drawn sketches poses a different set of challenges due to the large amount of variability among multiple instances of a figure. Lopresti et al. [89] have studied the problem and provided some initial experimental results that are encouraging. They describe the need for efficient algorithms to match components of sketches and utilize spatial arrangement of the components. Leung and Chen [80] represent the diagrams in each document using a feature vector, whose elements are confidence values corresponding to different primitive shapes derived from a shape estimator. One could also retrieve printed or hand-drawn sketches and images using sketches as queries. Jain et al. [60] addressed the problem of image retrieval using a deformable template, which is a binary edge image. The matching process takes into consideration the energy required to deform the model and the goodness of fit of the deformed model to the target image based on the gradient of the image. Del Bimbo and Pala [9] used a similar approach to match user drawn sketches to images in a database for retrieval. Manmatha et al. [90] used the edit distance to measure similarity between word shapes for retrieval. The problem of shape matching also involves the computation of a global transformation for aligning the two shapes. Belongie et al. [8] use a set of shape descriptors named 'shape context' to align two shapes for matching, which could be applied in the context of matching sketches.

In spite of many practical applications of indexing document databases, very few researchers have reported the results of their word matching algorithms for indexing. Manmatha et al. [90] used their word-spotting algorithm for indexing the handwritten document databases of E.D. Hudson and George Washington [67] and reported that the method was feasible. However, the error rates of the indexing algorithm were not reported. Since the basic problem of word matching in handwritten data is the same in both indexing and retrieval problems, we will concentrate on the retrieval problem from here onwards.

As mentioned above, the representation of handwritten data in the document database plays an important role in the efficiency and effectiveness of matching during the retrieval process. The level of abstraction increases as the handwritten data is transformed from point sequences, through multiple steps, into recognized text. Figure 2.4 shows different levels of abstraction of the data during recognition and the corresponding matching problems as described by Lopresti and Tomkins [88]. The two columns represent the processing of the keyword (pattern ink) and the words in the database (text ink) as they pass through different stages of recognition. Correspondingly, the matching problem changes from one of point matching between two words to matching the corresponding ASCII text. Matching at a higher level of abstraction such as the recognized text (see [127, 30]) can be efficient as far as the matching time is concerned. In addition, one can handle multi-writer databases better as the level of abstraction increases. Note that the approach used by Leung and Chen [80] for matching hand-drawn diagrams does recognition of shapes and hence it is user-independent. However, as the level of abstraction increases, the system becomes more constrained. For example, a recognition based approach can handle only text documents and that too in a specific language. Moreover, the choice of data representation, such as feature vectors used, becomes critical in the matching process and a poor choice of features can affect the performance of the system.

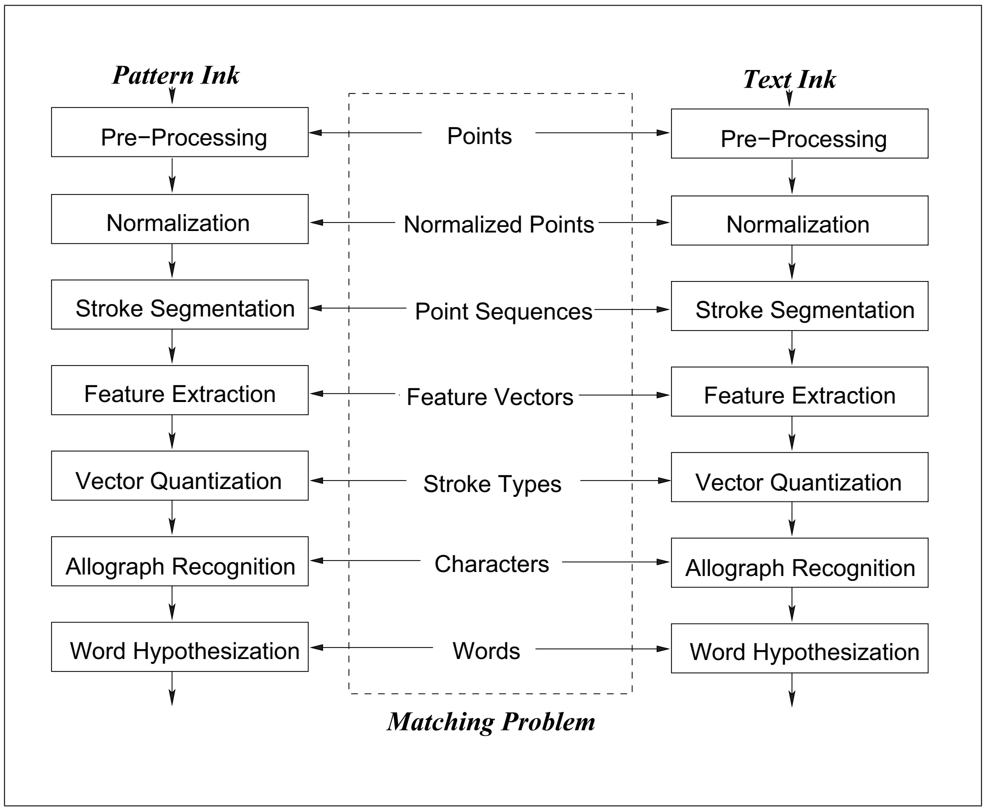


Figure 2.4: Word Matching at multiple levels of abstraction (from Lopresti and Tomkins [88]).

A lower level of abstraction in the data representation, such as a set of data points or a set of feature vectors derived from point sequences (see [54]) can lead to a generic content-based retrieval system. However, one needs to employ more powerful matching algorithms such as Dynamic Time Warping (DTW) on longer data sequences, leading to a slower retrieval system. Lopresti and Tomkins [88] use an intermediate level of abstraction, viz. features derived from stroke sequences, which the authors claim to be the best suited for word matching. However, a comparison of the retrieval results between their method and ones with lower and higher levels of abstraction reported in [54] and [127], leaves a reader unconvinced of the claim.

## 2.4 Matching Queries

In this section, we explore different methods for query matching employed in on-line handwritten document retrieval. The algorithms can be divided primarily into two classes: (i) recognition-based and (ii) recognition-free approaches. One might notice that the problem of on-line signature recognition, which has been studied extensively in the biometrics community (see Nalwa [96]), poses a similar problem of matching handwritten data. However, since the intra-class variability in the signature verification problem is generally lower than in generic word-matching, the latter calls for representations and matching techniques that are less affected by those variations.

We describe the algorithms used by Russell et al. [127], Leung and Chen [80], Lopresti and Tomkins [88] in detail here as they are representative of different classes of matching algorithms.

### 2.4.1 Recognition-based Text Matching

Russell et al. [127] proposed a text recognition based solution for retrieval of handwritten documents containing English text. Their algorithm runs an HMM-based text recognizer (reported in [144]) that returns the top $N$ word candidates, with the associated confidence values. This list of words is referred to as the N-best list. The matching can be performed in five different ways:

1. Typed vs. Top: In this case, the query is a typed keyword and it is compared with the best (top) candidate from recognition of each word in the database. Any of the exact text matching algorithms, such as Knuth-Morris-Pratt algorithm [71], could be used for text matching.

2. Top vs. Top: The query in this case is a handwritten keyword. The top candidate from the recognizer is used to retrieve documents from a database and the matching is similar to the previous case.

3. N-Best vs. Top: This is similar to case 2, except that each of the $N$ candidates from recognition results of the keyword is matched against the top candidate from recognition of words in the database.

4. Typed vs. N-Best: In this case, the query word is typed and is compared to each of the N-best candidates from recognition of words in the database.

5. Dot Product: In this case, the query is a handwritten keyword. The distance metric is the normalized dot product of the N-best confidence values of the keyword and database word recognition results.

Figure 2.5 shows the precision vs. recall plots of Russell et al.'s algorithm on a database of $35,172$ words. A total of $1,600$ independent query words were used for this experiment.

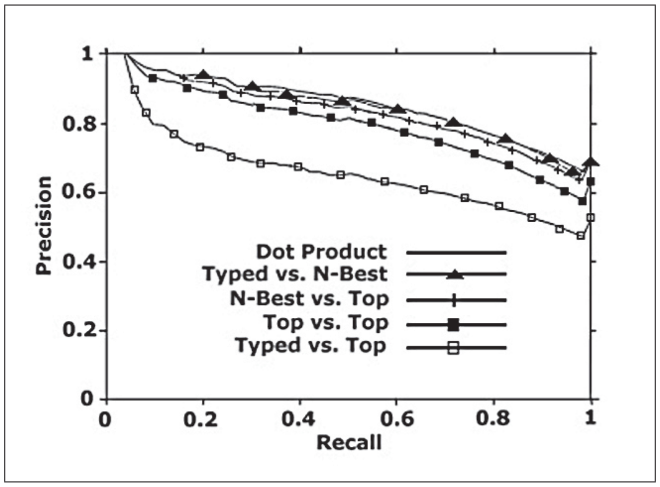


Figure 2.5: Precision vs. Recall curves for the algorithm reported in Russell et al. [127].

### 2.4.2 Recognition-based Picture Matching

As noted in the beginning of this chapter, the query for document retrieval need not necessarily be text. Leung and Chen [80] proposed a recognition based approach for matching on-line hand drawn sketches by identifying basic shapes such as circles, straight lines and polygons in the sketch. A more generic system of picture representation has been attempted by Sinha [138].

To identify the basic shapes, the strokes whose end points are close to each other are connected together. A set of features such as the center of the stroke, the perimeter, the area, the convex hull and perimeter efficiency are computed for each of the resulting strokes. The perimeter efficiency of a stroke is defined as $k = \frac{2\sqrt{\pi A}}{p}$, where A is the area enclosed by the stroke (assuming a closed figure) and P is the perimeter.

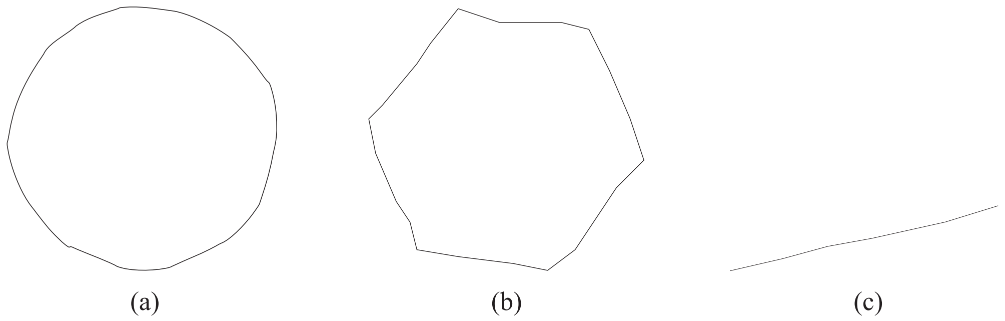


Figure 2.6: Examples of basic shapes used in the algorithm by Leung and Chen [80]. (a) circle, (b) polygon, and (c) line.

Once the features are computed, estimators of basic shapes are used to recognize the shapes. An estimator is a function that combines the features for classification of the strokes as one of the basic shapes. The three basic shapes (Figure 2.6) used and the associated properties used by estimators are given below.

1. Circle: perimeter efficiency is close to 1; large number of points in convex hull, and points traverse $360^o$ around the center of the stroke.

2. Polygon: Ratio of area enclosed by the stroke to the area of the convex hull is close

to 1; fewer number of points in the convex hull, and points traverse $360^o$ around the center of the stroke.

3. Straight line: Ratio of sum of distances between neighboring points in a stroke and the distance between end points is $1$, and the height of the triangle formed between the end points and any point on the line is small.

For each stroke, the estimators return a confidence value for each stroke type. The matching score between two strokes is the product of the confidence values for the stroke type, which maximizes that value. The matching score is also multiplied with the product of the feature vectors for each type to handle shapes that are not mentioned above. The matching score of two sketches is computed by weighting the matching scores of individual strokes, inversely, with the distance between their centers.

A database of $35$ shapes was collected from $4$ writers for testing, with each writer drawing the shape $20$ times. A second database of the same shapes was collected after $8$ months with $5$ repetitions. Figure 2.7 shows examples of on-line hand-drawn sketches from the database.

Figure 2.8 gives the precision vs. recall curves for the experiments.

### 2.4.3 Recognition-free Text Matching at the Stroke Level

Lopresti and Tomkins [88] proposed a recognition-free approach for matching text data in on-line handwritten documents. Even though the algorithm does not do recognition, the first stage of stroke segmentation assumes that the handwriting trace, referred to as ink, is a sequence of regular up and down movements of the pen as in the case of cursive English handwriting.

Stroke segmentation is the process of dividing the ink into smaller segments, called strokes. The ink is split at every sample point that is a local minimum for the $y$-coordinate along the trace. This might result in spurious strokes due to pen vibrations. However, the

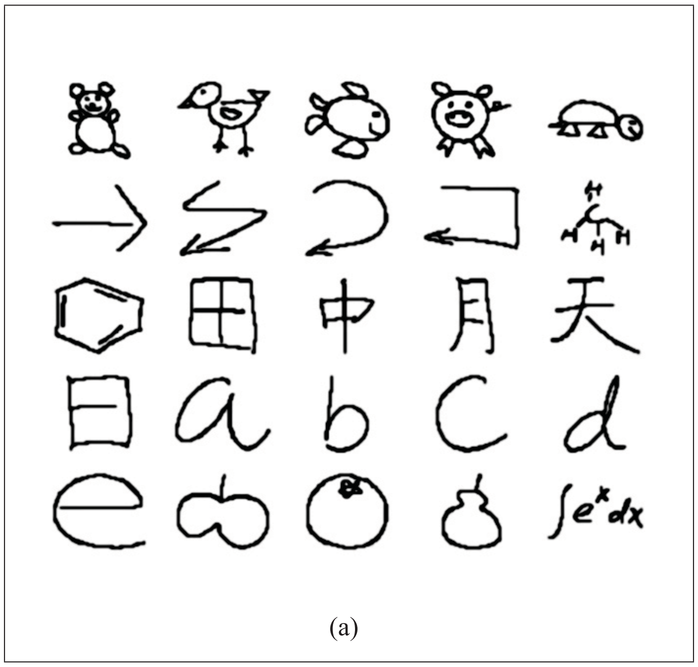


Figure 2.7: Examples of hand-drawn sketches from the database used by Leung and Chen [80].

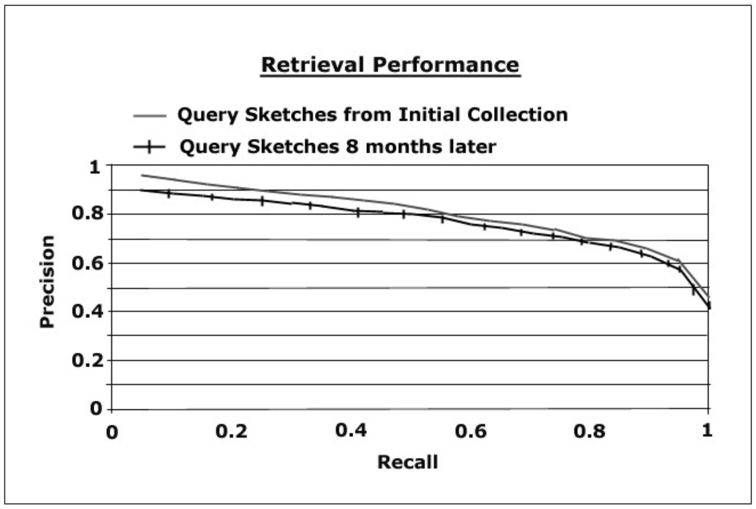


Figure 2.8: Precision vs. Recall curves for the algorithm reported in Leung and Chen [80].

matching stage is designed to handle a small amount of such spurious strokes. A set of 13 features are then extracted from each of the strokes, which together characterizes the stroke type. The feature values extracted are described by Rubine in the context of gesture recognition [124]. A handwritten document is thus converted to a sequence of feature vectors, each corresponding to a stroke. Each feature vector is then classified into one of $c$ classes, where the classes are determined by clustering the strokes of the particular user from a training set. Figure 2.9 shows a sample set of on-line words that are segmented into strokes by the algorithm reported in Lopresti and Tomkins [88].

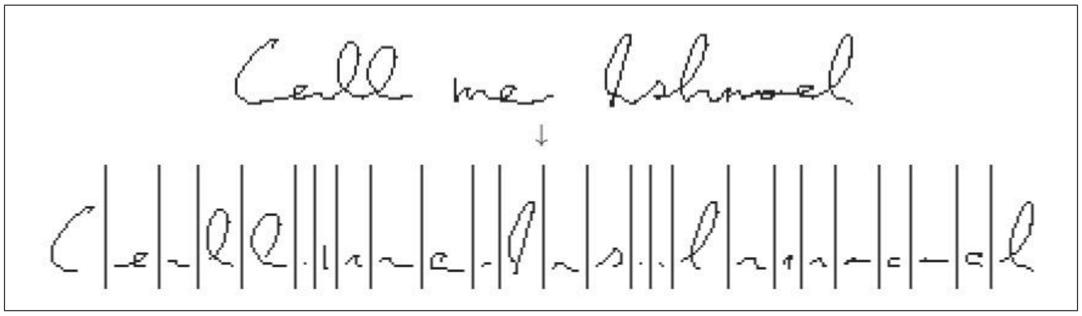


Figure 2.9: Segmentation of words into strokes (Lopresti and Tomkins [88]).

To carry out the matching, the query word is also converted to a set of strokes and the feature vectors corresponding to each stroke is extracted. The feature vectors are then classified into one of the $c$ classes as before. The matching is done using a dynamic programming based string matching algorithm. To match two stroke sequences $P$ and $T$ of lengths $m$ and $n$, respectively, we initialize an $m \times n$ array $d$, as follows:

$$d_{0,0} = 0$$

$$d_{i,0} = d_{i-1,0} + c_{del}(p_i) \quad 1 \leq i \leq m$$

$$d_{0,j} = d_{0,j-1} + c_{ins}(t_j) \quad 1 \leq j \leq n,$$

where $c_{del}()$ is the cost of deleting a stroke from $P$ and $c_{ins}()$ is the cost of inserting a stroke into $P$.

The dynamic programming recursion used for matching can be written as:

$$d_{i,j} = min \begin{cases} d_{i-1,j} + c_{del}(p_i) \\ d_{i,j-1} + c_{ins}(t_j) \\ d_{i-1,j-1} + c_{sub\ 1:1}(p_i, t_j) \\ d_{i-1,j-2} + c_{sub\ 1:2}(p_i, t_{j-1}t_j) \\ d_{i-2,j-1} + c_{sub\ 2:1}(p_{i-1}p_i, t_j) \end{cases}$$

The three additional costs, $c_{sub\ 1:1}()$, $c_{sub\ 1:2}()$ and $c_{sub\ 2:1}()$ are included to handle the spurious strokes mentioned before. $c_{sub\ i:j}()$ is the cost of replacing $i$ strokes in $P$ with $j$ strokes in $Q$.

A related work that uses a similar approach to data representation is by Kamel [65]. In this work, the author starts out with 11 of the 13 features described above. To improve the efficiency of matching, the feature vectors are inserted into a $R - tree$. A word is hence represented by a set of points in the 11-dimensional space, with each point representing a stroke. During the matching stage, each feature vector from the query word votes for the words in the database. Any word with a feature vector close to that in the query word gets one vote. The words in the database that get the most votes are selected for retrieval. The matching rate was reported to be $84\%$ when the top three matches for each query are considered.

### 2.4.4 Recognition-free Matching at Point Level

In this case, the matching algorithm performs a direct comparison of the sample points of the strokes in a word. The algorithm works at the lowest level of abstraction and hence can handle multiple languages effectively. This class of algorithms is also capable of handling non-text data. However, the applicability of a specific algorithm to non-text data depends on the specifics of the matching algorithm used. We explore this approach in more detail in this work (see Chapter 6).

### 2.4.5 Comparing the Matching Techniques

One of the main obstacles in comparing the results of various retrieval algorithms is the absence of a standard database. However, judging by the nature and size of the databases used in each of the experiments reported, one could conclude that the recognition-based approach described in Section 2.4.1 performs the best on databases containing only English text. The stroke-based matching is good in cases where we want to avoid recognition, while making use of the fact that the data contains only English text. The point-based matching approach is powerful and flexible, but is more suited for single-writer databases. One could use an R-tree structure, similar to the one used by Kamel [65], to increase the speed of the algorithm, by reducing the number of words that need to be compared.

Algorithms that combine matchers that work at multiple levels have also been proposed. Hull et al. [46] describe a system that combines three matchers, each comparing words as point sequences, sequence of stroke shapes and overall word shape, respectively. The authors report an accuracy of $98.1\%$ for the best match on a small database of $200$ words.

Finally, the problem of determining the relevance of the retrieved documents based on query words is very important, especially in the case of retrieval of documents from a larger database, such as the Internet [150]. The critical problem here is to determine the meaning of words or expressions in a document using higher level context analysis [129]. The meanings of individual words and phrases in a document are then combined using statistical methods to determine the topics of discussion of the document. A retrieval system could then use the topic of the document to rank the documents that match a query. The problem of document relevance needs to be addressed to make the retrieval systems more useful.

## 2.5 Script Recognition

Most of the published work on automatic script recognition deals with off-line documents, i.e., documents that are either handwritten or printed on a paper and then scanned to obtain a two-dimensional digital representation.

For printed documents, Hochberg et al. [43] used cluster-based templates for discriminating 13 different scripts. Spitz [141] proposed a language identification scheme where the words of 26 different languages are first classified into Han-based and Latin-based scripts. The actual languages are identified using projection profiles of words and character shapes. Jain et al. [59] used Gabor filter based texture features to segment a page into regions containing Han and Roman scripts. Tan [147] describes another texture based approach for language classification in printed documents. Pal and Chaudhuri [102] have developed a system for identifying Indian scripts using horizontal projection profiles and looking for the presence or absence of specific shapes in different scripts. Other approaches to script and language identification in printed documents are reported in [145, 146, 108].

There have been very few attempts on handwritten script identification in off-line documents. Hochberg et al. [42] used features of connected components to classify six different scripts (Arabic, Chinese, Cyrillic, Devnagari, Japanese and Roman) and reported a classification accuracy of 88% on document pages. Note that some of the previous work on on-line documents (e.g., [99]) uses the term *on-line* to refer to documents on the World Wide Web where the goal is to infer the language of a character-coded text document.

Currently, there are a few algorithms available for on-line text recognition for individual scripts, but there have been no attempts to automatically recognize the script in on-line documents. The only work in processing multi-lingual on-line documents that we are aware of is by Lee et al. [78], which attempts to do recognition of multiple languages simultaneously using a hierarchical Hidden Markov Model. A script identification system can improve the utility and performance of on-line data capturing devices, and also aid in the search and retrieval of on-line documents on the Internet containing a specific script.

## 2.6 Document Security and Watermarking

Watermarking techniques have been studied extensively. Petitcolas et al. [111] provide a comprehensive survey of digital watermarking techniques. The most common variety of watermarking techniques involve robust watermarking for copyright protection [40]. However, for the purpose of authentication, we need to assure that a digital document has not been tampered with from the time it was created and signed by the author to the time it was received at the destination. The specific problem of watermarking using biometric traits has been studied in the context of authenticating a biometric template by Jain et al. [57]. A detailed description of various data hiding techniques for authentication and other applications can be found in Wu and Liu [159].

Kundur and Hatzinakos [76] proposed a fragile watermarking technique for wavelet compressed still images for the purpose of document authentication. Yeung and Mintzer [160] proposed a fragile watermarking system for image verification. A modification of the above algorithm was used by Yeung and Pankanti [161] for the specific purpose of authenticating biometric data.

## 2.7 Summary

There is an extensive body of literature that deals with the different problems involved in document understanding and retrieval of printed document images. The work on off-line handwritten documents has been limited, and concentrates on specific applications such as postal address recognition, indexing specific databases, etc. The work on on-line data mainly concentrates on the recognition problem, where the data is assumed to be text in a specific language. Issues such as retrieval of documents based on structure and non-text content need to be addressed for facilitating the use of pen as an interaction and data input mechanism for hand-held devices.

# Chapter 3

# On-line Data Model

In this chapter, we develop the mathematical basis for representation of the on-line handwritten data. The model used to describe the handwriting should be able to closely approximate the data collected from users, while being compact in representation. In addition, the model should be efficient to compute, and should easily lend to operations such as comparison, modification and computation of properties like intersections and bounding boxes of handwriting samples. We will examine the desirable properties of such a model, later in this chapter.

On-line handwriting is a two-dimensional process, unfolding in the $x - y$ plane as the time $t$ progresses. The process can be described by the function:

$$\mathcal{H} : \quad \mathcal{R} \rightarrow \mathcal{R} \times \mathcal{R}, \tag{3.1}$$

where $\mathcal{R}$ is the set of real numbers. The function $\mathcal{H}$ is defined only for those values of $t$ for which the pen is in contact with the digitizing surface. Hence, we divide the handwritten data into a sequence of $N$ curves:

$$\mathcal{H} = \ < \xi_1, \xi_2 \cdots , \xi_N > . \tag{3.2}$$

Each curve $\xi_i$ is defined in the closed interval $[t_i, t_i + d_i]$, where $t_i$ is the time of pen-down and $d_i$ is the duration between the pen-down and the pen-up (see Figure 3.1).

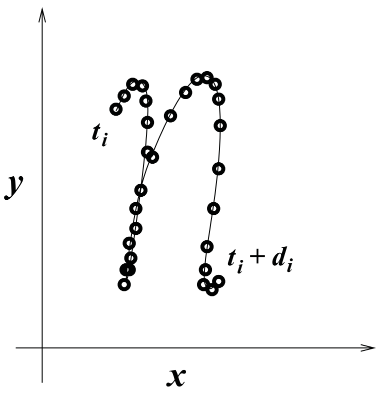


Figure 3.1: Example of on-line handwriting. The dots indicate pen positions that are equidistant in time.

The temporal order of the curves is enforced by the following relationship.

$$t_{i+1} > (t_i + d_i), \quad \forall i < N. \tag{3.3}$$

In many applications, the exact values of $t_i$ are not important as long as the above inequality is satisfied.

The data generated by an on-line data capturing device is a sequence of points, $(x_k, y_k)$, called a stroke, which is obtained by sampling the curve at regular intervals of time.

$$stroke_i(k) = Q(\xi_i(t_i + kT)), \;\; k = 0 \cdots d_i/T, \tag{3.4}$$

where $T$ is the period of sampling. The sampling frequency is hence given by $1/T$. The resulting sequence of points is then mapped to a finite set of values for representation inside the computer. This mapping, $Q$, is usually a staircase function and the process is referred to as quantization. In practice, $d_i$ is always a multiple of $T$ and the stroke contains $n_i + 1$ samples, where $n_i = d_i/T$. Equation (3.4) gives the ideal output of the digitizing device.

However, in practice the observed signal is corrupted by noise from various sources and errors in the digitizing device. Details of this process are discussed later in this chapter.

The models for representing handwriting attempt to describe the functions $\xi_i$ for the purpose of storage, recognition, rendering, etc. In the next section, we look at the different models that are currently used for representation of handwritten data, and some of their properties.

## 3.1 Handwriting Models

There are a variety of properties that are desirable for a handwriting model. Some of them are related to data representation, while others are important for recognition. In this work, we are trying to develop a curve model that best approximates the data collected from a user, while being efficient for processing. Hence the desired properties might be different from those of a model appropriate for recognition, which tries to model character or word classes. The following is a list of desirable properties:

1. *Flexibility*: The curve model should be able to represent arbitrarily complex curves. It should be able to precisely reproduce handwriting data collected from users. A close approximation of a real handwriting sample should also retain as much of information of the handwriting sample as possible. Note that this is opposed to the requirements in recognition models, which try to eliminate user specific variations of individual characters or words. However, it is desirable for the model to incorporate assumptions that are derived from the mechanics of the handwriting process. This will enable us to eliminate some of the noise introduced by the digitizer, while computing the parameters of the stroke for the model.

2. *Compactness*: The representation of the model should be compact so as to occupy the least amount of memory while processing. This means that a specific curve should be represented using as few parameters as possible.

3. *Stability*: The representation should be stable with respect to the parameters. This means that small variations in the parameter values should not produce large changes to the curve it represents.

4. *Uniqueness*: There should be a one-to-one mapping between a curve and the set of parameters that represents it. This requirement, along with the stability of representation helps us in comparing two curves by comparing the parameters used to represent them.

5. *Affine Invariance*: An affine transformation (translation, rotation, and scaling) of the model representation should not change the shape of the curve. The resulting curve should be the same as the original curve, with the transformation applied.

6. *Local Control*: It is the ability to manipulate a portion of the curve, without affecting the whole curve. This is useful in computations on the curve, for example, by splitting a curve into two parts.

7. *Computability*: The model should be easy to compute (least amount of processor requirements) from the point sequence representation, provided by the digitizer.

8. *Ease of Rendering*: One should be able to render the curves (compute the points on the curve with arbitrary precision), without doing complex computations. This involves interpolating the input curve between the sample points.

In addition to the above properties, it is desirable to have a representation that yields to simple computation of different local and global properties of a curve, such as curvature, tangents, cusps, bounding box, etc. It is also desirable to be able to compute the intersection of two curves or parts of a curve. In the remainder of this section, we examine some of the popular models for representing curves in handwriting and discuss their properties.

### 3.1.1 Handwriting as a Point Sequence

The simplest and the most popular representation of a curve is a sequence of points:

$$\xi_i \;=\; < (x_k, y_k) >; \quad k = 0, 1, \cdots, n_i \text{ and } \; x_k, y_k \in [0, max_{xy}], \tag{3.5}$$

where $max_{xy}$ is the maximum value that $x_i$ or $y_i$ can take. The representation is widely used because most digitizing devices generate the handwriting data in such a format. Hence this representation also retains all the information that is captured by the digitizer. The point sequence representation is equivalent to a piecewise linear curve obtained by connecting the sample points with straight lines.

### 3.1.2 The Oscillation Model

The oscillation model, proposed by Eden [29], tries to explain the generation of cursive English handwriting by modelling the pen tip movement as modulation of an oscillatory motion. The model considers the handwriting as a result of two orthogonal oscillations in the writing plane, which is superimposed on a constant horizontal velocity in the direction of writing. The oscillations can be described as:

$$\begin{aligned} \dot{x}(t) &= a\, sin(\omega_x(t - t_0)\phi_x) \;+\; c \\ \dot{y}(t) &= b\, sin(\omega_y(t - t_0)\phi_y), \end{aligned} \tag{3.6}$$

where $\omega_x$, $\phi_x$, $\omega_y$ and $\phi_y$ are the horizontal and vertical frequencies and phases, respectively, $a$ and $b$ are the horizontal and vertical velocity amplitudes, $t_0$ is the reference time and $c$ is the horizontal velocity, which creates the writing sweep. The letters are formed by modulating the above parameters at appropriate times in the writing. The parameter $a$ controls the letter height, and by modulating $\omega$ and $\phi$, one can control the slant and direction of the cycloidal pen movement (see Hollerbach [44]). A modification of this model was proposed by Chen et al. [16], who model the parameters $\omega$ and a damping factor of oscillation, $\zeta$ as piecewise linear functions.

The model is a simplistic explanation of cursive English writing, specifically for the Palmer style of writing [103], and could be useful for its recognition. However, it fails to model drawings, scripts other than English, or even printed letters in English. Hence the oscillation model is not appropriate as a generic representation for on-line handwriting.

### 3.1.3 The Delta Lognormal Model

The Delta Lognormal theory, proposed by Plamondon [112, 113, 114, 115] is one of the most powerful generational models of handwriting. It assumes that each curve is composed of a sequence of primitives, called strokes. Note that, unlike our previous definition, the term stroke, as used by Plamondon, refers to a part of the trace of the pen between a pen-down and pen-up. The strokes are formed by the hand motion resulting from two competing muscle activities, the agonist and antagonist. The velocity resulting from each activity is modelled as a lognormal function and the resulting velocity is the difference of the two.

$$\begin{aligned} V_\sigma(t) &= V_{\sigma_1}(t) - V_{\sigma_2}(t) \\ V_{\sigma_i}(t) &= \frac{D_i}{\sqrt{2\pi}\sigma_i(t-t_0)} \; e^{-\frac{1}{2\sigma_i^2}(ln(t-t_0)-\mu_i)^2}, \quad i = 1, 2, \end{aligned} \tag{3.7}$$

where $D_1$ and $D_2$ are the amplitudes of the two activities, and $t_0$ is the start time of the activities. The parameters $\mu_i$ and $\sigma_i$ characterize the speed of the activities. In addition to the above 7 parameters, the total curvature $C_0$ and the angle of incidence of the stroke $\theta_0$, completely characterizes a stroke. The curvature of a stroke is assumed to be a constant, resulting in the following expression for the tangent of a curve.

$$\theta_i \;=\; \theta_{0(i)} \;+\; C_0 \int_{t_{0(i)}}^{t} V_{\sigma(i)}(t)\, dt. \tag{3.8}$$

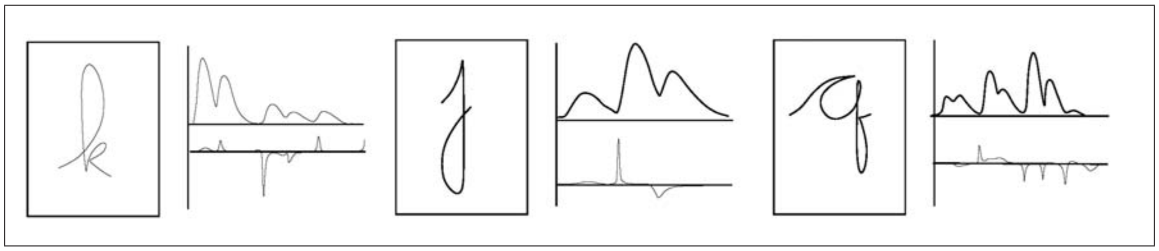


Figure 3.2: Letters generated by the Delta LogNormal model: Typical characters with their corresponding curvilinear and angular velocities [39].

Figure 3.2 illustrates the characters $k$, $j$, and $g$, generated by the Delta LogNormal model along with their corresponding curvilinear and angular velocities [39]. The velocity of the curve at the join between two strokes is computed as the norm of the sum of the individual velocity vectors. The model works well for modelling character classes and helps in segmentation of curves into physiologically meaningful strokes. However, it throws away a significant amount of user-specific variations in handwriting. Moreover, the applicability of the model to drawing strokes is not demonstrated. Hence the model is not appropriate for representing on-line data in general.

### 3.1.4 Graphical Models

Graphical models represent the shape of a curve as a function of a set of free parameters (usually one free parameter for planar curves). There are different ways to represent a function:

1. Explicit Functions: The functions are of the form

$$y \;=\; f(x).$$

The main disadvantages of using explicit functions to represent curves are that there can be only one y for a given value of x and that you cannot have vertical tangents to the curves.

2. Implicit functions: Implicit functions are of the form:

$$f(x, y) \;=\; 0.$$

   Implicit functions overcome the disadvantages mentioned above with explicit function forms. However, they have multiple solutions and it is difficult to determine consecutive points on a curve, which excludes the possibility of representing temporal information. In addition, the tangent direction is difficult to estimate.

3. Parametric functions: A third class of functions is the parametric form. There functions are of the form:

$$x(t) = f(t); y(t) = g(t),$$

   where t is the parameter that is used to compute the points on the curve $(x(t), y(t))$.

   It is relatively easy to estimate tangent and curvature at a point in the parametric representation. In addition, it is simpler to split a parametric curve at a point $t$, and there are no ambiguities as in the case of explicit functions. However, it is difficult to select the appropriate parameter to represent a particular family of curves.

There are different parametric models available to represent curves. However, the class of functions called *splines* [23] offer definite advantages for representing handwriting. In the following section, we examine different spline models, and describe our choice for modelling on-line data.

Table 3.1: Properties of spline functions.

| Spline Model | Properties |
|---|---|
| Catmull-Rom Spline | * Passes through all the data points<br>* $C^1$ is continuous, but $C^2$ is not continuous<br>* Curve can oscillate beyond control points |
| Interpolating (Cubic) Spline | * Smoothest curve<br>* Passes through all the data points |
| Approximating (Cubic) Spline | * Can control smoothness<br>* Knots are points of division along the curve<br>* Curve need not pass through all knots additional control points between knots<br>* Efficient to compute<br>* Local control |

## 3.2 Splines for Handwriting

Splines form a class of parametric functions, which are defined and computed in different ways [25, 132]. We first examine some of the commonly used spline models to decide on a particular variety of splines.

### 3.2.1 Spline Representation

A parametric function that approximates a given set of data points $p_t,\ t = 1 \cdots n$ can be written as, $p_t = f(t) + \epsilon_t$, where $\epsilon_t$ is the error term. Any model for the function $f(t)$, imposes a set of constraints on the nature of $f(t)$ and $\epsilon_t$. Splines impose the smoothness constraint, which minimizes the integral of the squared second derivative along the curve, i.e.,

$$\int_a^b s''(t)^2 dt \ \leq \ \int_a^b g''(t)^2 dt,$$

where $s(t)$ is the spline function, $g(t)$ is the set of all functions that have continuous second derivatives, and $a$ and $b$ are the end points of the curve. Depending on the additional constraints that we impose on the function, we get different spline models. Table 3.1 presents the properties of the commonly used spline functions.

Approximating splines allow for non-zero values for the error of approximation ($\epsilon$) at the data points. They offer the best compromise between computational efficiency and flexibility for our application. We will explore this class of splines in more detail here, since it is useful to our problem of modelling handwritten data. B-splines represent a spline curve using a set of basis functions, which is defined based on the degree of the curve [25]. A curve $S(t)$ is represented as a linear combination of the basis functions.

$$S(t) = \sum_{i=1}^{n} c_i.N_{i,k}(t), \;\; t_{min} \leq t < t_{max}, \tag{3.9}$$

where $N_{i,k}$ is the $i^{th}$ basis function of order $k$ (degree of the curve is $k-1$), $c_i, i = 1..n$ are the coefficients, and $[t_{min}, t_{max})$ represents the support of the basis function. Note that the function is defined piece-wise, and is continuous. Further, the basis functions ensure continuity up to the $(k-2)^{th}$ derivative.

The basis functions are defined recursively as follows:

$$\begin{aligned} N_{i,1}(t) &= \begin{cases} 1, & t_i < t < t_{i+1} \\ 0, & otherwise \end{cases} \\ N_{i,k}(t) &= \frac{t - t_i}{t_{i+k-1} - t_i} N_{i,k-1}(t) + \frac{t_{i+k} - t}{t_{i+k} - t_{i+1}} N_{i+1,k-1}(t). \end{aligned} \tag{3.10}$$

One can manipulate a B-spline curve by changing the number of control points, moving the control points, changing the knot vector and changing the order of the basis function.

### 3.2.2 Computing the Spline Model

To model handwritten data using B-splines, we need to determine the following parameters:

1. Order of the basis functions

2. The number and position of control points, and

3. The position of knot vectors.

The control points in our application are given directly by the handwriting digitizer, which provides a sequence of $x$ and $y$ positions of the pen tip. The following subsections describe the selection of each of the parameters mentioned above.

**Order of Basis**

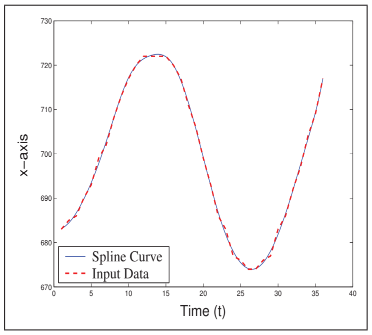
The choice of order of the basis functions is the result of continuity constraints placed on the derivatives. Since the curve is generated by force exerted by the muscles, the acceleration is always finite. In mathematical terms, this ensures continuity of the first derivative. Further, if we wish to impose a smoothness constraint on the curve, we need the second derivatives also to be continuous. The lowest order basis that satisfies these constraints is $3$, which forms the cubic B-spline basis. Note that the smoothness constraint is not always satisfied for handwritten curves (e.g., the sharp change in pen direction in the digit $3$ in Figure 3.4(c).) However, the functions $f_x(t)$ and $f_y(t)$ (Figures 3.4(a) and (b)), which are functions of time, will always be smooth due to the limit on force applied to the pen.

**Knot Vectors**

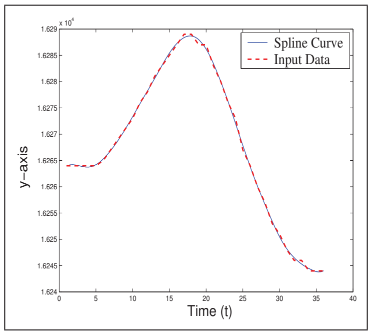
The choice of knot vectors will define the nature of the resulting spline. We select an open uniform knot vector sequence, where the distance between two knot vectors is kept constant. This ensures that all parts of the stroke are given equal importance (in absence of evidence to the contrary). The only parameter that needs to be selected is the inter-knot distance, which will decide the smoothness of the curve. The open uniform knot vector is of the form: $[1\ 1\ 1\ 1d\ 2d\ 3d\ ...\ kd = n\ n\ n\ n]$. The extreme points are repeated as many times as the order of the basis functions.

We notice from the above discussion that the order of the basis functions and the position of knots (in the knot vector) are inter-related in the context representing a specific curve. This is because, if we define the knots to be closer, the curve in between would be simpler (linear as two knots approach each other). When the knots are father apart, the curve between them is potentially more complex and we need higher order curves to approximate the data. Conversely, if the order of the curve that we need to fit between two knots is $d$, the number of control points between any two knots should be atleast $d - 1$. In addition, the choice of $d$ is affected by the sampling rate of the digitizing device. In our experiments with the CrossPad, with a sampling rate of $128$ samples per second, selecting every third sample as a knot vector seems to give good approximation. Note that this is the smallest value of $d$ for cubic splines.

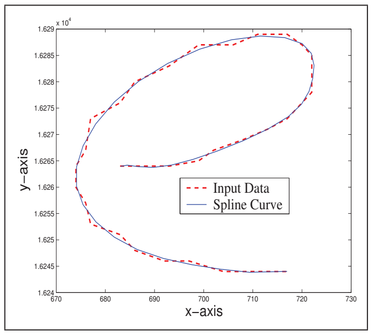
Figure 3.3 shows examples of two handwritten characters, *e* (Figures 3.3(a)-(c)) and *n* (Figures 3.3(d)-(f)) and the corresponding spline curves. These spline curves clearly demonstrate their ability to be tolerant to noise in the input data, while closely approximating the shape of the characters.



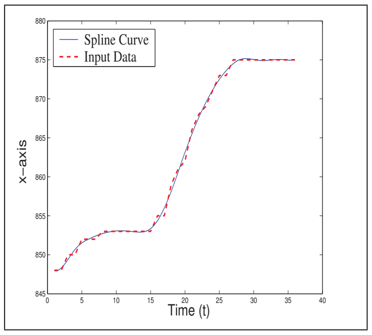
(a)



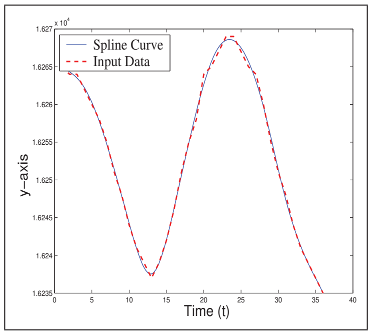
(b)



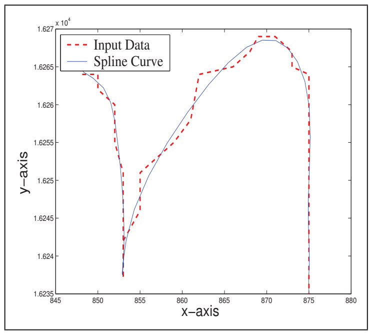
(c)



(d)



(e)



(f)

Figure 3.3: Noise removal by splines: Figures (a) through (c) and (d) through (f) show examples of two characters, *e* and *n*, respectively, where the spline curves closely approximate the shape of the characters while reducing the noise in the input data. The input data points are shown as dotted lines and the spline curves are shown as solid lines.

Figure 3.4 shows the example of a spline curve fitted to a stroke forming the digit 3. We notice that the curve resulting from the spline fit (Figure 3.4(c)) successfully captures the abrupt change in the stroke direction in the middle of the curve, while preserving the smoothness of the curve at other regions.

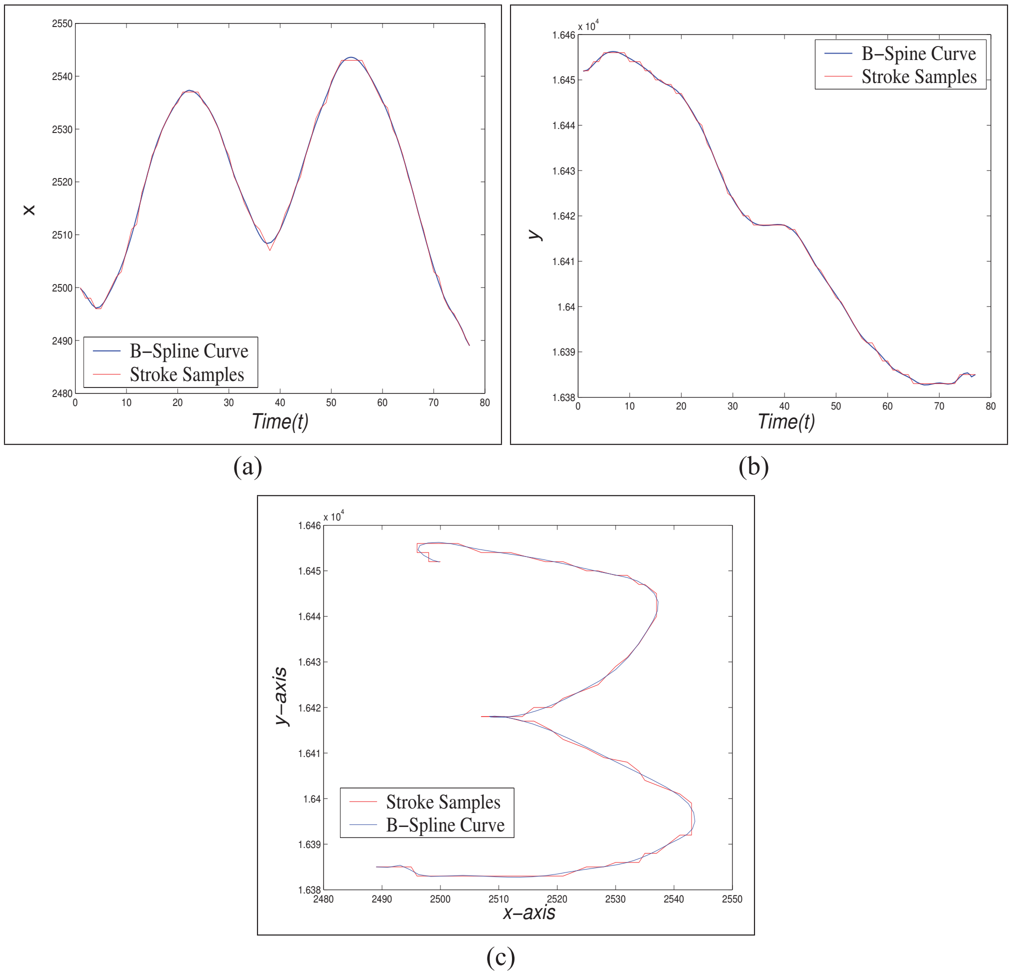


Figure 3.4: Spline approximation of a stroke: (a) and (b) shows the spline curves fitted to the $x$ and $y$ point sequences, respectively. (c) shows the resulting curve overlaid on the input digit '3'. The input data points are shown as dotted lines and the spline curves are shown as solid lines.

In spite of the fact that the model is based on the mechanics of the handwriting process, the process of fitting the model to noisy data could introduce some artifacts in the resulting spline representation. The most common artifact is the oscillation of the spline curve around the data samples in case of abrupt changes in the input curve, which is usually the result of spurious noise. Figure 3.5 shows an example of a spline curve fitted to a handwritten stroke that is corrupted by spurious noise. We note that the curve resulting from the spline fit oscillates around the input curve near the noisy data. Such oscillations are often referred to as ‘ringing artifacts’.

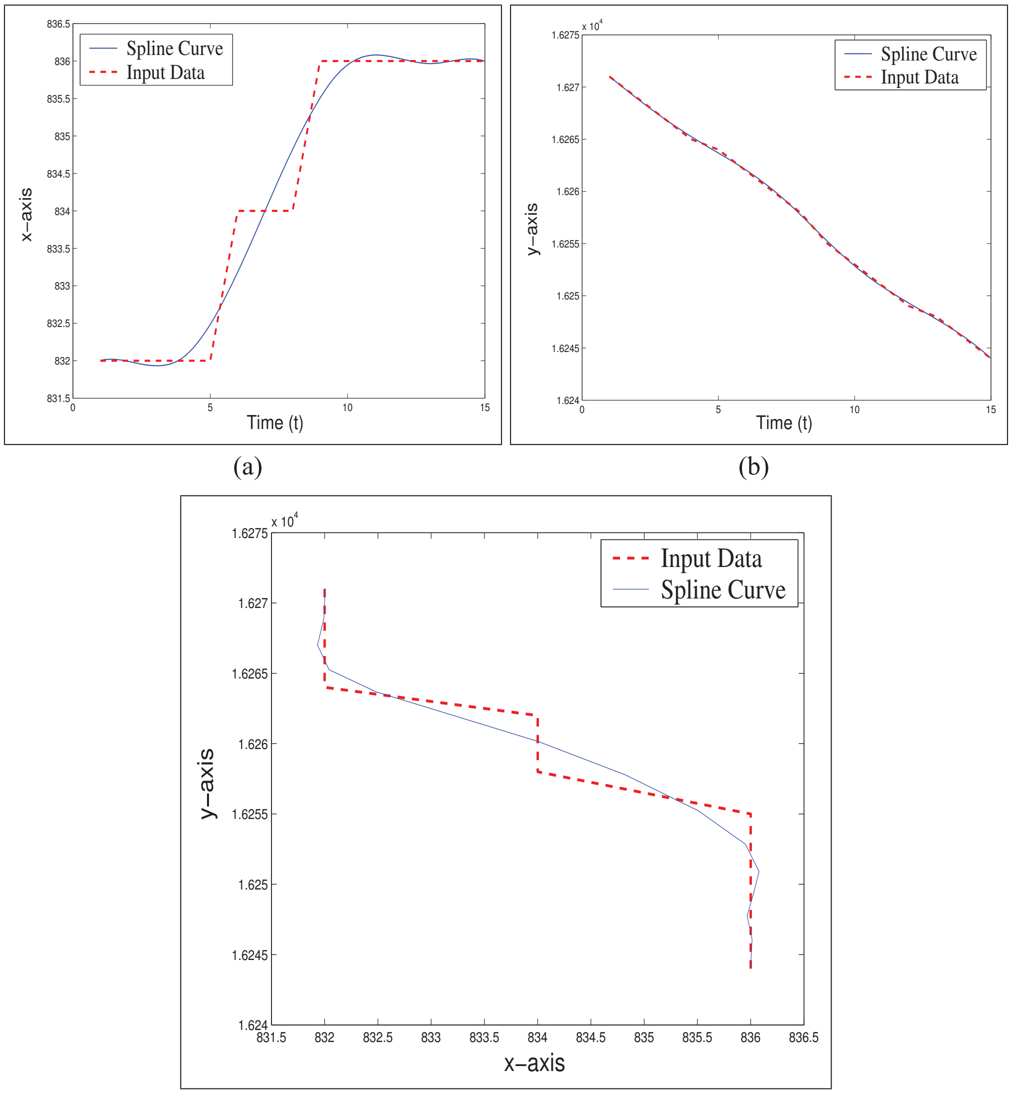


Figure 3.5: Ringing artifacts in spline fitting. (a) and (b) show the spline curves fitted to the $x$ and $y$ point sequences, respectively of a curve and (c) shows the resulting curve overlaid on the input. Note that the spline curve oscillates about the input curve towards its end points. The input data points are shown as dotted lines and the spline curves are shown as solid lines.

Figure 3.6 shows four additional examples of spline approximations to handwritten

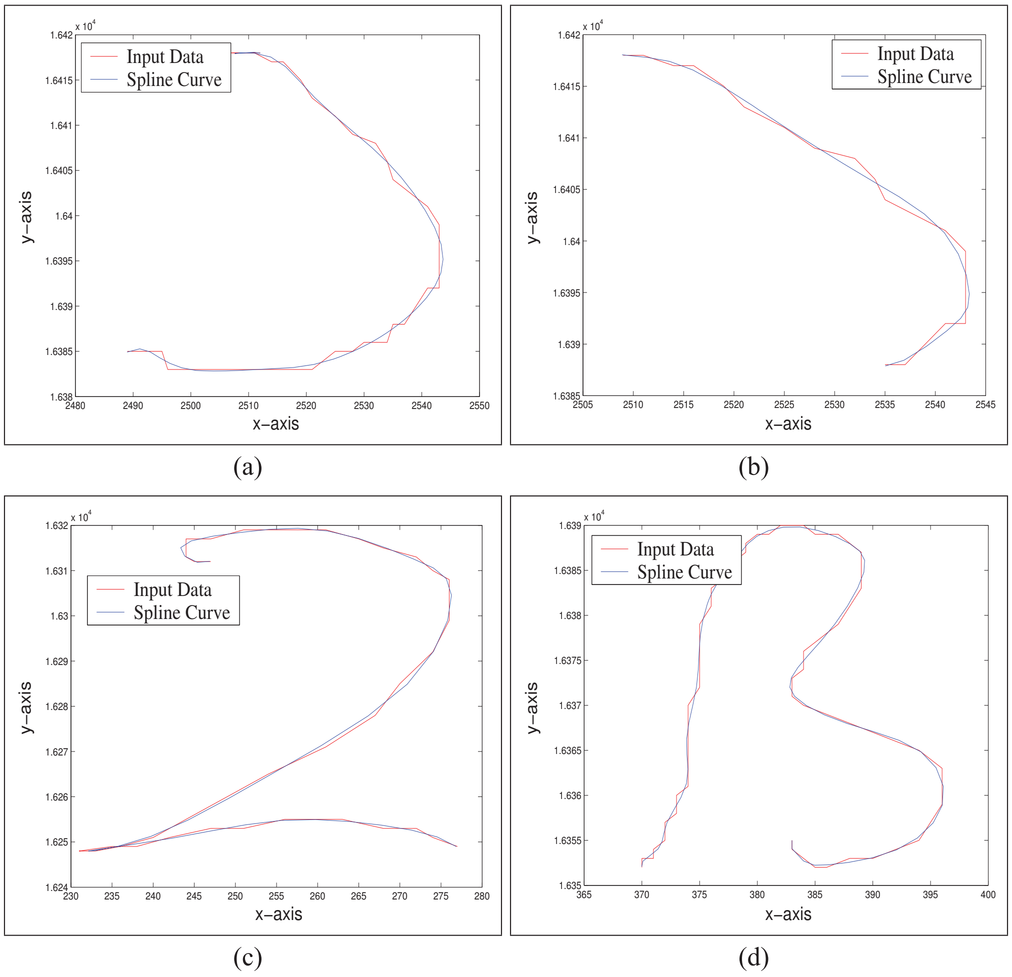


Figure 3.6: Examples of spline approximation of four strokes. The input data points are shown as dotted lines and the spline curves are shown as solid lines.

strokes. The average error of the approximation function (spline curve) on a typical document was about $1.6\%$ of the stroke size (height). We now take a closer look at the nature of the approximation error for the spline curve and its relationship to the digitizer noise.

### 3.2.3 Properties of Splines

We have described several desirable properties of a stroke model in Section 3.1. We will now examine some of the properties of the spline model described above.

The existence and uniqueness of B-spline coefficients for a specific curve, given a uniform knot vector and the order $k$, is guaranteed by the Schoenberg-Whitney theorem

[25]. The basis functions have compact support, which provides local control to manipulate curves represented as splines. In addition, the transformation of a stroke to the spline-based representation has the advantage of being affine invariant. This means that the function $F(stroke)$, which transforms a stroke from the point-based representation to its spline-based representation, is commutative with any affine transformation $A()$. In other words:

$$F(A(stroke)) = A(F(stroke)). \tag{3.11}$$

The result of the function $F()$ is a set of coefficients of the spline basis functions. This property helps us to compute the result of an affine transformation to a stroke directly from it's spline-based representation. Note that this is different from an affine invariant representation, where the parameters of the representation of a curve remains unchanged when an affine transformation is applied to the handwritten curve.

### 3.2.4 Approximation Error and Noise

The digitization process introduces noise into the handwriting curve, which comes from a variety of sources. These include electromagnetic interference, errors in pen position detection, errors in analog to digital conversion, etc. An ideal model for handwriting should be able to fit the digitized data in such a way that the noise is eliminated in the curve represented by the model. However, this is an extremely difficult task as there is no easy way to characterize the exact nature of the noise or the signal. We used the mechanics of the handwriting process to define the spline model, which should ensure that the model is able to fit all handwritten data. Any error in approximation should solely result from noise in the data. We try to verify this assertion by comparing the approximation error against a noise model, which is described below.

We consider three different types of noise that are present in most digitizing devices. These include a white noise that is introduced by the electronic components in the digitizing device such as amplifiers, position sensors, etc. The nature of this type of noise is easy to characterize and is often well approximated by a zero-mean normal distribution. The second source of noise, which is the most difficult to characterize, is the error that is introduced by the analog-to-digital (A/D) converters that are present in the digitizing tablets. The errors are mostly due to the approximation algorithms used by the A/D converters. An interesting nature of this error is that it occurs intermittently, and is dependent on the specific digitizer and sometimes on the velocity and curvature at that point. We model the noise as a gaussian noise that is added to a fraction of the samples provided by the digitizer. The third source of noise is the quantization process that introduces a uniformly distributed noise, $U(-0.5, 0.5)$, in the data. However, this noise is not independent of the previous two noise sources, as the quantization error is dependent on the value of the input signal, which includes the different noise types described earlier.

In order to compare the noise model against the approximation error, we need to compute the distribution of the resulting noise. However, there is no closed form expression to the distribution of the resulting noise, which involves a gaussian noise, and a non-independent uniformly distributed quantization noise. Hence we compute an empirical distribution by simulating the three noise processes on a random signal. The signal itself is uniformly distributed ($U(0, 1)$) to avoid any bias to the quantization process. We generate $50,000$ data samples and add a white noise of variance $\sigma_1^2$ to each data sample. Further a gaussian noise of variance $\sigma_2^2$ is added to a fraction, $k$ of the resulting data. The noisy data is then rounded to the nearest integer to simulate the quantization process. The values of $\sigma_1, \sigma_2$ and $k$ are dependent on the specific digitizer and data samples.

To compare the approximation error against the noise, we plot the distribution of the approximation error of the data (called observed error) against the simulated noise distribution (called expected error). Figure 3.7 shows the plots of the two distributions. The data comes from samples of text collected using a Tablet PC.

A second experiment is conducted to compare the distributions of errors when the na-

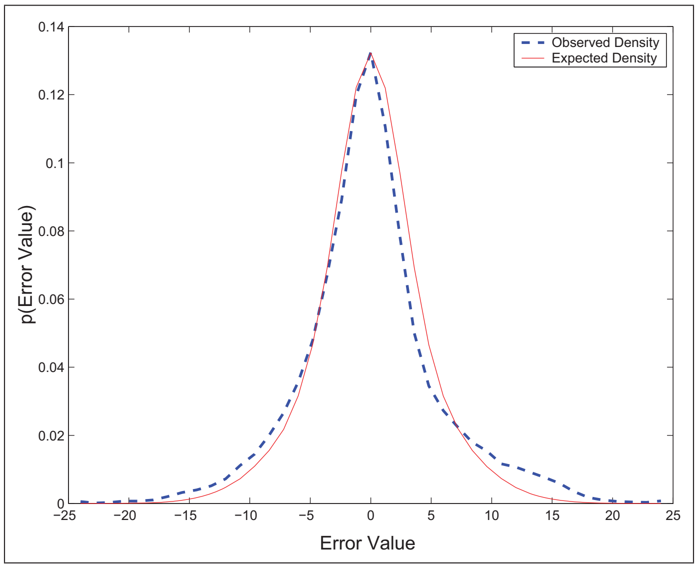


Figure 3.7: Plot of the observed and expected distributions of the approximation error for text data collected using the Tablet PC.

ture of data changes. We note that the noise introduced by the A/D converter is dependent on the data; the variance of the noise increases for strokes with high curvatures in it. Figure 3.8 shows a comparison of the the error distributions, plotted against the corresponding simulated distribution for text and non-text data. We notice that the distribution of the observed error approximates the expected distribution very well.

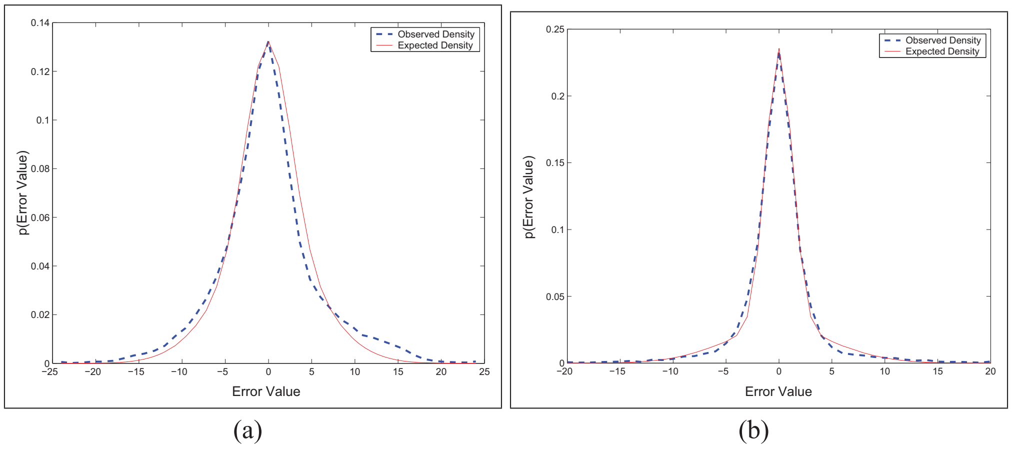


Figure 3.8: Comparison of error distributions for (a) text and (b) non-text data collected using the Tablet PC (Toshiba Protege 3505).

We also compared the nature of the error for the CrossPad, using text data collected from it. Tablet PC has a much larger number of electronic components compared to the CrossPad, which introduces noise, and hence the variance of the error distribution of the latter is lower. The fact that the observed distribution has a higher peak around zero might indicate the fact that the spline curve is approximating the data, including the noise, much closer. Figure 3.9 shows the plots of the observed and expected distributions in the case of text data collected from the CrossPad.

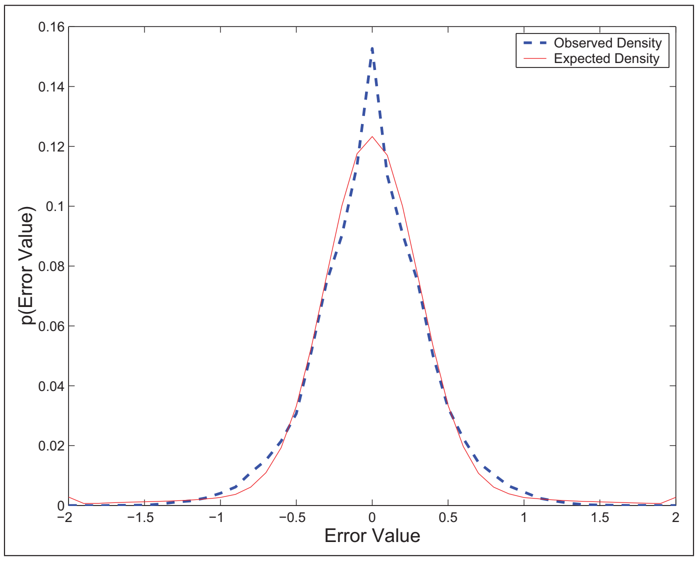


Figure 3.9: Plot of the observed and expected distributions of the approximation error for text data collected from the CrossPad.

We performed a Chi-squared goodness-of-fit test to determine if the observed noise in data fits the theoretical distribution. The hypothesis test has a null hypothesis, $H_0$: The observed noise follow the simulated distribution, and an alternate hypothesis, $H_a$: The noise do not follow the simulated distribution. To perform the test, we divided the data into $k(= 20)$ bins and compute the chi-squared statistic:

$$\chi^2 = \sum_{i=1}^{k} \frac{(O_i - E_i)^2}{E_i}, \tag{3.12}$$

where $O_i$ is the observed frequency of bin $i$ and $E_i$ is the frequency of bin $i$ in the simulated distribution. If the computed statistic is greater than $\chi^2_{(\alpha,k-c)}$, then the null hypothesis is rejected. $\alpha$ is the level of significance level and $c$ is the number of parameters estimated from the data. In our experiments, the degrees of freedom, $k - c$ is $18$, since the variance of two normal distributions are estimated from the data. The chi squared value for $18$ degrees of freedom at a level of significance of $0.99$ is approximately $7.02$ and the value computed according to Equation 3.12 was around $800$. This rejects the hypothesis that the observed distribution strictly follows the simulated noise distribution. However, the value of the computed statistic is not too high considering the number of samples (approximately $50,000$) and the difference is distribution could be primarily attributed to the noise model of A/D converter and other noise sources that are not considered.

To summarize the experiments, we notice that the approximation error follows a distribution that is very close to that of the noise model that we described. This is not a conclusive proof that the approximation process eliminates noise and only noise. However, since the restrictions on our model are primarily defined based on the handwriting process, we have reasons to believe that the approximation error is primarily composed of noise introduced by the digitizer. In other words, the spline model closely approximates the uncorrupted handwritten data.

## 3.3 Computing Stroke properties

Once the strokes are approximated using a spline, we can reliably compute different properties of the stroke. We will now derive the formulas for some of the common properties of strokes that need to be computed.

- **Stroke Length:** The stroke length is defined as the total length of the curve represented by the stroke. Let the stroke be defined as $S(t) = (S_x(t), S_y(t))$. The length of the stroke between two points, $t_i$ and $t_j$ is the arc-length integral of the stroke, given by:

$$L(t_i, t_j) = \int_{t_i}^{t_j} ((S_x^{'}(t))^2 + (S_y^{'}(t))^2)^{1/2} \ dt.$$

However, there is no closed form solution for the above integral. A numerical solution is often adopted to compute this integral. In the case of handwriting, we take advantage of the relationship between writing speed and stroke curvature to compute this integral efficiently. Since the writing process slows down at points of high curvature, we can approximate the curve between any two sample points using a straight line. With this assumption, we can convert the integral into the following sum.

$$\sum_{i=1}^{n} L_1(t_i, t_{i+1}),$$

where $L_1$ is the euclidean distance between points $t_i$ and $t_{i+1}$.

- **Stroke Curvature:** The curvature $k$ of a stroke $S$, represented using the arc-length parametrization $s$, is the magnitude of the curvature vector $k(s)$, given by:

$$k(s) = \frac{d^2 S(s)}{ds^2}. \tag{3.13}$$

The curvature vector is perpendicular to the tangent vector at the point $s$ [83]. However, this definition of curvature is difficult to use in computations since the second derivative in Equation (3.13) is not defined for points on a stroke that are not smooth. Hence we use an alternate property as a measure of the curvature of the stroke.

Intuitively, the curvature is a measure of deviation from linearity of the stroke at a point. Hence we use the derivative of tangent directions, $\Theta(t_i)$, instead of the second derivative in Equation (3.13).

$$k_1(s) = \frac{d}{ds}\left(\arctan(S'_y(s)/S'_x(s))\right). \tag{3.14}$$

Note that $k_1(s)$ is still not defined for points that are not smooth in a stroke. However, we can compute the total curvature of a stroke, which is given by the limiting sum:

$$C(t_i, t_j) = \lim_{\delta t \to 0} \sum_{k=0}^{(t_j - t_i)/\delta_t} |\Theta_{t_i+(k+1)dt} - \Theta_{t_i+k.dt}|, \tag{3.15}$$

where $\Theta(t)$ is the *slope* of the curve (computed as the angle with respect to the $x$ axis) at time $t$, given by:

$$\Theta(t) = \arctan(S'_y(t)/S'_x(t)).$$

The change in parameter from arc-length $s$ in Equation (3.14) to time $t$ in Equation (3.15) does not affect the value of $C()$. Further, we can approximate the limit by the sum:

$$C(t_i, t_j) = \sum_{k=i}^{j-1} |\Theta_{t_{k+1}} - \Theta_{t_k}|.$$

The approximation is exact under the assumption that the second derivative of the curve has no zero crossings between $t_i$ and $t_j$. The derivative of a spline function of order $n$ is another spline of order $n-1$. The derivative of $S(t)$ is given by:

$$\frac{d}{dt}S(t) = S'(t) = \sum_{i=0}^{n-1} N_{i,p-1}(t)K_i,$$

where $K_i$ is given by:

$$K_i = \frac{p}{u_{i+p+1} - u_{i+1}}(P_{i+1} - P_i)$$

- **Centroid:** The centroid, $M(S)$ of a stroke is computed as the integral:

$$M(S) = \frac{1}{L(t_1, t_n)} \int_{t_1}^{t_n} S(t)\ dt.$$

We need to adopt a numerical solution for the integration as there is no closed form solution to the integral. However, there is an efficient way to compute an approximate value of this integral, which is given by the sum:

$$\frac{1}{L(t_1, t_n)} \sum_{i=1}^{n} S\left(\frac{t_i + t_{i+1}}{2}\right) \, . \, L_1(t_i, t_{i+1}),$$

where $L()$ and $L_1()$ are defined above.

Once again the summation approximates the stroke as a piecewise linear curve. The approximation is valid due to the inverse relationship between the stroke speed and curvature.

## 3.4 File Formats for On-line Data

We have been discussing the representation of handwritten data in memory for the purpose of computing stroke properties. In this section, we take a look at the common file formats used to represent the data collected from a digitizing tablet.

The CrossPad uses two different representations; a binary format and an ASCII format for the data it transfers to a computer. The work in this thesis uses the ASCII format from the CrossPad or IBM Thinkpad Transnote, often referred to as the *Rcard* format. A second format, which is becoming an industry standard, is the InkML format [36]. The two formats are briefly described below.

The *Rcard* format represents a document in a file as a set of regions. Each region can represent a single page or a part of it. The regions are designated by a '*=R*' marker at the beginning of each region. This is followed by the number of strokes in that region and a set of device tags. A '*=T*' tag accompanies every '*=R*' tag, which is used to assign a name for the region. Each stroke begins on a new line and is marked by a '*=S*' tag, which also specifies the number of sample points in the stroke. This is followed by the pairs of $x$ and $y$ coordinates of the strokes. Figure 3.10 (a) shows part of a Rcard file.

The Tablet PC uses a proprietary binary representation for the data it collects in a file.

The format is not publicly available and hence we do not use the Tablet PC format for storage of handwritten data. The information from the stokes are collected during the process of writing and are stored in the Rcard format, whenever needed.

The InkML representation is adopted as an open standard by the W3C [36], and the first working draft was published in August 2003 [126]. The *trace* tags denote the strokes in the InkML format, which encloses a sequence of $(x, y)$ co-ordinate pairs of the sample points in a stroke. The regions are formed by grouping traces using the *traceGroup* tag. There are several tags that can be used to represent the device properties, author details, pen color, etc. Details of the representation can be found in the working draft of the format [126]. Figure 3.10 (b) shows a simple example of the InkML file format.

```
=R 2 999 0 0 210 C A
=T 29 This is a test page with text
=S 18 TN BE 0 0 0 0 0 0 0 13
  1069 16525 1067 16525 1069 16525 1069 16523
  1067 16522 1067 16520 1066 16517 1064 16513
  1062 16508 1059 16501 1055 16492 1050 16485
  1045 16476 1039 16466 1034 16455 1029 16445
  1024 16438 1020 16431
=S 23 TN BE 0 0 0 0 0 0 0 14
  1046 16522 1046 16525 1046 16529 1048 16531
  1052 16532 1055 16536 1059 16537 1064 16539
  1071 16539 1076 16539 1080 16537 1083 16536
  1088 16534 1090 16531 1092 16527 1092 16523
  1092 16520 1090 16517 1087 16515 1083 16511
  1080 16508 1074 16506 1071 16506
```

(a)

```
<ink>
  <trace>
    10 0 9 14 8 28 7 42 6 56 6 70 8
    84 8 98 8 112 9 126 10 140 13
    154 14 168 17 182 18 188 23 174
    30 160 38 147 49 135 58 124 72
    121 77 135 80 149 82 163 84 177
    87 191 93 205
  </trace>
  <trace>
    227 50 226 64 225 78 227 92 228
    106 228 120 229 134 230 148 234
    162 235 176 238 190 241 204
  </trace>
  <trace>
    366 130 359 143 354 157 349 171
    352 185 359 197 371 204 385 205
    398 202 408 191 413 177 413 163
    405 150 392 143 378 141 365 150
  </trace>
</ink>
```

(b)

Figure 3.10: Examples of (a) Rcard and (b) InkML representations of on-line data.

## 3.5 Document Database

The data used in this thesis was collected using the *CrossPad*. The pen position is sampled at a constant rate of $128$ samples per second and the device has a resolution of $250$ $dpi$ along the $x$ and $y$ axes [1]. Part of the data was collected on ruled paper with an inter-line distance of $8.75mm$ although some writers wrote on alternate lines. We must point out that the actual device for data collection is not important as long as it can generate a temporal sequence of $x$ and $y$ positions of the pen tip. Most of the data used in the document segmentation work was collected by the Pen Computing group at IBM T.J. Watson Research Center.

For the script identification work, the users were asked to write one page of text in a particular script, with each page containing approximately $20$ lines of text. No restriction was imposed on the content or style of writing. The details of the database used for this work are given in Table 6.9. Multiple pages from the same writer were collected at different times. Appendix B shows some examples of the documents in the database used in various experiments reported in this thesis.

## 3.6 Summary

In this chapter we have developed a generic representation for on-line handwritten data using cubic splines. We have reviewed the popular models currently used for on-line handwriting and explained the advantages and drawbacks of each. The suitability of the model is demonstrated based on the assumptions on hand motion and experimental results are provided to support the assertion. We derived formulas for some of the commonly used stroke properties, based on the spline representation. We have also described the commonly used file formats for on-line data.

There are no assumptions made regarding the nature of the data being written or drawn in defining the model. This makes the model suitable for a variety of applications such

[1] Users have reported that the actual resolution is only about half of this value [123].

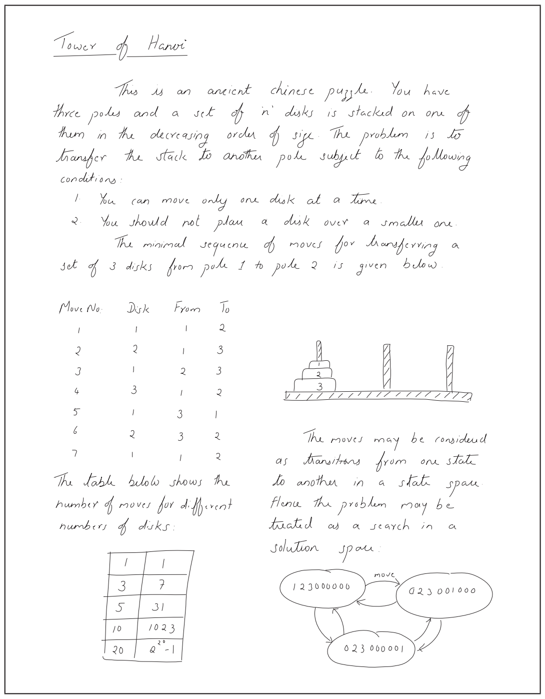
as handwriting recognition, writer identification, script or language identification, stroke matching, handwritten data compression, etc.

# Chapter 4

# Structure Identification

The problem of segmenting document pages into homogeneous regions containing unique semantic entities is of prime importance in automatic document understanding systems [94]. Several algorithms exist that recognize printed or handwritten text. However, most of these algorithms assume that the input is a plain text and the text lines and words in the text have been properly identified and segmented by a preprocessor. A typical handwritten document page may contain several regions of interest such as underlined keywords, different types of tables, diagrams, sketches and text (see Figure 4.1(a)). The main task of a segmentation algorithm is to identify contiguous regions of text, graphics and tables in such a document for document understanding and retrieval based on semantic entities. Figure 4.1(b) shows the desired output of a document understanding system after segmentation and full transcription of the handwritten document.

Different approaches to segmentation of document images were mentioned in Section 2.2. The top-down approaches are more suitable for printed documents such as journal papers, where documents are well structured, and the prior knowledge can be used in identifying higher level structures. In the case of handwritten documents, especially on-line documents, a bottom-up approach is more appropriate due to the large variations in the layout of the documents. In addition, the lower level primitives of on-line documents (strokes)



Tower of Hanoi

This is an ancient chinese puzzle. You have three poles and a set of 'n' disks is stacked on one of them in the decreasing order of size. The problem is to transfer the stack to another pole subject to the following conditions:

1 You can move only one disk at a time

2 You should not place a disk over a smaller one

The minimal sequence of moves for transferring a set of 3 disks from pole 1 to pole 2 is given below

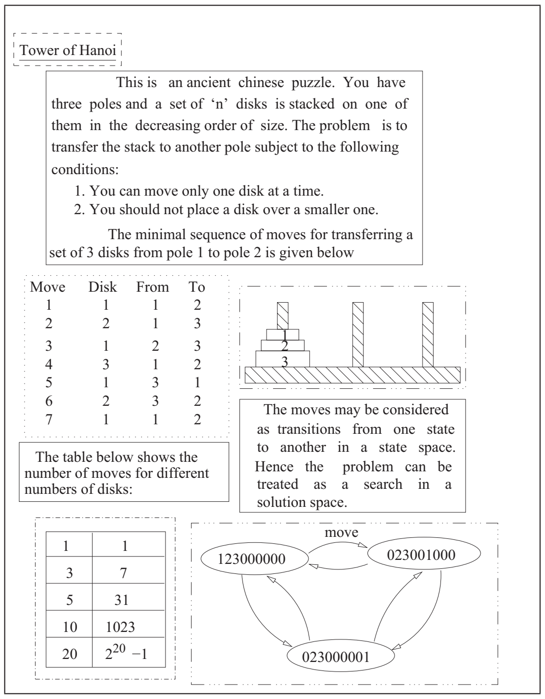
| Move No | Disk | From | To |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 2 | 2 | 1 | 3 |
| 3 | 1 | 2 | 3 |
| 4 | 3 | 1 | 2 |
| 5 | 1 | 3 | 1 |
| 6 | 2 | 3 | 2 |
| 7 | 1 | 1 | 2 |

The table below shows the number of moves for different numbers of disks:

| | |
|---|---|
| 1 | 1 |
| 3 | 7 |
| 5 | 31 |
| 10 | 1023 |
| 20 | $2^{20}-1$ |

The moves may be considered as transitions from one state to another in a state space. Hence the problem may be treated as a search in a solution space.

(a)



Tower of Hanoi

This is an ancient chinese puzzle. You have three poles and a set of 'n' disks is stacked on one of them in the decreasing order of size. The problem is to transfer the stack to another pole subject to the following conditions:

1. You can move only one disk at a time.
2. You should not place a disk over a smaller one.

The minimal sequence of moves for transferring a set of 3 disks from pole 1 to pole 2 is given below

| Move | Disk | From | To |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 2 | 2 | 1 | 3 |
| 3 | 1 | 2 | 3 |
| 4 | 3 | 1 | 2 |
| 5 | 1 | 3 | 1 |
| 6 | 2 | 3 | 2 |
| 7 | 1 | 1 | 2 |

The table below shows the number of moves for different numbers of disks:

| | |
|---|---|
| 1 | 1 |
| 3 | 7 |
| 5 | 31 |
| 10 | 1023 |
| 20 | $2^{20}-1$ |

The moves may be considered as transitions from one state to another in a state space. Hence the problem can be treated as a search in a solution space.

(b)

Figure 4.1: Segmentation of on-line documents: (a) input and (b) desired output.

are easily identified due to the nature of data collection, which records the data as a set of strokes.

We adopt a hierarchical, bottom-up approach to analyze on-line documents (see Figure 4.2). First, the individual strokes are classified as text or non-text strokes. The non-text strokes are then grouped into homogeneous regions based on their proximity to identify ruled tables and diagram regions. In the third stage, we focus on the (supervised) classification of tables, text, diagrams, and underlined keywords. In addition to facilitating text recognition, understanding the structure of an on-line document opens up many applications. The spatial relationship of pictures and text in the document may be used to identify captions and labels. Page layout information allows the use of digitizing tablets as an interface for designing web pages. Search and retrieval of documents can be done based on an underlined keyword or a diagram specified by the user as a query.

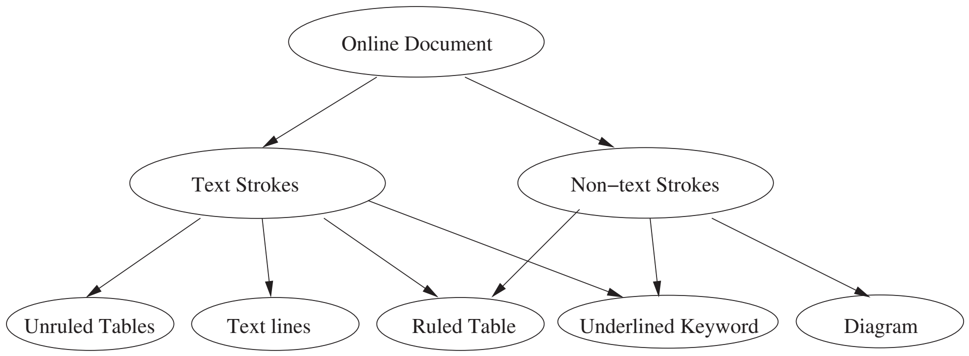


Figure 4.2: Classification of on-line documents.

### 4.0.1 Document Segmentation

The process of document segmentation in on-line documents can be defined as partitioning a set of $n$ strokes (representing a page) into $k$ subsets, $\{R_1, R_2, \cdots, R_k\}$, where $R_i$ represents a region and $|R_1| + |R_2| + \cdots + |R_k| = n$. In this paper, we develop a Stochastic Context Free Grammar (SCFG) based solution that computes an optimal partition of the strokes, given a criterion function. The method assumes that the strokes of a region are contiguous in time and spatially compact. We will define the compactness more precisely in Section 4.2, which forms part of the criterion function. The assumption of temporal contiguity leads to over-segmentation. A post-processing stage is used to combine such fragments into a single region. An overview of the segmentation algorithm is given in Figure 4.3.

The first step is to represent the individual strokes using a stroke model, which helps in the removal of noise in the data as well as computation of stroke properties. These properties are then used to classify the individual strokes as text or non-text. The classification retains the uncertainty in the decision process, which is used by the subsequent stages in defining word and region boundaries. Each of the above steps are described in detail in the following sections.

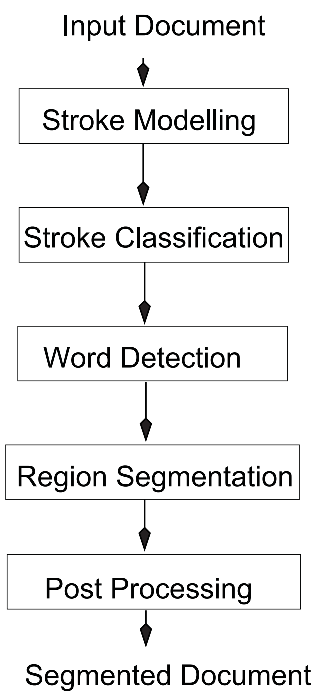


Figure 4.3: Steps of segmentation in the proposed algorithm.

## 4.1 Text versus Non-text Strokes

We use the *Stroke Length* and *Stroke Curvature* as features for classifying individual strokes as text or non-text. Chapter 3 provided an extensive description of computation of the stroke properties from the individual strokes.

A two-dimensional feature space representation of the text and non-text strokes of the document page in Figure 4.1(a) is shown in Figure 4.4. The plot indicates that a linear decision boundary would separate the two classes. We compute the linear decision boundary that minimizes the mean squared error. Assume that the decision boundary is given by $g(x) = 0$.

$$g(x) : w_0 + \sum_{i=1}^{d} w_i x_i, \tag{4.1}$$

where $w_i s$ are the weights associated with each of the $d$ ($d = 2$ in this problem) features. The weights are initialized to random numbers between $0$ and $1$. We define the error $e_x$, for the input pattern $x$ as:

$$e_x = o(x) - g(x), \tag{4.2}$$

where $o(x)$ is $+1$ for text and $-1$ for non-text. The mean squared error of a decision boundary for a set of points is given by:

$$E = \frac{1}{|M|} \sum_{x \in M} e_x^2, \tag{4.3}$$

where $M$ is the set of misclassified samples. The weights are updated by classifying all the training patterns and updating the weights using the rule:

$$w_i(t+1) \leftarrow w_i(t) + \alpha . e_x . x_i, \tag{4.4}$$

where $\alpha$ is the learning rate. The learning rate is reduced slowly over multiple iterations. Figure 4.5 gives some examples of stroke classification.

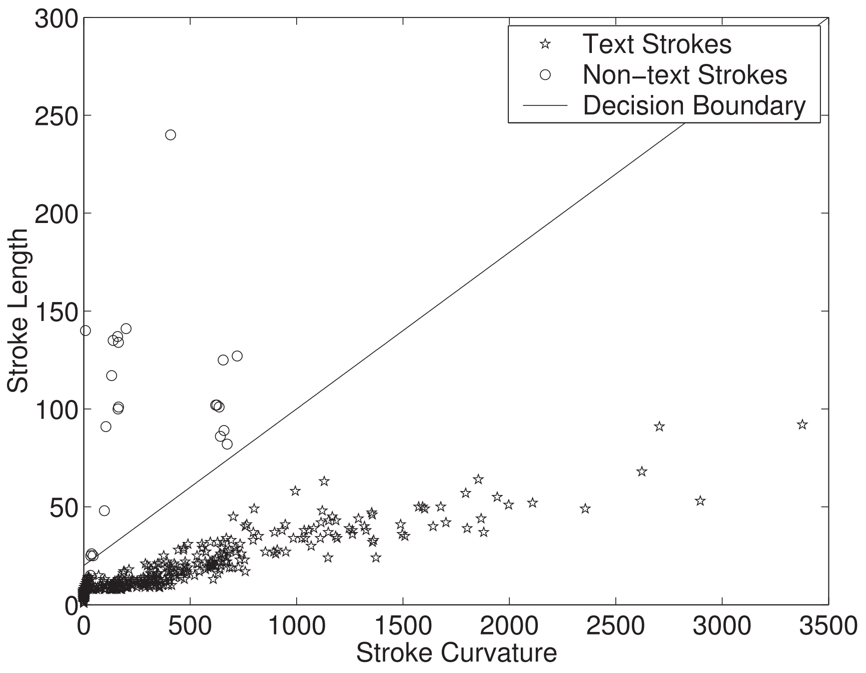
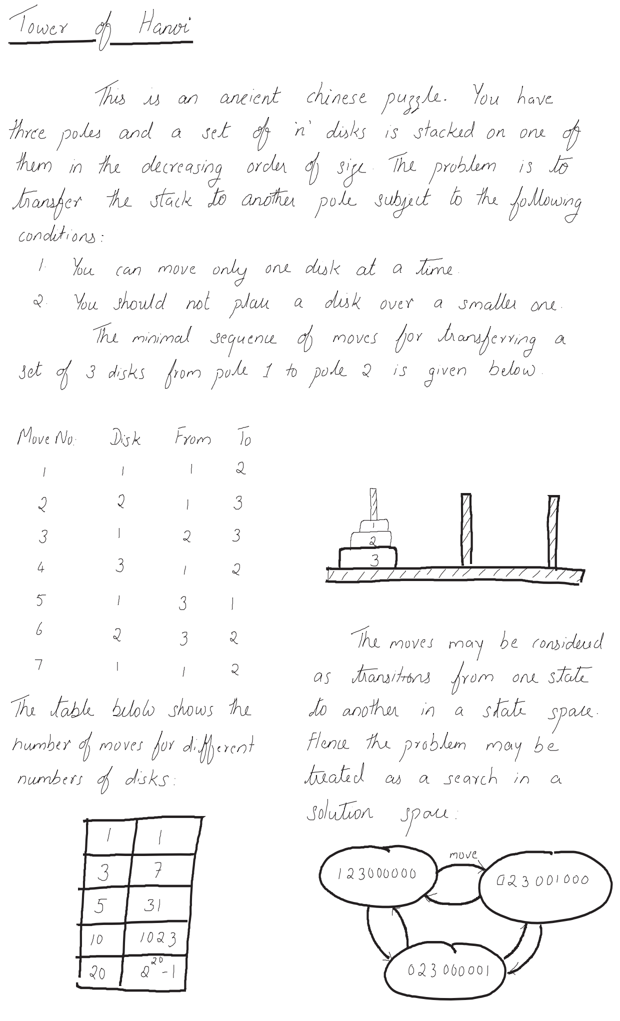


Figure 4.4: Text vs. non-text strokes.



Tower of Hanoi

This is an ancient chinese puzzle. You have three poles and a set of 'n' disks is stacked on one of them in the decreasing order of size. The problem is to transfer the stack to another pole subject to the following conditions:

1. You can move only one disk at a time.
2. You should not place a disk over a smaller one.

The minimal sequence of moves for transferring a set of 3 disks from pole 1 to pole 2 is given below.

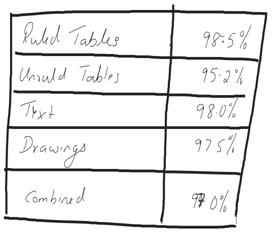
| Move No. | Disk | From | To |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 2 | 2 | 1 | 3 |
| 3 | 1 | 2 | 3 |
| 4 | 3 | 1 | 2 |
| 5 | 1 | 3 | 1 |
| 6 | 2 | 3 | 2 |
| 7 | 1 | 1 | 2 |

The table below shows the number of moves for different numbers of disks:

| | |
|---|---|
| 1 | 1 |
| 3 | 7 |
| 5 | 31 |
| 10 | 1023 |
| 20 | $2^{20}-1$ |

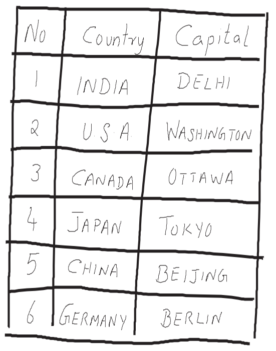
The moves may be considered as transitions from one state to another in a state space. Hence the problem may be treated as a search in a solution space:

(a)



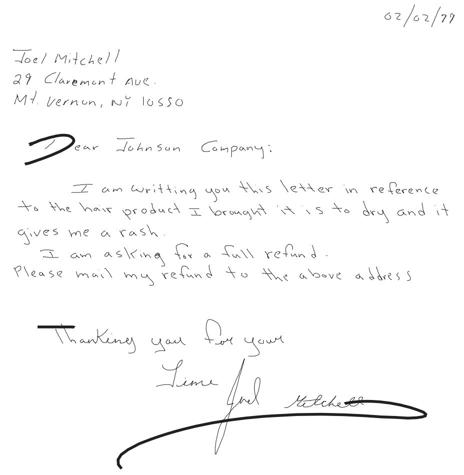
| | |
|---|---|
| Ruled Tables | 98.5% |
| Unruled Tables | 95.2% |
| Text | 98.0% |
| Drawings | 97.5% |
| Combined | 97.0% |

(b)



| No | Country | Capital |
|---|---|---|
| 1 | INDIA | DELHI |
| 2 | U.S.A | WASHINGTON |
| 3 | CANADA | OTTAWA |
| 4 | JAPAN | TOKYO |
| 5 | CHINA | BEIJING |
| 6 | GERMANY | BERLIN |

(c)



02/02/77

Joel Mitchell
29 Claremont Ave.
Mt. Vernon, NY 10550

Dear Johnson Company:

I am writting you this letter in reference to the hair product I brought it is to dry and it gives me a rash.
I am asking for a full refund.
Please mail my refund to the above address

Thanking you for your
Time
Joel Mitchell

(d)

Figure 4.5: Examples of stroke classification. Non-text strokes are shown in bold. In (d) three text strokes are misclassified as non-text strokes because they have large stroke length and/or small curvature.

## 4.2 Word and Region Segmentation

Once the strokes are classified into text and non-text strokes, we group adjacent text strokes into words. The word detection module is used to identify tokens that form the terminals of the SCFG, which is used to describe the page layout. One could define the grammar rules to generate the strokes directly. However, such an approach will increase the complexity of the grammar and affect the parsing performance.

The word detection module itself is a deterministic finite state automaton (FSA), which is augmented to output a confidence value to each word detected. Figure 4.6 shows the structure of the FSA. The node $S_2$ denotes the end state, where a word is detected. Along with the word detection, we compute a confidence value for the assigned label. This is computed as the posterior probability of the label, given a word model. The probability of observing the text class is the product of the probabilities of the individual strokes being text and the probability of orientation of the stroke centers.

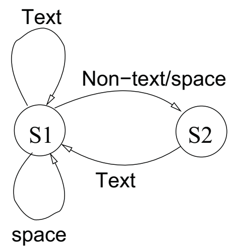


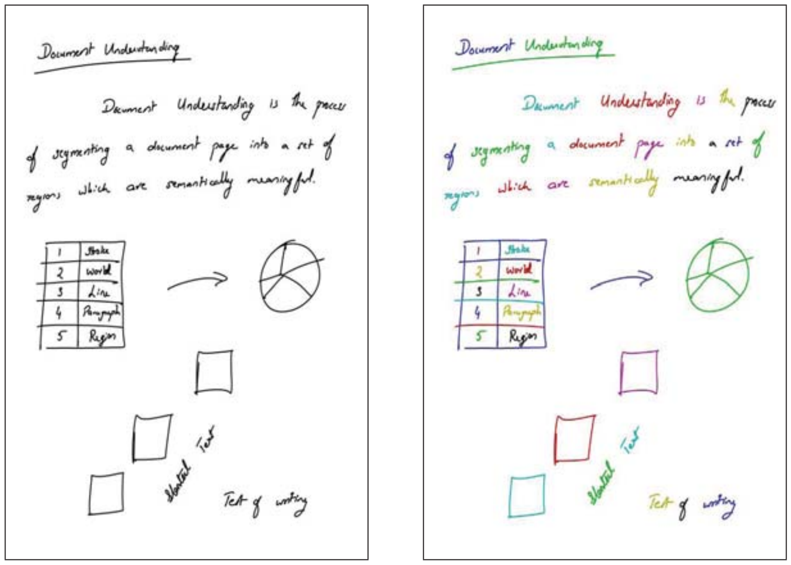
Figure 4.6: The finite state automaton that detects words in the document.

$$P(word/strokes) = \prod_{i=1}^{k} P(text/stroke_i) * P(position_i),$$

where $P(position)$ is a normal distribution on the inter-stroke distance with zero mean and variance set based on the resolution of the tablet. A similar value is computed for non-text strokes. Once the conditional class probabilities are estimated, the confidence value for the label is computed using the Bayes rule:

$$P(text/word) = \frac{p(word/text)P(text)}{p(word/text)P(text) + p(word/nontext)P(nontext)}.$$

We assume equal priors ($P(text) = P(nontext)$) for the labels. However, one could modify this based on the neighboring regions and their spatial relationships. Figure 4.7(b) shows example of word detection for the on-line document in Figure 4.7(a). The word segmentation algorithm does make errors in segmentation, especially when neighboring words are close to each other. In addition, many of the non-text regions are incorrectly clustered (lines in the table in Figure 4.7(b)). However, these errors does not significantly affect the segmentation performance since the non-text strokes are treated as individual strokes during the region segmentation phase.


Document Understanding
Document Understanding is the process
of segmenting a document page into a set of
regions which are semantically meaningful.
1 Stroke
2 Word
3 Line
4 Paragraph
5 Region
Slanted Text
Text of writing

(a) (b)

Figure 4.7: Results of word detection: (a) an on-line document and (b) the words detected by the algorithm. Each word is shown in a different color.

### 4.2.1 Region Segmentation

The segmentation of regions is inherently an ambiguous problem. Even human experts could differ in their judgement about region boundaries in an unstructured handwritten page. Hence we need to define a criterion that captures the generally accepted properties of a distinct region. The criterion that we use to define a region uses its spatial and temporal compactness. In other words, a region is a set of strokes that are spatially and temporally close. We define the penalty of clustering a set of strokes into a single region based on the difference in area of its bounding box and that of the individual strokes. However, the compactness criterion by itself tend to fragment regions to reduce the total area. Hence we introduce an opposing criterion that penalizes the splitting of regions. Hence the probability associated with a particular partition is inversely proportional to both the number of regions and the sum of the areas enclosed by the bounding boxes of the individual regions. Hence the criterion or cost function that we want to minimize, C(S), for the segmentation $S = \{s_1, s_2, \cdots, s_n\}$, is:

$$C(S) = \frac{n^k \ . \ area(S)}{\sum_{i=1}^{n} area(s_i)} . \prod_{i=1}^{n} C(s_i),$$

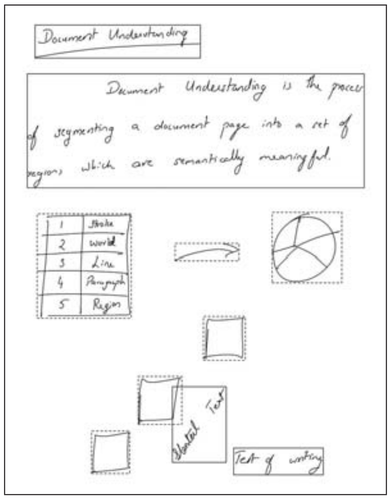
where $s_i$ denotes the individuals regions, and $k$ is a constant that controls the penalty for the division of a page into $n$ regions (computed empirically). The cost of the terminal symbols is set to $1$.

The goal of our system is to come up with a partition of the strokes that globally minimizes the cost function. We define a stochastic context-free grammar (SCFG) to parse and segment the document into a hierarchy of regions. Since the intermediate nodes in this tree refer to regions of text or sketches, the nodes should carry additional information about the nature and content of the region. Such a grammar, which incorporates attributes of nodes in addition to their labels, is referred to as an *attributed grammar*. In our system, the attributes include the bounding box of the region and the classification confidence of the region. Figure 4.8 shows the grammar used in this work for region segmentation. In addition to the grammar rules, each production is associated with a probability. The probabilities for all productions, except that of the text line are computed based on the cost function, $C(S)$ described above. The probability for text lines also incorporates the direction between the adjacent words. Note that only the important rules are listed here to improve readability.

| | | |
|---|---|---|
| page | ::= | regions [numberOfRegions] |
| region | ::= | paragraph \| nonTextRegion [boundingBox] |
| paragraph | ::= | textLines [separation] |
| textLine | ::= | words [direction] |
| nonTextRegion | ::= | words \| nonTextStrokes [boundingBox] |

Figure 4.8: Grammar rules used for segmentation. The attributes are shown in square brackets.

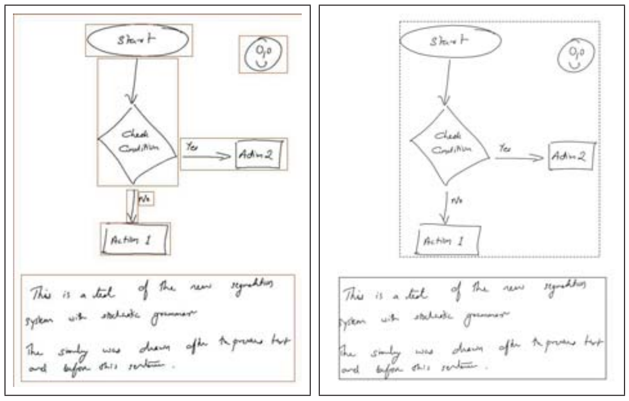
The parsing is done using the CYK algorithm [64], which is a dynamic programming method that computes the most probable parse of an observation sequence, given a grammar. The result of parsing the page shown in Figure 4.7 is given in Figure 4.9.



Document Understanding
Document Understanding is the process
of segmenting a document page into a set of
regions which are semantically meaningful.
1 Stroke
2 Word
3 Line
4 Paragraph
5 Region
Vertical Text
Text of writing

Figure 4.9: Result of parsing of the document in Figure 4.7. The solid boxes indicate text regions and dotted boxes indicate non-text regions.

The parsing might lead to over-segmentation in some cases, where dividing a region could reduce the area of its bounding box. Figure 4.10 (a) shows an example of a document that is over-segmented by the parsing algorithm. This could also arise from temporal discontinuities in a region. To overcome these difficulties, we introduce a post processing stage that combines spatially overlapping regions. We also combine spatially and temporally adjacent non-text regions, based on a threshold on spatial adjacency (set to 120 units for the resolution of Tablet PC). The result of post processing of the document in Figure 4.10 (a) is given in Figure 4.10 (b). Note that the fragmented flowchart is combined into a single region. However, the sketch at the top right corner is also included in the region, and could be labelled as an error.



(a) (b)

Figure 4.10: Output of (a) segmentation and (b) post-processing of an on-line handwritten document.

## 4.3 Table Identification and Processing

Handwritten tables can be classified into two categories: ruled and unruled. The ruled tables have lines separating the rows and columns of the table whereas an unruled table is an arrangement of text into different columns, without any explicit lines separating the columns (see Figure 4.5(a)). These two types of tables are treated separately as the identification of ruling lines provides an additional cue for ruled tables.

### 4.3.1 Ruled Tables

A ruled table is identified based on the presence of horizontal and vertical lines, which are detected using the Hough transform. The ruled tables in the 2-dimensional Hough space $(r, \theta)$ have significant peaks around angles of $\theta = 0^o$ and $\theta = 90^o$. Each bin corresponds to a span of $50$ pixels on the $r$-axis and $10^o$ on the $\theta$-axis. The threshold for identifying a peak is computed based on the width or height of the region. The threshold of bin strength for a region $R$ is defined as:

$$Bin\ Strength(R) = w_{threshold} \cdot \frac{min(width(R), height(R))}{ResamplingDistance}, \tag{4.5}$$

where $w_{threshold}$ is the weight that controls the proportion of points that will appear in a bin (experimentally set to $0.2$).

A region is classified as a table or a diagram based on the lengths of the lines (size of the cluster in the transform space) and their orientations. A further restriction that tables contain at least $5$ lines (four borders and at least one partition) helps to distinguish ruled tables from underlined key words. Figure 4.11 shows a typical ruled table and its $(r, \theta)$ representation. A total of $9$ lines are detected, which have been redrawn as straight lines in Figure 4.11 (c). An underlined keyword is detected when a region has a single horizontal line in the Hough transform space and has some text above it. Note that this method can detect broken underlines as can be seen in Figure 4.12.

### 4.3.2 Unruled Tables

The text strokes contain both plain text and unruled tables. To identify the unruled tables, first individual text lines are identified and adjacent lines are incrementally grouped. The lines are detected as part of the segmentation by the SCFG-based parser.

Identification of unruled tables is primarily based on the x-axis projection of the region to be classified. The table regions tend to have prominent peaks and valleys in this pro-

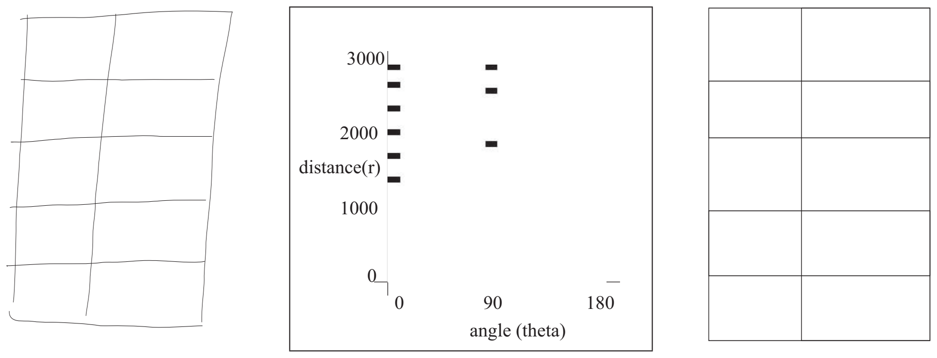


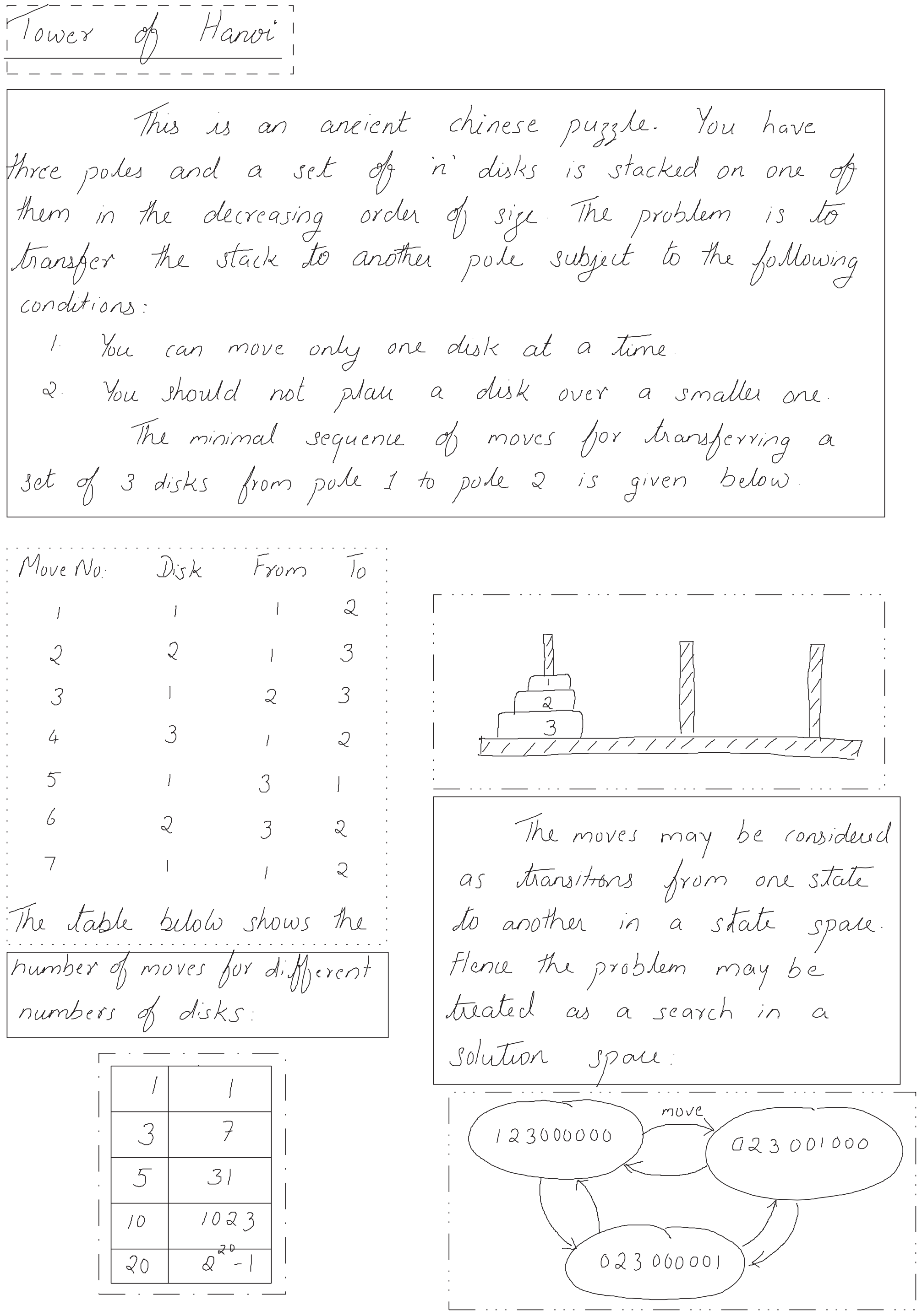
Figure 4.11: Finding lines in ruled tables: (a) a typical table, (b) peaks in the 2-Dimensional Hough transform, and (c) table cleaned up by identifying the lines.

jection, while text regions have a more uniform projection (see Figures 4.13 (c) and (d)). Text also varies in appearance due to differences in inter-word and inter-line spacings. We assume that the inter-word distances are more than double the inter-stroke distances within a word.

### 4.3.3 Table Processing

In the case of a ruled table, the horizontal and vertical lines provide the cell boundaries. The cell boundaries in unruled tables are determined by dividing each line at vertical cell boundaries given by the valleys in the projection histogram. The cell contents are easily identified by collecting the text strokes within each cell boundary. The text in each cell is supplied to a text recognizer [1] and the results are written into an ASCII table for exporting it to *Microsoft Excel* ©. Figure 4.14 shows the result of exporting the ruled table in Figure 4.5 (b) into *Excel*. Note that the numerical data in individual cells were correctly identified although the text recognizer incorrectly recognized the word ‘Unruled Tables’ in the second row as ‘Annual Tables’. We used the text recognizer supplied along with the CrossPad, by IBM for this experiment. The final result of processing the on-line document in Figure 4.1 (a) is shown in Figure 4.12.

[1] *IBM Ink Manager* © software shipped with the *CrossPad*©.



Tower of Hanoi

This is an ancient chinese puzzle. You have three poles and a set of 'n' disks is stacked on one of them in the decreasing order of size. The problem is to transfer the stack to another pole subject to the following conditions:

1. You can move only one disk at a time.
2. You should not place a disk over a smaller one.

The minimal sequence of moves for transferring a set of 3 disks from pole 1 to pole 2 is given below.

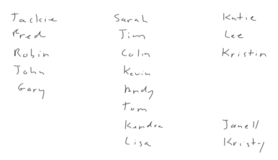
| Move No. | Disk | From | To |
|---|---|---|---|
| 1 | 1 | 1 | 2 |
| 2 | 2 | 1 | 3 |
| 3 | 1 | 2 | 3 |
| 4 | 3 | 1 | 2 |
| 5 | 1 | 3 | 1 |
| 6 | 2 | 3 | 2 |
| 7 | 1 | 1 | 2 |

The table below shows the number of moves for different numbers of disks:

| | |
|---|---|
| 1 | 1 |
| 3 | 7 |
| 5 | 31 |
| 10 | 1023 |
| 20 | $2^{20}-1$ |

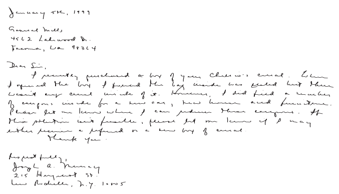
The moves may be considered as transitions from one state to another in a state space. Hence the problem may be treated as a search in a solution space.

Figure 4.12: Segmented document of Figure 4.1 (a).



| | | |
|---|---|---|
| Jackie | Sarah | Katie |
| Fred | Jim | Lee |
| Robin | Colin | Kristin |
| John | Kevin | |
| Gary | Andy | |
| | Tom | |
| | Kendra | Janell |
| | Lisa | Kristy |

(a)



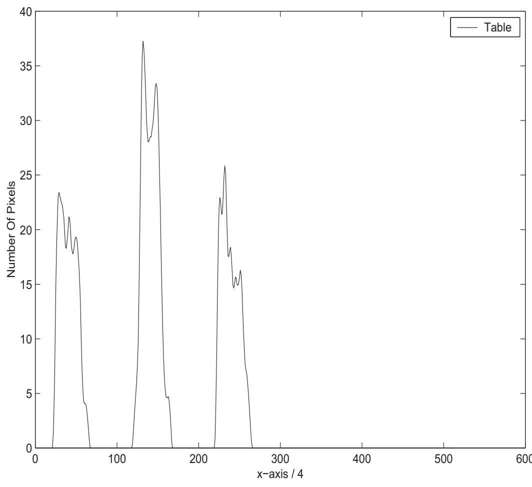
January 5th, 1999

General Mills
4562 Lakewood Dr.
Tacoma, WA 98364

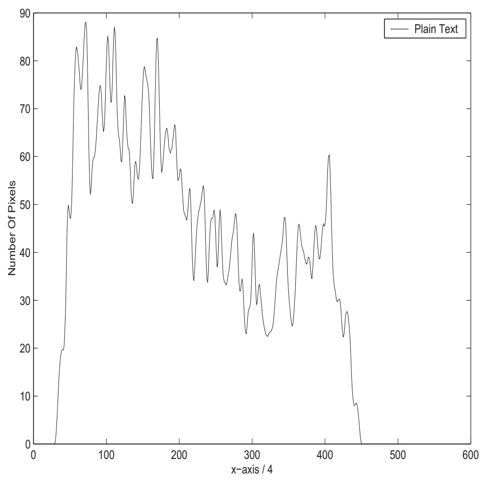
Dear Sir,
I recently purchased a box of your Cheerio's cereal. When
I opened the box I found the bag inside was sealed but there
wasn't any cereal inside of it. However, I did find a number
of coupons inside for a new car, new house and furniture.
Please let me know where I can redeem these coupons. If
this solution isn't feasible, please let me know if I may
either receive a refund or a new box of cereal.
Thank You.

Respectfully,
Joseph A. Murray
215 Haymarket St.
New Rochelle, N.Y. 10805

(b)

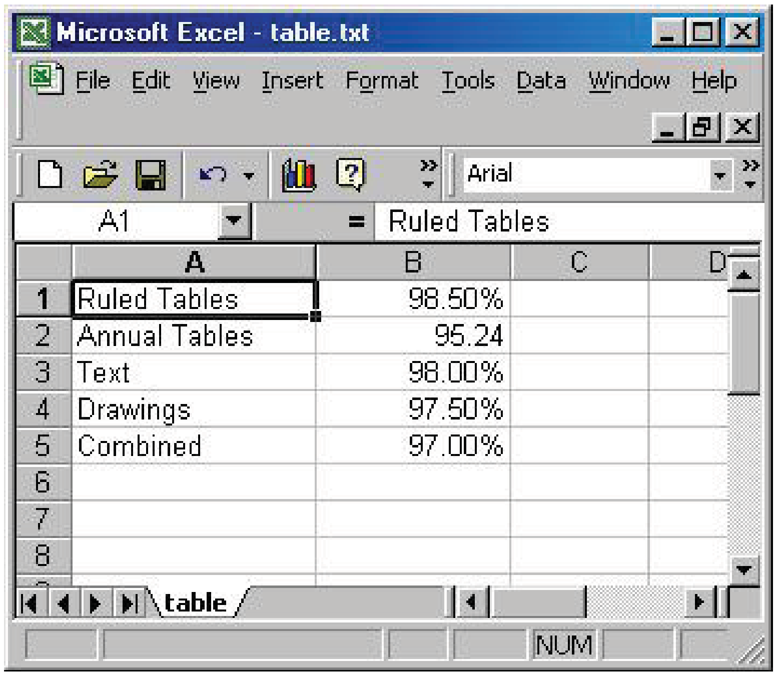


(c)



(d)

Figure 4.13: Projections of (a) table and (b) text.


Microsoft Excel - table.txt
File Edit View Insert Format Tools Data Window Help
Arial
A1
= Ruled Tables
table
NUM

| | A | B | C | D |
|---|---|---|---|---|
| 1 | Ruled Tables | 98.50% | | |
| 2 | Annual Tables | 95.24 | | |
| 3 | Text | 98.00% | | |
| 4 | Drawings | 97.50% | | |
| 5 | Combined | 97.00% | | |
| 6 | | | | |
| 7 | | | | |
| 8 | | | | |

Figure 4.14: A table imported into Microsoft Excel.

## 4.4 Experimental Results

The experimental data for document segmentation and table identification was collected from $12$ different people [2] without any restriction on the style or content of data. Most of the data consisted of meeting notes collected over a period of three months that contained both text and sketch regions. Figure B.1 in Appendix B shows examples of documents in the database used for this experiment. Text vs. non-text classifier was trained on a set of $1,304$ strokes collected from $5$ writers ($1,202$ text strokes and $102$ non-text strokes). A classification accuracy of $99.1\%$ was achieved on an independent test set of $36,812$ strokes collected from $10$ writers ($35,882$ text strokes and $930$ non-text strokes). Most of the misclassifications were due to very short strokes in diagrams that were incorrectly identified as text strokes. About $99.9\%$ of the text strokes were correctly classified.

The document segmentation algorithm described above achieves an accuracy of $88.3\%$

[2]We would like to thank Dr. Jayashree Subrahmonia and the Pen Computing group at IBM T.J. Watson Research Center for generously providing this data.

on a data set of 40 documents containing a total of 154 regions. The accuracy refers to the percentage of regions that was correctly identified. Each error corresponds to either two separate regions that were merged or a single region that was split into two. The table detection algorithm achieved a classification accuracy of 84.8% on a data set containing 105 ruled tables. The errors in table detection were primarily due to skewed lines and the regions were classified as sketches. The unruled table detection algorithm is highly sensitive to the alignment of text and it achieves a classification accuracy of only 60% on a set of 30 regions containing text and unruled tables. Detection of unruled tables is an extremely challenging problem and a projection-based approach is not very promising.

Figures 4.15 (a) - 4.15 (c) show examples of three on-line documents after the stroke classification. The results of stroke classification is presented after making a hard decision based on the probabilities computed as described before. Note that there are several classification errors such as detection of strokes of characters ’T’ and ’p’ as non-text and the arrow heads in Figure 4.15 (a) being detected as text. These errors are primarily because of the overlap of short text and non-text stokes in the feature space representation (see Figure 4.4). The results of segmentation and ruled table detection of these three documents are shown in Figures 4.15 (d) - 4.15 (f). Note that all the regions and ruled tables are correctly detected in spite of the errors in stroke classification. The grammar-based segmentation algorithm is capable of tolerating many of the errors in stroke classification.

Figures 4.16 (a) - 4.16 (c) show examples of errors in segmentation of three on-line documents. In Figure 4.16 (a), A flowchart is divided into two segments due to the compactness of the individual segments and the separation between them. Figure 4.16 (b) shows an example, where three regions were merged together into a single region, due to its spatial and temporal proximity. Figure 4.16 (c) shows the result of segmentation, where a single region is divided into six smaller components due to the spatial separation of the individual components.

Figures 4.17 (a) - 4.17 (c) show examples of three on-line documents with ruled tables

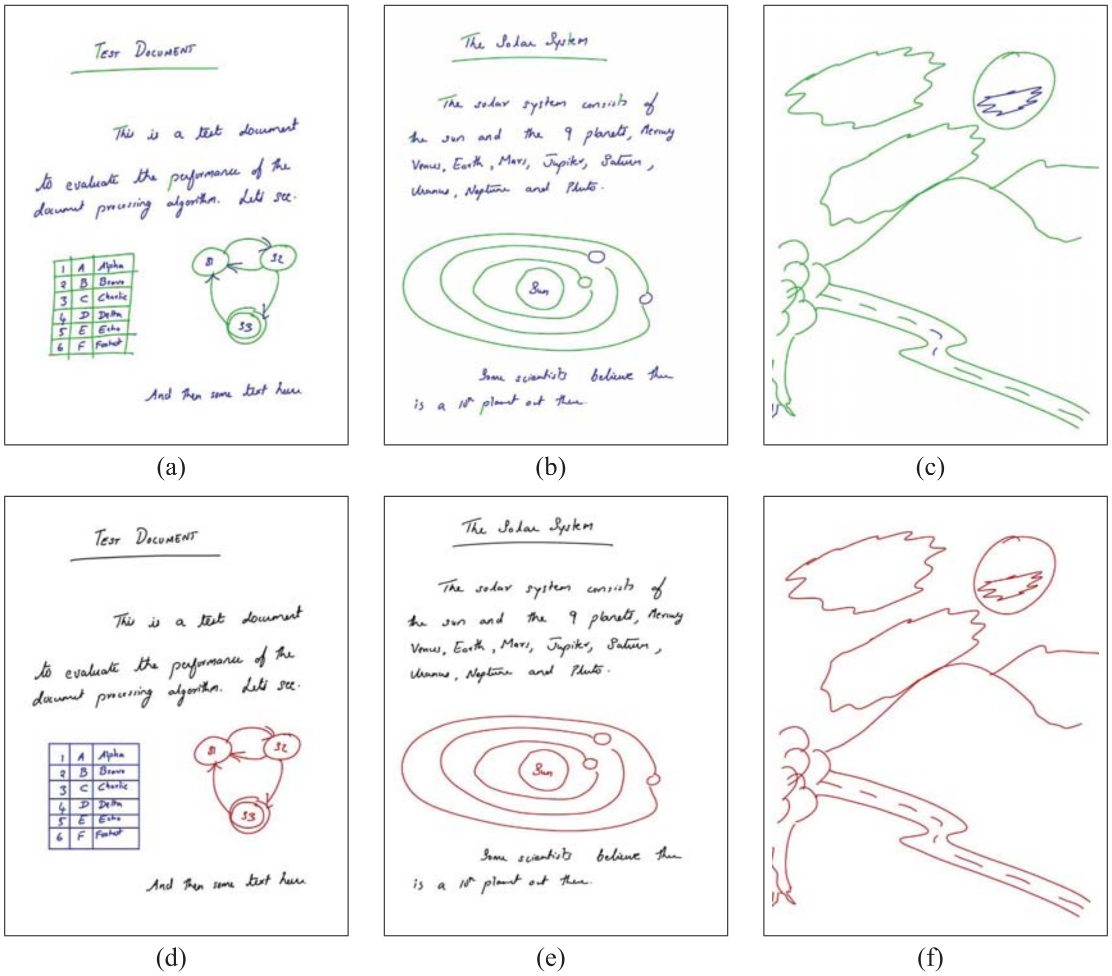
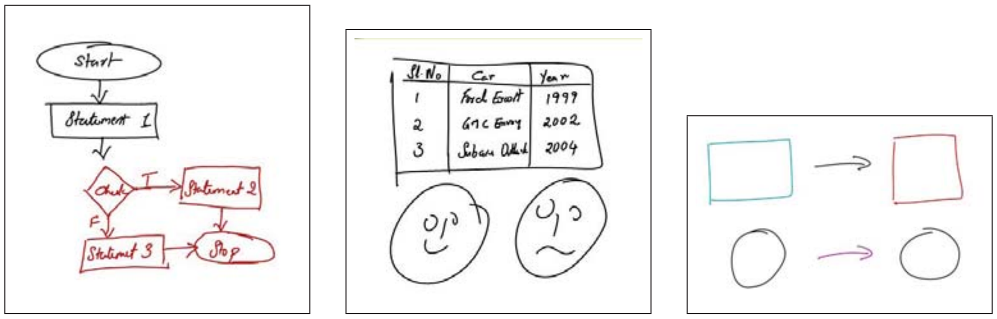


Figure 4.15: Results of stroke classification and document processing. (a) - (c) three on-line documents after stroke classification. Text strokes are shown in blue and non-text strokes are shown in green. (d) - (f) results of segmentation and ruled table detection of the corresponding documents in (a) - (c).

after segmentation. Note that the regions containing the ruled tables are correctly segmented from the text along with them. Figures 4.17 (d) - 4.17 (f) show the results of ruled table detection for these documents. The table in Figure 4.17 (a) is not detected by our algorithm. This is primarily because the lines of the ruled table are not straight and does not result in a peak in the Hough transform space. The tables in Figures 4.17 (b) and 4.17 (c) are correctly detected by the proposed algorithm. However, there are errors in detection of some of the lines of the table and their positions. The position of vertical line in the second table is not correctly detected due to the slant of the line, while some of the



(a) (b) (c)

Figure 4.16: Errors in region segmentation. (a) - (c) results of segmentation of three on-line documents. Each region is shown in a different color.

horizontal lines in the third table are not detected since the individual strokes that form the line are not oriented in the same direction. Detection and processing of ruled tables is an extremely challenging problem and a more comprehensive solution need to be developed for detection of complex ruled tables.

## 4.5 Summary

We have proposed a framework for segmentation of unstructured handwritten documents. The stroke properties are computed using a spline model, which approximates the observed data. A stochastic context free grammar-based solution is proposed for region segmentation. A quantitative comparison of the results to those reported in the literature is not feasible due to the lack of standardization in the test databases and error reporting. However, the proposed approach is capable of segmentation of a variety of writing styles and has the potential of learning from observed data, unlike rule-based methods. Initial results of this work has been published in [55] and the SCFG-based segmentation work is currently under review for publication.

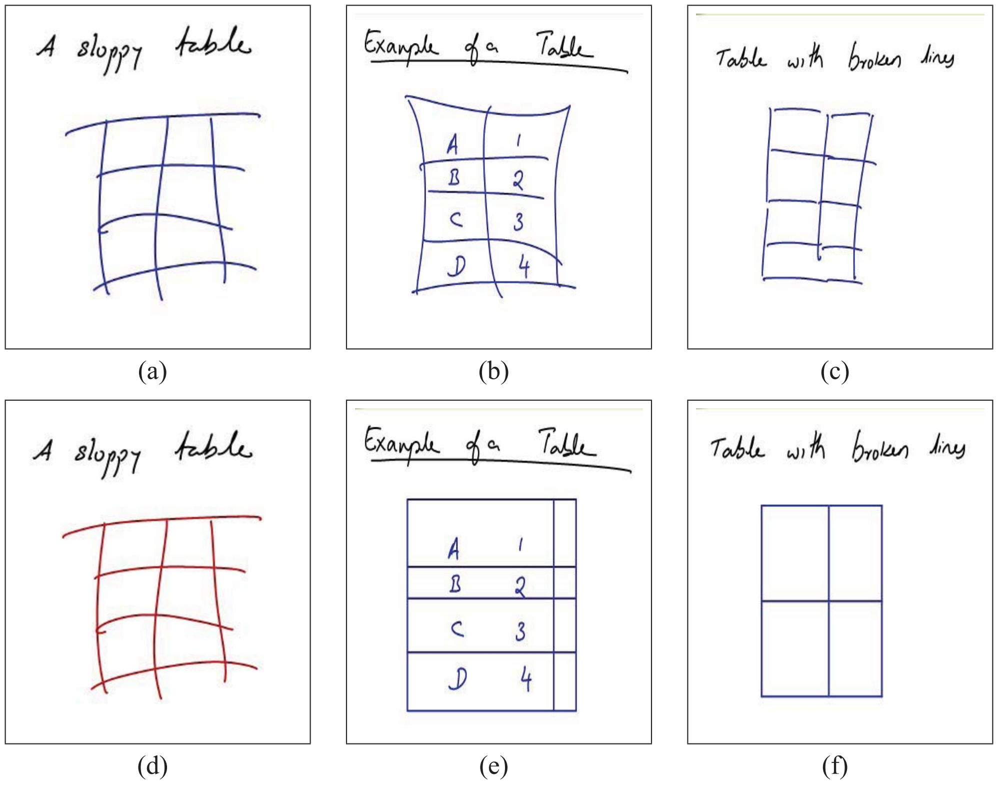


Figure 4.17: Errors in table detection and processing. Figures (a) - (c) show examples of three tables and (d) - (f) show the corresponding outputs of the ruled table processing algorithm.

# Chapter 5

# Script Identification

On-line documents may be written in different languages and scripts [1]. A single document page in itself may contain text written in multiple scripts. For example, a document in English may have some annotations or edits in another language. Such documents are commonly encountered in note taking applications, where users who write multiple languages might use more than one script in a document page. Databases of handwritten documents might also contain different documents that are written in different scripts. Most of the text recognition algorithms are designed to work with a particular script [20, 130, 136] and, therefore, they treat any input text as being written only in the script under consideration. An on-line document analyzer must first identify the script before employing a particular algorithm for text recognition. Figure 5.1 shows an example of a document page (created by us for illustrating the results) containing six different scripts. Note that a typical multi-lingual, on-line document contains two or three scripts [78]. For lexicon-based recognizers, it may be helpful to identify the specific language of the text if the same script is used by multiple languages [38].

[1]Multiple languages may use the same script. See Section 5.1 for explanation.

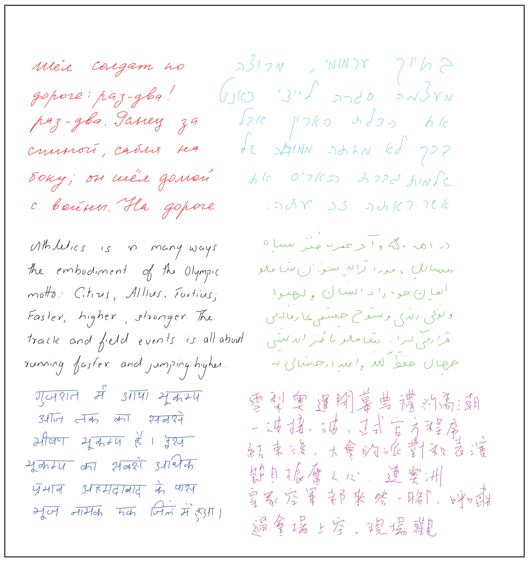

Шёл солдат по
дороге: раз-два!
раз-два. Ранец за
спиной, сабля на
боку; он шёл домой
с войны. На дороге
Athletics is in many ways
the embodiment of the Olympic
motto: Citius, Altius, Fortius;
Faster, higher, stronger. The
track and field events is all about
running faster and jumping higher
गुजरात में आया भूकम्प
आज तक का सबसे
भीषण भूकम्प है। इस
भूकम्प का सबसे अधिक
प्रभाव अहमदाबाद के पास
भुज नामक एक जिले में हुआ।

Figure 5.1: A multi-script on-line document containing Cyrillic, Hebrew, Roman, Arabic, Devnagari and Han scripts.

## 5.1 Scripts and Languages

A script is defined as a graphic form of a writing system [21]. Different scripts may follow the same writing system. For example, the *alphabetic* system is adopted by scripts like Roman and Greek, and the *phonetic-alphabetic* system is adopted by most Indian scripts, including Devnagari. A specific script like Roman may be used by multiple languages such as English, German and French. Figure 5.2 shows examples of three different languages, English, German and French, all written using the same script.

During the evolution of languages, existing scripts were adopted or modified by many languages to suit their specific words and sounds. Due to such interactions, the scripts of many languages are either identical or have only minor variations. Table 5.1 shows the main scripts used by various languages around the world [95]. A detailed study of the history and evolution of scripts can be found in Jensen [63] and Lo [86]. The first six scripts in Table 5.1 - Arabic, Cyrillic, Devnagari, Han, Hebrew and Roman - cover the languages used by a majority of the world population. Other scripts, marked as language specific scripts in Table 5.1, are used exclusively by certain languages. Based on the above observation of

Table 5.1: A classification of the major scripts of the world with examples of languages that use individual scripts [95]. The first column shows the direction of writing as an ordered pair: (Inter-word direction, Inter-line direction). The letters L, R, T and B stand for Left, Right, Top and Bottom, respectively and the string L-R stands for "Left to Right".

| Writing direction | Script | Examples of Languages |
|---|---|---|
| (L-R, T-B) | Cyrillic | Russian, Ukrainian, Bulgarian, Byelorussian |
| | Devnagari | Sanskrit, Hindi, Marathi, Nepali |
| | Roman | English, French, German Polish, Icelandic, Swahili, Spanish, Vietnamese |
| | (Language specific scripts) | Bangla, Georgian, Greek, Gujarati, Kannada, Korean, Malayalam, Punjabi, Tamil, Telugu, Thai |
| (R-L, T-B) | Arabic | Arabic, Farsi, Urdu, Pashto |
| | Hebrew | Hebrew, Yiddish |
| (L-R, T-B) | Han | Chinese, Japanese |
| (T-B, R-L) | Kanji/Kana | Japanese |

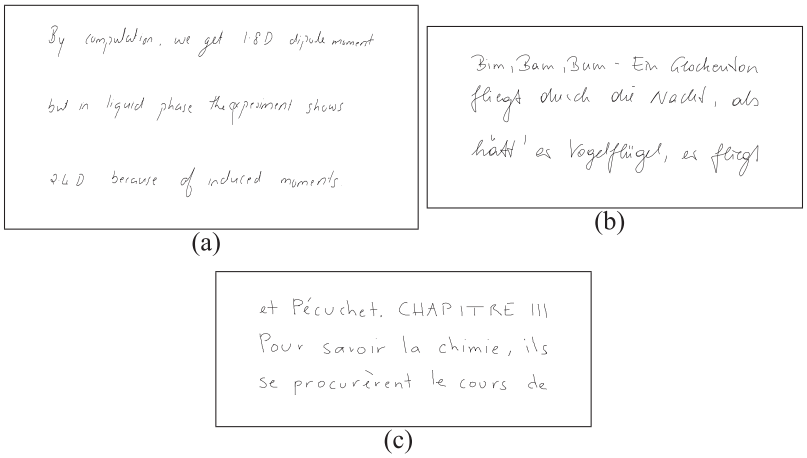

By computation, we get 1.8D dipole moment
but in liquid phase the experiment shows
2.4D because of induced moments.
(a)
Bim, Bam, Bum - Ein Glockenton
fliegt durch die Nacht, als
hätt' es Vogelflügel, es fliegt
(b)
et Pécuchet. CHAPITRE III
Pour savoir la chimie, ils
se procurèrent le cours de
(c)

Figure 5.2: Multiple languages using the same script. (a) English, (b) German, and (c) French, all use the same *Roman* script.

the relationship between languages and their scripts, the six most popular scripts, Arabic, Cyrillic, Devnagari, Han, Hebrew and Roman, were chosen for our on-line classification study (see Figure 5.1). The general class of Han-based scripts include Chinese, Japanese and Korean. Japanese and Korean languages augment the main script with their own set of characters (Kana and Han-Gul). In this work, we have used only Chinese characters and hence we use the term *Han Script* to refer to the Chinese character set. This includes both the traditional and simplified Chinese character sets. Devnagari script is used by many Indian languages, including Hindi, Sanskrit, Marathi and Rajasthani. Arabic script is used by Arabic, Farsi, Urdu, etc. Roman script is used by many European languages like English, German, French and Italian. We attempt to solve the problem of script recognition, where either an entire document or a part of a document (upto word level) is classified into one of the six scripts mentioned above with the aim of facilitating text recognition. The problem of identifying the actual language often involves recognizing the text and identifying specific words or sequences of characters, which is beyond the scope of this paper.

## 5.2 Multi-Script Data

The users were asked to write one page of text in a particular script, with each page containing approximately 20 lines of text. No restriction was imposed on the content or style of writing. The details of the database used for this work are given in Table 6.9. Multiple pages from the same writer were collected at different times. Figures B.2 and B.3 in Appendix B shows examples of documents in the database used for this experiment.

Table 5.2: Data description for the script classification problem.

| Script | Number of Writers | Number of Pages | Number of Lines | Number of Words |
|---|---|---|---|---|
| Arabic | 15 | 15 | 209 | 1423 |
| Devnagari | 12 | 18 | 382 | 3173 |
| Han | 12 | 15 | 282 | 1981 |
| Roman | 45 | 45 | 722 | 3539 |
| Cyrillic | 10 | 10 | 276 | 1002 |
| Hebrew | 10 | 10 | 284 | 2261 |

## 5.3 Feature Extraction

Each sample or pattern that we attempt to classify is either a word or a set of contiguous words in a line. Figure 5.3 shows examples of each of the six scripts under consideration. It is helpful to study the general properties of each of the six scripts for feature extraction.

1. *Arabic*: Arabic is written from right to left within a line and the lines are written from top to bottom. A typical Arabic character contains a relatively long main stroke, which is drawn from right to left, along with one to three short strokes. The character set contains three long vowels. Short markings (diacritics) may be added to the main character to indicate short vowels [128]. Due to these diacritical marks and the dots in the script, the length of the strokes vary considerably.

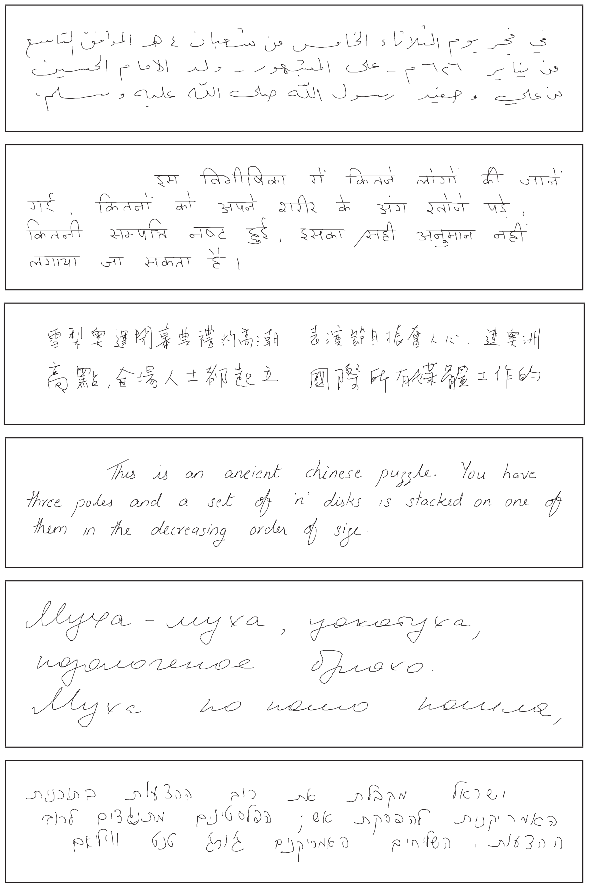

This is an ancient chinese puzzle. You have three poles and a set of 'n' disks is stacked on one of them in the decreasing order of size.
Муха - муха, цокотуха, позолочённое брюхо. Муха по полю пошла,

Figure 5.3: Examples of lines from each script that were used to train and test the classifiers.

2. *Cyrillic*: Cyrillic script looks very similar to the cursive Roman script. The most distinctive features of Cyrillic script, compared to Roman script are: (i) individual characters, connected together in a word, form one long stroke, and (ii) absence of delayed strokes (see Figure 5.5). Delayed strokes cause movement of the pen in the direction opposite to the regular writing direction.

3. *Devnagari*: The most important characteristic of Devnagari script is the horizontal line present at the top of each word, called '*Shirorekha*' (see Figure 5.4). These lines are usually drawn after the word is written and hence are similar to delayed strokes in Roman script. The words are written from left to right in a line.

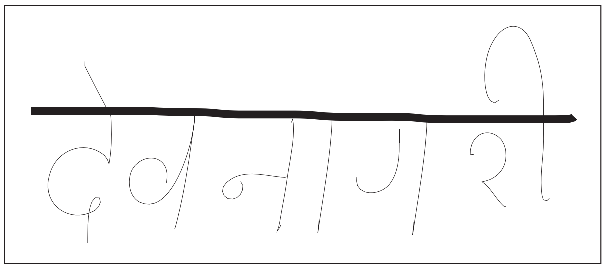


Figure 5.4: The word '*devnagari*' written in the Devnagari script. The *Shirorekha* is shown in bold.

4. *Han*: Characters of Han script are composed of multiple short strokes. The strokes are usually drawn from top to bottom and left to right within a character. The direction of writing of words in a line is either left to right or top to bottom. The database used in this study contains Han script text of the former type (horizontal text lines).

5. *Hebrew*: Words in a line of Hebrew script are written from right to left and hence the script is temporally similar to Arabic. The most distinguishing factor of Hebrew from Arabic is that the strokes are more uniform in length in the Hebrew script.

6. *Roman*: Roman script has the same writing direction as Cyrillic, Devnagari and Han scripts. We have already noted the distinguishing features of these scripts compared to the Roman script. In addition, the length of the strokes tend to fall between that of Devnagari and Cyrillic scripts.

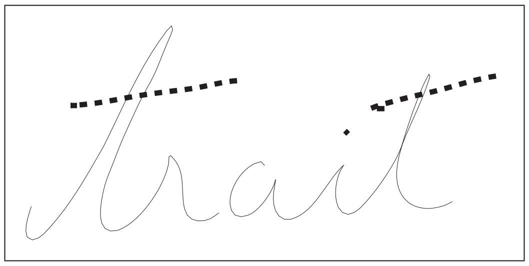


Figure 5.5: The word '*trait*' contains three *delayed strokes*, shown as bold dotted lines.

The features are extracted either from the individual strokes or from a collection of strokes. Here, we describe each of the features and its method of computation. The features are presented in the order of their saliency in the final classifier (as determined by the feature selection algorithm described in Section 5.5).

1. *Horizontal Inter-stroke Direction (HID)*: This is the sum of the horizontal directions between the starting points of consecutive strokes in the pattern. The feature essentially captures the writing direction within a line.

$$HID = \sum_{i=1}^{n-r} dir(i, i+r), \tag{5.1}$$

$$dir(i, j) = \begin{cases} +1 & \text{if } X_{start}(stroke_i) < X_{start}(stroke_j) \\ -1, & \text{otherwise,} \end{cases} \tag{5.2}$$

   where $X_{start}(.)$ denotes the $x$ coordinate of the pen-down position of the stroke, $n$ is the number of strokes in the pattern and $r$ is set to $3$ to reduce errors due to abrupt changes in direction between successive strokes. The value of $HID$ falls in the range $[r - n, n - r]$.

2. *Average Stroke Length (ASL)*: The computation of the stroke length for an individual stroke is described in Chapter 3. The Average Stroke Length is defined as the average length of the individual strokes in the pattern.

$$ASL = \frac{1}{n} \sum_{i=1}^{n} length(stroke_i), \tag{5.3}$$

   where $n$ is the number of strokes in the pattern. The value of $ASL$ is a real number that falls in the range $(0.0, R_0]$, where the value of $R_0$ depends on the resampling distance used during pre-processing ($R$ = $600$ in our experiments).

3. *Shirorekha Strength*: This feature measures the strength of the horizontal line component in the pattern using the Hough transform. The value of this feature is computed

as:

$$ShirorekhaStrength = \frac{\sum\limits_{\forall\ r, -10<\theta<10} H(r,\theta)}{\sum\limits_{\forall\ r,\theta} H(r,\theta)}, \tag{5.4}$$

where $H(r, \theta)$ denotes the number of votes in the $(r, \theta)^{th}$ bin in the two-dimensional Hough transform space. The numerator is the sum of the bins corresponding to line orientations between $-10°$ and $10°$ and the denominator is the sum of all the bins in the Hough transform space. Note that it is difficult to constrain the values of $r$ in the transform space due to variations introduced by sampling and the handwriting itself. The value of *Shirorekha Strength* is a real number that falls in the range $[0.0, 1.0]$.

4. *Shirorekha Confidence*: We compute a confidence measure for a stroke being a Shirorekha (see Figure 5.4). Each stroke in the pattern is inspected for three different properties of a Shirorekha; Shirorekhas span the width of a word, always occur at the top of the word and are horizontal. Hence the confidence ($C$) of a stroke ($s$) is computed as:

$$C(s) = \frac{width(s)}{width(pattern)} * \frac{\bar{Y}(s)}{height(pattern)} * \left(1 - \frac{height(s)}{width(s)}\right), \tag{5.5}$$

where $width(s)$ refers to the length along the $x$-axis (horizontal), $height(s)$ is the length of a stroke along the $y$-axis (vertical), and $\bar{Y}(s)$ is the average of the $y$-coordinates of the stroke points. Note that $0 \leq C(s) \leq 1$ for strokes with $height < width$. For vertical strokes, the value of $C(s)$ will be negative. To avoid abnormal scaling of the values of this feature, $C(s)$ was set to zero, if its computed value is negative. For an $n$-stroke pattern, the *Shirorekha Confidence* is computed as the maximum value for $C$ among all its component strokes.

5. *Stroke Density*: This is the number of strokes per unit length ($x$-axis) of the pattern. Note that the Han script is written using short strokes, while Roman and Cyrillic are

written using longer strokes.

$$\text{Stroke Density} = \frac{n}{width(pattern)}, \tag{5.6}$$

where $n$ is the number of strokes in the pattern. The value of *Stroke Density* is a real number and can vary within the range $(0.0, R_1)$ where $R_1$ is a positive real number. In practice, the value of $R_1$ was observed to be $0.5$.

6. *Aspect Ratio*: This is the ratio of the width to the height of a pattern.

$$AspectRatio(pattern) = \frac{width(pattern)}{height(pattern)}. \tag{5.7}$$

The value of *Aspect Ratio* is a real number and can vary within the range $(0.0, R_2)$, where $R_2$ is a positive number. In practice, the value of $R_2$ was observed to be $5.0$. Note that this feature is more meaningful for word-level classification than for the classification of a complete line.

7. *Reverse Distance*: This is the distance by which the pen moves in the direction opposite to the normal writing direction. The normal writing direction is different for different scripts as we noted at the beginning of this section. The value of *Reverse Distance* is a non-negative integer and its observed values are in the range $[0, 1200]$.

8. *Average Horizontal Stroke Direction*: Horizontal Stroke Direction ($HD$) of a stroke, $s$, can be understood as the horizontal direction from the start of the stroke to its end. Formally, we define $HD(s)$ as:

$$HD(s) = \begin{cases} +1 & \text{if } X_{pen-down}(s) < X_{pen-up}(s) \\ -1, & \text{otherwise,} \end{cases} \tag{5.8}$$

where $X_{pen-down}(.)$ and $X_{pen-up}(.)$ are the $x$-coordinates of the pen-down and pen-up positions, respectively. For an $n$-stroke pattern, the Average Horizontal Stroke Direction is computed as the average of the $HD$ values of its component strokes. The value of *Average Horizontal Stroke Direction* falls in the range $[-1.0, 1.0]$.

9. *Average Vertical Stroke Direction*: It is defined similar to the Average Horizontal Stroke Direction. The Vertical Direction ($VD$) of a single stroke $s$ is defined as:

$$VD(s) = \begin{cases} +1 & \text{if } Y_{pen-down}(s) < Y_{pen-up}(s) \\ -1, & \text{otherwise,} \end{cases} \tag{5.9}$$

where $Y_{pen-down}(.)$ and $Y_{pen-up}(.)$ are the $y$-coordinates of the pen-down and pen-up positions, respectively. For an $n$-stroke pattern, the Average Vertical Stroke Direction is computed as the average of the $VD$ values of its component strokes. The value of *Average Vertical Stroke Direction* falls in the range $[-1.0, 1.0]$.

10. *Vertical Inter-stroke Direction (VID)*: The Vertical Inter-stroke Direction is defined as:

$$VID = \sum_{i=1}^{n-1} dir(i, i+1), \tag{5.10}$$

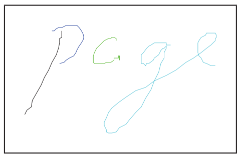
where

$$dir(i,j) = \begin{cases} +1 & \text{if } \bar{Y}(stroke_i) < \bar{Y}(stroke_j) \\ -1, & \text{otherwise.} \end{cases} \tag{5.11}$$

$\bar{Y}(s)$ is the average of the $y$-coordinates of the stroke points and $n$ is the number of strokes in the pattern. The value of *VID* is an integer and falls in the range $(1-n, n-1)$.

11. *Variance of Stroke Length*: This is the variance of the length of the individual strokes within a pattern. The value of *Variance of Stroke Length* is a non-negative integer and its observed value is between $0$ and $250,000$ in our experiments.

Figure 5.6: Feature vector computation: (a) strokes of the word ‘page’. Each stroke is shown in a different color. (b) the feature vector extracted from the word in (a).



(a)

| |
|---|
| 1, 51.75, 0.15, 0.036, 0.023, 2.27, 0, 0.5, $-1.0$, 3.0, 1891.69 |

(b)

Figure 5.6 shows the on-line word ‘page’ and the feature vector extracted from it.

Figure 5.7 shows $500$ word samples from each of the six scripts in a two dimensional feature space. The features used are the two most discriminating features derived from linear discriminant analysis.

## 5.4 Classifier Design

Experiments were conducted with different classifiers to determine, which classifier is the best for the script classification problem. Here we describe the details of each of the classifiers employed in our experiments.

1. *k-Nearest Neighbor (*KNN*) classifier*: The distance between two feature vectors was computed using the Euclidean distance metric. Since the features computed differ drastically in their range of values (see Section 5.3), features were normalized so that they have zero mean and a standard deviation of $1$ (in the training set). Experiments were conducted with different values of $k$, varying from $1$ to $11$.

2. *Bayes Quadratic classifier*: The distribution of the feature vectors for each of the classes was assumed to be Gaussian, and the mean and the covariance matrix were estimated from the training data. A Bayesian classifier, with equal priors was used to classify the test patterns into one of the six script classes. The features were normalized as described before.

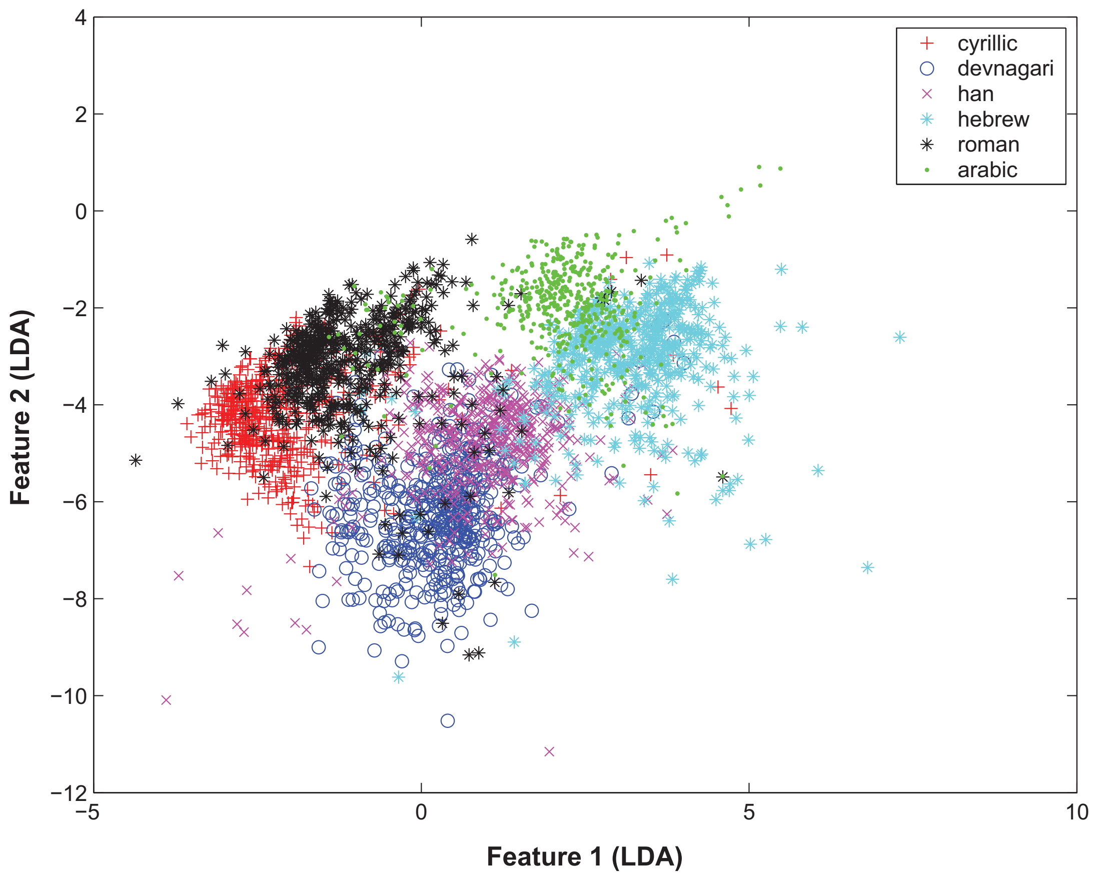


Figure 5.7: Feature space representation of the six scripts. 500 word samples of each script class is presented along the two most discriminating features derived from LDA.

3. *Bayesian classifier with Mixture of Gaussian Densities*: The densities of each of the six classes were assumed to be a mixture of Gaussians. The number of Gaussians within a mixture and their parameters were estimated using the EM algorithm [33]. The classifier itself is similar to the *Bayes Quadratic* classifier and the features were normalized.

4. *Decision Tree-based classifier*: The decision tree-based classifier partitions the feature space into a number of sub-regions by splitting the feature space, using one feature at a time (axis-parallel splits). The regions are split until each subregion contains patterns of only one class with a small number of possible outliers. We used the *C5.0* package [125] to train and test the decision tree classifier.

5. *Neural Network-based classifier*: We used a three-layer neural network (one hidden layer) for the script classifier. The input layer contains $11$ nodes, corresponding to each of the $11$ features used. The output layer contains $6$ nodes, corresponding to each of the six script classes. The number of nodes in the hidden layer was experimentally determined as $25$. Increasing the number of hidden nodes above $25$ resulted in very little improvement in the classification accuracy. During the training phase, the desired output for the node corresponding to the label of the pattern is set to $1$ and all other node outputs are set to $0$. When a test pattern is fed to a trained neural net, the class corresponding to output node with the highest output value is determined to be the correct result.

6. *Support Vector Machine (SVM)*: Support vector machines map an $n$-dimensional feature vector to an $m$-dimensional feature space, where $m > n$, with the assumption that the patterns belonging to different classes are linearly separable in the $m$-dimensional feature space. The mapping is implicitly defined by a *kernel function* [22]. There are a number of commonly used kernel functions, and for the script classifier, we used linear, polynomial and radial basis function (RBF) kernels. The classifier using RBF kernel performed the best among these kernels.

Table 5.3 compares the performance of different classifiers with all the $11$ features described in Section 5.3. The data set containing $13,379$ words was randomly divided into $5$ (approximately equal) groups and a $5$-fold cross validation was done for each of the classifiers. The error rates reported are the averages of these five trials. We notice that the *KNN* ($k = 5$), neural net and *SVM* based classifiers give similar performances. The standard deviation of the error rate was about $0.3\%$ for all the three classifiers.

Table 5.4 gives the confusion matrix for one of the realizations of the neural net classifier for word level classification. Most of the errors are due to short words, where the number of strokes are not sufficient to detect the writing direction. Shorter words also cause a large variance in aspect ratios. The similarity of Roman and Cyrillic scripts is one

Table 5.3: The error rates for different classifiers with $5$-fold cross-validation using $11$ features.

| Classifier | Remarks | Error Rate |
|---|---|---|
| Nearest Neighbor | No Normalization | 35.8 % |
| Nearest Neighbor | Normalized Features | 17.6 % |
| 5-Nearest Neighbor | Normalized Features | 15.4 % |
| 11-Nearest Neighbor | Normalized Features | 15.2 % |
| Bayes Quadratic | Gaussian with Full Covariance | 22.9 % |
| Mixture of Gaussian Distr. | Diagonal Covariance | 25.5 % |
| Decision Tree | C5.0 | 16.1 % |
| Neural Network | One hidden layer with 25 Nodes | 14.3 % |
| SVM | RBF Kernel | 13.5 % |

of the major sources of error in the system. The primary feature that distinguishes the two scripts is based on delayed strokes and is not reliable since not all words in roman script has delayed strokes. Table 5.5 gives the confusion matrix for the classifier that discriminates individual text lines into one of the six scripts. It is difficult to compare these results with any of the reported results in the existing literature, as the script identification problem for on-line documents has not been attempted before. In the case of off-line documents, Hochberg et al. [42] reported an overall accuracy of $88\%$ for script identification at the page level, using six different scripts. Our algorithm correctly classifies all of the $108$ test pages correctly. However, one should be forewarned that the off-line and on-line script identification problems involve very different challenges. The above comparison is made only for the purpose of understanding the relative complexities of the two similar, but non-identical problems.

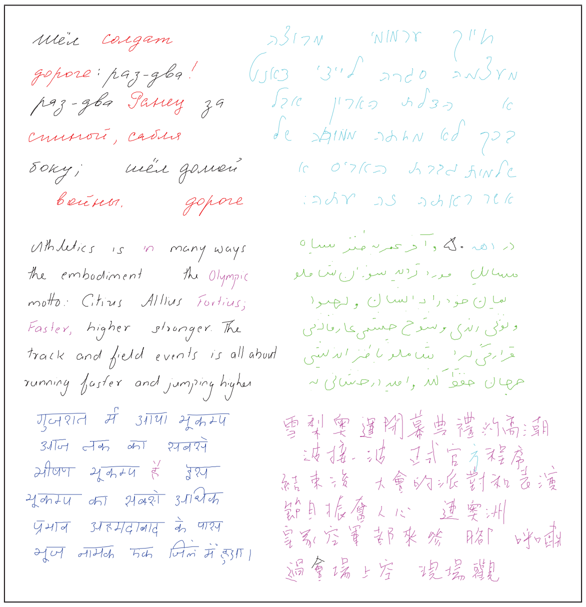
Figure 5.8(a) shows the output of script recognition of the page in Figure 5.1 and Figures 5.8(b), 5.8(c) and 5.8(d) show the output of three other pages.

Table 5.4: Confusion matrix of the script classifier at the word level. The overall accuracy on a test set consisting of $13,379$ words is $86.0\%$.

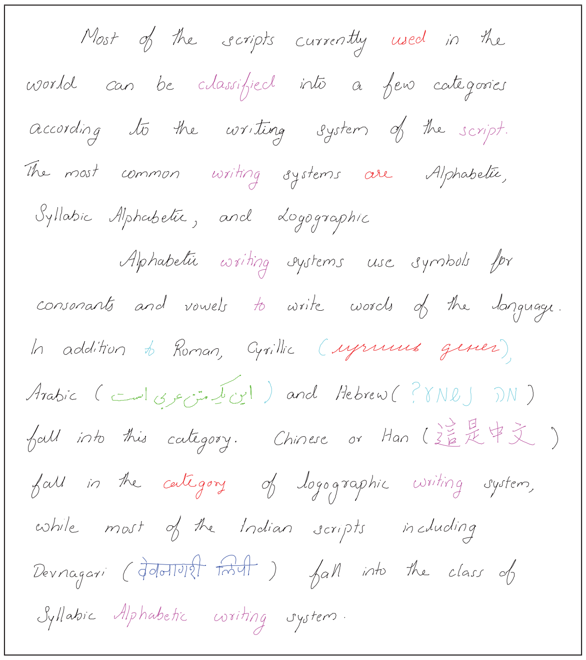
|  | Classified |  |  |  |  |  |
|---|---|---|---|---|---|---|
| True | Arabic | Devnagari | Han | Roman | Cyrillic | Hebrew |
| Arabic | 1151 | 15 | 27 | 47 | 1 | 182 |
| Devnagari | 29 | 2924 | 97 | 108 | 7 | 8 |
| Han | 8 | 87 | 1695 | 157 | 23 | 11 |
| Roman | 32 | 136 | 128 | 3018 | 190 | 35 |
| Cyrillic | 13 | 8 | 26 | 304 | 646 | 5 |
| Hebrew | 126 | 4 | 17 | 39 | 0 | 2075 |

Table 5.5: Confusion matrix of the script classifier for text lines. The overall accuracy is $95.4\%$ on $2,155$ individual text lines.

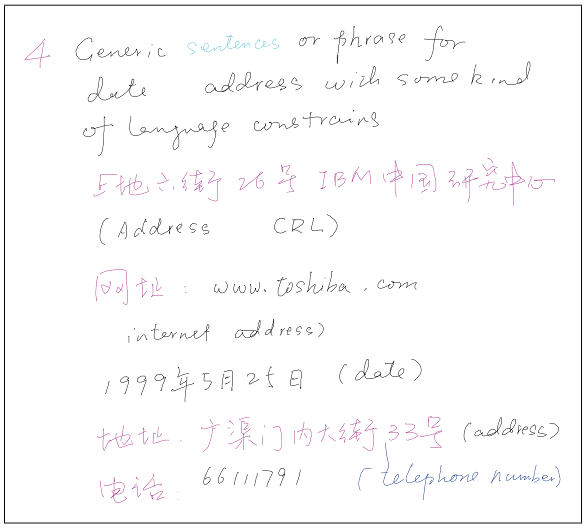
|  | Classified |  |  |  |  |  |
|---|---|---|---|---|---|---|
| True | Arabic | Devnagari | Han | Roman | Cyrillic | Hebrew |
| Arabic | 204 |  |  | 1 |  | 4 |
| Devnagari |  | 373 | 1 | 7 |  | 1 |
| Han |  |  | 276 | 5 | 1 |  |
| Roman |  | 3 | 1 | 694 | 24 |  |
| Cyrillic | 1 | 3 | 1 | 40 | 230 | 1 |
| Hebrew | 5 |  |  |  |  | 279 |


Athletics is in many ways the embodiment the Olympic motto: Citius Altius Fortius; Faster, higher stronger. The track and field events is all about running faster and jumping higher
雪梨奧運閉幕典禮

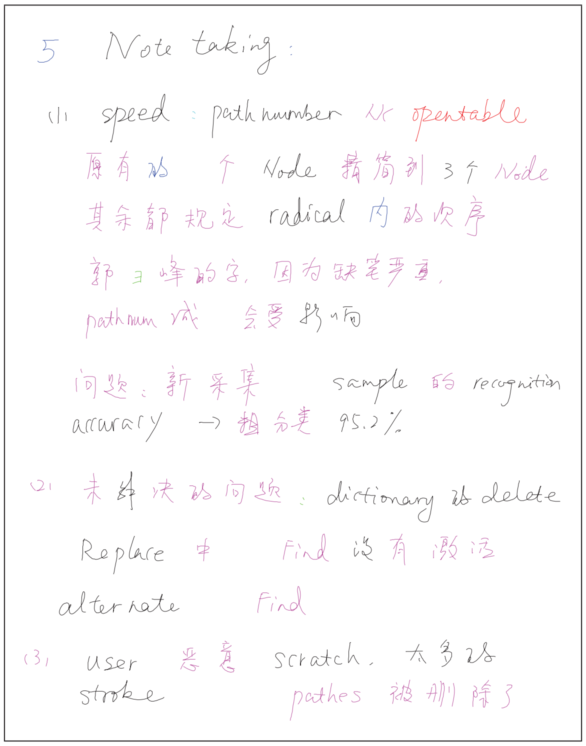
(a)


Most of the scripts currently used in the world can be classified into a few categories according to the writing system of the script.
The most common writing systems are Alphabetic, Syllabic Alphabetic, and Logographic
Alphabetic writing systems use symbols for consonants and vowels to write words of the language.
In addition to Roman, Cyrillic (кириллица), Arabic (), and Hebrew () fall into this category. Chinese or Han (這是中文) fall in the category of logographic writing system, while most of the Indian scripts including Devnagari (देवनागरी लिपी) fall into the class of Syllabic Alphabetic writing system.

(b)


4 Generic sentences or phrase for date address with some kind of language constrains
上地小街 26号 IBM中国研究中心
(Address CRL)
网址: www.toshiba.com
internet address)
1999年5月24日 (date)
地址: 广渠门内大街33号 (address)
电话: 66111791 (telephone number)

(c)


5 Note taking:
(1) speed: pathnumber 从 opentable
原有的 个 Node 精简到 3个 Node
其余都规定 radical 内的次序
郭 ヨ 峰的字, 因为缺笔严重,
pathnum 减 会受影响
问题: 新采集 sample 的 recognition
accuracy → 相差 95.2%
(2) 未解决的问题: dictionary 的 delete
Replace 中 Find 没有激活
alternate Find
(3) user 恶意 scratch, 太多的 stroke pathes 被删除了

(d)

Figure 5.8: Output of the script classifier. (a) the classification results of the document page in Figure 5.1. Figures in (b), (c), and (d) show the outputs of the classifier on three other pages. The color scheme to denote each script is the same as that used in Figure 5.1.

### 5.4.1 Combining Multiple Classifiers

The results of different classifiers may be combined to obtain better classification accuracy [69]. Multiple classifiers can be combined at different stages of the classification process such as combining feature vectors, combining posterior class probabilities, combining classifier decisions, etc. We have used a confidence level fusion technique where each classifier generates a confidence score for each of the six scripts. The confidence score is a number in the range $[0, 1]$, where $0$ indicates that the test pattern is least likely to be of the script type associated with the score while a confidence score of $1$ indicates that the test pattern is most likely to be the corresponding script type. The confidence scores generated by the individual classifiers are added up and normalized to the range $[0, 1]$ to generate the final confidence score. The script that has the highest score is selected as the true class (refereed to as the Sum Rule). In our experiments, we combined the results obtained from three best performing classifiers for script recognition - *SVM*, *KNN* (k=5), and neural network. For the SVM classifier, the output value of the discriminant function is used to generate the confidence value. This value is normalized using a sigmoid squashing function. The confidence score for the KNN classifier was computed as the proportion of neighbors that belong to the decided class. The output value of the node corresponding to the decided class gives the confidence value for the neural net classifier. The combined word-level classifier could attain an accuracy of $87.1\%$ on $5$-fold cross validation, as opposed to $86.5\%$ for the best performing individual classifier (SVM-based). The line level classification accuracy of the combined classifier was $95.5\%$, compared to $95.4\%$ accuracy of the neural net and SVM classifiers.

## 5.5 Feature Selection

An interesting question to ask in any classification system is, how many features are adequate for classification. A very effective, although sub-optimal, way to determine the best subset of features for classification, given a particular classifier, is the sequential floating search method (SFSM) [61, 27]. The features described in Section 5.3 are ordered according to their contribution to the classification accuracy. The plot in Figure 5.9 shows the increase in performance as each feature is added by the SFSM. None of the features were eliminated during the selection process even though the removal of final two features only marginally degrade the performance of the classifier. Other approaches to feature selection such as sequential forward and sequential backward searches were also used in the experiment, which resulted in a similar ordering of features. The features in the order of significance correspond to: i)Horizontal Inter-stroke Direction, ii) Average Stroke Length, iii) Shirorekha Strength, iv) Shirorekha Confidence, v) Stroke Density, vi) Aspect Ratio, vii) Reverse Distance, viii) Average Horizontal Stroke Direction, ix) Average Vertical Stroke Direction, x)Vertical Inter-stroke Direction, and xi) Variance of Stroke Length. We note that the first feature encodes the writing direction, which is important in the classification of Arabic and Hebrew scripts against the rest. The second most significant feature, the Average Stroke Length helps to distinguish the Han script from the rest. Shirorekha Strength is the most important feature that distinguishes Devnagari script. The three least significant features selected by the sequential forward method and the SFSM were the same, while the sequential backward search contained one feature that was different.

## 5.6 Classification of Contiguous Words and Text Lines

In many practical applications, contiguous words belonging to the same script are available for classification. We expect that the script recognition accuracy will improve as the number of consecutive words of text in a test sample increases. In the case of on-line script recognition, this boils down to the number of words of text that is required to make an accurate prediction. The plot in Figure 5.10 shows the increase in accuracy of the classifier as a function of the number of words in a test sample. In the case of multiple words, all the

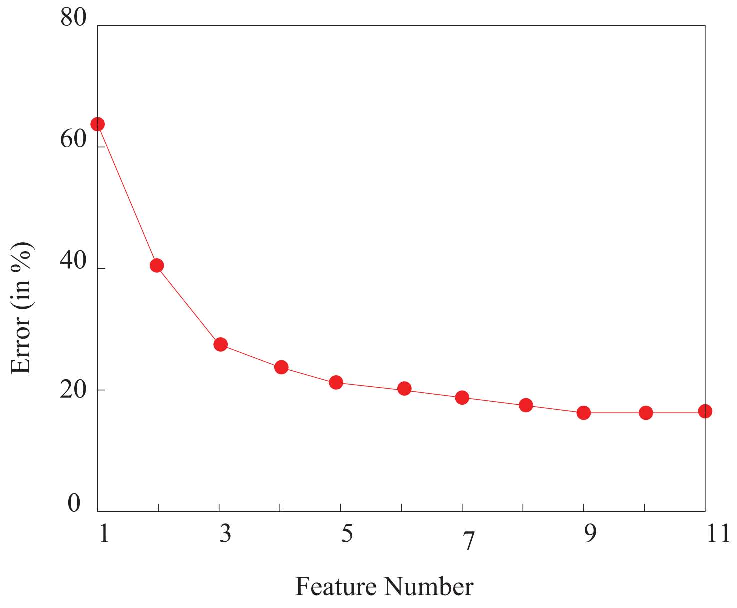


Figure 5.9: Word-level classification performance as individual features are added by the sequential floating search method.

strokes in the words are used to compute a single feature vector, and hence the computation of the feature vector is more reliable. We notice that with five words, we can make a highly

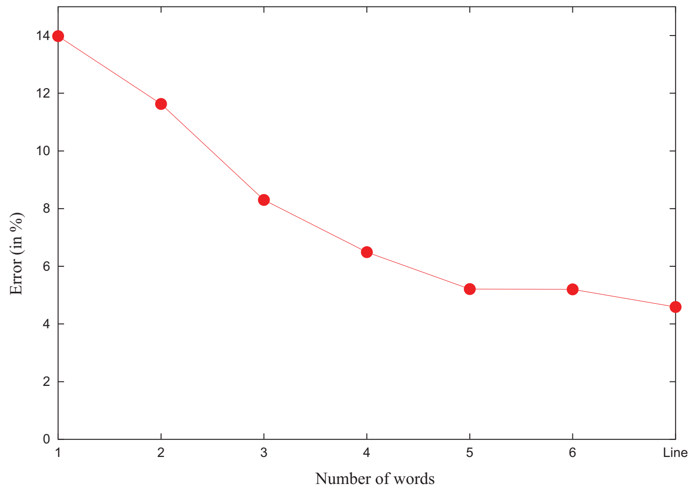


Figure 5.10: Multi-word classification performance with increasing number of words in a test sample (neural net classifier).

accurate (95% accurate) classification of the script of the text. The script classification accuracy improves to 95.5% when we use an entire text line, consisting of, on average, seven words. The error rate was computed on 5-fold cross validation with a standard deviation of 0.6%.

The accuracy of prediction of the script of a single word also depends on the length of the word. A measure of word length, which can be employed in the case of on-line data, is the number of strokes in the word. The plot in Figure 5.11 shows the error rate of classification of single words according to the number of strokes in it. We notice that most of the errors are made in the case of single-stroke words and the accuracy of classification improves considerably as the number of strokes in the word increases (upto 89% for 5-stroke words). These results give us an indication of the improvement in performance as the amount of data increases (evidence accumulation).

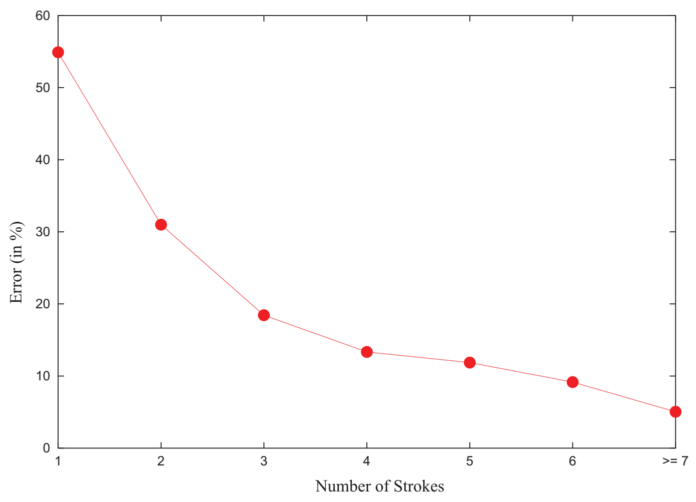


Figure 5.11: Classification performance with increasing number of strokes in a word. The x-axis shows the number of strokes in the word and the y-axis shows the classification error in percentage (neural net classifier).

## 5.7 Summary

We present a script identification algorithm to recognize six major scripts in an on-line document. The aim is to facilitate text recognition and to allow script-based retrieval of on-line handwritten documents on the Internet. The classification is done at the word level, which allows us to detect individual words of a particular script present within the text of another script. The classification accuracies reported here are much higher than those reported in the case of script identification of off-line handwritten documents till date, although the reader should bear in mind that the complexities of the two problems are different.

# Chapter 6

# Matching and Retrieval

The most efficient method for storage of handwritten text would be to convert the text into ASCII code using a text recognizer and store the resulting text document. This method enables efficient storage and retrieval of documents based on keywords. However, the accuracy of text recognizers is highly dependent on the individual writing styles. Text recognizers also need user intervention for reliable translation. Moreover, handwritten data often contain diagrams and sketches and so it is desirable to store the raw data (*digital ink*) as well. The limitations of recognition algorithms and the expressive power of handwriting over text have led researchers to explore the idea of using digital ink as a basic data type. Another reason for requiring a recognition-free solution is the multitude of languages and symbols that an application using pen-based input needs to handle.

An alternate approach to retrieval of handwritten data is to use a search technique that can find a document in a database using a handwritten keyword as a query. The process of comparing handwritten or spoken words in a document, without explicit recognition is referred to as 'word-spotting' [90]. In this chapter, we describe a retrieval algorithm that is based on word-spotting of on-line words. The technique assumes that the document structure identification, described in Chapter 4, is used to identify words in an on-line document.

## 6.1 Indexing versus Retrieval

Indexing is the problem of dividing the words in a database into equivalence classes, each consisting of instances of a specific word. Indexing algorithms use a similarity measure for pairwise comparison of words to form clusters of words. Representative instances from each cluster/class are then used to create a word index for the database. In contrast, the problem of word-spotting deals with retrieval of documents by comparing a keyword with individual words in the documents in the database (see Figure 6.1(a)). Although the indexing process may seem functionally similar to retrieval, there are many aspects of indexing applications that make it different from retrieval [90]. However, the critical factor in solving both the indexing and retrieval problems is the choice of a distance measure between words, which has a small value (dissimilarity) for the instances of the same word and a large value for different words.

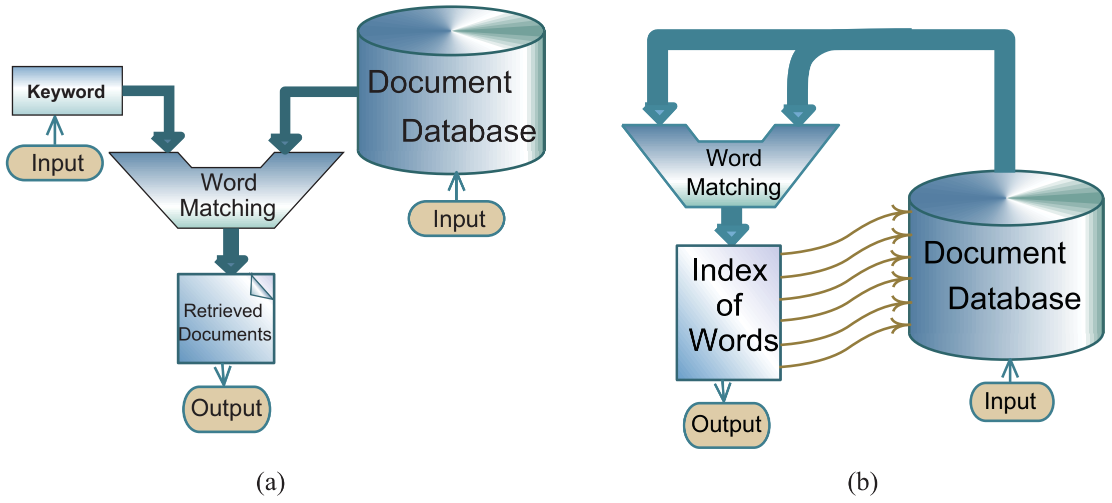


Figure 6.1: Indexing and retrieval problems. Schematic diagram of (a) a word-spotting based retrieval system and (b) an indexing system.

The retrieval and indexing systems based on word-spotting is independent of the language/script of the document. Further, the keyword need not even be a legal text as long as the 'writing' to be spotted is properly segmented from the pages in the database.

Document retrieval refers to the problem of selecting a set of documents from a database that matches a specific query. The query can be in the form of a set of typed words or an example of a document, such as an image or a handwritten word. Figure 6.2 shows examples of the different query types. Figure 6.1(b) shows a schematic diagram of the retrieval process.

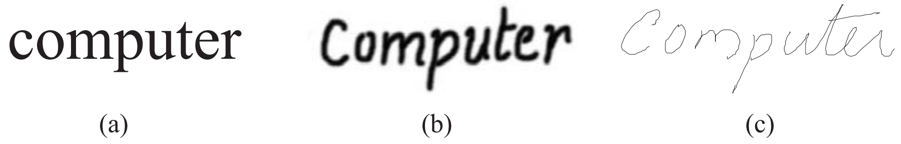


Figure 6.2: Query types: (a) a typed query, (b) an off-line handwritten query, and (c) an on-line handwritten query. The text in (a) is in ASCII, (b) is an image, and (c) is on-line data.

Indexing is the process of dividing the words in a database into equivalence classes, each containing occurrences of a single word. Once the classes are generated, an index of the database can be created using representative words from each class. Although the indexing process may seem functionally similar to retrieval, there are many aspects of indexing applications that make it different from retrieval.

1. The matching process during indexing involves classifying each word instance in the database into one of *n* classes, where *n* is the number of distinct words in the database. In contrast, the retrieval involves classifying each word as a *match* or *no-match* to the keyword being searched.

2. The matching process in retrieval, i.e. during the creation of the database index, is done off-line (not in real-time). Hence we could potentially employ more powerful and time-consuming methods for matching.

3. Multiple instances of a word may be available while doing the indexing, which can be used to construct models of words and refine the classification.

4. Since the document index is typically used multiple times for a particular database, user feedback can be used to correct the errors in the database, once it is generated.

## 6.2 Word Matching

The method used for word matching is based on a string matching technique, called *dynamic time warping*. Dynamic time warping (DTW) is a powerful method for comparing two sequences of data points. To employ DTW, we first preprocess the keyword that is provided by the user as described in Section 5.2. The processed keyword is used as a template string for matching purposes. The words in the documents to be searched are extracted using the techniques mentioned in the previous section. The following three features are computed at each sample point in the word, resulting in a sequence of feature vectors (see Figure 6.3):

1. The height ($y$) of the sample point: This is the distance of the sample point from the base of the word. The base of the word is defined as the horizontal line passing through the lowest sample point in the word.

2. The stroke direction at $p$: The stroke direction at $p$ is defined as the positive or counter clock-wise angle between horizontal axis and the tangent of stroke at $p$. The computation of the tangent direction is described in Chapter 3. The writing order is important in the computation of this feature.

3. The curvature of the stroke at point $p$: The computation of curvature of a stroke at point $p$ is described in Chapter 3. However, the curvature is not defined for points where the stroke direction changes abruptly. Hence we use the angle subtended by the neighboring points at $p$ as a measure of the curvature.

Each word in a document is compared with the keyword. The word to be compared is first scaled so that it is of the same height as the keyword, and translated so that both the words have the same centroid. The DTW technique then aligns the feature vector sequence from a database word to that of the keyword using a dynamic programming-based algorithm. The algorithm computes a distance score for matching points by finding

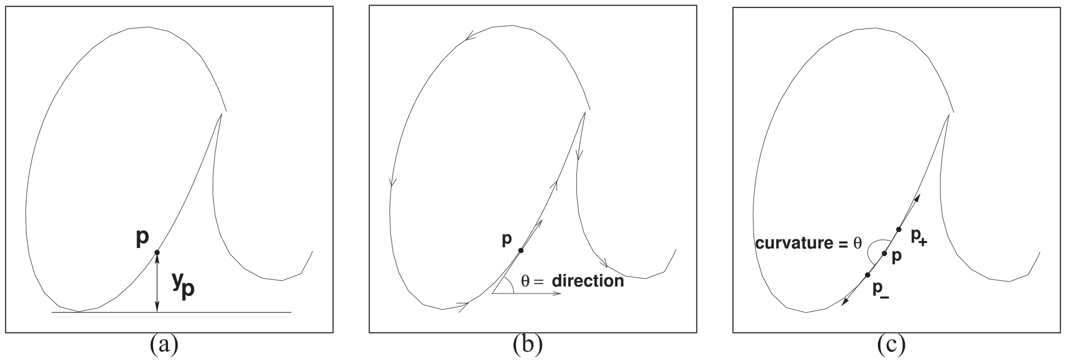


Figure 6.3: Computing word features. (a) height ($y_p$) at point $p$, (b) the direction feature, and (c) computation of curvature at $p$.

the Euclidean distance between corresponding feature vectors and penalizes missing or spurious points in the word being tested. The optimal alignment between the two words is computed, with each feature weighted differently. The weights are learned during the training phase of the algorithm ($W_y = 3.0$, $W_{dir} = 3.0$ and $W_{curv} = 1.0$).

The DTW-distance between two point sequences, $W_1$: $p_1, p_2, \ldots, p_n$ and $W_2$: $q_1, q_2, \ldots, q_m$, denoted by $D(W_1, W_2)$ is given by the recurrence relation:

$$D(i,j) = min \begin{cases} D(i, j-1) + p_m \\ D(i-1, j) + p_s \\ D(i-1, j-1) + dist(p_i, q_j), \end{cases} \tag{6.1}$$

where $p_m$ and $p_s$ are the penalties for missing and spurious points in the test word ($W_2$). The penalties $p_m$ and $p_s$ are set to be equal and experimentally determined to be $200$ for the sampling resolution of the CrossPad ($250$ $dpi$). The function $dist(p_i, q_j)$ is the distance measure between two points $p_i$ and $q_j$. As mentioned above, the distance measure used for word-spotting was the weighted euclidian distance.

Figure 6.4 illustrates the correspondence between two words in the database. Note that our algorithm does not match the spatial coordinates, but matches feature vectors computed at the corresponding points. The distance score computed is scaled based on the length of the keyword. Since longer strings tend to have larger values of distances from the keyword, we normalize the distance measure by dividing it with the sequence length.

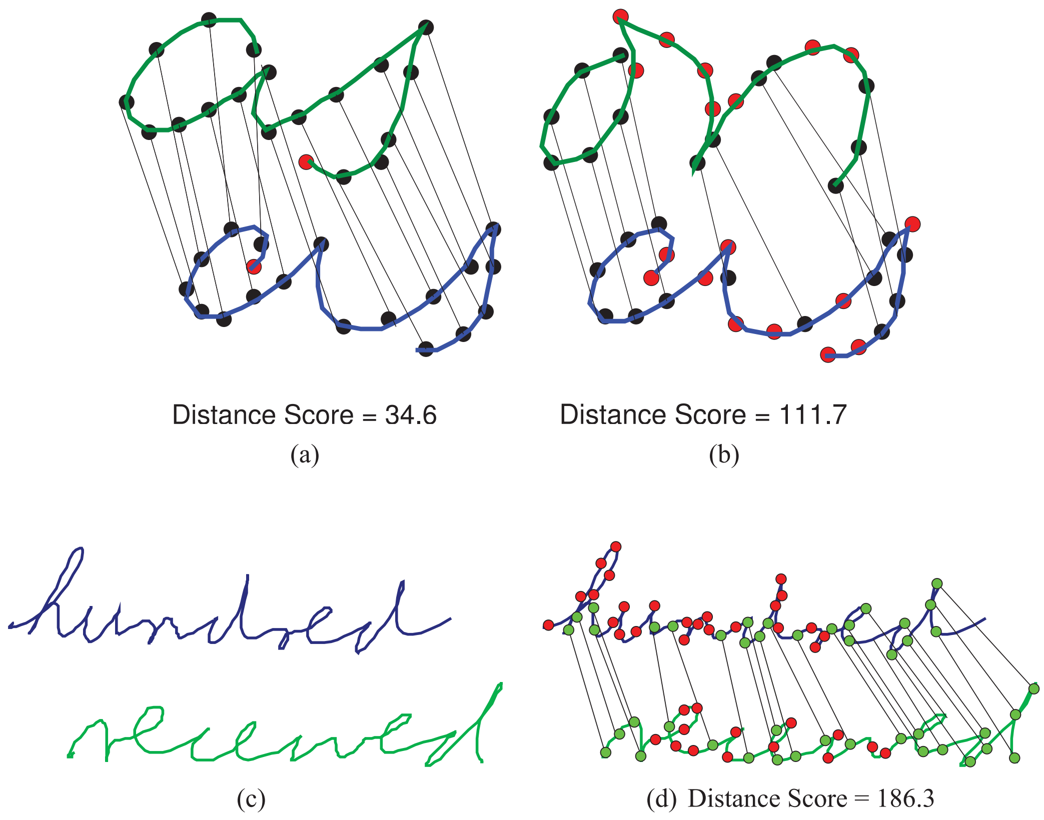


Figure 6.4: Correspondences between (a) two instances of the word *as*, and (b) between the words *on* and *as*. Partial match between two words (c) is shown in (d). The matching sample points are shown in green and the non-matching ones in red.

The string matching algorithm uses a number of parameters including the weights for each of the features and the penalties for missing (or spurious) points in the sequence. These parameters were determined from an independent training set collected from a single user. The training set consisted of 5 occurrences of 56 different words and were written in a single session.

## 6.3 Sketch Matching

The problem of matching on-line sketches is specifically interesting due to the challenges it poses, and its differences with off-line matching of shapes. On-line data captures the temporal sequence of strokes[1] while drawing a sketch. Most of the off-line sketch matching algorithms deal with the problem of matching a hand-drawn shape to an image. The intra-class variability of shapes in images is usually small, since the images are formed from natural objects. However, in the case of hand-drawn sketches, the intra-class variability can be much higher due to a variety of reasons. They include variability due to drawing styles, drawing sequence, overwriting or corrections, etc. We can convert the on-line sketch to an off-line image to avoid some of these variations. However, we will lose the ability to use the stroke information for efficient matching in that case.

### 6.3.1 On-line Sketch Matching

A sketch matching algorithm could compare two sketches using a distance measure such as dynamic time warping. However, such an approach will not be able to deal with the variability in hand-drawn sketches due to changes in the order of drawing the strokes. Figure 6.5 shows two similar shapes, drawn using different stroke orders and with different number of strokes. In addition, a holistic matching of sketches would be computationally expensive for retrieval applications, where we need to compare a query sketch against a large set of sketches in a database. To deal with this problem, the sketch matching algorithm should be able to represent each sketch in a compact form, which can be compared efficiently and effectively. Representing a sketch using a set of global properties, such as total stroke length, perimeter efficiency, etc. would be compact, but it cannot capture the salient details of all the possible sketches and hence, will not be an effective representation.

A common approach, used by Lopresti et al. [89] and Leung and Chen [80] is to represent a sketch in terms of a set of basic shapes. Lopresti et al. [89] refer to these basic shapes as 'motifs', whose identification and matching are not provided in detail in their paper. Leung and Chen [80] proposed a recognition-based approach for matching hand drawn sketches by identifying basic shapes such as circles, straight lines and polygons in

[1]A stroke is defined as the locus of the tip of the pen from pen-down to the next pen-up position.

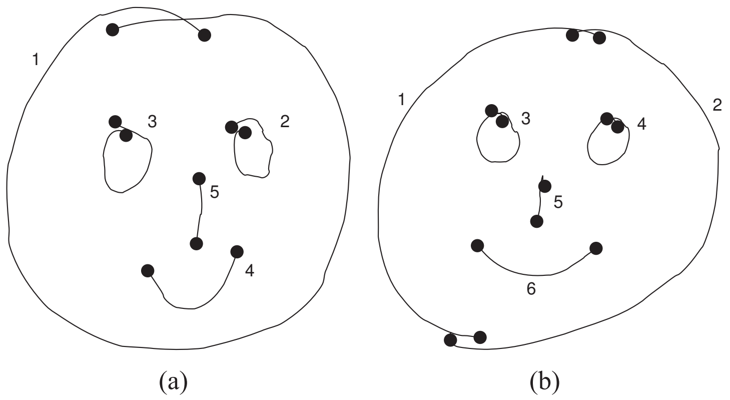


Figure 6.5: On-line sketch variability: Two similar sketches, drawn using different stroke orders and with different number of strokes. The numbers indicate the order of the strokes and the dots indicate the end points of the strokes.

the sketch. The problem of developing a set of basic shapes for representing a generic set of sketches is difficult at best. Once the basic shapes in a sketch are identified, one needs to compare the two sketches, based on the identified shapes. This involves a comparison of two shapes and their spatial arrangement in the two sketches. We note that an exhaustive comparison of two sets of shapes to arrive at the best matching is exponential in time. We present a matching algorithm that tries to overcome the problems with an efficient representation and similarity measure.

### 6.3.2 Proposed Algorithm

The matching algorithm consists of three stages. The first stage tries to eliminate the noise introduced in the sketches during data capture due to noise from the sensor, quantization, etc. In the second stage, we divide the traces in the sketch into smaller units. This is followed by a matching stage, where the sketches are compared based on their global properties and the properties of the individual portions of the strokes in the two sketches. The input data for the matching algorithm can be noisy due to a variety of factors. We use the spline-based representation described in Chapter 3 to smoothen the strokes in the sketch.

One of the main problems in comparing sketches based on a set of basic shapes is the identification of the basic shapes from a sketch. The problem is similar to a structural pattern recognition [105, 106] problem, where a pattern class is described in terms of a set of structural primitives. We overcome this problem by representing the sketches as a set of stroke segments instead of a pre-defined set of shapes. After pre-processing, each stroke is divided at the points where the $x$ or $y$ direction of a stroke changes. Each resulting stroke segment is represented using a line segments. Figure 6.6 shows a sketch, where the stroke segments are identified and represented as a set of line segments.

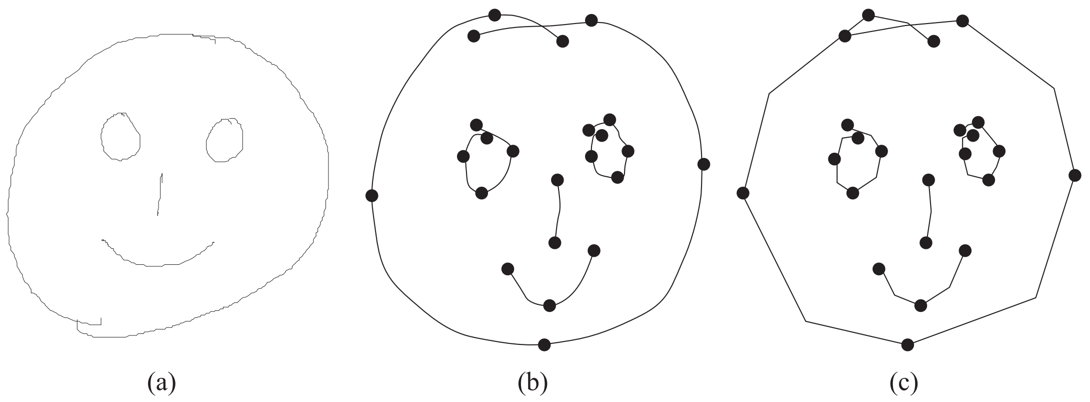


Figure 6.6: Segmentation of strokes: (a) an input sketch, (b) the critical points in the sketch, and (c) the representation of the sketch as a set of straight lines.

Let $\{\zeta_1, \zeta_2, \cdots, \zeta_n\}$ be the set of segments identified from a stroke $\xi$ in the sketch $S$. A sketch with $m$ strokes is hence represented as:

$$S = \xi_1 \cup \xi_2 \cup \cdots \cup \xi_m = \{\zeta_1, \zeta_2, \cdots, \zeta_k\}, \tag{6.2}$$

where $k$ is the total number of stroke segments identified from all the $m$ strokes in the sketch. Each stroke segment $\zeta_i$ is represented using its position, direction, length, and curvature, $<P_i, D_i, L_i, C_i>$. Consider a stroke segment, $\zeta$, containing $p$ sample points.

$$\zeta =< (x_1, y_1), (x_2, y_2), \cdots, (x_p, y_p) > . \tag{6.3}$$

The position, direction and length of the segment are computed as:

$$\begin{aligned} Position &= \left( \sum_{j=1}^{p} x_j/p, \sum_{j=1}^{p} y_j/p \right) \\ Direction &= arctan\left( \frac{y_p - y_1}{x_p - x_1} \right) \\ Length &= \sqrt{}((x_p - x_1)^2 + (y_p - y_1)^2) \end{aligned} \qquad (6.4)$$

The $curvature$ is measured as the angle subtended by the end points of the stroke segment, $(x_1, y_1)$ and $(x_p, y_p)$ at its mid point, $(x_{(1+p)/2}, y_{(1+p)/2})$. The value of direction, in degrees, is restricted to the range $[-180, 180]$ to avoid the difference between strokes drawn in opposite directions during comparisons. The value of position and length features are normalized using the size of the sketch, which is defined as ($length + width$). The value of curvature, in degrees, falls in the range $[-180, 180]$.

The comparison of two sketches involves computing the best match between the two sets of stroke segments. The matching distance between two stroke segments is defined as a weighted euclidian distance.

$$\begin{aligned} & d(\zeta_i, \zeta_j) = d(< P_i, D_i, L_i, C_i >, < P_j, D_j, L_j, C_j >) \\ & = w_p.|P_i - P_j| + w_d.|D_i - D_j| + w_l.|L_i - L_j| + w_c.|C_i - C_j|, \end{aligned} \qquad (6.5)$$

where $w_p, w_d, w_l,\ and\ w_c$ are the weights corresponding to each feature. The values of the weights were determined empirically to be 3.5, 5.0, 5.5 and 4.0.

The computation of the optimal match between two sets of line segments is exponential in terms of the number of lines in the sets. Let the number of stroke segments in the two sketched to be compared be $p$ and $q$. An optimal matching requires $\frac{p!}{(p-q)!}$ comparisons. To reduce the time complexity of this computation, we use a greedy algorithm, which is possibly sub-optimal. In this method, we repeatedly select the most similar line segment pair between the two sets. The matching of two sketches takes only $O(p^2.q)$ comparisons with this algorithm. One could further refine this algorithm by incorporating heuristics for selecting the matching segment pair at each iteration, thus moving towards an optimal solution. However, the algorithm works well in practice as indicated by the experiments. We use the number of stroke segments, the total length of the strokes in a sketch to avoid matching sketches that are markedly dissimilar and thus reducing the average time for matching two sketches.

## 6.4 Experimental Results

The test set consisted of a passage of $30$ pages ($410$ lines and $3,872$ words) collected from a single writer and a set of $280$ keywords ($5$ instances each of $56$ words) collected from $10$ writers (a total of $2,800$ words). The database was collected over a period of two weeks in over five sessions. To measure the performance of the algorithm, the words in the database were manually labelled (ground truth) after detection of lines and words (a total of $6,672$ words). Figure 6.7 shows examples of pages from our on-line database. More examples of documents in the database can be found in Appendix B.

The performance of a retrieval algorithm is typically measured using two quantities: *precision* and *recall*. Consider an algorithm that retrieves $n$ items (words or documents) from a database of $N$ items for a particular query. Further, assume that $k$ of the retrieved items match the query, and the actual number of items in the database that match the query is $K$. The precision and recall are defined as follows:

*Precision* refers to the fraction of the retrieved items that match the query. $Precision = k/n$.

*Recall* refers to the fraction of items that match the query in the database, which was retrieved by the algorithm. $Recall = k/K$.

### 6.4.1 Document Retrieval

Every keyword in the database was compared with every other word written by the same writer and the results were used to measure the performance of the algorithm. For the passage, the set of keywords written by the same writer was used for document retrieval. Figure B.4 in Appendix B shows examples of documents in the database used for this experiment.

Figure 6.8 shows the Precision-Recall curves for ten different users who contributed to the database. The results reported by Lopresti and Tomkins [88] are also plotted along with these curves for comparison (not for the same data). This is the only work, to our knowledge, that has reported the Precision-Recall curve for on-line documents. The accuracy of our algorithm was $92.3\%$ at a recall rate of $90\%$ averaged over the whole database ($6,672$ words). This is significantly better than the results in [88].

### 6.4.2 Document Indexing

The indexing experiment was carried out on the passage of $30$ pages. We discard short words (consisting of $3$ characters or less) to avoid articles, prepositions, etc. The word matching algorithm computes the distance between every pair of words; word pairs with a distance less than a threshold (determined by $90\%$ recall rate) were joined together to form word classes. The label (ground truth) of every class is set to the word that is in majority in the class. Each label occurs only once in the index. Every word instance that is not classified under its label is counted as an error. Our algorithm achieved an accuracy of $87.5\%$ with $1,461$ words (after removing short words, the original database of $3,872$ words was reduced to $1,461$ words) being clustered into $943$ word classes. Most of the errors were due to grouping/clustering of similar words (e.g., *foot* and *feet*).

The algorithm takes approximately $9\ msecs$, on an average, for a word comparison on a machine with an Intel P-III 650 MHz CPU and $192$ MB of RAM.

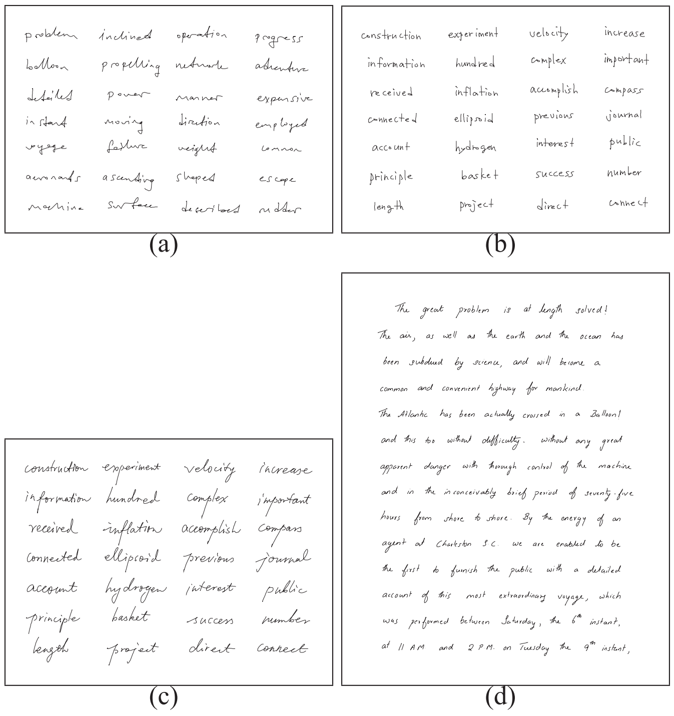


Figure 6.7: Sample pages from the database. (a)-(c) samples of keywords from 3 different writers. (d) sample page from the passage.

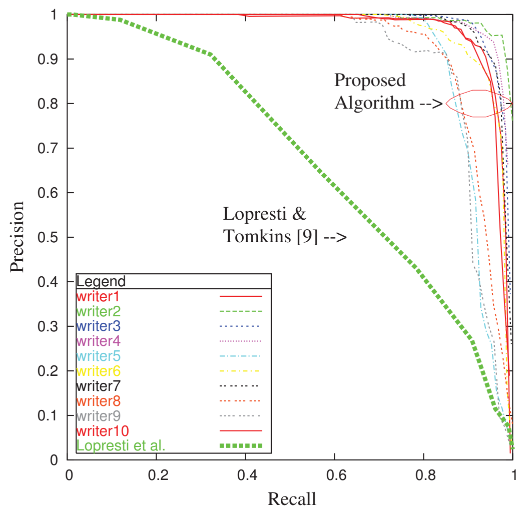


Figure 6.8: The precision vs. recall curves. Note that the results of Lopresti and Tomkins are not for the same data as the work done in this thesis.

### 6.4.3 Sketch Matching

The database consists of $60$ instances of $15$ different sketches, collected from $20$ users (a total of $900$ sketches), with each user contributing two instances of each sketch over a period of one week. Figure 6.9 shows an example of each of the sketches in the database. The data was collected using the *CrossPad*® . The CrossPad has a pen and paper interface along with the ability to digitally capture the (x,y) position of the pen tip at a resolution of $254\ dpi$. The pen position is sampled at a constant rate of $128\ Hz$.

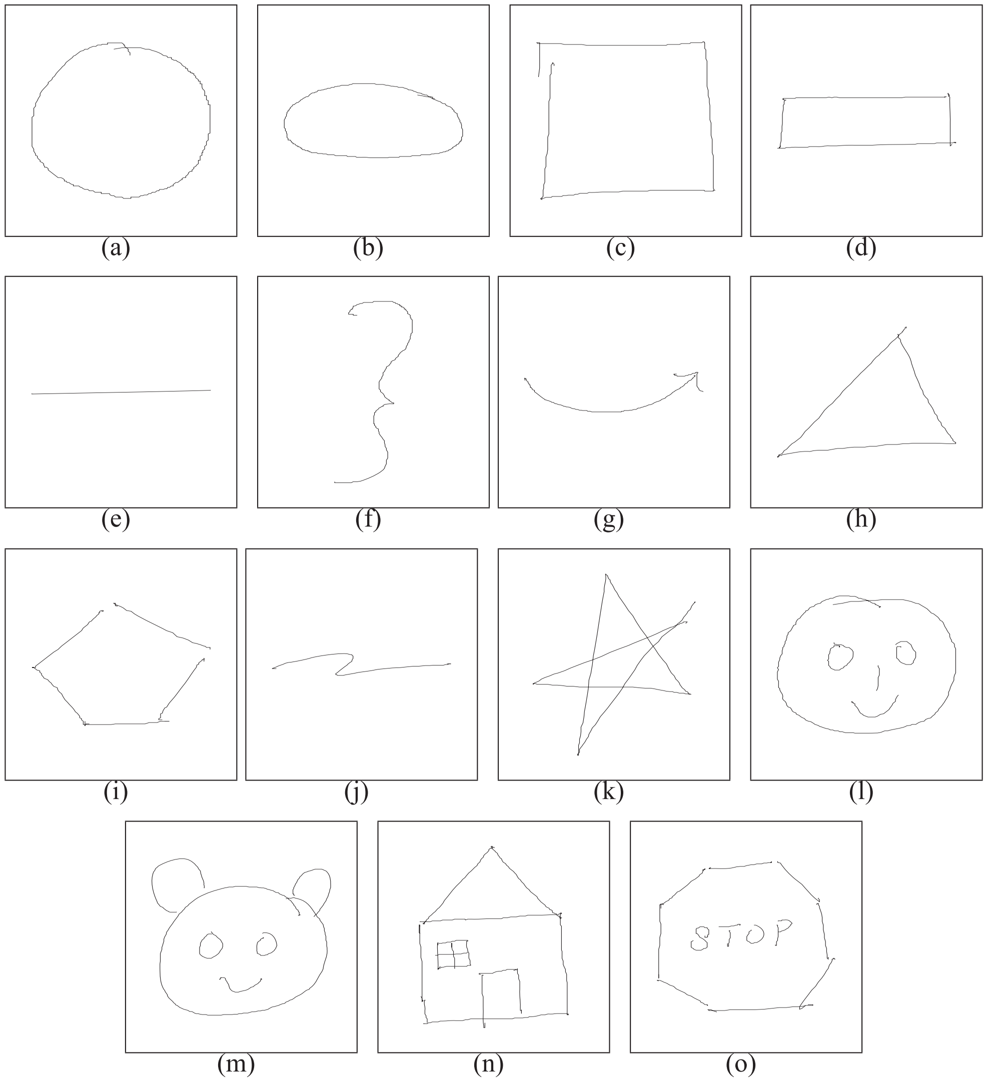


Figure 6.9: Examples from the on-line database of sketches.

During testing, each sketch in the database was used as a query to search for similar

| Top $N$ Retrieval | Accuracy |
|---|---|
| 1 | 98.3 % |
| 2 | 97.9 % |
| 3 | 97.2 % |
| 4 | 96.8 % |
| 5 | 96.6 % |

Table 6.1: Retrieval results: Accuracy of matching among the top $N$ retrievals.

sketches. The query sketch was matched against every other sketch in the database. The results of each query was sorted in increasing order of the matching distance. The matching accuracy is measured as the percentage of correct retrievals in the top $N$ sketches in the sorted list. Table 6.1 shows the results of our retrieval experiments.

The results of matching revealed that most of the errors were committed by false matching between similar looking shapes such as the circle and the ellipse. Figures 6.10(a) and 6.10(b) show the example of such a retrieval. A second source of error was query figures in the database that were poorly drawn. Figure 6.10 shows examples of incorrect matches among the top-$5$ retrievals from the database. The query sketch in Figure 6.10(c) is a star, such as in Figure 6.9(k), which was incorrectly matched with a triangle. The accuracy of retrieval could also be measured in terms of the precision and recall rates. Precision refers to the proportion of relevant (or correct) sketches among those retrieved, and recall refers to the proportion of relevant sketches in the database that was retrieved using a query. The proposed algorithm achieved a precision of $80\%$ at a recall rate of $73\%$. Leung and Chen [80], performed their experiments on a set of $2800$ sketches, collected from $4$ writers belonging to $35$ sketch types. They reported a precision of $80\%$ at a recall rate of $50\%$. However the results are not directly comparable due to the differences in the databases. The proposed algorithm takes approximately $75$ $msec$ to compare two sketches, which includes the time for reading in the sketches from a file.

The algorithm is also sufficiently fast to be used in real time searches of databases. The time to compare two sketches is approximately $75$ $msecs$ on a Sun Ultra 10 machine, which

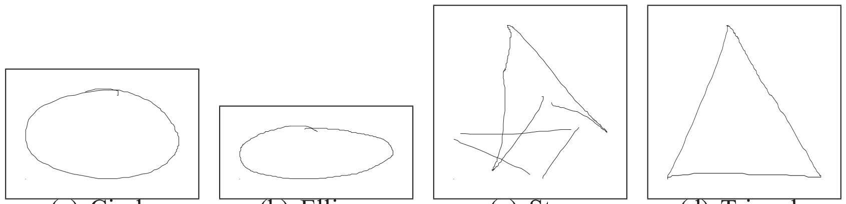

(a) Circle (b) Ellipse (c) Star (d) Triangle

Figure 6.10: Examples of incorrect matches among the top-$5$ retrievals. (a) and (c) show two query sketches and (b) and (d) show the corresponding retrievals from the database.

includes the time for reading in the sketches from a file. In practice, the matching can be much faster as the pre-processing and feature extraction of the sketches in the database can be done off-line.

## 6.5 Summary

We have proposed a string matching based word-spotting algorithm using features computed from the strokes of individual words. The algorithm achieves a precision of $93.2\%$ at a recall rate of $90\%$ averaged over $10$ writers. Indexing experiments show an accuracy of $87.5\%$ using a database of $3,872$ words. These accuracies are quite good considering the large intra-class variability encountered in on-line handwritten words. We are currently extending our algorithm to match non-text regions such as line drawings. Computation of the feature vectors can be adapted to handle devices of different resolutions. The algorithm may also be optimized to improve the run time performance [70].

We have also proposed an on-line sketch retrieval algorithm that computes the similarity between two sketches based on a line-based representation of the sketches. The algorithm achieves a precision of $88.5\%$ at the same recall rate. The representation allows us to overcome the difficulties associated with shape-based matching algorithms. We note that the high-level temporal features of the sketches are not very useful for the purpose of matching in a database of generic sketches. The algorithm can also be used for retrieval of printed or off-line hand-drawn sketches using an appropriate line detection algorithm.

# Chapter 7

# Securing On-line Documents

Authentication of digital documents is an important concern as digital documents replace the traditional paper-based documents. This is especially important as digital documents are exchanged over the Internet and can easily be accessed or modified by intruders. Multimedia data contains information in the form of audio, video, still images, etc. Large amounts of multimedia data is being made available in many digital repositories such as newspaper and television web sites and museum databases that archive historic documents. This increases the need of authentication and verification of document integrity for users of such data. One of the well-known methods used for authentication of digital documents is the public key encryption-based authentication [131]. However, the encryption-based method is not suitable for widespread distribution of a document since it needs to be decrypted by each recipient, before using it or additional data should be tagged along with the document. An alternate approach uses digital watermarking [159] to ascertain the source/origin of the document, where a signature string is embedded in the document in such a way that the contents of the document are not altered. Watermarking can also be used in conjunction with encryption-based authentication techniques to provide an additional level of security in document authentication.

The work till now in the field of document watermarking, deals with off-line documents and images. In this chapter, we propose a technique that uses on-line signatures for watermarking off-line documents. The work will also be extended to watermark on-line documents in the future.

## 7.1 Establishing Document Integrity

Authentication of a document image using watermarking should ensure that the document has not been altered from the time it was created and signed by the author to the time it was received at the destination. This means that any alteration in the document, however small, should be detectable during the watermark extraction stage. For the purpose of detecting tampering, it is desirable for a fragile watermarking system to have the following properties:

1. The authentication system should be able to detect areas of document image that have been altered after watermarking, in addition to, detecting whether the document has been altered at all.

2. The watermark should be invisible to a person who does not know the secret key and it should be secure against attacks to extract the watermark.

3. The watermarking process should not perceptually alter the document image. In the case of on-line documents, this also means that the watermarking should not affect any of the processing steps mentioned in earlier chapters.

To accomplish the first goal, the watermark detection scheme should compute a watermark function $WM()$ for every pixel of the document image and compare the function value against a reference value (typically the watermark image). To detect any altered pixels, most verification (fragile) watermarking algorithms (e.g., Yeung and Mintzer [160]) use a binary image of the same size as the host image as a watermark. The watermark image is any binary image, whose pixels could be shuffled to reduce the chances of watermark detection (see Voyatzis and Pitas [154]) using techniques such as vector quantization. This technique was used by Yeung and Pankanti [161] for watermarking fingerprint images. In addition, the pixel values in the host image should be altered as little as possible to preserve the appearance of the host document.

### 7.1.1 Watermarking

Traditional watermarking techniques embed a character string, such as the name of the author in a document. However, this does not guarantee the document source, since anyone can watermark a document with a particular name. The use of a biometric trait, such as signatures, can increase the utility of watermarking techniques for authentication. In addition to the long-standing acceptance of signatures for (paper) document authentication, watermarking techniques using signatures can assure that the document was not tampered with after it was signed. The embedding of a biometric characteristic such as signature in a document also enables us to verify the signature of the author using a database of signatures, thus reducing the chance of a forgery.

In this work, we use the on-line signature [53] of the author, captured at the time of signing the document, to generate a watermark image. The use of a biometric characteristic such as signature as the watermark image, provides an added level of security in the authentication process. In addition to ensuring that the document has not been altered, one could also use the extracted watermark, namely the on-line signature, to verify the identity of the claimed author. Moreover, the need of a secret watermark image is avoided by using the signature as a watermark. Signatures have the advantage over other biometric characteristics that it has a long history of being accepted for document authentication in the past. However, the intra-class variability of the signatures (see Figure 7.1) poses a new challenge, since the watermark image would be different every time a new instance of a user's signature is used to create a watermark image.

Figure 7.2 gives a schematic diagram of the watermark embedding and detection stages

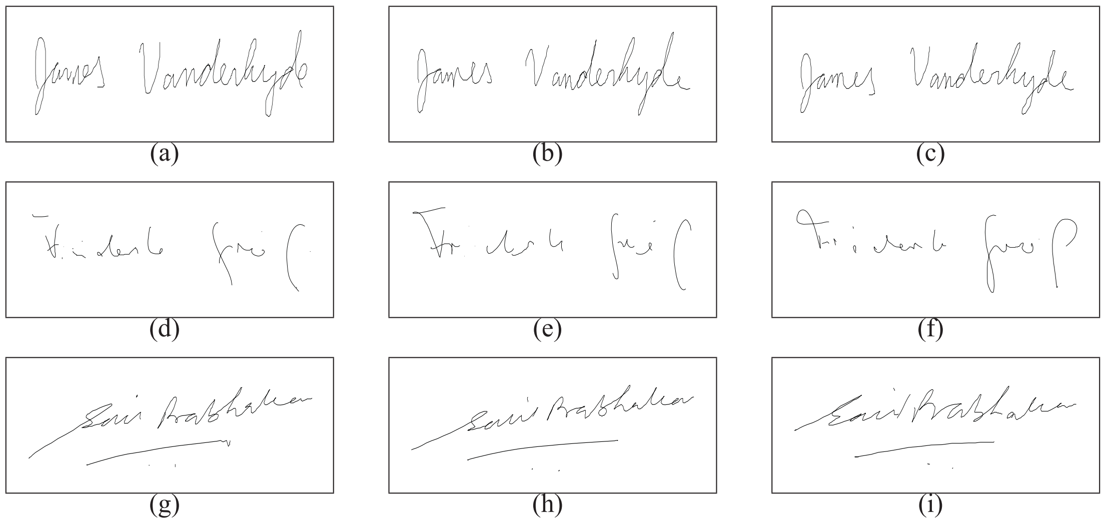


Figure 7.1: Intra-class variability of on-line signatures: Signatures from three different users collected at different times. In addition to the shape of the signatures, there are variations in the speed at various points of the signatures.

of our algorithm.

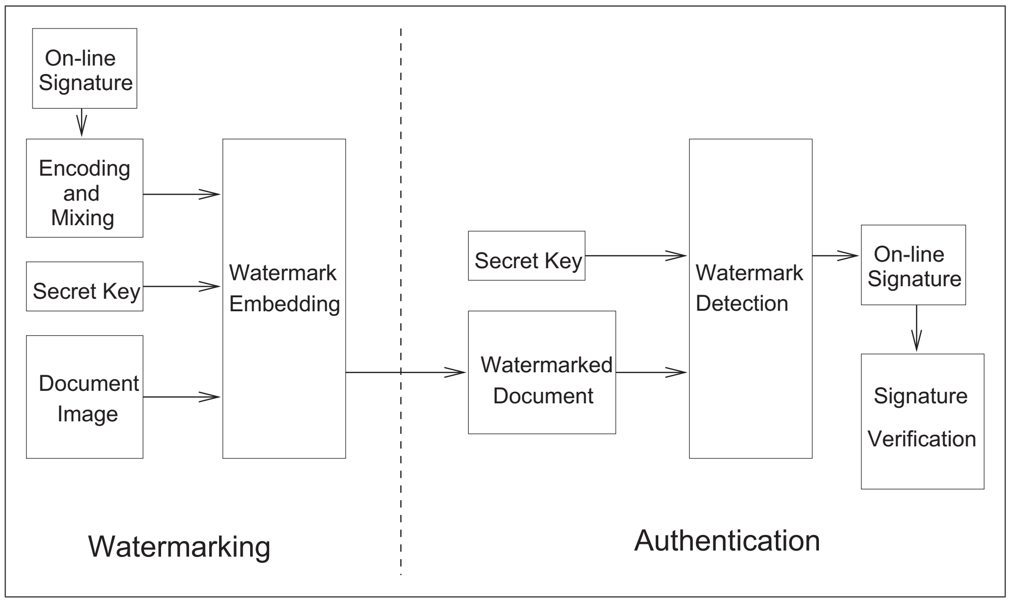


Figure 7.2: Schematic diagram of a fragile watermarking and authentication system.

### 7.1.2 Generating the Watermark Image

The on-line signature of the author is re-sampled so that consecutive sample points along the signature are atmost $8$ pixels apart. This ensures that a differential coding of the signature can use $4$-bit numbers. A typical signature has around $200$ sample points in our experiments. Each sample point is encoded using a single byte based on its distance from the previous sample point. The signature is then converted to a bit stream that is repeated to form an image of the same size as the document image. A toral automorphism is then applied to the watermark image to make it random [154]. Since the toral automorphisms are applied on square lattices, the watermark image for a host image of size $M \times N$ is constructed to be of size $N \times N$, where $N \leq M$.

A toral automorphism is a bijection from a rectangular array (or lattice) to another array of the same size. Let the square lattice under consideration be denoted by: $\mathcal{L} : [0, N-1] \times [0, N-1]$. The grid $\mathcal{L}$ is a special case, where the point locations are integers. In this case, the range of the bijection is the integer lattice itself. In addition, it can be shown that the sequence of points obtained by applying the automorphism can be shown to be periodic [5].

A common family of automorphisms is characterized by a single parameter $k$, and can be represented by the generic form given in Equation (7.1) [110]. We use the automorphism obtained by setting $k = 1$ in Equation (7.1), *i.e.*, $\left(\begin{smallmatrix}1 & 1\\ 1 & 2\end{smallmatrix}\right)$. The inverse of the automorphism is used to recover the watermark image during extraction.

$$A_N(k) : \mathcal{L} \rightarrow \mathcal{L} : \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} = \begin{pmatrix} 1 & 1 \\ k & (k+1) \end{pmatrix} \begin{pmatrix} x_{n+1} \\ y_{n+1} \end{pmatrix} (modN). \tag{7.1}$$

We denote the watermark image generated from the signature as $S(i, j)$. Figure 7.3 shows the generation of the watermark image from an on-line signature.

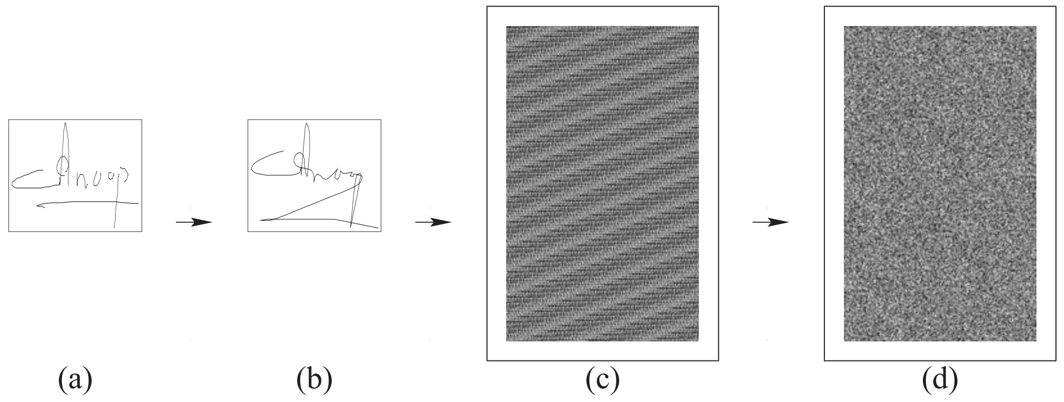


Figure 7.3: Generation of watermark image from an on-line signature: (a) on-line signature of the author, (b) the signature, whose strokes are connected together to form a single stroke, and (c) the binary image obtained from (b). The binary image in (b) is shuffled to get the watermark image in (d).

### 7.1.3 Embedding the Watermark

The process of embedding the watermark image into the host image (document image) should ensure that the pixel values are modified by the least amount possible to avoid any perceptual change in the document. The watermark function, $WM()$, is generated using a pseudo random sequence, $R_i$, of length $256$. The mapping is defined as: $WM(i) = R_i mod 1$, which is a binary number. The watermark function is controlled by a secret key, which is used as a seed to generate the random sequence.

For each pixel in the image $I(i, j)$, we compute the watermark function $W(I(i, j))$ and compare the value against the watermark image $S(i, j)$. If the values match, the pixel in the document is left unchanged. If they do not match, the pixel value is modified by the smallest amount, such that the watermark function and the binary pixel value of the watermark image are identical. One might notice that the change in value required might be quite high in case when the binary random sequence generated contains a long series of $zeros$ or $ones$. We avoid this problem by modifying the random sequence, so that there are atmost $8$ consecutive $zeros$ or $ones$. The watermark embedding we use is similar to the one proposed by Yeung and Pankanti [161].

### 7.1.4 Watermark Detection and Authentication

During the detection stage, the watermark image is recovered using the watermark function, generated using the secret key. Once the watermark image is recovered, a reverse mapping of the mixing function used during the encoding stage will yield back the embedded signature. Note that there are multiple copies of the input signature in the watermark image (152 copies of the signature in the example in Figure 7.3). The corresponding bits in different signatures included in the watermark image are then compared to see if they agree. If the bits agree in all the signatures, we conclude that the pixel is unaltered. Note that there should be at least two instances of the signature in the document image for this approach to work. To locate the pixels that were altered, we need to recover the original watermark image at the receiving end. We use the majority among the corresponding pixel values of the different signatures as an estimate of the original signature string. Figure 7.4 shows a watermarked document that was modified after watermarking, and the result of the watermark extraction process. Those pixels where tampering was detected are shown in red. The on-line signature is then reconstructed from the watermark image. Figure 7.5 shows a watermarked document image and the recovered on-line signature.

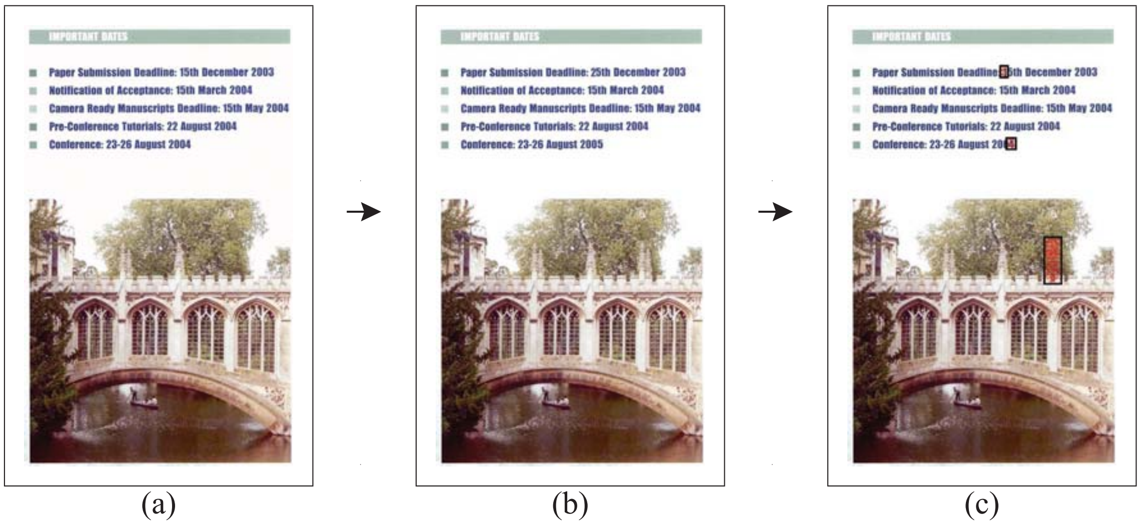

IMPORTANT DATES
Paper Submission Deadline: 15th December 2003
Notification of Acceptance: 15th March 2004
Camera Ready Manuscripts Deadline: 15th May 2004
Pre-Conference Tutorials: 22 August 2004
Conference: 23-26 August 2004
(a)
IMPORTANT DATES
Paper Submission Deadline: 25th December 2003
Notification of Acceptance: 15th March 2004
Camera Ready Manuscripts Deadline: 15th May 2004
Pre-Conference Tutorials: 22 August 2004
Conference: 23-26 August 2005
(b)
IMPORTANT DATES
Paper Submission Deadline th December 2003
Notification of Acceptance: 15th March 2004
Camera Ready Manuscripts Deadline: 15th May 2004
Pre-Conference Tutorials: 22 August 2004
Conference: 23-26 August 20
(c)

Figure 7.4: Detecting tampered locations in a document image: (a) a watermarked document image, (b) the image in (a) that has been modified at three locations, and (c) the result of watermark detection. Note that the pixels where a change is detected are shown in red.

## 7.2 Writer Identification

Once the integrity of the document is established, the extracted signature is used for verification of the author's identity. The signature matching algorithm reported in Jain et al. [53] was used for this purpose. In cases where the document in not large enough to accommodate multiple copies of the watermark image, one cannot verify its integrity. However, even in the cases where there is only a single copy of the watermark image in the document, the extracted signature would be distorted in case of alterations to the watermarked document. Hence the signature verification stage acts as an additional level of security.

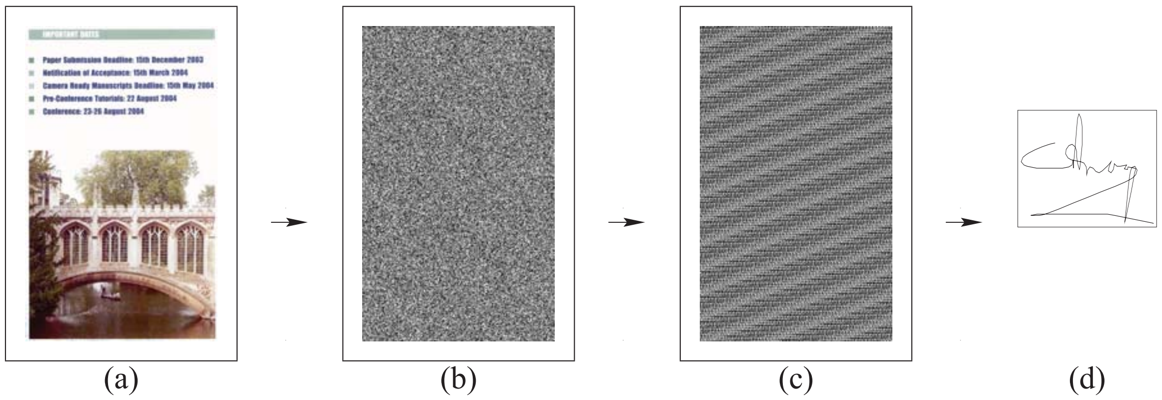


Figure 7.5: Recovering the embedded signature: (a) a watermarked document image, (b) the watermark image extracted from (a), (c) the binary image obtained by the inverse shuffling function, and (d) the on-line signature recovered from (c).

### 7.2.1 On-line Signature Verification

Handwritten signatures are commonly used to certify the contents of a document or authenticate legal transactions. Signature verification is usually done by visual inspection. A given signature is visually compared to a reference signature, say filed in a bank, and is accepted as authentic if they are sufficiently similar. In automatic signature verification, a computer takes over the task of comparing two signatures to determine if the similarity between the signatures exceeds some pre-specified threshold.

The handwritten signature is a well-known biometric attribute. Other biometric attributes that are commonly used for authentication include iris, hand geometry, face and fingerprints (See Jain et al. [56]). While attributes like iris and fingerprints generally do not change over time and thus have very small intra-class variation, they require special and relatively expensive hardware to capture the biometric image. An important advantage of the signature over other biometric attributes is that it has been traditionally used in authenticating documents and hence is socially more acceptable.

An on-line signature captures the dynamics of signing in addition to the spatial information contained in an off-line signature. This enables us to use the speed and direction of movement of the pen tip along with its spatial co-ordinates to carry out the matching. Devices such as Tablet PCs and PDAs allows us to interact using a pen or stylus and can capture the dynamic information when a user signs on the screen. The dynamic information in on-line signatures makes it extremely difficult to forge a signature. Figure 7.6 shows part of a signature by the genuine user and a forger, who was copying the signature from an off-line sample on a paper. The signature was drawn with three strokes, $blue$, $green$, and $red$ in that order. The direction of drawing of the individual strokes is denoted by an arrow, and the dots on the lines represent sample points. Note that the distance between two sample points is directly proportional to the speed of writing as the samples are collected at equal intervals of time. We can observe that the forger's signature, although similar in spatial appearance, is very different when we consider the order and direction of drawing the strokes, and the speed of writing at the corresponding spatial locations.

Figure 7.7 shows a schematic diagram of a signature verification system [53]. During enrollment of a new user, a set of reference signatures is provided by the user. This data is saved in a database together with a unique identifier (ID) for the user. During the verification stage, the signature, which is extracted from the watermarked image, is input to the verification module, along with the claimed identity of the author. The extracted signature is then compared to each of the reference signatures in the database based on the claimed identity (ID). The signature is accepted as genuine or rejected as a forgery, based

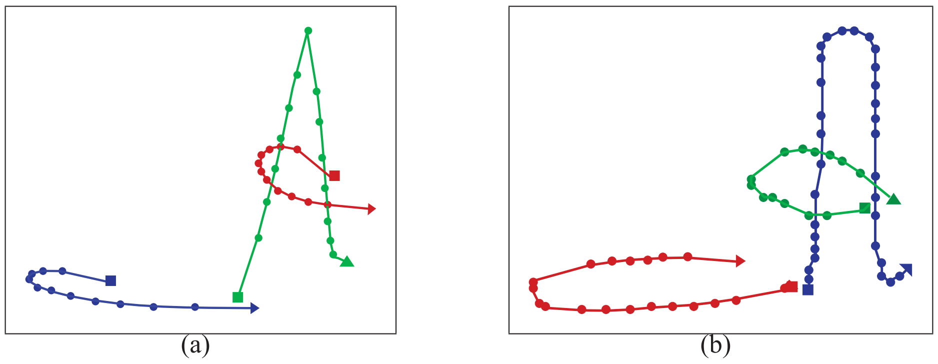


Figure 7.6: Genuine and forged signatures: (a) on-line signature by genuine user, and (b) an attempt to forge (a) by copying from an off-line sample. The strokes were drawn in the order $blue$, $green$, and $red$. The dots on the strokes represent sample points and arrows represent the writing direction.

on the matching. Figure 7.8 shows a plot of the false accept rate (FAR) versus the false reject rate (FRR) of the signature verification system (also called the Receiver Operating Characteristic (ROC) curve).

The system was tested on a set of $500$ signatures collected from $100$ users, each user providing $5$ samples of their signature and has an equal error rate of $6.8\%$. Figure 7.1 shows examples of signatures from our database and Figure 7.9 shows examples of documents used in our experiments.

## 7.3 Summary

The proposed method or authentication and verification of writer identity is applied to off-line documents. We use a fragile watermarking algorithm, where a document image is embedded with an on-line signature of the author. The algorithm can retrieve the embedded signature at the receiving end, and detect changes in the document at the same time. The retrieved signature can then be used for verifying the identity of the author. Results of authentication, carried out on a set of $15$ document images, each tested for $10$ different cases of alterations, demonstrate the ability of the algorithm to detect changes. This work has been published in [97].

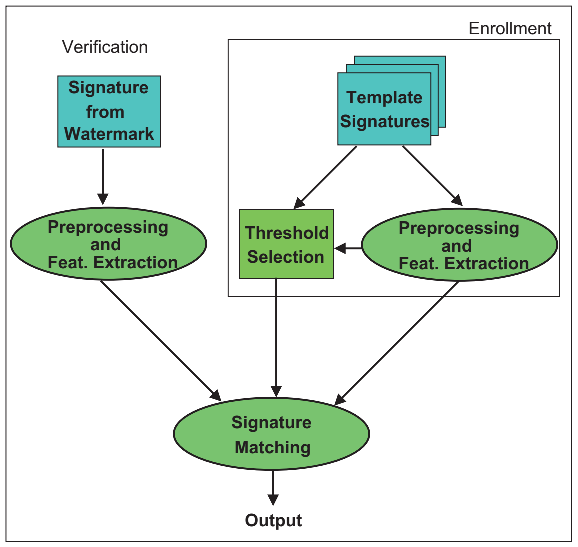


Figure 7.7: Schematic diagram of a signature verification system.

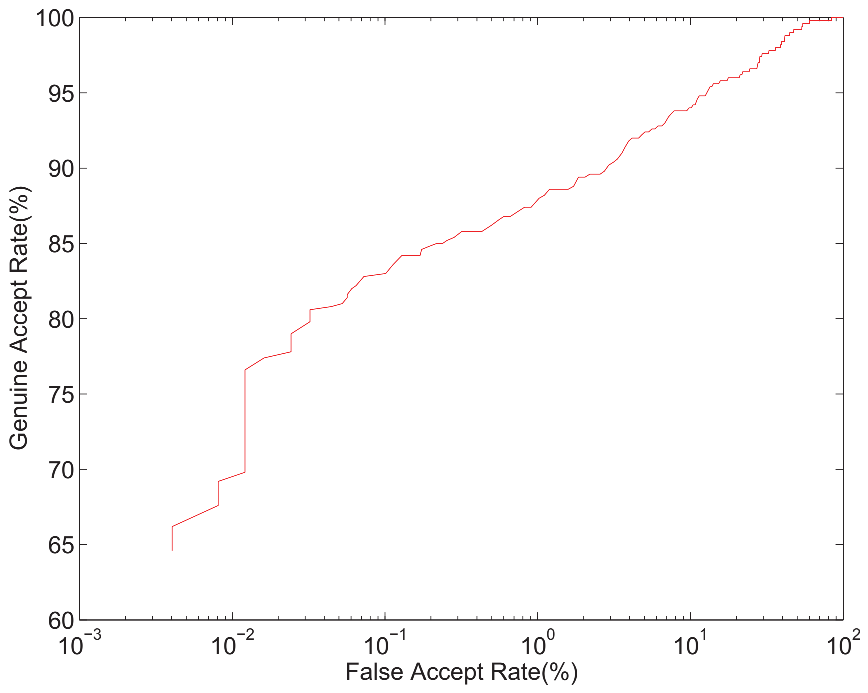


Figure 7.8: ROC curve of the signature verification system.

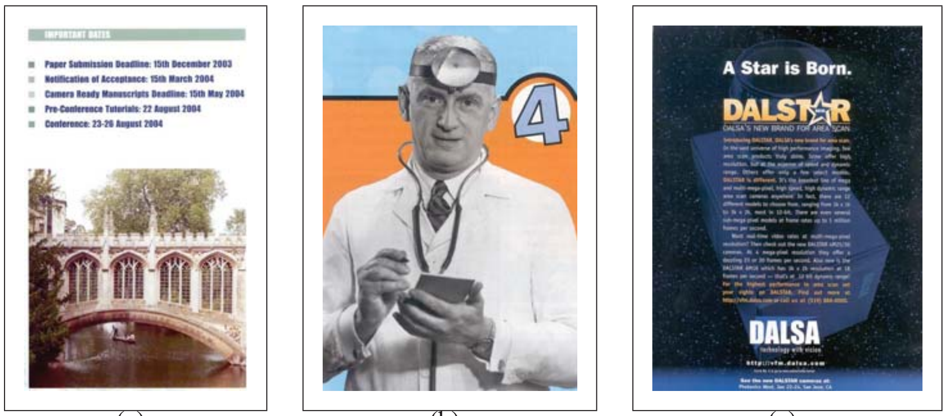

IMPORTANT DATES
Paper Submission Deadline: 15th December 2003
Notification of Acceptance: 15th March 2004
Camera Ready Manuscripts Deadline: 15th May 2004
Pre-Conference Tutorials: 22 August 2004
Conference: 23-26 August 2004
4
A Star is Born.
DALSTAR
DALSA'S NEW BRAND FOR AREA SCAN
DALSA
technology with vision
http://vfm.dalsa.com
(a)
(b)
(c)

Figure 7.9: Examples of document images used in the watermarking experiments.

# Chapter 8

# Summary

The primary objective of the work presented in this thesis was to develop a principled approach to solve the problem of on-line handwritten document understanding. In spite of the increased popularity and availability of pen-based devices, the research in this field has been limited to the problem of on-line handwriting recognition. However, the problem of document understanding in the domain of on-line documents is a very challenging one due to the highly unstructured nature of handwritten documents. Our aim was to highlight the significant challenges that need to be addressed to develop a comprehensive solution to this problem, and to describe and prove the viability of promising approaches to the individual challenges. The goal also included the development of an application that could highlight the potential of an on-line document understanding system.

## 8.1 Summary of the work

The problem of document understanding in the domain of handwritten documents poses significantly different challenges than there in the case of printed documents. This thesis proposes promising solutions to several of the problems in the field of on-line document understanding. We also describe a retrieval framework based on recognition-free matching of digital ink. Figure 8.1 shows a schematic diagram of the different problems attempted in the thesis and their interrelationship within the retrieval framework (reproduced from the first chapter). The salient parts of the work include:

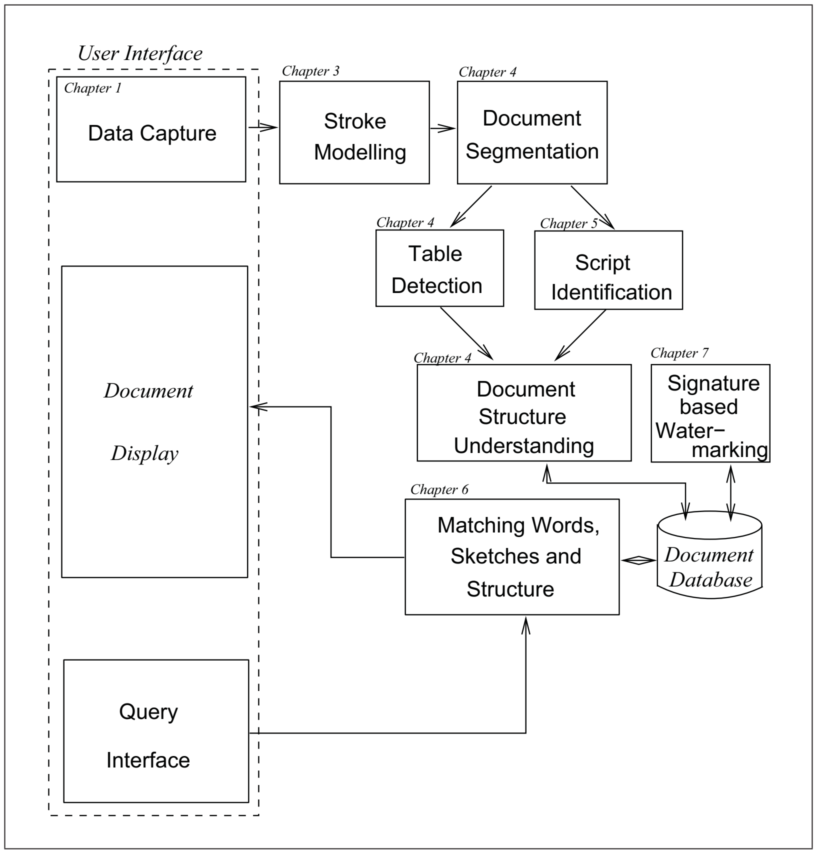


Figure 8.1: A schematic diagram of the on-line document understanding and retrieval system proposed in this thesis.

- *Representation of handwritten data*: The thesis proposes a spline-based representation for handwritten data. The representation is based on the mechanics of the handwriting process and can effectively approximate the handwritten strokes while removing the noise introduced by the digitizing devices. Experiments on text and non-text data from different digitizing devices show the generality of the representation. The average error of approximation at the data points was $1.6\%$ of the stroke size (height). A comparison of the distribution of the approximation error for the model and the expected distribution of noise in the data, based on a noise model derived from the digitization process, demonstrates the validity of the assumptions made while defining the stroke model.

- *Segmentation of handwritten documents*: The problem of segmentation of unconstrained on-line handwritten documents, to the best of our knowledge, has not been attempted before. We have proposed a segmentation algorithm based on the optimization of a compactness criterion defined on the regions in a document. The solution is based on a parsing technique using a stochastic context free grammar. We also present an algorithm for the detection of ruled and unruled tables and sketches within the regions identified by the segmentation process. The accuracy of segmentation was $88.3\%$ on a set of $154$ regions, while the ruled table detection algorithm achieved an accuracy of $84.8\%$ on a test set of $105$ tables and sketches.

- *Identification of scripts in handwritten data*: We have developed an algorithm for identification of the script of on-line handwritten text. The algorithm classifies every word in the input text into one of the following six scripts: Arabic, Cyrillic, Devnagari, Han, Hebrew, and Roman. We propose a set of features derived from the strokes of a handwritten word for the purpose of script identification. The effectiveness of the proposed features was tested using a variety of classifiers and also using a multi-classifier approach. We achieved an average classification accuracy of $87.1\%$ using 5-fold cross validation on a test set of $13,379$ words. The classification accuracy increased to $95.5\%$ when we used an entire text line, consisting of, on average, seven words.

- *Matching and retrieval of digital ink*: The thesis proposes matching algorithms for indexing and retrieval of on-line documents. The matching algorithms are recognition-free and hence make few assumptions about the nature of the data. The segmentation algorithm detects text regions and sketches that are used by the retrieval algorithm to automatically apply the appropriate matching algorithm for the query. The word-spotting algorithm achieves an overall accuracy of $92.3\%$ at a recall rate of $90\%$

averaged over a database containing $6,672$ words with an equal error rate of about $9\%$. The equal error rate of the sketch matching algorithm is $11.56\%$.

- *Security issues in on-line documents*: One of the fundamental obstacles of using digital documents for legal purposes is the ability to manipulate them effortlessly on a computer. We also present a solution to the problem of document integrity verification and writer identification. An on-line signature of the author is used to watermark digital documents (both on-line and off-line), which can be recovered and used for authentication by the receiver of the document. The algorithm can detect changes to a document image, which can be as small as $5$ pixels (or $5$ samples in the case of on-line documents) with a probability of $97\%$.

## 8.2 Future Research

There are several promising directions to extend the work presented in this thesis:

- *Improvements to the stroke representation*: The spline based representation of the stroke proposed in this work could be improved in several aspects. One could use a limit on the speed of pen movement to detect spurious errors and reduce their influence on the spline fitting process. However, computation of such a limit should involve extensive experiments on the various handwriting styles, nature of data and the digitizing hardware. In addition, one could also extend the representation in order to model additional stroke properties such as pen pressure and tilt.

- *Individuality of on-line handwritten data*: A study of correlation between various properties of the strokes could give valuable insights to the problem of writer identification. The spline-based representation could be used as the basis for computing stroke properties.

- *Improvement to page segmentation*: The page segmentation algorithm currently uses a grammar that models text lines, paragraphs, and non-text regions. This subset was chosen to include the most generic region types in any handwritten document. One could extend the grammar to incorporate additional stroke and region types. A more comprehensive grammar for the generation of handwritten documents is given in Appendix A, which was developed in consultation with experts in handwriting recognition in the Pen Computing Group at IBM T.J. Watson Research Center. One has to define appropriate criterion functions that need to be optimized for any set of grammar rules that are adopted, which is a non-trivial task. One could also create a user interface that facilitates interactive updates of the segmentation by the user.

- *Document genre identification*: Another interesting direction of research is that of document genre identification. The genre identification problem involves classifying a given document into a set of genre classes (e.g., letters, articles, driving directions, itemized lists, etc.). The problem involves defining the appropriate set of document genres and identifying or learning properties of each genre based on their structure and content.

- *Sketch matching in specific domains*: The matching of sketches could be improved dramatically with certain assumptions about the nature of the sketch. For example, the interesting parts of a sketch that need to match for the molecular structure of a chemical would be different from that which is interesting in the case of a circuit diagram.

- *Retrieval using strokes as primitives*: Since the basic primitives on on-line handwritten data are strokes, one could model the retrieval framework with strokes as primitives. Such a framework would model the generation of strokes in a document and could do retrieval based on a representation of the strokes or a higher level understanding of the documents developed from the strokes. A related problem is the generation of an inverted index for a set of document pages using the strokes.

- *Developing a comprehensive document analyzer*: The ultimate goal of a document understanding system is to develop a complete understanding of the structure and contents of a document page. The structure identification and script identification functionalities need to be integrated with text recognition to develop such a system. A comprehensive document analyzer should consider the structure and content recognition functionalities as part of a single optimization problem, since the correct interpretation of the content is often closely related to its structure or layout. However, a specific implementation of such a system might encode the functionalities in different modules that communicate all the required information between them effectively and efficiently.

# Appendices

# Appendix A

# Document Layout Description Rules

The grammar rules for segmentation of a generic handwritten document are given below. The grammar was constructed by considering the commonly occurring structures in handwritten documents. The grammar divides a document page into a set of regions, where each region is one of the following types: (i) text, (ii) table, (iii) list, (iv) figure, or (v) void (empty region). Each region could be decorated with an annotation that stands for a bullet or underline or a cross-out (strike through line). In addition, the regions could be related to each other by one of the following: a grouping (e.g., a box surrounding the regions), a connection (e.g., an arrow from one region to another), or a separator (e.g., a line dividing two regions). The grammar contains a total of 19 rules, which are divided into 3 primary rules, 10 intermediate rules, and 6 primitive rules. The primary rules define the page as a set of regions, while the intermediate rules define the components of each region type. The 6 primitive rules define the different stroke types that form the basis of the region definitions.

## A.1 Grammar Rules

*Primary rules* :

| | | |
|---|---|---|
| <inkDocument> | ::= | [InkPage] . . . |
| <inkPage> | ::= | <region> . . . |
| <region> | ::= | <annotation \| textBlock \| table \| list \| figure \| void \| relation> . . . |

*Intermediate rules* :

| | | |
|---|---|---|
| <textBlock> | ::= | [relation] . . . <[annotation] textLine> . . . |
| <textLine> | ::= | <[annotation] phrase> . . . |
| <table> | ::= | [relation] . . . [ruling] <region> . . . |
| <list> | ::= | [relation] . . . <annotation region> . . . |
| <figure> | ::= | [relation] . . . <figure \| strokes> . . . [textBlock] . . . |
| <annotation> | ::= | <bullet \| underline \| crossOut> . . . |
| <relation> | ::= | <grouper \| connection \| separator> |
| <grouper> | ::= | <strokes> <region> . . . |
| <connection> | ::= | <strokes> <region> <region> |
| <separator> | ::= | <strokes> <region> <region> |

*Primitive rules* :

| | | |
|---|---|---|
| <phrase> | ::= | <strokes> |
| <bullet> | ::= | <strokes> |
| <underLine> | ::= | <strokes> |
| <crossOut> | ::= | <strokes> |
| <ruling> | ::= | <strokes> |
| <void> | ::= | <empty bounding box> |

*Notes*: $i)$ $< X >$ means $X$ is mandatory; $ii)$ $[X]$ means $X$ is optional; $iii)$ $[X]\ldots$ means $0$ or more $Xs$; and $iv)$ $< X > \ldots$ means $1$ or more $Xs$.

## A.2 Region Properties

The general region properties are: $extent$ (in x and y axes), $bulleted$, $grouped$, $crossedOut$ and $underlined$. These are applicable to all entities. Self referencing properties (such as the property 'crossOut' for a the region crossOut) are defined to be true. The following is a list of properties associated with specific region types in the grammar. Note that some of the types are not listed (e.g., annotation) since they have the same properties as the specific region it points to.

| | | |
|---|---|---|
| inkDocument | : | number of pages. |
| inkPage | : | number of regions, author, scripts. |
| textBlock | : | number of lines. |
| table | : | number of rows, number of columns, ruled. |
| list | : | number of bulleted items. |
| figure | : | number of text blocks. |
| void | : | none, other than general region properties. |
| textLine | : | number of phrases. |
| phrase | : | transcription. |
| bullet | : | type (star, dot, arrow, other). |
| underline | : | type (eg: single, double, or other). |
| crossOut | : | type (single line, others). |
| grouper | : | type (braces, box, line, other). |
| connection | : | type (arrow, line, other). |
| separator | : | type (single line, double line, other). |

# Appendix B

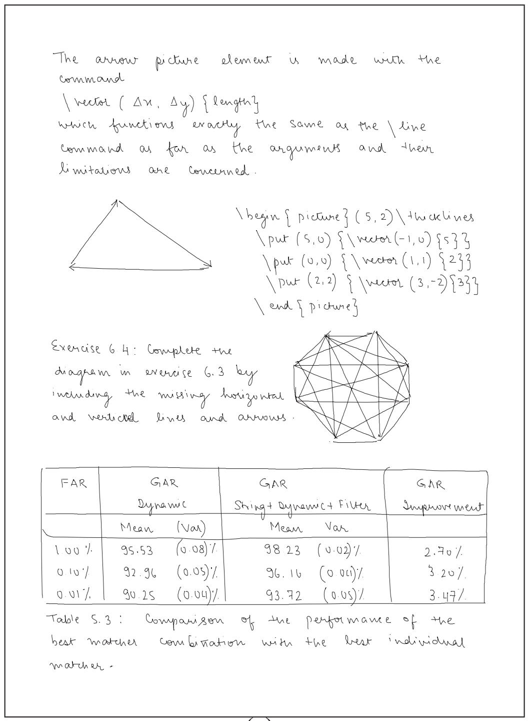
# Sample Documents from Database

This section provides some examples of the documents used in the various experiments reported in this thesis. A brief description of the figures is given below.

Figure B.1 shows examples of documents used in testing the document structure analysis work.

Figures B.2 and B.3 provide examples of the documents used for the script classification work. The scripts are: B.2(a): Arabic, B.2(b): Hebrew, B.2(c): Cyrillic, B.2(d): Han, B.3(a): Devnagari, and B.3(b): Roman. Figures B.3(c) and B.3(d) show samples of multilingual pages containing both Han and Roman scripts.

Figure B.4 shows examples of documents used for word-spotting-based indexing and retrieval.



The arrow picture element is made with the command

\vector (Δx, Δy) {length}

which functions exactly the same as the \line command as far as the arguments and their limitations are concerned.

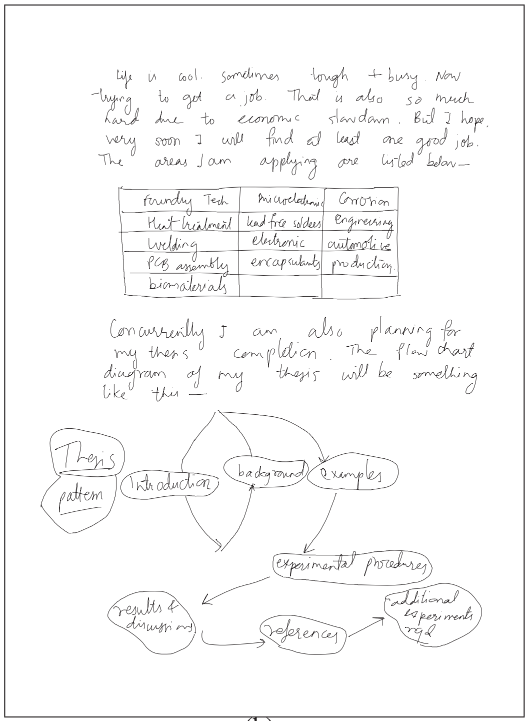
```
\begin{picture}(5,2)\thicklines
  \put (5,0) {\vector(-1,0) {5}}
  \put (0,0) {\vector (1,1) {2}}
  \put (2,2) {\vector (3,-2){3}}
\end{picture}
```

Exercise 6.4: Complete the diagram in exercise 6.3 by including the missing horizontal and verticle lines and arrows.

| FAR | GAR Dynamic | | GAR String+ Dynamic+ Filter | | GAR Improvement |
|---|---|---|---|---|---|
| | Mean | (Var) | Mean | Var | |
| 1.00% | 95.53 | (0.08)% | 98.23 | (0.02)% | 2.70% |
| 0.10% | 92.96 | (0.05)% | 96.16 | (0.01)% | 3.20% |
| 0.01% | 90.25 | (0.04)% | 93.72 | (0.05)% | 3.47% |

Table 5.3: Comparison of the performance of the best matcher combination with the best individual matcher.

(a)

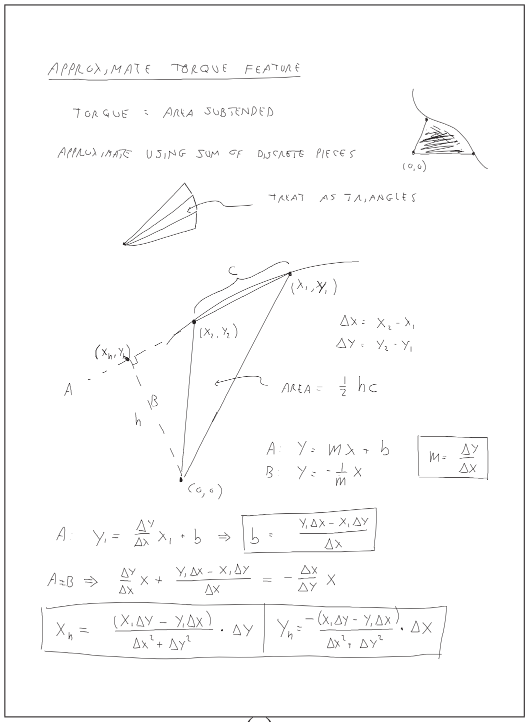


Life is cool. sometimes tough + busy. Now trying to get a job. That is also so much hard due to economic slowdown. But I hope, very soon I will find at least one good job. The areas I am applying are listed below—

| | | |
|---|---|---|
| Foundry Tech | Microelectronic | Corrosion |
| Heat treatment | lead free soldees | Engineering |
| Welding | electronic | automotive |
| PCB assembly | encapsulants | production |
| biomaterials | | |

Concurrently I am also planning for my thesis completion. The flow chart diagram of my thesis will be something like this—

(b)



APPROXIMATE TORQUE FEATURE

TORQUE = AREA SUBTENDED

APPROXIMATE USING SUM OF DISCRETE PIECES

(0,0)
TREAT AS TRIANGLES
C
$(X_1, Y_1)$
$(X_2, Y_2)$
$(x_h, y_h)$
A
B
h
(0,0)

$\Delta X = X_2 - X_1$

$\Delta Y = Y_2 - Y_1$

AREA = $\frac{1}{2} hc$

A: $Y = mX + b$

B: $Y = -\frac{1}{m} X$

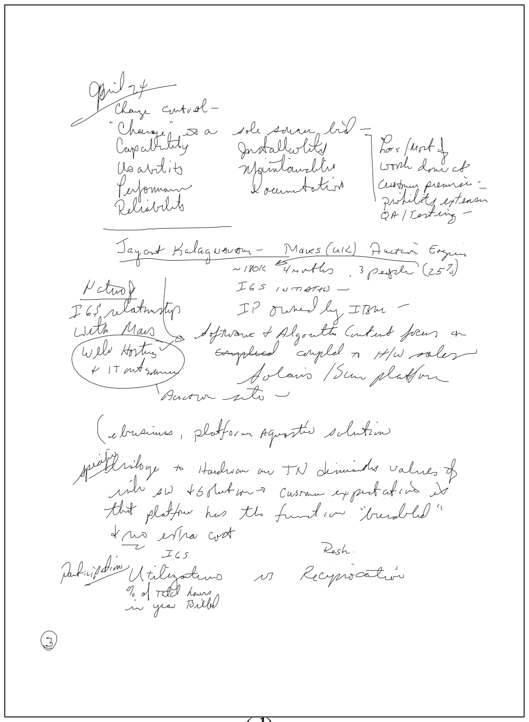
$\boxed{m = \frac{\Delta Y}{\Delta X}}$

$$A: \quad Y_1 = \frac{\Delta Y}{\Delta X} X_1 + b \Rightarrow \boxed{b = \frac{Y_1 \Delta X - X_1 \Delta Y}{\Delta X}}$$

$$A = B \Rightarrow \frac{\Delta Y}{\Delta X} X + \frac{Y_1 \Delta X - X_1 \Delta Y}{\Delta X} = -\frac{\Delta X}{\Delta Y} X$$

$$\boxed{X_h = \frac{(X_1 \Delta Y - Y_1 \Delta X)}{\Delta X^2 + \Delta Y^2} \cdot \Delta Y \quad \Big| \quad Y_h = \frac{-(X_1 \Delta Y - Y_1 \Delta X)}{\Delta X^2 + \Delta Y^2} \cdot \Delta X}$$

(c)



April 24

Change control—

"Change" is a sole source bid—

| | | |
|---|---|---|
| Capability | Installability | has / Most of work done at customer premise—prohibits extensive QA / Testing— |
| Usability | Maintainability | |
| Performance | Documentation | |
| Reliability | | |

Jayant Kalagnanam— Mars (UK) Auction Engine

~180K 4 months, 3 people (25%)

IGS INITIATED—

IP owned by IBM—

Network

IGS relationship with Mars

Web Hosting + IT outsourcing

Software + Algorithm Content focus on ~~coupled~~ coupled to H/W sales

Solaris / Sun platform

Auction site—

( ebusiness, platform Agnostic solution

specify Linkage to Hardware or TN diminishes values of [illegible] SW + Solution → Customer expectation is that platform has the function "bundled" + no extra cost

Participation

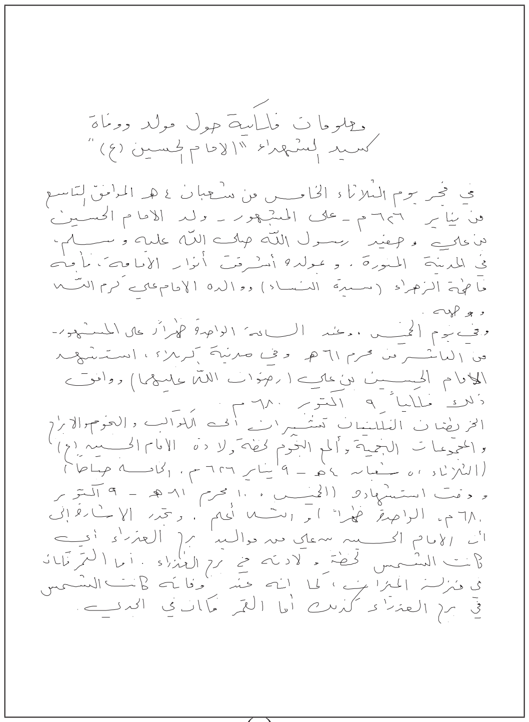
IGS Utilization vs Resh. Reciprocation

% of total hours in year Billed

3

(d)

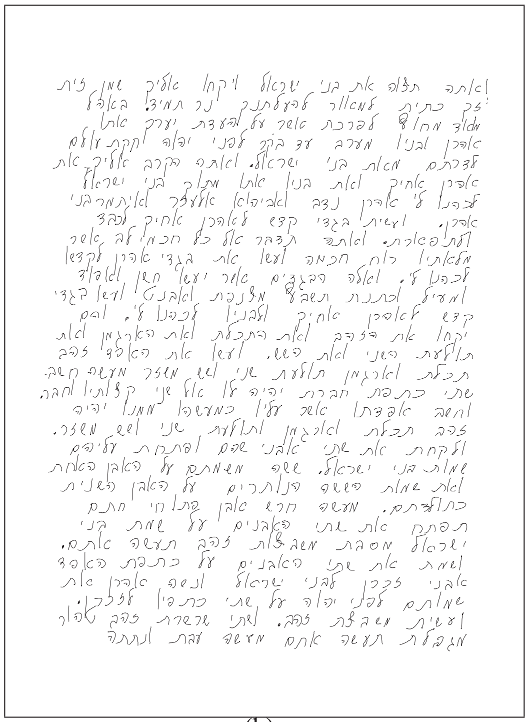
Figure B.1: Examples of on-line documents used in the structure detection experiments.



معلومات فلكية حول مولد ووفاة
سيد الشهداء "الامام الحسين (ع)"

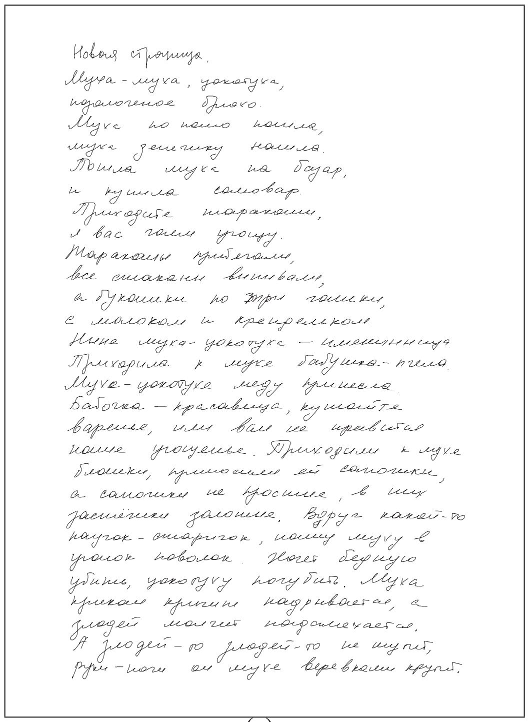
في فجر يوم الثلاثاء الخامس من شعبان ٤ هـ الموافق التاسع
من يناير ٦٢٦ م - على المشهور - ولد الامام الحسين
بن علي ، وحفيد رسول الله صلى الله عليه و سلم ،
في المدينة المنورة . و بمولده أشرقت أنوار الامامة ، فأمه
فاطمة الزهراء (سيدة النساء) و والده الامام علي كرم الله
وجهه .
وفي يوم الخميس ، عند الساعة الواحدة ظهراً على المشهور -
من العاشر من محرم ٦١ هـ وفي مدينة كربلاء ، استشهد
الامام الحسين بن علي (رضوان الله عليهما) ووافق
ذلك فلكياً ٩ اكتوبر ٦٨٠ م
الخريطتان الفلكيتان تشيران الى الكواكب والنجوم والابراج
والمجموعات النجمية وأهم النجوم لحظة ولادة الامام الحسين (ع)
(الثلاثاء ٥ شعبان ٤هـ - ٩ يناير ٦٢٦ م ، الخامسة صباحاً)
ووقت استشهاده (الخميس ، ١٠ محرم ٦١ هـ - ٩ اكتوبر
٦٨٠ م ، الواحدة ظهراً) والله أعلم . وتجدر الاشارة إلى
أن الامام الحسين عليه السلام من مواليد برج العذراء أي
كانت الشمس لحظة ولادته في برج العذراء . أما القمر فكان
في منزلة الميزان ، كما انه عند وفاته كانت الشمس
في برج العذراء كذلك أما القمر فكان في الجدي .

(a)



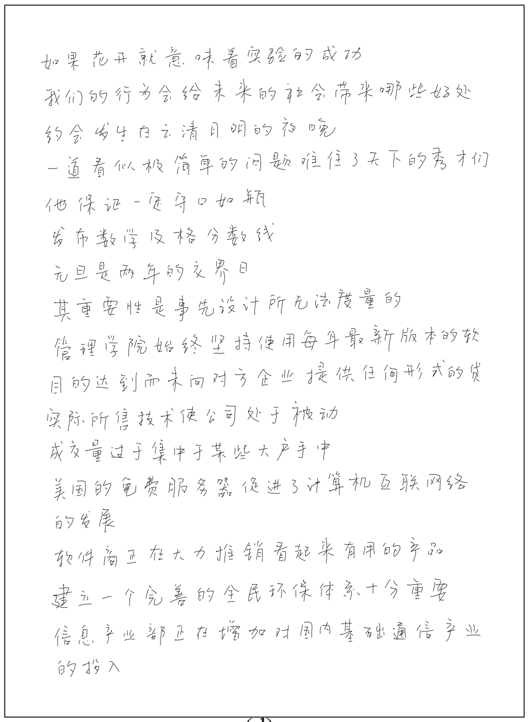
ואתה תצוה את בני ישראל ויקחו אליך שמן זית
זך כתית למאור להעלת נר תמיד. באהל
מועד מחוץ לפרכת אשר על העדת יערך אתו
אהרן ובניו מערב עד בקר לפני יהוה חקת עולם
לדרתם מאת בני ישראל. ואתה הקרב אליך את
אהרן אחיך ואת בניו אתו מתוך בני ישראל
לכהנו לי אהרן נדב ואביהוא אלעזר ואיתמר בני
אהרן. ועשית בגדי קדש לאהרן אחיך לכבוד
ולתפארת. ואתה תדבר אל כל חכמי לב אשר
מלאתיו רוח חכמה ועשו את בגדי אהרן לקדשו
לכהנו לי. ואלה הבגדים אשר יעשו חשן ואפוד
ומעיל וכתנת תשבץ מצנפת ואבנט ועשו בגדי
קדש לאהרן אחיך ולבניו לכהנו לי. והם
יקחו את הזהב ואת התכלת ואת הארגמן ואת
תולעת השני ואת השש. ועשו את האפד זהב
תכלת וארגמן תולעת שני ושש משזר מעשה חשב.
שתי כתפת חברת יהיה לו אל שני קצותיו וחבר.
וחשב אפדתו אשר עליו כמעשהו ממנו יהיה
זהב תכלת וארגמן ותולעת שני ושש משזר.
ולקחת את שתי אבני שהם ופתחת עליהם
שמות בני ישראל. ששה משמתם על האבן האחת
ואת שמות הששה הנותרים על האבן השנית
כתולדתם. מעשה חרש אבן פתוחי חתם
תפתח את שתי האבנים על שמת בני
ישראל מסבת משבצות זהב תעשה אתם.
ושמת את שתי האבנים על כתפת האפד
אבני זכרן לבני ישראל ונשא אהרן את
שמותם לפני יהוה על שתי כתפיו לזכרן.
ועשית משבצת זהב. ושתי שרשרת זהב טהור
מגבלת תעשה אתם מעשה עבת ונתתה

(b)



Новая страница.
Муха-муха, цокотуха,
позолоченное брюхо.
Муха по полю пошла,
муха денежку нашла.
Пошла муха на базар,
и купила самовар.
Приходите тараканы,
я вас чаем угощу.
Тараканы прибегали,
все стаканы выпивали,
а букашки по три чашки,
с молоком и крендельком.
Ныне муха-цокотуха — именинница.
Приходила к мухе бабушка-пчела.
Мухе-цокотухе меду принесла.
Бабочка — красавица, кушайте
варенье, или вам не нравится
наше угощенье. Приходили к мухе
блошки, приносили ей сапожки,
а сапожки не простые, в них
застёжки золотые. Вдруг какой-то
паучок-старичок, нашу муху в
уголок поволок. Хочет бедную
убить, цокотуху погубить. Муха
криком кричит надрывается, а
злодей молчит подсмехается.
А злодей-то злодей-то не шутит,
руки-ноги он мухе веревками крутит.

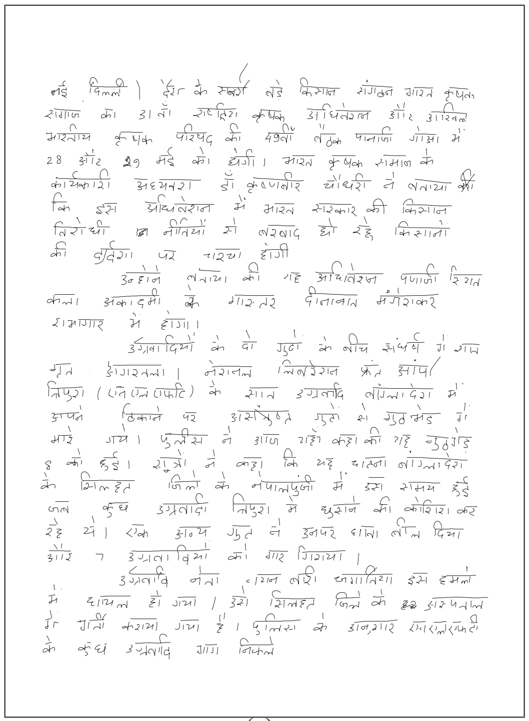
(c)



如果花开就意味着实验的成功
我们的行为会给未来的社会带来哪些好处
约会发生在云清月明的夜晚
一道看似极简单的问题难住了天下的秀才们
他保证一定守口如瓶
发布数学及格分数线
元旦是两年的交界日
其重要性是事先设计所无法度量的
管理学院始终坚持使用每年最新版本的软
目的达到而未向对方企业提供任何形式的贷
实际所售技术使公司处于被动
成交量过于集中于某些大户手中
美国的免费服务器促进了计算机互联网络
的发展
软件商正在大力推销看起来有用的产品
建立一个完善的全民环保体系十分重要
信息产业部正在增加对国内基础通信产业
的投入

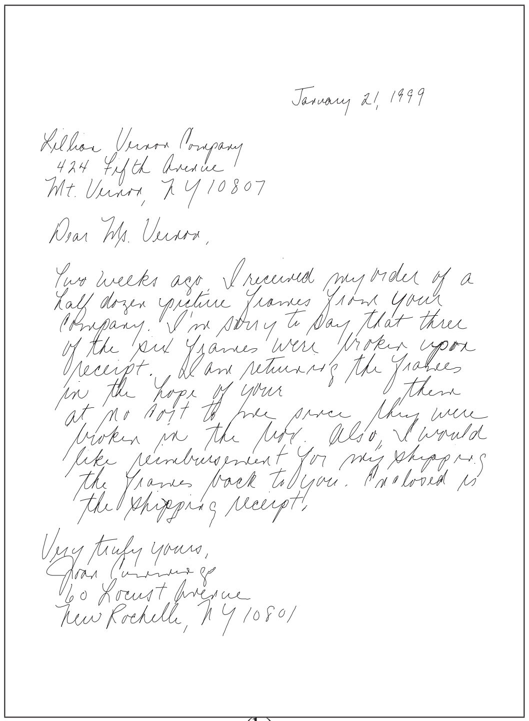
(d)

Figure B.2: Examples of on-line documents used in the script recognition experiments. (a) Arabic, (b) Hebrew, (c) Cyrillic, and (d) Han scripts.



नई दिल्ली। देश के सबसे बड़े किसान संगठन भारत कृषक
समाज का 31वाँ राष्ट्रीय कृषक अधिवेशन और अखिल
भारतीय कृषक परिषद की 49वीं बैठक पनाजी गोआ में
28 और 29 मई को होगी। भारत कृषक समाज के
कार्यकारी अध्यक्ष डॉ कृष्णवीर चौधरी ने बताया ~~की~~
कि इस अधिवेशन में भारत सरकार की किसान
विरोधी ~~न~~ नीतियों से बरबाद हो रहे किसानों
की दुर्दशा पर चर्चा होगी
उन्होंने बताया की यह अधिवेशन पणजी स्थित
कला अकादमी के भारतर दीनानाथ मंगेशकर
सभागार में होगा।
उग्रवादियों के दो गुटों के बीच संघर्ष में शान्
गत डोगारतला। नेशनल लिबरेशन फ्रंट ऑफ/
त्रिपुरा (एनएलएफटी) के सात उग्रवादि बांग्लादेश में
अपने ठिकाने पर आसोकृत्ते गुटों से मुठभेड़ में
मारे गये। पुलीस ने आज यहां कहा की यह मुठभेड़
8 को हुई। सूत्रों ने कहा कि यह घटना बांग्लादेश
के सिलहट जिले के नेपालपुंजी में इस समय हुई
जब कुछ उग्रवादी त्रिपुरा मे घुसने की कोशिश कर
रहे थे। एक अन्य गुट ने उनपर धावा बोल दिया
और 7 उग्रवादियों को मार गिराया।
उग्रवादि नेता बासन बरि जमातिया इस हमलों
में घायल हो गया। उसे सिलहट जिले के ~~इल~~ अस्पताल
में भर्ती कराया गया है। पुलिस के अनुसार एनएलएफटी
के कुछ उग्रवादि भाग निकले

(a)



January 21, 1999

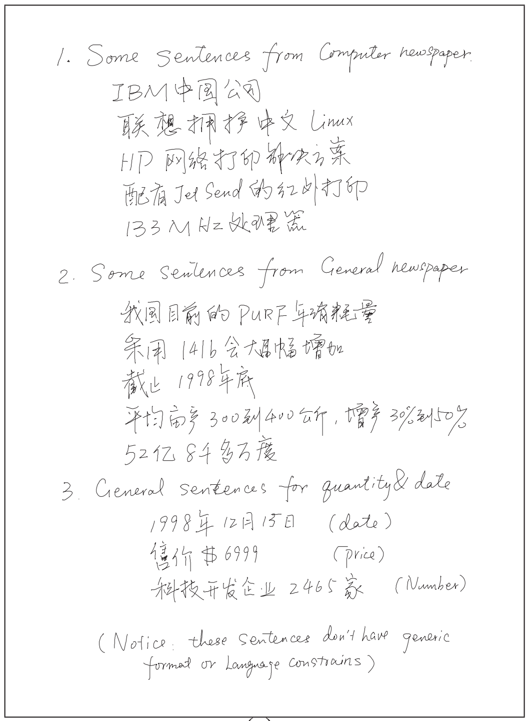
Lillian Vernon Company
424 Fifth Avenue
Mt. Vernon, NY 10807

Dear Ms. Vernon,

Two weeks ago, I received my order of a half dozen picture frames from your company. I'm sorry to say that three of the six frames were broken upon receipt. I am returning the frames in the hope of your them at no cost to me since they were broken in the box. Also, I would like reimbursement for my shipping the frames back to you. Enclosed is the shipping receipt.

Very truly yours,
Joan Cummings
60 Locust Avenue
New Rochelle, NY 10801

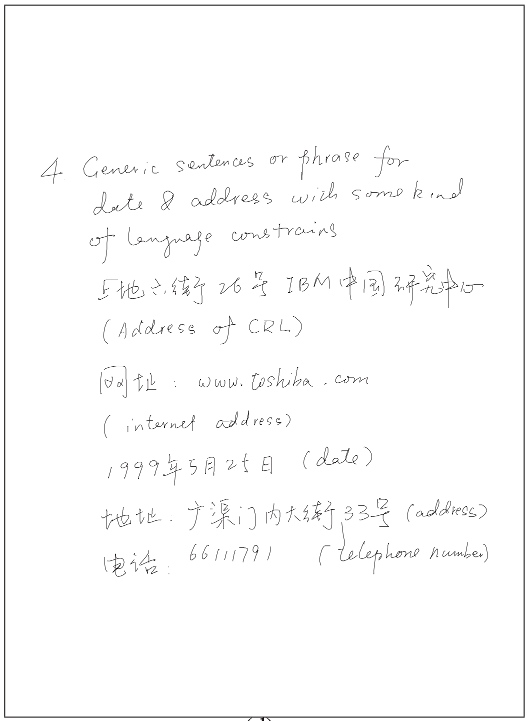
(b)



1. Some sentences from Computer newspaper.
   IBM中国公司
   联想拥抱中文 Linux
   HP 网络打印解决方案
   配有 Jet Send 的红外打印
   133 MHz处理器
2. Some sentences from General newspaper
   我国目前的PURF年消耗量
   采用 1416 会大幅增加
   截止 1998年底
   平均亩产 300到400公斤，增产 30%到50%
   52亿 8千多万度
3. General sentences for quantity & date
   1998年 12月15日 (date)
   售价 $6999 (price)
   科技开发企业 2465家 (Number)

(Notice: these sentences don't have generic format or Language constrains)

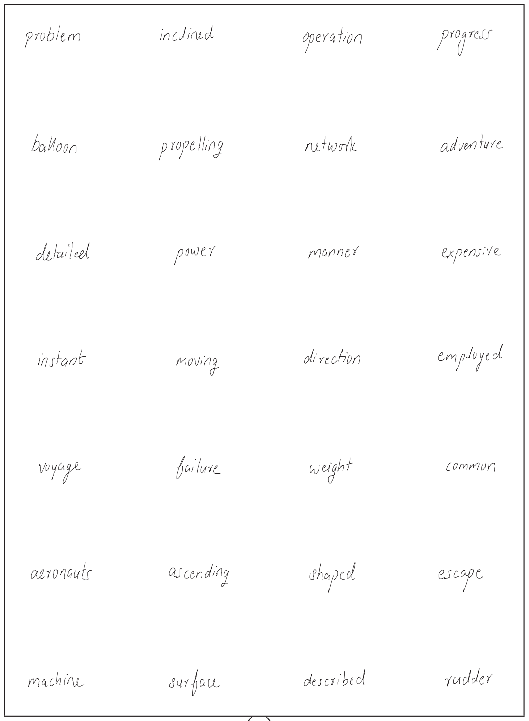
(c)



4. Generic sentences or phrase for date & address with some kind of language constrains
   上地六街 26号 IBM中国研究中心
   (Address of CRL)
   网址: www.toshiba.com
   (internet address)
   1999年5月25日 (date)
   地址: 广渠门内大街 33号 (address)
   电话: 66111791 (telephone number)

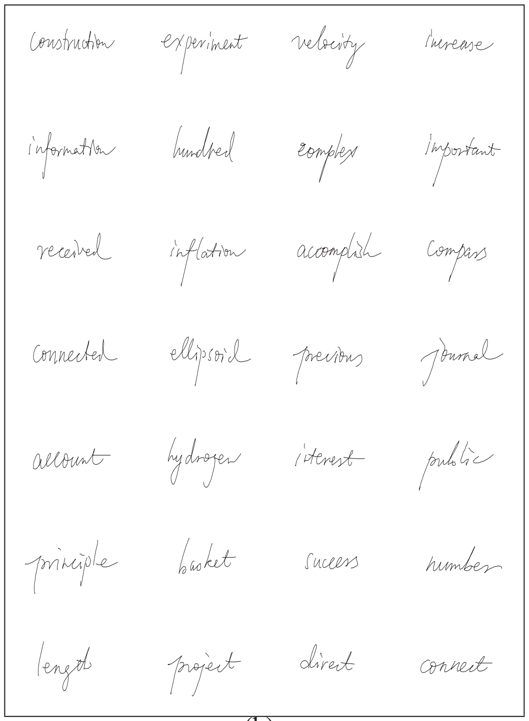
(d)

Figure B.3: Examples of on-line documents used in the script recognition experiments. (a) Devnagari, (b) Roman, (c,d) multilingual with both Roman and Han scripts.



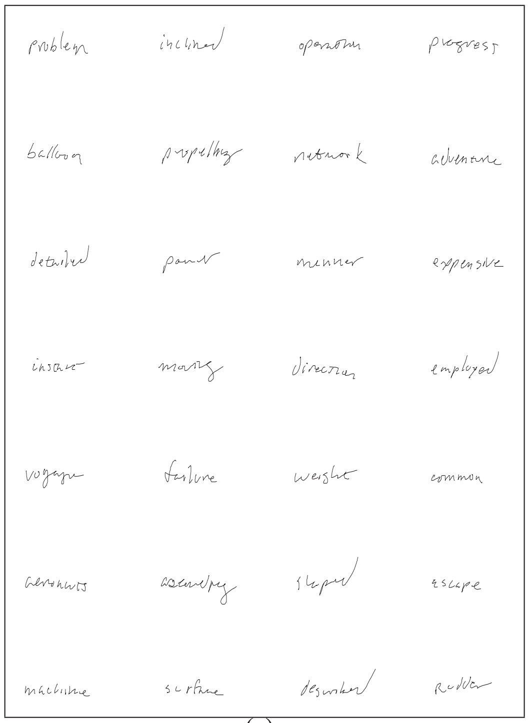
| | | | |
|---|---|---|---|
| problem | inclined | operation | progress |
| balloon | propelling | network | adventure |
| detailed | power | manner | expensive |
| instant | moving | direction | employed |
| voyage | failure | weight | common |
| aeronauts | ascending | shaped | escape |
| machine | surface | described | rudder |

(a)



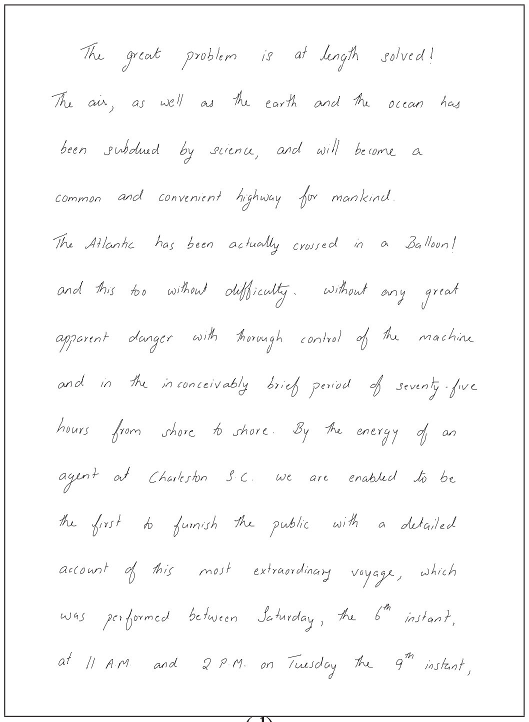
| | | | |
|---|---|---|---|
| Construction | experiment | velocity | increase |
| information | hundred | complex | important |
| received | inflation | accomplish | compass |
| Connected | ellipsoid | previous | journal |
| account | hydrogen | interest | public |
| principle | basket | success | number |
| length | project | direct | connect |

(b)



| | | | |
|---|---|---|---|
| problem | inclined | operation | progress |
| balloon | propelling | network | adventure |
| detailed | power | manner | expensive |
| instant | moving | direction | employed |
| voyage | failure | weight | common |
| aeronauts | ascending | shaped | escape |
| machine | surface | described | rudder |

(c)



The great problem is at length solved!
The air, as well as the earth and the ocean has
been subdued by science, and will become a
common and convenient highway for mankind.
The Atlantic has been actually crossed in a Balloon!
and this too without difficulty. without any great
apparent danger with thorough control of the machine
and in the inconceivably brief period of seventy-five
hours from shore to shore. By the energy of an
agent at Charleston S.C. we are enabled to be
the first to furnish the public with a detailed
account of this most extraordinary voyage, which
was performed between Saturday, the 6th instant,
at 11 A.M. and 2 P.M. on Tuesday the 9th instant,

(d)

Figure B.4: Examples of on-line documents used for word-spotting based indexing and retrieval.

# Bibliography

# Bibliography

[1] riteMail note taking application, 2003. http://www.ritemail.net, 2004.

[2] M. Aiello, C. Monz, L. Todoran, and M. Worring. Document understanding for a broad class of documents. *International Journal on Document Analysis and Recognition*, 5(1):1–16, 2002.

[3] Anoto. The technologies behind anoto functionality. http://www.anoto.com/?url=/technology/, 2004.

[4] Walid G. Aref, Ibrahim Kamel, and Daniel P. Lopresti. On handling electronic ink. *ACM Computing Surveys (CSUR)*, 27(4):564–567, 1995.

[5] D. K. Arrowsmith and C. M. Place. *An Introduction to Dynamic Systems*. Cambridge University Press, 1990.

[6] T. Artieres. Poorly structured handwritten documents segmentation using continuous probabilistic feature grammars. In *Proceedings of the Third International Workshop on Document Layout Interpretation and its Applications*, pages 5–8, Edinburgh, Scotland, August 2003. http://www.science.uva.nl/events/dlia2003/program/.

[7] J. Bach, C. Fuller, A. Gupta, A. Hampapur, B. Horowitz, R. Humphrey, R. Jain, and C. Shu. The Virage image search engine: An open framework for image management. In *SPIE Conference on Storage and Retrieval for Image and Video Databases*, volume 2670, pages 76–87, August 1996.

[8] Serge Belongie, Jitendra Malik, and J. Puzicha. Matching shapes. In *Proceedings of the* $8^{th}$ *International Conference on Computer Vision*, volume 1, pages 454–461, Vancouver, Canada, July 2001.

[9] Alberto Del Bimbo and Pietro Pala. Visual image retrieval by elastic matching of user sketches. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):121–132, February 1997.

[10] E. Bruzzone and M. C. Coffetti. An algorithm for extracting cursive text lines. In *Proceedings of the* $5^{th}$ *International Conference on Document Analysis and Recognition*, pages 749–752, Bangalore, India, September 1999.

[11] Chad Carson, Megan Thomas, Serge Belongie, Joseph M. Hellerstein, and Jitendra Malik. Blobworld: A system for region-based image indexing and retrieval. In *Proceedings of the $3^{rd}$ International Conference on Visual Information Systems*, volume 1614, pages 509–516, Amsterdam, The Netherlands, June 1999. Springer.

[12] R. Cattoni, T. Coianiz, S. Messelodi, and C. M. Modena. Geometric layout analysis techniques for document image understanding: A review. citeseer.nj.nec.com/cattoni98geometric.html, 2004.

[13] CDLI. The Cuneiform Digital Library Initiative. http://cdli.ucla.edu/, 2004.

[14] K. F. Chan and D. Y. Yeung. Mathematical expression recognition: A survey. *International Journal on Document Analysis and Recognition*, 3(1):3–15, 2000.

[15] Kam-Fai Chan and Dit-Yan Yueng. An efficient syntactic approach to structural analysis of online handwritten mathematical expressions. *Pattern Recognition*, 33(3):375–384, March 2000.

[16] Hao Chen, Oscar E. Agazzi, and Ching Y. Suen. Piecewise linear modulation model of handwriting. In *Proceedings of the $4^{th}$ International Conference on Document Analysis and Recognition*, pages 363–367, Ulm, Germany, August 1997.

[17] Hui Cheng, Charles A. Bouman, and Jan P. Allebach. Multiscale document segmentation. In *Proceedings of the IS&T $50^{th}$ Annual Conference*, pages 417–425, Cambridge, MA, May 1997.

[18] P.A. Chou and G.E. Kopec. A stochastic attribute grammar model of document production and its use in document recognition. In *Proceedings of the First International Workshop on Principles of Document Processing*, Washington, DC, October 1992.

[19] M. Clowes. A generative picture grammar. In *Proceedings of Computing Research Section*, Melbourne, Australia, 1967. Commonwealth Scientific and Industrial Research Organization.

[20] Scott D. Connell, R. M. K. Sinha, and Anil K. Jain. Recognition of unconstrained on-line devanagari characters. In *Proceedings of the $15^{th}$ International Conference on Pattern Recognition*, volume 2, pages 368–371, Barcelona, Spain, September 2000.

[21] Florian Coulmas. *The Blackwell Encyclopedia of Writing Systems*. Blackwell Publishers, Malden, Massachusetts, 1999.

[22] Nello Cristianini and John Shawe-Taylor. *An Introduction to Support Vector Machines*. Cambridge University Press, Cambridge, UK, March 2000.

[23] H. B. Curry and Isaac J. Schoenberg. On PLYA frequency functions IV: The fundamental spline functions and their limits. *Journal d'Analyse Math.*, 17:71–107, 1966.

[24] J. De Curtins. Comparison of OCR versus word shape recognition for key word spotting. In *Proceedings of the Symposium on Document Image Understanding Technology*, pages 205–213, Annapolis, MD, 1997.

[25] C. de Boor. *A Practical Guide to Splines*. Springer-Verlag, Berlin, $1^{st}$ edition, 1978.

[26] David Doermann. The indexing and retrieval of document images: A survey. Technical Report, University of Maryland, February 1998. LAMP-TR-0013.

[27] R. Duda, P. Hart, and D. Stork. *Pattern Classification*. John Wiley and Sons, New York, $2^{nd}$ edition, 2000.

[28] E-pen. E-PEN: Natural writing in a digital world. http://www.e-pen.com/, 2003.

[29] M. Eden. Handwriting and pattern recognition. *IEEE Transactions on Information Theory*, 8(2):160–166, February 1962.

[30] A. El-Nasan and George Nagy. Ink-link. In *Proceedings of the $15^{th}$ International Conference on Pattern Recognition*, volume 2, pages 573–576, Barcelona, Spain, September 2000.

[31] Kamran Etemad, David S. Doermann, and Rama Chellappa. Multiscale segmentation of unstructured document pages using soft decision integration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(1):92–96, January 1997.

[32] Kevin Featherly and Mary Van Beusekom. Medicine on the move. *Healthcare Informatics*, pages 28i–28vi, February 2004.

[33] Mario T. Figueiredo and Anil K. Jain. Unsupervised learning of finite mixture models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(3):381–396, March 2002.

[34] M. Flickner, H. Sawhney, W. Niblack, J. Ashley, Q. Huang, B. Dom, M. Gorkani, J. Hafner, D. Lee, D. Petkovic, D. Steele, and P. Yanker. Query by image and video content: The QBIC system. *IEEE Computer*, 28(9):23–32, September 1995.

[35] Jonathan Foote. An overview of audio information retrieval. *ACM Multimedia Systems*, 7(1):2–10, 1999.

[36] Max Froumentin, Dave Raggett, and Philipp Hoschka. Ink Markup Language. http://www.w3.org/2002/mmi/ink, 2004.

[37] Utpal Garain and B.B.Chaudhuri. On machine understanding of online handwritten mathematical expressions. In *Proceedings of the $7^{th}$ International Conference on Document Analysis and Recognition*, volume 1, pages 349–353, Edinburgh, Scotland, August 2003.

[38] E. Gokcay and D. Gokcay. Combining statistics and heuristics for language identification. In *Proceedings of the $4^{th}$ Annual Symposium on Document Analysis and Information Retrieval*, pages 423–433, Las Vegas, USA, April 1995.

[39] Wacef Guerfali and Réjean Plamondon. The delta lognormal theory for the generation and modeling of cursive characters. In *Proceedings of the* 3$^{rd}$ *International Conference on Document Analysis and Recognition*, pages 495–498, Montréal, Canada, August 1995.

[40] F. Hartung and M. Kutter. Multimedia watermarking techniques. *Proceedings of the IEEE*, 87(9):1079–1107, July 1999.

[41] Y. Hirayama. A method for table structure analysis using DP matching. In *Proceedings of the* 3$^{rd}$ *International Conference on Document Analysis and Recognition*, pages 583–586, Montreal, Canada, August 1995.

[42] Judith Hochberg, Kevin Bowers, Michael Cannon, and Patrick Kelly. Script and language identification for handwritten document images. *International Journal on Document Analysis and Recognition*, 2(2):45–52, February 1999.

[43] Judith Hochberg, Patrick Kelly, Timothy Thomas, and Lila Kerns. Automatic script identification from document images using cluster-based templates. In *Proceedings of the* 3$^{rd}$ *International Conference on Document Analysis and Recognition*, pages 378–381, Montreal, Canada, August 1995.

[44] John M. Hollerbach. An oscillation theory of handwriting. *Biological Cybernetics*, 39:139–156, 1981.

[45] Jianying Hu, Ram Kashi, Daniel Lopresti, and Gordon Wilfong. Table detection across multiple media. In *Proceedings of the Workshop on Document Layout Interpretation and its Applications (ICDAR'99)*, Bangalore, India, September 1999. http://www.science.uva.nl/events/dlia99/.

[46] R. Hull, D. Reynolds, and D. Gupta. Scribble matching. In *Proceedings of the* 4$^{th}$ *International Workshop on Frontiers in Handwriting Recognition*, pages 285–295, Taipei, Taiwan, December 1994.

[47] IBM. IBM Pen Technologies. http://www.research.ibm.com/handwriting/, 2004.

[48] IBM. IBM ThinkPad TransNote. http://www.pc.ibm.com/us/thinkpad/transnote/, 2003.

[49] Seiko Instruments. Inklink Handwriting System. http://www.siibusinessproducts.com/products/link-ir-p.html, 2004.

[50] ISC. Internet Software Consortium: Internet Domain Survey. http://www.isc.org/ds/, 2004.

[51] Stefan Jaeger, Stefan Manke, Jrgen Reichert, and Alex Waibel. Online handwriting recognition: The NPen++ recognizer. *International Journal on Document Analysis and Recognition*, 3(3):169–180, 2001.

[52] Anil K. Jain and Sushil K. Bhattacharjee. Text segmentation using Gabor filters for automatic document processing. *Machine Vision and Applications*, 5:169–184, 1992.

[53] Anil K. Jain, Friederike D. Griess, and Scott D. Connell. On-line signature verification. *Pattern Recognition*, 35(12):2963–2972, December 2002.

[54] Anil K. Jain and Anoop M. Namboodiri. Indexing and retrieval of on-line handwritten documents. In *Proceedings of the* $7^{th}$ *International Conference on Document Analysis and Recognition*, volume 2, pages 655–659, Edinburgh, Scotland, August 2003.

[55] Anil K. Jain, Anoop M. Namboodiri, and Jayashree Subrahmonia. Structure in online documents. In *Proceedings of the* $6^{th}$ *International Conference on Document Analysis and Recognition*, pages 844–848, Seattle, Washington, September 2001.

[56] Anil K. Jain, S. Pankanti, and R. Bolle (eds.). *BIOMETRICS: Personal Identification in Networked Society*. Kluwer, 1999.

[57] Anil K. Jain, Umut Uludag, and Rein-Lein Hsu. Hinding a face in a fingerprint image. In *Proceedings of the* $16^{th}$ *International Conference on Pattern Recognition*, volume 3, pages 756–759, Quebec City, Canada, August 2002.

[58] Anil K. Jain and Bin Yu. Document representation and its application to page decomposition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):294–308, March 1998.

[59] Anil K. Jain and Y. Zhong. Page segmentation using texture analysis. *Pattern Recognition*, 29(5):743–770, May 1996.

[60] Anil K. Jain, Yu Zhong, and Sridhar Lakshmanan. Object matching using deformable templates. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 18(3):267–278, March 1996.

[61] Anil K. Jain and D. Zongker. Feature-selection: Evaluation, application, and small sample performance. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(2):153–158, 1997.

[62] Rob Jarrett and Philip Su. *Building Tablet PC Applications*. Microsoft Press, Redmond, WA, 2002.

[63] Hans Jensen. *Sign, Symbol and Script: An Account of Man's Effort to Write*. George Allen and Unwin, London, $3^{rd}$ edition, 1970.

[64] Daniel Jurafsky and James H. Martin. *Speech and Language Processing: An Introduction to Natural Language Processing*. Computational Linguistics and Speech Recognition. Prentice-Hall, New Jersey, 2000.

[65] Ibrahim Kamel. Fast retrieval of cursive handwriting. In *Proceedings of the* $5^{th}$ *International Conference on Information and Knowledge Management*, pages 91–98, Rockville, MD, November 1996.

[66] Junichi Kanai and Henry S. Baird (ed.). Document image understanding and retrieval. *Special issue of Computer Vision and Image Understanding*, 70(3), June 1998.

[67] S. Kane, A. Lehman, and E. Partridge. Indexing George Washington's handwritten manuscripts. Technical report, Center for Intelligent Information Retrieval, University of Massachussetts, 2001.

[68] Thomas G. Kieninger and Andreas Dengel. The T-RECS approach for table structure recognition and table border determination. In *Proceedings of the Workshop on Document Layout Interpretation and its Applications (ICDAR'99)*, Bangalore, India, September 1999. http://www.science.uva.nl/events/dlia99/.

[69] Josef Kittler, Mohamad Hatef, Robert P. W. Duin, and Jiri Matas. On combining classifiers. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(3):226–239, March 1998.

[70] K. M. Knill and S. J. Young. Fast implementation methods for Viterbi-based word-spotting. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, pages 522–523, Atlanta, GA, May 1996.

[71] Donald E. Knuth, J. H. Morris, and V. R. Pratt. Fast pattern matching in strings. *SIAM Journal of Computing*, 6(2):323–350, 1977.

[72] Aleksander Kolcz, Joshua Alspector, Marijke Augusteijn, and Robert Carlson. Word-spotting studied in cursive writing using query-by-example search approach. *Electronic Imaging: Optical Engineering Reports*, 8(2), December 1998. http://www.spie.org/web/oer/december/dec98/eitg.html, 2004.

[73] Gary E. Kopec and Phil A. Chou. Document image decoding using markov source models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(6):602–617, June 1994.

[74] Gary E. Kopec and Philip A. Chou. Stochastic attribute grammar model of document production and its use in document image decoding. In *Proceedings of the SPIE Vol. 2422, Document Recognition II*, pages 66–73, 1995.

[75] W. Kornfeld and J. Wattecamps. Automatically locating, extracting and analyzing tabular data. In *Proceedings of the* $21^{st}$ *Annual International ACM SIGIR Conference*, pages 347–349, Melbourne, Australia, August 1998.

[76] D. Kundur and D. Hatzinakos. Digital watermarking for telltale tamper proofing and authentication. *Proceedings of the IEEE*, 87(9):1167–1180, July 1999.

[77] Shyh Shiaw Kuo and Oscar E. Agazzi. Keyword spotting in poorly printed documents using pseudo 2-D hidden markov models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(8):842–848, August 1994.

[78] Jay J. Lee and Jin H. Kim. A unified network-based approach for online recognition of multi-lingual cursive handwritings. In *Proceedings of the* $5^{th}$ *International Workshop on Frontiers in Handwriting Recognition*, pages 393–397, Colchester, UK, September 1996.

[79] Daniel Leong. A history of PDAs. http://www.pdawear.com/news/ article_pda_beginning.htm, 2003.

[80] Wing Ho Leung and Tsuhan Chen. User-independent retrieval of free-form hand-drawn sketches. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, volume 2, pages 2029–2032, Orlando, FL, May 2002.

[81] Jia Li and Robert M. Gray. Text and picture segmentation by distribution analysis of wavelet coefficients. In *Proceedings of the* $5^{th}$ *International Conference on Image Processing*, pages 790–794, Chicago, IL, October 1998.

[82] Xiaolin Li, Marc Parizeau, and Rjean Plamondon. Segmentation and reconstruction of on-line handwritten scripts. *Pattern Recognition*, 31(6):675–684, June 1998.

[83] Martin M. Lipschutz. *Differential Geometry*. McGraw-Hill, 1969.

[84] Jinhui Liu and Anil K. Jain. Image-based form document retrieval. *Pattern Recognition*, 33(3):503–513, March 2000.

[85] Y. Liu. An automation system: generation of digital map data from pictorial map resources. *Pattern Recognition*, 35(9):1973–1987, September 2002.

[86] Lawrence K. Lo. Ancientscripts.com. http://www.ancientscripts.com/, 2004.

[87] Matt Loney. Get ready for a bigger dose of Tablet PCs. http://insight. zdnet.co.uk/specials/tabletpcs/0,39025068,39147361,00.htm, 2004.

[88] Daniel Lopresti and Andrew Tomkins. On the searchability of electronic ink. In *Proceedings of the* $4^{th}$ *International Workshop on Frontiers in Handwriting Recognition*, pages 156–165, Taipei, Taiwan, December 1994.

[89] Daniel Lopresti, Andrew Tomkins, and J. Zhou. Algorithms for matching hand-drawn sketches. In *Proceedings of the* $5^{th}$ *International Workshop on Frontiers in Handwriting Recognition*, pages 233–238, Colchester, UK, September 1996.

[90] R. Manmatha, C. Han, and E. Riseman. Word spotting: A new approach to indexing handwriting. In *Proceedings of the IEEE Computer Vision and Pattern Recognition Conference*, pages 631–637, San Francisco, June 1996.

[91] A. Meyer. *Pen Computing: A Technology Overview and a Vision*, volume 27, pages 46–90. SIGCHI Bulletin, July 1995.

[92] Microsoft. Windows XP Tablet PC Edition Homepage. http://www.microsoft.com/windowsxp/tabletpc/, 2004.

[93] Mimio. Mimio: Interactive whiteboards. http://www.mimio.com, 2004.

[94] George Nagy. Twenty years of document image analysis in PAMI. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):38–62, January 2000.

[95] Akira Nakanishi. *Writing Systems of the World*. Charles E. Tuttle Company, Tokyo, 1999.

[96] Vishvjit S. Nalwa. Automatic on-line signature verification. *Proceedings of the IEEE*, 85(2):215–239, February 1997.

[97] Anoop M. Namboodiri and Anil K. Jain. Watermarking document images for authentication using on-line signature. In *SPIE Vol. 5306, Security, Steganography and Watermarking of Multimedia Content VI*, pages 653–662, San Jose, California, January 2004.

[98] NDL. Library of congress: National Digital Library. http://www.loc.gov, 2004.

[99] N. Nobile, S. Bergler, C Y. Suen, and S. Khoury. Language identification of on-line documents using word shapes. In *Proceedings of the* $4^{th}$ *International Conference on Document Analysis and Recognition*, pages 258–262, Ulm, Germany, August 1997.

[100] Jeffrey C. O'Neill, Alfred O. Hero, and William J. Williams. Word spotting via spatial point processes. In *Proceedings of the* $3^{rd}$ *IEEE International Conference on Image Processing*, pages 215–219, Lausanne, Schweiz, September 1996.

[101] OTMTech. Optical translation measurement. http://www.otmtech.com/otm.asp, 2004.

[102] U. Pal and B B. Chaudhuri. Script line separation from Indian multi-script documents. In *Proceedings of the* $5^{th}$ *International Conference on Document Analysis and Recognition*, pages 406–409, Bangalore, India, September 1999.

[103] Austin Norman Palmer. *The Palmer Method of Business Writing*. A. N. Palmer Company, New York, 1930.

[104] P. W. Palumbo, S. N. Srihari, J. Soh, R. Sridhar, and V. Demjanenko. Postal address block location in real-time. *IEEE Computer*, 25(7):34–42, July 1992.

[105] Theo Pavlidis. *Structural Pattern Recognition*. Springer-Verlag, December 1977.

[106] Theo Pavlidis. Problems in recognition of drawings. In G. Ferrante, T. Pavlidis, A. Sanfeliu, and H. Bunke, editors, *Syntactic and Structural Pattern Recognition*, volume F 45 of *ASI Series*, pages 103–113. Springer-Verlag, 1988.

[107] Theo Pavlidis. Page segmentation by white streams. In *Proceedings of the* $1^{st}$ *International Conference on Document Analysis and Recognition*, pages 945–953, St. Malo, France, September 1991.

[108] G S. Peake and T N. Tan. Script and language identification from document images. In *Proceedings of the* $3^{rd}$ *Asian Conference on Computer Vision*, pages 96–104, Hong Kong, China, January 1998.

[109] Pen2Net. Compupen. http://www.compupen.com/Compupen/compupen.html, 2004.

[110] I. Percival and F. Vivaldi. Arithmetical properties of strongly chaotic systems. *Physica*, 25D:105–130, 1987.

[111] F. A. P. Petitcolas, R. J. Anderson, and M. G. Kuhn. Information hiding - A survey. *Proceedings of the IEEE*, 87(9):1062–1078, July 1999.

[112] Rejean Plamondon. A kinematic theory of rapid human movements: Part I: Movement representation and generation. *Biological Cybernetics*, 72(4):295–307, 1995.

[113] Rejean Plamondon. A kinematic theory of rapid human movements: Part II: Movement time and control. *Biological Cybernetics*, 72(4):309–320, 1995.

[114] Rejean Plamondon. A kinematic theory of rapid human movements: Part III: Kinetic outcomes. *Biological Cybernetics*, 78(2):133–145, 1998.

[115] Rejean Plamondon, R. Feng, and A. Woch. A kinematic theory of rapid human movements: Part IV: A formal mathematical proof and new insights. *Biological Cybernetics*, 89(2):126–138, 2003.

[116] Rjean Plamondon and Sargur N. Srihari. On-line and off-line handwriting recognition: A comprehensive survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(1):63–84, 2000.

[117] Y. Pu and Z. Shi. A natural learning algorithm based on Hough transform for text lines extraction in handwritten documents. In *Proceedings of the* $6^{th}$ *International Workshop on Frontiers in Handwriting Recognition*, pages 637–646, Taejon, Korea, August 1998.

[118] D. Blostein R. Zanibbi and J.R. Cordy. A survey of table recognition: Models, observations, transformations and inferences. *International Journal on Document Analysis and Recognition*, 7(1), 2004.

[119] M. Armon Rahgozar and Robert Cooperman. Graph-based table recognition system. In Luc M. Vincent and Jonathan J. Hull, editors, *Proceedings of SPIE Vol. 2660, Document Recognition III*, pages 192–203, San Jose, CA, January 1996.

[120] T. M. Rath and R. Manmatha. Word image matching using dynamic time warping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 521–527, Madison, WI, June 2003.

[121] Eugene H. Ratzlaff. Inter-line distance estimation and text line extraction for unconstrained online handwriting. In *Proceedings of the* $7^{th}$ *International Workshop on Frontiers in Handwriting Recognition*, pages 33–42, Nijmegen, Netherlands, September 2000.

[122] F. Rossant. A global method for music symbol recognition in typeset music sheets. *Pattern Recognition Letters*, 23(10):1129–1141, August 2002.

[123] Sam T. Roweis. CrossPad Under Linux Information. http://www.cs.toronto.edu/ r̃oweis/crosspad.html, 2004.

[124] Dean Rubine. *The Automatic Recognition of Gestures*. PhD thesis, Carnegie Mellon University, Pittsburg, PA, 1991.

[125] RuleQuest. RuleQuest Research Data Mining Tools. http://www.rulequest.com/ index.html, 2004.

[126] Gregory Russell, Yi-Min Chee, Giovanni Seni, Larry Yaeger, Christopher Tremblay, Katrin Franke, Sriganesh Madhvanath, and Max Froumentin. Ink Markup Language: Current draft. http://www.w3.org/TR/InkML/, 2004.

[127] Gregory Russell, Michael P. Perrone, Yi min Chee, and Aiman Ziq. Handwritten document retrieval. In *Proceedings of the* $8^{th}$ *International Workshop on Frontiers in Handwriting Recognition*, pages 233–238, Ontario, Canada, August 2002.

[128] Mamoun Sakkal. The Art of Arabic Calligraphy. http://www.sakkal.com/ ArtArabicCalligraphy.html, 2004.

[129] Gerard Salton. *Automatic Text Processing*. Addison-Wesley, Reading, MA, 1989.

[130] M. Schenkel, I. Guyon, and D. Henderson. On-line cursive script recognition using time delay neural networks and hidden markov models. In *Proceedings of the IEEE International Conference on Acoustics Speech and Signal Processing*, pages 637–640, Adelaide, Austrailia, April 1994.

[131] B. Schneier. *Applied Cryptography*. John Wiley & Sons, 1996.

[132] Larry L. Schumaker. *SPLINE FUNCTIONS: Basic Theory*. John Wiley & Sons, New York, 1981.

[133] Abigail J. Sellen and Richard H. R. Harper. *The Myth of the Paperless Office*. MIT Press, $1^{st}$ edition, 2001.

[134] A. C. Shaw. A formal picture description scheme as a basis for picture processing systems. *Information and Control*, 14(1):9–52, 1969.

[135] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):888–905, August 2000.

[136] Bruno Simard, Birendra Prasada, and R. M. K. Sinha. On-line character recognition using hand-writing modelling. *Pattern Recognition*, 26(7):993–1007, 1993.

[137] Y. Singer and N. Tishby. Dynamical encoding of cursive handwriting. *Biological Cybernetics*, 71(3):227–237, 1994.

[138] R. M. K. Sinha. PLANG - A picture language schema for a class of pictures. *Pattern Recognition*, 16:373–383, 1983.

[139] SmartTech. SMART Technologies Inc. Homepage. http://www.smarttech.com/, 2004.

[140] Arnold W. M. Smeulders, Marcel Worring, Simone Santini, Amarnath Gupta, and Ramesh Jain. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(12):1349–1380, December 2000.

[141] A. L. Spitz. Determination of the script and language content of document images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(3):235–245, March 1997.

[142] Birgit Stehno and Gregor Retti. Modelling the logical structure of books and journals using augmented transition network grammars. *Journal of Documentation*, 59(1):69–83, January 2003.

[143] Jayashree Subrahmonia. Pen computing: Challenges and applications. In *Proceedings of the 15th International Conference on Pattern Recognition*, volume 2, pages 60–66, Barcelona, Spain, September 2000.

[144] Jayashree Subrahmonia, K. Nathan, and Michael Perrone. Writer dependent recognition of on-line unconstrained handwriting. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 3478–3481, Atlanta, GA, May 1996.

[145] C. Y. Suen, S. Bergler, N. Nobile, B. Waked, C. P. Nadal, and A. Bloch. Categorizing document images into script and language classes. In *Proceedings of the International Conference on Advances in Pattern Recognition*, pages 297–306, Plymouth, UK, November 1998.

[146] C. L. Tan, P Y. Leong, and S. He. Language identification in multilingual documents. In *Proceedings of the International Symposium on Intelligent Multimedia and Distance Education*, pages 59–64, Baden-Baden, Germany, August 1999.

[147] T. N. Tan. Rotation invariant texture features and their use in automatic script identification. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7):751–756, July 1998.

[148] Taku A. Tokuyasu and Philip A. Chou. An iterative decoding approach to document image analysis. In *Proceedings of the Workshop on Document Layout Interpretation and its Applications (ICDAR'99)*, Bangalore, India, September 1999. http://www.science.uva.nl/events/dlia99/.

[149] C. I. Tomai, B. Zhang, and Venu Govindaraju. Transcript mapping for historic handwritten document images. In *Proceedings of the $8^{th}$ International Workshop on Frontiers in Handwriting Recognition*, pages 413–418, Ontario, Canada, August 2002.

[150] TREC. Text REtrieval Conference (TREC). http://trec.nist.gov/, 2004.

[151] UDL. The Universal Library. http://www.ul.cs.cmu.edu/, 2004.

[152] Remco C. Veltkamp and Mirela Tanase. Content-based image retrieval systems: A survey. Technical report, Department of Computing Science, Utrecht University, UU-CS-2000-34, October 2000. UU-CS-2000-34.

[153] Christopher Verplaetse. Can a pen remember what it has written using inertial navigation? an evaluation of current accelerometer technology. http://xenia.media.mit.edu/ verp/projects/smartpen/, 2004.

[154] G. Voyatzis and I. Pitas. Chaotic mixing of digital images and applications to watermarking. In *Proceedings of the European Conference on Multimedia Applications Services and Techniques*, volume 2, pages 687–695, Louvain-La-Neuve, Belgium, May 1996.

[155] Wacom. Wacom products. http://www.wacom.com/productinfo/, 2004.

[156] T. Wakahara, H. Murase, and K. Odaka. On-line handwriting recognition. *Proceedings of the IEEE*, 80(7):1181–1194, July 1992.

[157] Dacheng Wang and Sargur N. Srihari. Classification of newspaper image blocks using texture analysis. *Computer Vision Graphics, and Image Processing*, 47(3):327–352, September 1989.

[158] Jia-Ping Wang. Stochastic relaxation on partitions with connected components and its application to image segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(6):619–636, June 1998.

[159] Min Wu and Bede Liu. *Multimedia Data Hiding*. Springer-Verlag, New York, NY, 2002.

[160] Minerva M. Yeung and F. C. Mintzer. Invisible watermarking for image verification. In *Proceedings of the International Conference on Image Processing*, volume 2, pages 680–683, Washington, DC, October 1997.

[161] Minerva M. Yeung and Sharath Pankanti. Verification watermarks on fingerprint recognition and retrieval. *Journal of Electronic Imaging*, 9(4):468–476, October 2000.

[162] Bin Yu and Anil K. Jain. A robust and fast skew detection algorithm for generic documents. *Pattern Recognition*, 29(10):1599–1630, October 1996.

[163] Yu Zhong, Kalle Karu, and Anil K. Jain. Locating text in complex color images. *Pattern Recognition*, 28(10):1523–1535, October 1995.