Analysis of Multi-lingual Documents with Complex Layout & Content

A Thesis

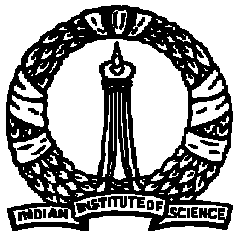
Submitted for the Degree of

Doctor of Philosophy

in the Faculty of Engineering

by

Peeta Basa Pati



Department of Electrical Engineering
Indian Institute of Science
Bangalore – 560 012

April 2007

Synopsis

A document image, beside text, may contain pictures, graphs, signatures, logos, barcodes, hand-drawn sketches and/or seals. Further, the text blocks in an image may be in Manhattan or any complex layout. Document Layout Analysis is an important preprocessing step before subjecting any such image to OCR. Here, the image with complex layout and content is segmented into its constituent components. For many present day applications, separating the text from the non-text blocks is sufficient. This enables the conversion of the text elements present in the image to their corresponding editable form.

In this work, an effort has been made to separate the text areas from the various kinds of possible non-text elements. The document images may have been obtained from a scanner or camera. If the source is a scanner, there is control on the scanning resolution, and lighting of the paper surface. Moreover, during the scanning process, the paper surface remains parallel to the sensor surface. However, when an image is obtained through a camera, these advantages are no longer available. Here, an algorithm is proposed to separate the text present in an image from the clutter, irrespective of the imaging technology used.

This is achieved by using both the structural and textural information of the text present in the gray image. A bank of Gabor filters characterizes the statistical distribution of the text elements in the document. A connected component based technique removes certain types of non-text elements from the image.

When a camera is used to acquire document images, generally, along with the structural and textural information of the text, color information is also obtained. It can be assumed that text present in an image has a certain amount of color homogeneity. So, a graph-theoretical color clustering scheme is employed to segment the iso-color components of the image. Each iso-color image is then analyzed separately for its structural and textural properties. The results of such analyses are merged with the information obtained from the gray component of the image. This helps to separate the colored text areas from the non-text elements.

The proposed scheme is computationally intensive, because the separation of the text from non-text entities is performed at the pixel level. Since any entity is represented by a connected set of pixels, it makes more sense to carry out the separation only at specific points, selected as representatives of their neighborhood. Harris' operator evaluates an *edge-measure* at each pixel and selects pixels, which are locally rich on this measure. These points are then employed for separating text from non-text elements.

Many government documents and forms in India are bi-lingual or tri-lingual in nature. Further, in school text books, it is common to find English words interspersed within

sentences in the main Indian language of the book. In such documents, successive words in a line of text may be of different scripts (languages). Hence, for OCR of these documents, the script must be recognized at the level of words, rather than lines or paragraphs.

A database of about 20,000 words each from 11 Indian scripts[1] is created. This is so far the largest database of Indian words collected and deployed for script recognition purpose. Here again, a bank of 36 Gabor filters is used to extract the feature vector which represents the script of the word. The effectiveness of Gabor features is compared with that of DCT and it is found that Gabor features marginally outperform the DCT. Simple, linear and non-linear classifiers are employed to classify the word in the feature space.

It is assumed that a scheme developed to recognize the script of the words would work equally fine for sentences and paragraphs. This assumption has been verified with supporting results. A systematic study has been conducted to evaluate and compare the accuracy of various feature-classifier combinations for word script recognition. We have considered the cases of bi-script and tri-script documents, which are largely available. Average recognition accuracies for bi-script and tri-script cases are 98.4% and 98.2%, respectively. A hierarchical blind script recognizer, involving all eleven scripts has been developed and evaluated, which yields an average accuracy of 94.1%.

The major contributions of the thesis are:

- A graph theoretic color clustering scheme is used to segment colored text.
- A scheme is proposed to separate text from the non-text content of documents with complex layout and content, captured by scanner or camera.
- Computational complexity is reduced by performing the separation task on a selected set of locally edge-rich points.
- Script identification at word level is carried out using different feature classifier combinations. Gabor features with SVM classifier outperforms any other feature-classifier combinations.
- A hierarchical blind script recognition algorithm, involving the recognition of 11 Indian scripts, is developed. This structure employs the most efficient feature-classifier combination at each individual nodal point of the tree to maximize the system performance.
- A sequential forward feature selection algorithm is employed to select the most discriminating features, in a case by case basis, for script-recognition.

[1]The 11 scripts are Bengali, Devanagari, Gujarati, Kannada, Malayalam, Odiya, Punjabi, Roman, Tamil, Telugu and Urdu.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

"Inventions have long since reached their limit, and I see no hope for further development." – Julius Sextus Frontinus (Highly regarded engineer in Rome, 1st century A.D.)

Summary:

Document processing has changed the way we look at paper documents and interpret them. It has become an inherent part of the office automation process. Though a lot of research work has been carried out in this field, large areas remain to be explored. Page Layout Analysis (PLA) is the branch of document processing which segments a document image into its various constituent regions, such as text, pictures and graphs.

Identification of script is one of the primary tasks for recognition of the document in a multi-script scenario. In India, such documents are a common occurrence. A script recognizer, therefore, eases the task of OCR by enhancing the accuracy of recognition and reducing the computational complexity.

1.1 Introduction

Historians draw a distinction between pre-history and history, with history defined by the advent of writing (1). With advent of writing came the documents which, according to *The Wikipedia*, refers to actual products of writing, intended to communicate or store collections of data. Traditionally, the medium of a document was paper and the ink was used for writing. Through time, documents have also been written with ink on papyrus or parchment, scratched as runes on stone using a sharp apparatus and on clay tablets. Thus, the document is a technology with millennia of advancement attached to it.

Presently, our daily life is governed by the efficiency with which we process information. We produce, transmit and consume information by documents. Millions of documents are being both produced and consumed every day, the world over. Documents organize and present information for human understanding. Technology to automate the process of producing and using documents helps us to achieve significant productivity enhancement.

Though the present day documents exist both in printed and electronic format, humans prefer consumption of paper documents. This is owing to the many advantages of a paper document. They are: (i) paper allows flexible navigation though documents, (ii) cross-referencing of several documents at the same time, (iii) allows annotations, and (iv) interweaving of reading with writing. Besides, paper is cheap, lightweight, thin, easily portable, un-powered, stable and provides us with better resolution of printing. Thus, paper is still the preferred medium of communication of ideas and information even with fancy technological stuff at hand.

Today's document, besides text, contains many other items to make it visually more appealing and for the ease of communication. Items such as photographs, graphs, barcodes, hand-drawn sketches are regularly used along with hand-written stuff and signatures. Many official documents contain logos and seals representing the office. The IEEE definition of a document is, "A medium, and the information recorded on it, that generally has permanence and can be read by a person or machine." A document could assume one of the following *avatars*.

(i) Text document – contains predominantly free flowing text, either printed or/and handwritten. It may contain some graphic objects such as a photograph, a logo or a signature or any combination of the above items. Example of these documents are newspaper, research articles and story books.

(ii) Form – is a representation of n-tuple data items in a document format. Each of these items contain at least two different categories of information. They are the information item name and its value. The value could contain one (*eg.* gender field) or more pieces of information (*eg.* address item). They could contain printed and handwritten matter; sometimes the printed matters could have a lot of variation in font size and style. They may or may not form complete sentences or a logical layout of text. However, they have a specific structure of representation.

(iii) Map or engineering drawing – mostly representative of a geographical locality or an object such as a machine, instrument or a building. They contain a lot of curves and lines and very little text to represent vital information about these lines. Such text is mostly uncorrelated text and doesn't form sentences according to strict grammatical

rules. These documents neither have a specific structure of text representation nor have a strict layout.

(iv) Postal mails and commercial documents such as bank cheques – these items have a strict logical format but have a large physical layout for their representation. They, mostly, contain non-text items either as a part of the background or as components of the document.

(v) Musical transcripts – these documents mostly contain a set of symbols to represent information. This is more graphical in nature and hardly contains any text.

(vi) Braille documents – documents used for imparting information to visually impaired people. Here, each alphabet in the set is formed out of a distinct spatial arrangement of dots.

Typically, a document has the following stages in its life cycle: (i) conception, (ii) creation, (iii) distribution, (iv) modification, (v) archival, (vi) retrieval, (vii) reuse, (viii) destruction. However, in the present age, it has become a common practice for a document to be scanned and stored in its digital form, re-used for various kinds of purposes in its varied forms, such as image and editable text.

Document image analysis is the process that performs the overall interpretation of document images. In other words, it converts the tuple of numbers, representing the pixels, to meaningful information. Document analysis concerns the various issues involving recognition of written language in images. It acts as a super-structure to the OCR and establishes an organization of the document. Many a times it considers application of the contextual prior knowledge to enhance the interpretation of a given document. Thus, a document analysis system involves a suitable amalgamation of the simpler mechanisms of image segmentation, component separation, layout understanding, symbol recognition and application of contextual rules.

The document image analysis systems provide fast storage, recall and distribution of documents in work-flow processing and other applications. Besides, they help in indexing for archiving and retrieval. The partitioning of the image into subregions of interest could also easily be achieved by these systems.

Office automation is becoming a booming technological factor for better efficiency and quality service, and product delivery. The factors such as customer satisfaction are governing the functioning of an office. Fortunately, the process of automation, supported by the information technology is rapidly being employed to attain this stage. Automated reading and interpretation of the hard-copy documents and other such existing media is

an integral part of this process. The system of office automation is incomplete without a fool-proof OCR and text analyzer.

In the recent past, some amount of progress has been achieved for development of OCR's which could deliver the output at specified quality levels. However, such OCR systems operate within constrained environments such as text only page and quality printing. An OCR system which could function like a human being is a far fetched dream and there are hundreds of hurdles to be cleared before the mankind manages to get such a system. The following are a list of such hurdles which tell why any OCR system is far from being used by a common man.

1. The present day OCR technology mostly deals with scanned images. The performance of these systems on camera captured images is far from being acceptable.

2. The OCR's work efficiently on text only pages – a normal situation would deal with documents containing more components than the text.

3. The OCR's deal with good quality printed material. The failure on the old and newspaper prints is high.

4. There are too many constraints adopted to recognize the handwritten material with acceptable efficiency.

5. The performance on degraded documents is poor.

6. The efficiency of recognition for low scanning resolution images is poor. If the scanning resolution is higher, we have problems of storage and time efficiency of operation. The outcome is that the system is far from being real time.

7. The methodologies adopted for the development of OCR technology is not human-like. Human beings don't seem to analyze the page first, segment the text areas and then read it. They do all things in parallel. Thus there is a good need for massive parallelization of the OCR technologies.

8. The OCR technologies are too much dependent on the script. Every technology is fine tuned for a particular script.

9. There is hardly any scope for automatic learning of the technology itself from the mistakes and new situations it faces. We are far from being able to develop a technology which will design its own algorithm looking at the situation.

Though the hurdles are too many, steady progress is being made by researchers around the globe. The aim is to attain a stage where a machine would be able to read a document like a human being. The present state of technology claims to have a good amount of understanding of the problem and a feasible solution for mono-script & text-only pages. So the obvious next step is a technological stage where we would be able to read the documents containing components such as pictures, handwritten stuff, logos, signatures, graphs and hand-drawn sketches. Interpretation of these images and their correlation to the context of the text to assimilate information may be difficult to achieve immediately. But a system with capability to extract the text only portions of the page and read them would be much better than the existing one.

On similar lines, an OCR's capability to read more than a single script makes it more usable. This is, especially, true for many parts of the world, including India, where many documents have more than a single script. Sometimes, each document contains text from a single script while the OCR engine is designed to read documents from different scripts. In such situations, a script specific information could be supplied, automatically or manually, to the system. However, there are situations, when the script alters at paragraph level, necessitating its recognition at that level. Research works for efficient automatic recognition of the script at these levels of the document are already reported. However, in the Indian context, there are situations, mostly found, where the script varies at the word level. It is a normal situation to have documents of an Indian script which contain intermittent words in Roman script. This could be done for emphasizing, for explaining the context or because the word is best represented using the Roman script. So it is not only necessary but important to recognize the script at the word level. This supports the system by enhancing the accuracy, reducing the search space of the library of reference patterns or functions and by adopting an optimal path for the OCR technology. For example, an OCR technology for recognition of Devanagari script using shiro-rekha removal approach could make use of this script information to remove the headline from the word.

The purpose of this work is to explore these two possibilities which would take the OCR technology to the next level of its existence. Here, adapting the same framework to accomplish the two different jobs has been explored. Various other studies associated with these two process blocks have also been presented here.

1.2 Page Layout Analysis (PLA)

Recognition of characters in a document is always preceded by a layout analysis of the document image. The layout analysis involves several operations such as determining the skew, separating picture from text, and partitioning the text into columns, lines, words, and connected components. However, for documents with a complex layout consisting of many different components, appropriate processing is needed to detect and extract text regions. This needs separation of text and non-text regions which finds many useful applications in document processing (2). Performance of an Optical Character Recognizer(OCR), greatly depends on the efficacy of this separating task.

Document Image Analysis(DIA) involves the automated interpretation of images in printed and handwritten documents, including forms, postal envelopes, bank cheques, engineering drawings and maps. Several systems which work in specific domains, like the ones mentioned above, have been developed. A majority of the work on camera captured data has processed text from video or still images. The processing of image text is a specific application which seeks to recognize text superimposed on the image such as subtitles, sports scores or movie credits and scene text on buildings, vehicles, name tags or in T-shirts.

Each of the above problems has its unique challenges but all are directed toward the ultimate goal of providing devices with text reading capabilities. An efficient scheme for detection, localization and extraction of textual information from images is necessary for such a task. The challenge is to deal with images which have text of various style and size on a complex background.

Text extraction from images has received a great deal of attention as it provides a supplemental way of automatic annotation and efficient indexing of images. Such text eventually helps in mining the relevant images from the database. However, to achieve a stage where one could separate the text areas from all other non-text elements, one needs to understand the various properties of the text present in such images. The following section briefly describes these properties.

1.2.1 Properties of Text in Images

1. Geometry:

    - Size: Although the character size can vary a lot, assumptions can be made depending on the application domain.
    - Alignment: The caption texts appear in clusters and usually lie horizontally, except when there are special effects. This does not apply to scene text, which

has various perspective distortions. Scene text can be aligned in any direction and can have geometric distortions.

- Inter-character distance: Characters in a text line have nearly a uniform distance between them.

2. Color: The characters tend to have the same or similar colors. This property makes it possible to use a connected component based approach for text detection. Most of the research reported till date has concentrated on finding text strings of a single color. However, video images and other complex color documents can contain text strings with more than two colors for effective visualization, *i.e.*, different colors within one word.

3. Edge: Most caption and scene text are designed to be easily read, thereby resulting in strong edges at the boundaries of text and background.

4. Compression: Many digital images are recorded, transferred and processed in a compressed format. Thus a faster text information extraction system can be achieved if one can extract text without decompression.

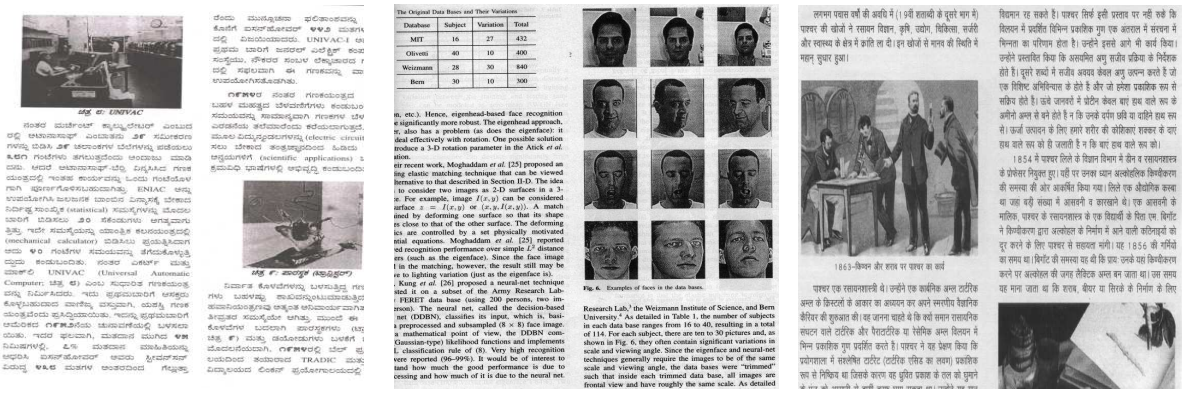


Figure 1.1: Sample Gray scale scanned document images.

Figures 1.1-1.3 show examples of text images. Page Layout Analysis(PLA) usually deals with document images as shown in Fig. 1.1 and these images have uniform background. Figure 1.2 shows some images, acquired by camera, which have embedded text, *i.e.*, exist naturally. Images with text are shown in Fig. 1.3 where text is artificially overlaid on the images.

Figure 1.2: Camera captured images with embedded text.

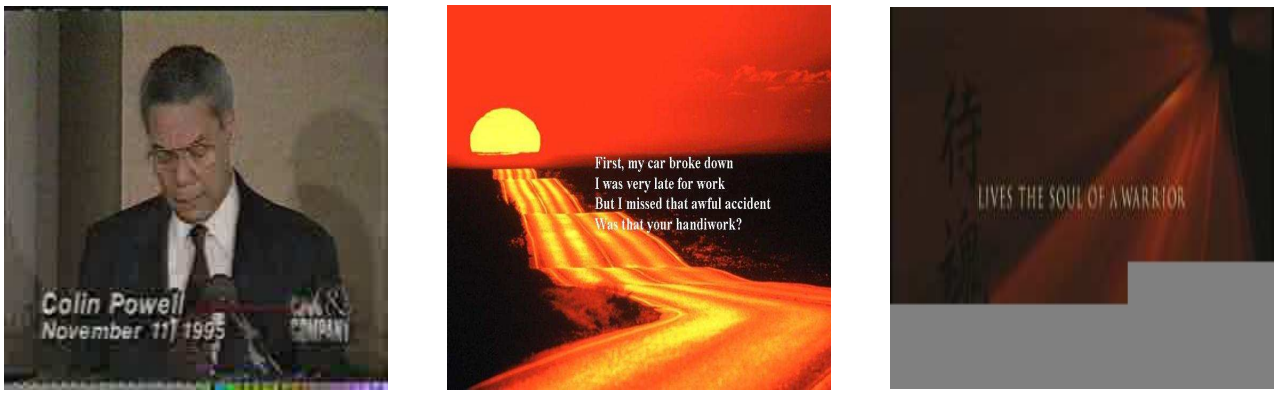


Figure 1.3: Images with superimposed text.

1.2.2 Overview of Text Extraction Methods

This section presents a literature survey of methods for extracting text from images. There has been growing interest in the development of methods for detecting, localizing and segmenting text from images. There are two primary methods for text extraction: (i) Connected Component(CC) based method and (ii) Texture based methods.

Connected Component Based Methods

A Connected component is a set of pixels grouped together based on connectivity. Thus, all pixels in a connected component share similar pixel intensity or color attributes, and, are in some way connected with each other. Extraction of various disjoint, connected components in an image is central to many automated image analysis applications. Connected component based methods identify all the components in the image and analyze their geometrical characteristics.

Physical page structure can be viewed in a hierarchical structure and thus can be

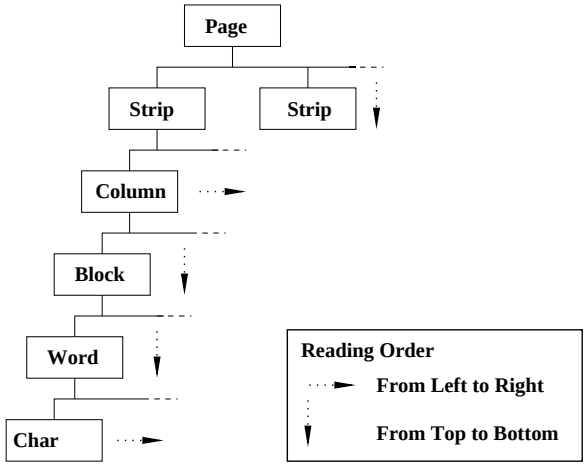


Figure 1.4: The typical physical layout of scientific journals.

represented by an n-ary tree (Fig. 1.4). With this representation, the reading order can be deduced from the level of depth of the layer. Dashed arrows on the figure mark the reading order (either from top to bottom or from left to the right). For example, strips are read from top to bottom on the page, while columns are read from left to the right within the strips.

Traditional techniques for CC based segmentation are bottom up (3) and top down approaches (4). In bottom up technique, connected components are grouped together to form progressively higher level descriptions. They are usually more flexible than top-down methods; however, they may suffer from the accumulation of errors when going from the small scale details up to the large scale features. Top down approach looks for global information in the page and splits the page from column level to word level. For the procedure to be effective, prior knowledge about the structure of the page is necessary. These methods are effective when the layout is constrained, such as is often the case when considering pages from scientific journals. Some techniques combine both the bottom-up and top-down approaches. Other techniques for page layout analysis are based on run length smoothing (5), run length smearing (6), and fractal signature (7).

In the case of color images, textual analysis is carried out based on similarity of color and size of the components. Messelodi *et al.* (8) have extracted connected components to characterize text objects, based on their size information, in book cover color images. They utilize the heuristics which depend both on the geometrical features of a single component as well as its geometrical and spatial relationship with other components.

Zhong *et al.* (9) locate text in complex color images. They propose a method based on quantizing the color space based on peaks in the histogram. Adjacent colors are merged into these peaks. This assumes that the text occupies a narrow region of color space and

a significant portion of the image. Color components are then labeled as text based on their geometrical properties and if there are at least three characters aligned horizontally. If the color quantization step fails because the text has low contrast or occupies a small portion of the frame, then a character may be broken into multiple segments. The other method uses the heuristic of high horizontal spatial variance to localize text. They report results for compact disc and book cover images and the spatial variance method on car scenes from video.

Jain and Yu (10) extract a set of images by analyzing the color space of the input image. They employ connected component analysis (CCA) on each of the derived images to locate possible text regions. Finally, they merge the information so obtained to locate text regions in the original image. Strouthpoulos *et al.* (11) have proposed a color reduction technique to determine the optimal number of unique colors present in the input image. This technique is based on an unsupervised neural network classifier and a tree-search procedure. *Split-and-merge* conditions are used, during an adaptive process, to decide whether color classes must be split or merged. They use a page layout analysis technique, on each of the obtained optimal color images. Finally, they add the information obtained from each color image to extract the text region.

Texture Based Methods

Wang and Srihari reported in 1989 (12) the maiden work in this field using texture analysis. Their work assumed rectangularity of page blocks. They also assumed that such blocks are well separated, by sufficient horizontal and vertical space, from each other. Besides, their documents were assumed to be aligned with the world co-ordinate system, *i.e.*, without any kind of skew being present.

Jain and Bhattacharjee (13) have reported a Gabor filter based technique for separating the text and non-text regions in a document image. They have assumed the non-text region to be consisting of only natural images and empty background areas. The reported method does not deal with line based drawings. Chan and Coghill have taken a similar approach for text analysis as well as script identification (14).

The technique reported by Antonacopoulos & Ritching (15) segments and classifies a document image based on white tiles. This is in contrast to other approaches, which rely on foreground information. White tiles provide a flexible data representation, combining advantages of both rectangles and polygons. Besides, it avoids the lack of efficiency in dealing with arbitrary layouts and page skew. This system gives robust representation of complex shaped regions and can only process binary image.

An approach for document classification based on Neural networks is proposed by Jain

and Zhong (16). They train a two layered network, with back-propagation algorithm, for generating a set of masks. They use these masks to best discriminate between text, background, line-drawing and pictures. The training is accomplished by using sample data from each of these classes. Convolving these masks with the input image produces textural features. These features are used by the network to classify each pixel into one of three classes: (i)text and graphics, (ii) picture, and (iii) background. The regions belonging to the text and graphics class are binarized and connected component analysis separates text and graphics. This method is robust to different scripts. The need for a large training set is the major disadvantage of this system, as it consumes a large amount of time to generate such a set.

A stationary Hidden Markov Model (HMM) based scheme has been reported by Chen & Ding (17). A model is generated for each texture type and each model is trained independent of the models corresponding to other classes. If a new texture class is added, a new model is created only on samples of this class. This is in contrast to many neural network based approaches, where a complete *re-training* of the networks is necessary for all samples including the ones for which it has already been trained.

1.3 Script Identification

Script is a particular orthography or writing system. Script may refer to any of various writing forms which carries in its meaning an aspect of precision, direction, and definition. In simple terms, script consists of a set of symbols and a set of rules to make use of these symbols such that information is represented in a graphic form on some medium. Such media could be paper, leaf, stones, clay-tablets, metal plates or electronic media. Thus, script is used as a representation of a language. A given script could be used to represent many languages; *e.g.*, Devanagari, an Indian script, is used to represent languages such as Sanskrit, Hindi, Marathi and Nepali. Roman script represents many European languages including English. There are also cases where a single language is represented by more than one script. For example, Manipuri language could be represented by either the Bengali script or by Manipuri script. But, generally we have a single script to represent a language. Some scripts have an alphabetic form of representation (Roman) while some others are phonetic in nature (most of the Indian scripts). Lo (18) has compiled the history and evolution of various scripts.

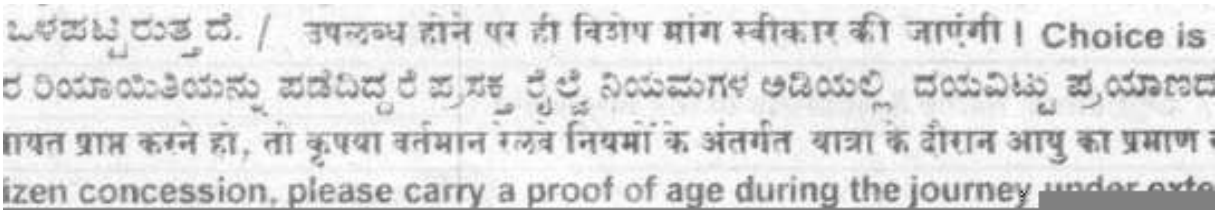
India is a multi-lingual country. Here most of the people use more than one language for communication. Thus it is imperative to say that most of the people, who use script, are able to use more than one script for communication. Many of the documents here

are multi-script in nature. Though English language and Roman script are the binding factors across the various geographical barriers of the country, most states have a language and script of their own. India does have a national language which is Hindi and it is represented with Devanagari script. So, most of the documents prepared by the various governmental offices, carry either two or three scripts. Figure 1.5 (a) shows a railway reservation form which has three scripts for representation of information. Here, Roman script is used besides Devanagari (script of the national language) and Kannada (the local script). Similarly, figure 1.5 (b) shows a portion of the passport application form containing two scripts. There are numerous examples of books and other printed matter in local script containing intermittent words in Roman script. This is resorted to because the word does not have an equivalent in the local language or the word is best represented in Roman script for easier understanding. Figure 1.5 (c) represents such an image where Devanagari is the primary script with intermittent English words.

Multi-script documents are a common occurrence in India. Besides, they appear in most of the Asian, African and Latin American countries. There are many countries in the world, such as Singapore, which have more than one language as state language. These countries have ample number of multi-script documents. Moreover, with globalization of economy, the instruction manuals of various industrial products are being printed in multi-language and multi-script configuration.

As has been mentioned in the previous sections, document image analysis has a big impact on the way people lead their lives. In future, it is also going to have an impact on the way we carry on with our everyday business. Script identification has the prospect of making the document analysis a much easier job in many respects. Some of the contributions of script identification to the domain of DIA are noted below.

- It helps reduce the algorithmic complexity of the DIA.
- It reduces the search space for the recognition system, hence, generating faster results.
- For some scripts such as Devanagari, the *shirorekha* is removed before segmenting the word to characters. Prior information about the script helps to detect the position of the shirorekha and its efficient removal.
- Sometimes, different scripts have similar looking characters, such as the character “O” in Roman script and “ṭH” in Odiya script. In these situations, the script of the word helps decide the actual identity of the character.



(a)

8. *यदि आवेदक निर्देश के पैरा (7) में दिए गए पते पर लगातार गत वर्ष से नहीं रहा है तो, कृपया उन स्थानों के पते और अवधि दें जहां इस अवधि के दौरान रहा हो। आवेदक ऐसे प्रत्येक अतिरिक्त स्थान के लिए जहां वह रहा/रही हो वैयक्तिक विवरण पत्र का एक अतिरिक्त सैट (फोटो लगी दो प्रति) जमा कराये। इस उद्देश्य के लिए वैयक्तिक विवरण पत्र की फोटो कापी करवा लें।*

If the applicant has not resided at the address given against para (7) continuously for the last two years, please furnish the other address(es) with duration where he/she has resided during the period. Applicant should furnish an additional set of Personal particulars form (in duplicate with photograph) for each additional place of stay during the last ONE YEAR. PP form may be photocopied for the purpose.

से/From..... *तक*/to .....*(दिनांक*/ Date) *से*/ From .....*तक*/to .....*(दिनांक*/Date)

मकान नाम/नं० / House Name/No. .....
गाँव/गली / Vill./Street .....
डाकघर / Post Office .....
पुलिस थाना / Police Station .....
जिला / Distt. .....
राज्य / State .....
पिन कोड / PIN Code .....
टेलीफोन सं० / Tel No. .....

मकान नाम/नं० / House Name/No. .....
गाँव/गली / Vill./Street .....
डाकघर / Post Office .....
पुलिस थाना / Police Station .....
जिला / Distt. .....
राज्य / State .....
पिन कोड / PIN Code .....
टेलीफोन सं० / Tel No. .....

(b)

पढ़ें और फिर किताबें पढ़ के एक लेक्चर दें। I am not lecturer कई कई लोग कहते, आज जी उनका Lecture होगा। मुझे बड़ा अजीब लगे। ये Lectures क्या हुआ भई? Lectures are prepared by lecturers.

यहाँ कोई आये हो कालेज Professor या स्कूल के Teacher स्कूल के टीचर को भी आज कल तैयारी करनी पड़ती है लेकिन कॉलेज के प्रोफेसर को, University के प्रोफेसर को एक Subject के ऊपर बोलने से पहले तैयारी करनी पड़ती, पढ़ना पड़ता है। Latest news क्या हैं? उनका सबका Viva करना पड़ता है। पर मेरे लिए यहाँ आ के बैठना ओर बोलना बस यही तैयारी कि यहाँ आ के बैठ गये।

(c)

Figure 1.5: Sample Indian bi-script and tri-script documents. (a) A railway reservation form showing three different scripts, (b) A bilingual governmental form with distinct scripts. (c) A sample bi-script document showing interspersed Hindi and English words.

- Automatic script recognition facilitates applications such as archiving of multilingual documents and searching on-line archives of such document images. This helps to select a script specific OCR in a multilingual environment.

1.3.1 Properties of Indian Scripts

The languages of India primarily belong to two major linguistic families. They are Indo-Aryan and Dravidian, spoken by about 74% and 24% of the population of the country, respectively (1). Though the number of spoken languages in India is as high as 1652, there are only 24 languages which are spoken by a million or more people. Hindi language, using the Devanagari script, has been identified to be the official language of the union of India. English language has also been accepted as a co-official language of the country. Besides, the different states of the country have recognized one or more official languages for their respective states. Today, the Constitution of India recognizes a total of *twenty-two* **national languages**, spoken throughout the country. They are: Assamese, Bengali, Bodo, Dogri, Gujarati, Hindi, Kannada, Kashmiri, Konkani, Maithili, Malayalam, Manipuri, Marathi, Nepali, Odiya, Punjabi, Sanskrit, Santhali, Sindhi, Tamil, Telugu and Urdu.

Most of these languages have a common alphabet set. However, the symbols used for the representation in different languages are different. Typically, each language is being represented by a distinct script. The script is divided into two major groups of characters: (i) vowels and (ii) consonants. Each script has about 12 vowels and 37 consonants. Each symbol has a phonetic representation and the combination of the vowels and consonants yield a very rich set of symbols, and hence, phonetic representations. The concept of upper and lower case characters is absent in the Indian system. The writing style of left to right is followed for most of the languages. The alphabet set for all these scripts are native to India. All these native scripts are derived from the ancient Brahmi script. Only the symbols used for representation of Urdu and Kashmiri have an Arabic origin & hence, are written from right to left. The Roman script is used to represent English and many other native languages of the north-eastern state, Nagaland.

The numerals in Indian scripts are the origin of the "Arabic" numerals used in European writing systems. This is because Arabic numerals, borrowed by Europeans, were themselves borrowed from India by the Arabs (19). The scripts used for most northern Indian languages are closely related to Devanagari. South Indian scripts generally have a much rounder shape. There is a lot of visual similarity between the (i) Telugu and Kannada, (ii) Kashmiri and Urdu, scripts. This similarity is to the extent that a reader of one script could read the other script without any difficulty. There is a large similarity between the (i) Devanagari – Bengali and (ii) Devanagari – Gurumukhi scripts, as well.

(a) (b)

Figure 1.6: (a) A sample image showing the *shiro-rekha* and vertical strokes. present in Devanagari script. The words shown here are "Devanagari Lipi", written using the Devanagari script. (b) The word "Bharatiya" scribed in twelve different Indian scripts, including Roman script.

Details of all these scripts could be obtained from (20) and (21).

The presence of a horizontal line running on top of the base characters in a word, joining them, is called a head-line or a *shirorekha* and is demonstrated in figure 1.6 (a). It could also be easily located in all the three images in figure 1.5. This is a feature with Bengali, Devanagari and Gurumukhi scripts. Besides, these scripts are dominated by the presence of a vertical straight line (shown in figure 1.6 (a) running from the shirorekha to the base of alphabet, present with a majority of the members of the set.

Though the characters in Gujarati script have vertical strokes, as in Bengali and Devanagari alphabets, they don't have a shirorekha joining the characters to form a word. Vertical strokes are also present in Odiya script. Similarly, the Kannada, Telugu, Malayalam, Odiya and Tamil scripts contain a lot of curves and loops in their characters.

Different Indian scripts have different structural and textural properties. This is amply shown in figure 1.6 (b), where the word "Bharatiya" has been represented using 12 different scripts. The scripts used here are: Assamese, Bengali, Gujarati, Devanagari, Kannada, Telugu, Malayalam, Odiya, Gurumukhi, Tamil, Manipuri and Roman, arranged in the row-wise fashion. Thus, analysis of the structure and texture content of documents has a lot of promise in separating the different scripts. This has already been exploited by many researchers for Indian and other scripts. A comprehensive review of these works is presented in the following section.

1.3.2 Survey of Script Recognition Algorithms

Quite a few results have been reported in the literature, identifying the scripts in multi-script documents. However, very few of these works deal with script identification at the word level. In the section presented below, we review the literature on script recognition at a paragraph and line level.

Script variation with Paragraphs/Blocks & Lines

Spitz (22) uses the spatial relationship between the structural features of characters for distinguishing Han from the Latin script. Asian scripts (Japanese, Korean and Chinese) are differentiated from Roman by an uniform vertical distribution of upward concavities. In the case of the above Asian scripts, the number of ON-pixels per unit area is employed to distinguish one from the other. Hochberg *et al.* (23) use cluster based templates for script identification. They consider thirteen different scripts including Devanagari. They cluster the textual symbols (connected components) and create a representative symbol or a template for each cluster. Identification is through comparison of textual symbols of the test documents with those of the templates. However, the requirement of the extraction of connected components makes this feature a local one. Wood *et al.* (24) suggest a method based on Hough transform, morphological filtering, and analysis of projection profile. Their work involves the global characteristics of the text.

Chaudhuri *et al.* (25) have proposed a method, based on a decision tree, for recognizing the script of a line of text. They consider Roman, Bengali and Devanagari scripts. They have used the projection profile, besides statistical, topological and stroke-based features. At the initial level, the Roman script is isolated from the other two, by examining the presence of the **headline**[1] (shirorekha). Devanagari is differentiated from Bengali by identifying the principal strokes (25). In (26), Pal & Chaudhuri have extended the the above work to the identification of the script from a given triplet consisting of Devanagari, Roman and a state language, where each line of text contains only a single script. Here, they have dealt with almost all the Indian scripts. Besides the headline, they have used some script-dependent structural properties, such as the distribution of ascenders and descenders, the position of the vertical line in a text block, and the number of horizontal runs.

Tan (27) has suggested a method for identifying six different scripts using a texture based approach. Textual blocks of 128×128 are taken and filtered with angular spacings

[1]Both Devanagari and Bengali scripts have a horizontal line at the top, known as ***shirorekha***, which connects the characters in a word.

of 11.25^o. This method requires image blocks containing text of same script. Roman, Persian, Chinese, Malayalam, Greek and Russian scripts, with multiple font sizes and styles (font invariance within the block), are identified. Chaudhuri and Seth (28) have proposed a technique using features such as the horizontal projection profile, Gabor transform and aspect ratio of connected components. They have handled Roman, Hindi, Telugu and Malayalam scripts.

On similar lines, Chan & Sivaswamy (29) and Joshi *et al.* (30) have used Gabor features for classification of most of the Indian script documents. They assume a block of text to consist of characters from the same script and employ a multi-channel log-Gabor filter bank to discriminate between the various scripts. Manthalkar and Biswas (31) have used rotation-invariant texture features, using Gabor filterbank, to differentiate between text blocks for various Indian scripts. Ablavsky and Stevens (32) make use of geometric properties of the text structures at connected component level to classify the script of lines of text. Gllavata and Freisleben (33) make use of a set of textural and structural features such as wavelet coefficients, horizontal projection profile, to separate the Roman script from Ideographic scripts.

Script variation with words

Recognition of the script, using statistical features, at word level has been reported by Dhanya *et al.* (34). Here the authors tried to differentiate Tamil and Roman scripts using various feature-classifier combinations. This work has been extended by Pati *et al.* (35) and (36). Here the authors have identified Odiya and Hindi scripts, besides Tamil, against the Roman script. They use a bank of Gabor filters for feature extraction with linear discriminant classifier for decision making. Besides, redesign of the Gabor functions has enhanced the system efficiency. Ma & Doermann (37) use a filterbank of Gabor functions to separate Roman script from non-Roman (Arabic, Chinese, Devanagari & Korean). Jaeger *et al.* (38) have combined the informational confidence values of the various classifiers to improve upon the above system, on the same set of scripts. In the both the above cases, the authors have dealt with bi-script documents.

On the contrary, Pal and Chaudhury (26) have tried to discriminate Roman, Devanagari and Bengali scripts at the word level using a set of structural features and a tree based classifier. They have extended their approach to identify scripts in another triplet, Roman, Devanagari and Telugu scripts (39). Later they have proposed a script identifier for 12 different Indian scripts (40) using the same structural features. Padma and Nagabhushan (41) have discriminated Roman, Devanagari and Kannada scripts on similar lines.

Classifiers for Script Identification

Researchers have reported use of various classifiers for the purpose of identifying script at different levels like blocks/paragraphs, lines and words. Spitz (22) has used linear discriminant analysis (LDA). Hochberg *et al.* (23) use a variant of nearest neighbor classifier (NNC) where instead of using the minimum distance to the prototype of a class, an average of the distances to all the representative patterns of any class is considered. Tan (27) uses NNC to the mean of script clusters with normalized features where the normalization is accomplished with their respective variance. Chaudhury & Sheth (28) also use NNC for classification. Joshi *et al.* (30) have experimented with a number of classifiers namely, Quadratic Bayes Normal Classifier, Neural networks, k-Nearest Neighbor, Support vector machines (with Polynomial, Radial basis Exponential kernels) and Parzen density based classifier. They have observed that of all these classifiers, the k-NN classifier delivers the best performance. Ablavsky (32) and Gllavata (33) use k-NN classifier. Doermann (37, 38) uses NNC, SVM's and Gaussian mixture models as classifiers. Pal and Chaudhuri (26, 40) use a tree-structured syntactical classifier. This is because they rely on the structural properties of the scripts, rather than the textural properties.

Though some works have been reported for identifying the script of the Indian documents, the work is still in its infancy. In the present work, we explore the effectiveness of our approach (36) in recognizing word-level change of script up to three or eleven scripts.

1.4 Conclusion

In this chapter, an introduction to automated document processing system is briefly described. The current state of the art in this field has been reviewed and the immediate requirements for taking the existing system to the next level of of its development are mentioned. The immediate goals are also marked and possible future developments have been mentioned. The motivation for development of a document analysis system is presented.

Two of the immediate goals of the document processing system, for which the thesis makes contribution, are inspected. The various properties of text elements present in document images are described. On similar lines, a survey of the Indian languages and scripts is given. The state of the art in research and available literature, for both page layout analysis and script recognition, has been discussed.

Chapter 2

Text Extraction from Gray Images

"...it doesn't matter how beautiful your theory is, it doesn't matter how smart you are – if it doesn't agree with experiment, it's wrong." – R.P. Feynman.

Summary:

Extraction of text areas is a necessary first step before subjecting a complex document image for character recognition task. Besides text, documents may contain varieties of non-text elements. Further, the layout may be Manhattan or complex. These document images are acquired either by scanner or using a camera. Besides, they might be stored in 2-tone, 8-bit per pixel or 24-bit per pixel format.

The non-text objects include graphics, pictures, logos, signatures and other such items. In general, each object has a different textural property. Hence, a text separation algorithm is proposed in this chapter based on texture information. This algorithm operates on the illumination component of the document image and combines the structural and textural properties of the objects to separate text from non-text objects. The structural properties are exploited with a connected component analyzer. A human vision system inspired Gabor filter bank extracts the textural information. Since the brute force mechanism of processing the textural properties at each pixel is computationally heavy, a selected set of information rich points are employed to yield the result. Such points are detected using a Harris' corner operator. Finally, a Delaunay triangulation based text selection mechanism is employed to filter the last stray pieces of non-text elements from the document image.

2.1 Introduction

Documents occur in all varieties of formats and layouts. Documents may be stored as color images with 24-bits, gray images (8 bits) or in binary format with 1-bit depth information. Image parameters such as scanning resolution and the bit-depth of pixel representation can be determined at the acquisition stage. Feasibility issues involving the business process, storage, transmission & availability of technology, also, determine the optimal parameters of image acquisition. The determination of these optimal parameters of an image are application specific. However, it is always better to have the best resolution and best bits-per-pixel representation for processing of document images. It is easy to throw away information and to go to a lower plane of existence than tread the reverse path. For example, conversion of a 24-bit color image to a 8-bit gray image is much simpler than vice-versa. This is because while color-to-gray is a loss of information, the reverse involves generation of information from almost no-where.

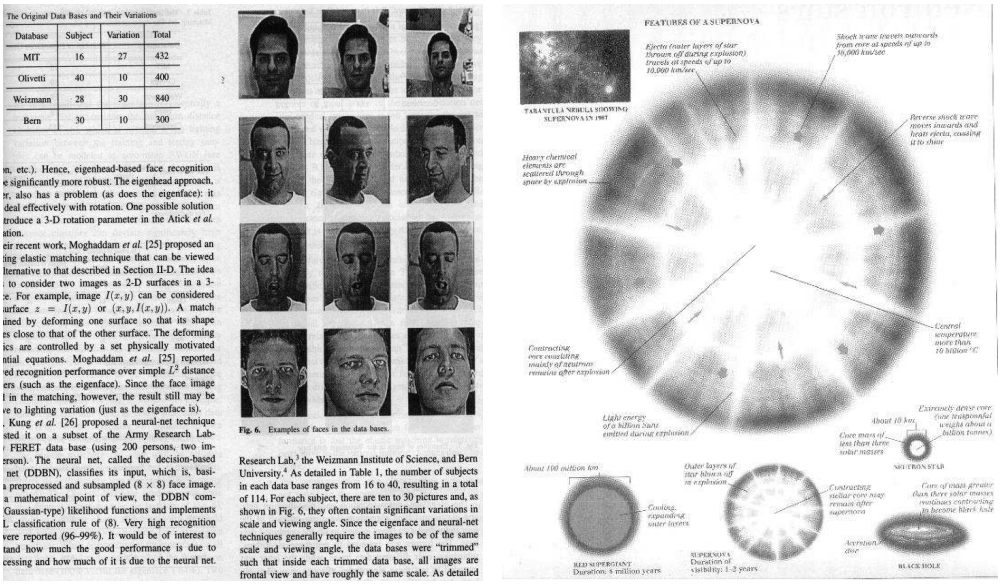
Most of the object specific information in the image are present in its illumination part. This is because, each object is assumed to have a specific texture and it is different from other objects. So, we propose to analyze the texture-only-information of an image to separate the text areas from their non-text counterparts. In the subsequent paragraphs, answer to some of the very common questions is provided.

- **What is a document and How is it acquired?**

  Document is a medium of expression and recording of facts. It has great use when the communicator and the person to be communicated are not available together. Traditionally, leaves, metal plates, clay/wood tablets, stones and paper have been used as media for documentation. However, with technology improvisation, documents could as well be in the form of e-mails, videos, photos and so on.

  Paper has been in existence since 2^{nd} century BC. It is also the most popular and widely used document form. This is owing to the various advantages it offers: (i) cheap, (ii) light, (iii) flexibility and ease of maneuver, (iv) powerless form of representation and (v) bulk availability.

  A document could contain elements besides text to add value to the information. Sometimes symbolic form of representation is used for ease & clarity of expression and security. The various non-text elements which could be found along with text in a document are: pictures, graphs, signatures, logos, hand-drawn sketches. In this thesis, all these non-text form of information representation in a document are collectively referred to as *clutter*. Figure 2.1 shows two sample document images with text and clutter.



The Original Data Bases and Their Variations

Database	Subject	Variation	Total
MIT	16	27	432
Olivetti	40	10	400
Weizmann	28	30	840
Bern	30	10	300

Fig. 6. Examples of faces in the data bases.

FEATURES OF A SUPERNOVA

(a) (b)

Figure 2.1: Sample documents containing text and clutter. (a) Document with text, image and a table with a Manhattan layout, (b) document with a complex, non-Manhattan layout.

- **What is text extraction or Page Layout Analysis?**

  Clutter may be present in a well separated rectangular layout, called Manhattan Layout. When there is no such clear rectangular border, it is referred to as non-Manhattan layout. Figure 2.1(a) shows a sample document with a Manhattan layout while one with a non-Manhattan layout is shown in 2.1(b).

  For many of the automation processes involving text interpretation, there is a need to separate the text areas from the non-text elements and such a separation is called *text extraction*. This enables an easy and efficient text interpretation. This text separation process is referred to as Page Layout Analysis (PLA).

- **Why do we need to separate text from non-text elements?**

  Automation of text interpretation involves character recognition (also known as OCR). An OCR engine is capable of recognizing the characters present in an image. If a document image with non-character elements is fed to an OCR, its performance degrades. So there is a need to separate the text elements from the non-text elements.

  Besides, lots of information may be present in the non-text elements such as pictures and graphs. These elements need independent and parallel interpretation systems. For example, if a face image is present in a text document, the face may be recognized

to extract the context of the document. Such information could also be used for classification, archival and retrieval of documents. This needs separate object class specific interpreters and they have poor performance with objects of text class. Thus, separation of text from non-text elements has double advantage.

- **What kind of information could we make use to extract text from the image?**

  Different approaches have been taken for text-extraction from documents. These approaches could, broadly, be classified into two groups: (i) structure related approaches and (ii) texture related approaches. Structure related approaches deal with the structural aspects such as homogeneity of the structure of the characters, spacing between the characters in the document and other such information. Shape and size information is also considered. Such systems mostly are based on a syntactic pattern recognition principle. Textural approaches consider the text areas to be of a kind of texture different from the non-text elements. These textural aspects of the text and clutter elements are quantitatively represented by a set of features and their classification is based on the principles of statistical pattern recognition.

  A more modern approach to this task is a combination of both these approaches at various levels. Such aspects are explored in the present thesis.

- **Is it computationally feasible? How do we enhance efficiency?**

  Till date, efforts have been mostly towards efficiency in separation and not time. However, researchers have tried to make some of the better established systems to be more time efficient.

  Presently, the available systems are far from being implementable in real-time to process video streams. For A4 documents scanned at 200 dpi, some of the most efficient systems take few seconds to complete the page layout analysis task on a P4 processor based desktop PC.

- **What is a Text Information Extraction System?**

  Text Information Extraction (TIE) involves extraction of information present in the textual part of the document. Such a system involves detecting and localizing the text blocks in the document, extracting the text only areas and recognizing the textual content. Post-processing rules may be applied to enhance context based accuracy and add meaning to the text content. Sometimes, natural language processing rules are applied to extract the knowledge/information present in the text content. Figure 2.2 presents a basic text information extraction system.

Figure 2.2: Schematic representation of a system for Extraction of Text Information.

- **What has already been accomplished in this area?**

  A near-complete review of the available literature and technology has been provided in section 1.2.2. At this stage, it is important to make a note that the available technology/algorithms are far from being able to reliably separate text from any kind of clutter, under all circumstances. Many of the algorithms have restrictions on the kind of clutter and/or the number of clutter objects. Some algorithms assume that text is the predominant object present in the document with some non-text objects to support the textual information. Many a time, the layout of the objects is assumed to be Manhattan type. Very few attempts are made to separate text in camera captured images. Thus, an algorithm to extract text objects in general class of images is yet to be realized.

- **What is being accomplished in this work?**

  In this thesis, an algorithm has been proposed to extract text from a general class of images. The algorithm is shown to work for a wide range of images with many co-existing non-text classes. Though we have not been able to address the timing constraints for real-time operation, there exists scope to optimize the algorithm & implementation for faster operation.

A list of terminologies often used in the literature and a brief description of each of them is provided below.

- **Input Document:** It refers to either a scanned or camera captured digital copy of a document. Sometimes it also refers to video sequences. The input document may be in binary, gray or color representation. Generally, the document is assumed to contain text. However, images without text are also used and such images are called *text-negative* images.

  When the input document is a video, it is split into its constituent frames and processed. So, here, they also are referred to as document images.

- **Text Detection:** Since the input image could either be text-positive or text-negative, we need to find the presence of text in it. This is called as *text detection*.

- **Text Localization:** There is a need to localize the co-ordinates of the text blocks present in text-positive images. This is referred to as *text localization*.
- **Text Extraction:** Removing all the non-text objects, from a document image, to retain only the text blocks is called as *text extraction*.
- **False Positives:** In the process of text-extraction, if some non-text objects get identified as text blocks, they are referred to as *false positives*.
- **False Rejection:** If some text blocks get identified as non-text elements, they are rejected in the text-extraction process. This is called *false rejection*.

2.2 Problem Formulation

A study of the literature reveals the absence of a text extraction system capable of handling multi-class clutter in a complex layout. It is also seen that no single algorithm is robust for detection of an unconstrained variety of text appearing in the document and scene images. A comprehensive study of the various methods is presented in section 1.2.2. The methods reviewed assume that the text regions: (i) are high in contrast against the background, (ii) are composed of one consistent color or gray-level, and (iii) form a major component in the image. In general, the use of a few rigid assumptions about the nature of text in camera images is a weak heuristic. This study aims to develop a framework for reliable extraction of text from documents as well as scenic images.

Formally, the problem can be defined as the development of algorithms for the reliable detection, localization and extraction of text from unconstrained, general purpose documents. We assume a gray image as its input. The result of the text-extraction step will be an image where the text pixels are retained while pixels representing non-text objects are made background. Tighter constraints regarding the nature of text can be applied at this stage, *i.e.,* about their texture consistency and contrast with the background. Figure 2.3(c) presents the extracted text from the input image (Fig. 2.3(a)) while Fig. 2.3(b) presents the expected output from a text extraction algorithm.

Unconstrained text in scenic images is too varied to allow solution by a single approach. A multi-modal solution that uses a number of simple filters whose outputs are combined in an intelligent way proves to be more robust. This approach offers promise of succeeding on a wide variety of images while reducing the risks of any particular problem-specific method. We also propose to optimize and reduce the computation of our algorithm by operating on a set of selected information rich points. A second pass to reassure the text detection process, enhances the quality of the output.

(a) (b) (c)

Figure 2.3: (a) Original image containing text, (b) Image with text localized, and (c) text extracted from image in (a).

2.3 Image Database Collection

Document images are scanned with, (a) **Hewlett Packard Scanjet 2200c**, or (b) **UMAX ASTRA 5400**, scanners. The spatial scanning resolution is set at 200 dots per inch (dpi) in both the horizontal and vertical directions. The images are scanned in 24-bit color representation and stored as Windows Device Independent Bitmap (BMP) format. For gray level processing of these images, they are converted to 8-bit gray level representation. The database contains about 350 such scanned document images, with variations in content, layout and script. Figure 2.1 shows two sample scanned document images containing both text and clutter.

In addition, some document images have been captured by, (i) **Minolta Dimage 2330 Zoom** and (ii) **Sony DSC-T3** digital cameras. The distance between the camera lens and the surface, on which the document is placed, ranges from 0.4 - 0.7 meters. The illumination of the room is typical study room condition, *i.e.*, 160 watts of tube light (4 $\times$ 40 watts) placed at an average distance of 1.5 meters from the surface the document in a room of size 8 $\times$ 5 meters. The focus of the zoom lens camera is auto adjusted. The captured color images are stored in JPEG image format. The database consists of about 100 such images.

The documents in the image database (both scanned & camera captured) are collected from a variety of books, newspapers and from the world wide web network. These documents are selected such that they contain various kinds of clutter (natural images, graphics, line drawings, hand made sketches, etc.). The fonts of the text regions vary both in size and style. Fonts sizes varying between 8 to 22 are accommodated in the dataset. The font sizes are as per the definitions of Microsoft Inc. and are used in MS

Word. The set of camera captured images contain photographs of paper documents and scenic images.

2.4 Technology Description

The proposed page text segmentation scheme involves a number of steps, each of which is examined in detail in the subsections below. Anisotropic diffusion based filtering is employed as a preprocessing step, to get rid of possible background noise from the document image. Connected Component Analysis (CCA) on a binary image is followed by a Gabor filter based local energy computation on the corresponding gray image. A Block Energy Analysis (BEA) is employed on the energy image, consisting of total local energy at each pixel, for separating text blocks from the blocks containing clutter.

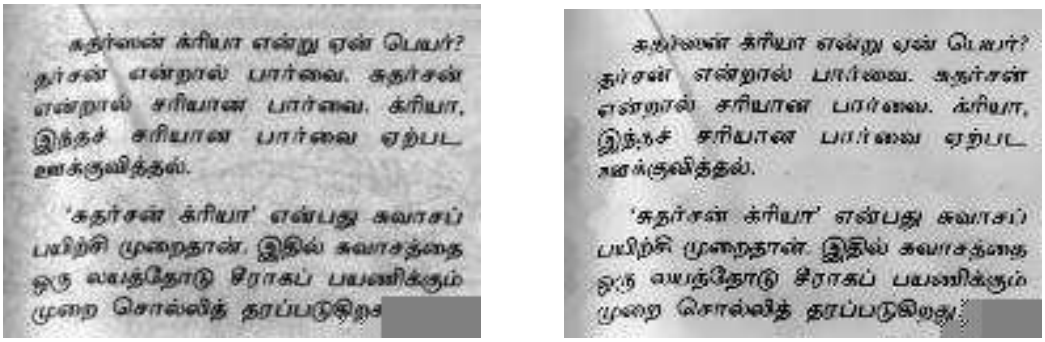
2.4.1 Preprocessing: Anisotropic Diffusion

For an image, $I(x, y)$ and a low-pass filter, $f(x, y)$, the filtering is accomplished by convolving the image with the filter. If $I_f(x, y)$ is the output image, then:

$$I_f(x, y) \;=\; I(x, y) \;*\; f(x, y) \tag{2.1}$$

where '*' denotes a convolution. The subscript 'f' in output image I_f makes mention of the filtering of the original image I with the filter, f. As can be noted from Eqn. 2.1, the intensity of blurring at any image point, (x, y), neither depends on the location nor the local image statistics. This is the major disadvantage in a conventional low-pass filtering. The price paid for removal of noise, using this mechanism, is the decrease of spatial resolution, caused by flattening of sharp edges or blur. When we low pass filter a document image, the edges of the characters are blurred. Many a times, this causes the particular character to be mis-recognized, in OCR systems.

This drawback of low-pass filtering is efficiently overcome by anisotropic filtering mechanism(42). In this mechanism, diffusion coefficient (blurring coefficient) at every image point is appropriately weighed. Such a weight is evaluated based on a suitable monotonically decreasing function, $g(||\nabla I||)$, of the local image gradient, ∇I. A careful choice of the $g(.)$ determines the performance of the algorithm. The final low-pass filtered image is obtained in an iterative process. At time instance t of this process, an estimate of the output image, $I(x, y, t + 1)$, is obtained in the following manner.



(a) (b)

Figure 2.4: (a) A sample document image with show-through effect which makes the background textured and the result of diffusion filtering on this image is shown in (b). It can be noted that the edges are preserved while the show-through effect is largely removed.

$$I(x,y,t+1) = div(c(x,y,t)\ \nabla I(x,y,t)) \quad (2.2)$$

$$= c(x,y,t)\Delta I(x,y,t) + \nabla c(x,y,t)\ .\ \nabla I(x,y,t) \quad (2.3)$$

where, ∇ and Δ represent the gradient and Laplacian operators. $c(x,y,t) = g(||\nabla I(x,y,t)||)$, denotes the monotonically decreasing function of the local image gradient. $I(x,y,t)$ is the filtered scale-space image of the previous step & $I(x,y,0)$ is the initial image.

The anisotropic diffusion based filtering removes noise at pixel locations where gradient doesn't have large oscillations. This could be noticed in figure 2.4 (b) which is the filtered output of Fig.2.4 (a). Here, it can be clearly noticed that the background noise, contributed due to the *show-through* effect, has been removed while the structural elements of the characters are preserved. Anisotropic diffusion filtering is employed only for noisy images.

2.4.2 Connected Component Analysis

While designing the Gabor filter bank, we have assumed that the text regions are richer in high frequency components than the regions containing natural images. However, this assumption of ours is violated by the presence of line based clutter. Line based clutter is also equally rich in high frequency components, since there are abrupt changes of the gray levels in the neighborhood of the pixels. Thus, it becomes difficult to separate these clutter from the text, purely based on frequency assumptions. Such clutter, however, has structural properties different from text elements. This led to the proposition of adopting

a different technique to separate these clutter. We have employed connected component analysis (CCA) to accomplish the job. CCA is able to discriminate the text and graphic portions based on component size information (graphics are assumed to be much bigger in size than individual characters).

The document image is binarized by local thresholding prior to CCA. The gray image is divided into 30 non-overlapping blocks (based on experience), 6 blocks along the direction of height and 5 divisions along the width. This is performed in order to compensate for the variations in the gray levels of the text regions, across different sections of the document image. Such a variation may occur owing to varying illumination intensity and quality of printing or the paper. This helps in adapting the threshold for binarization based on the local image statistics. Within each block, the image is assumed to have a bi-modal histogram, from which the threshold is appropriately chosen.

The whole binary image is analyzed for sizes of the connected components. Let the area, A_i, of a connected component, C_i, be the total number of ON-pixels present in C_i. h_i and w_i are the height and the width of the rectangular box that encloses the component C_i, respectively. If T_A is the threshold used to discriminate graphics from text based on area of components and H & W (H and W are the average height and width of the bounding boxes to C_i's respectively.), then

$$T_A = \frac{3}{M} \sum_{i=1}^{M} A_i \tag{2.4}$$

$$H = \frac{1}{M} \sum_{i=1}^{M} h_i \tag{2.5}$$

$$W = \frac{1}{M} \sum_{i=1}^{M} w_i \tag{2.6}$$

where M is the total number of connected components present. T_A is chosen to be thrice the average area of the C_i's.

Any C_i is considered a clutter if any one of the following conditions is satisfied.

(i) $A_i \geq T_A$ AND, $h_i \geq H$ or $w_i \geq W$

(ii) $h_i \geq 5 \times H$ AND $w_i \leq 0.1 \times W$

(iii) $w_i \geq 5 \times W$ AND $h_i \leq 0.1 \times H$

(iv) $h_i \leq 3$ AND $w_i \leq 3$

Here condition (ii) and (iii) identify the lines. Condition (i) takes care of big objects while (iv) ensures that the noisy dots are removed from the image. The ON-pixels in the clutter, detected by the above method, are set to the gray value equal to the threshold used for binarization of the document in that locality. Such a replacement avoids abrupt changes in the gray levels in the locality, which would defeat the purpose of lowering the energy content in that locality.

2.4.3 Gabor Filter Bank

In a document, the text regions are richer in high frequency components than the non-text regions. This is clearly demonstrated in figure 2.5, where, for the non-text image, the energy is concentrated at the low-frequency regions. It can be noted that the high frequency regions in the Fourier magnitude spectrum of the text image are brighter for the text image (Fig. 2.5(c)) than for the non-text image (Fig. 2.5(d)). This is owing to the fact that the regions consisting of natural images are relatively smooth. Moreover, the text regions contain a lot of abrupt changes in the gray values, in various directions. The directions of the above mentioned abrupt gray value changes are dependent on the properties of the script of the text for regular kinds of fonts. Thus, a block of the image is rich in edge information when it contains text. So, an ideal feature to discriminate such kinds of text and non-text areas would invariably involve directional frequency information.

Biological visual systems are known to involve a set of parallel quasi-independent filtering mechanisms at the cortical level (43; 44). This supports the theory of multi-channel filtering of signals in systems that model biological vision (45). Such a filter scheme employs a filter-bank with each filter modeling a single channel. Moreover, Morrone and Burr (46) have reported that the features, used in human visual system (HVS), responsible for discriminating different kinds of texture information, are direction dependent. Thus, modeling of the directional sensitivity of HVS for texture interpretation is best done by Gabor functions(47; 48). This scheme is popular in texture segmentation and researchers have tried to use it for text page segmentation.

The technique involving multi-channel filtering with Gabor function based filters, for page layout analysis, is developed due to convergence of biological and machine learning approaches. Such a method is meant to detect text regardless of the type of script, size of font or the layout it is embedded in. It is more robust than other kinds of feature detection models. Besides, it has a low sensitivity to various kinds of noise.

A bank of Gabor filters is chosen for extraction of the above mentioned features. This is because of the inherent advantages of the Gabor function. They are: (i) it is the only function for which the lower bound of space bandwidth product is achieved,

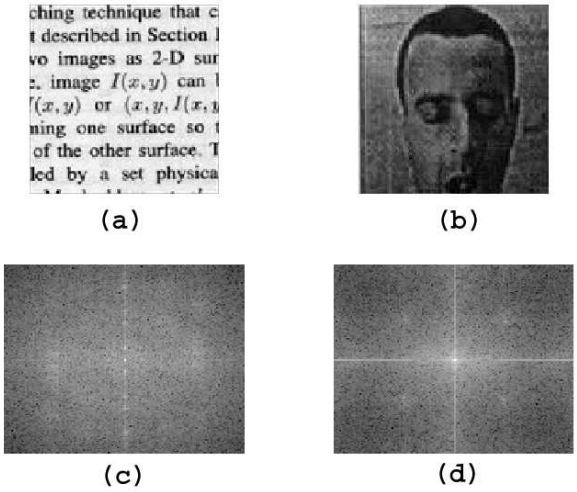


Figure 2.5: The Fourier magnitude spectra (magnitude is scaled logarithmically) for text and non-text images taken from figure 1. (a) Text region (b) non-text region (c) Fourier magnitude spectra of text region. (d) Fourier magnitude spectra of non-text region.

(ii) the shapes of Gabor filters resemble the receptive field profiles of the simple cells in the visual pathway, and (iii) they are direction specific band-pass filters. Gabor introduced the function in 1946 in one-D case which was later extended to 2D case by Daugman (48). A Gabor function is a Gaussian modulated by a complex sinusoid. In the 2D case, we represent it mathematically as,

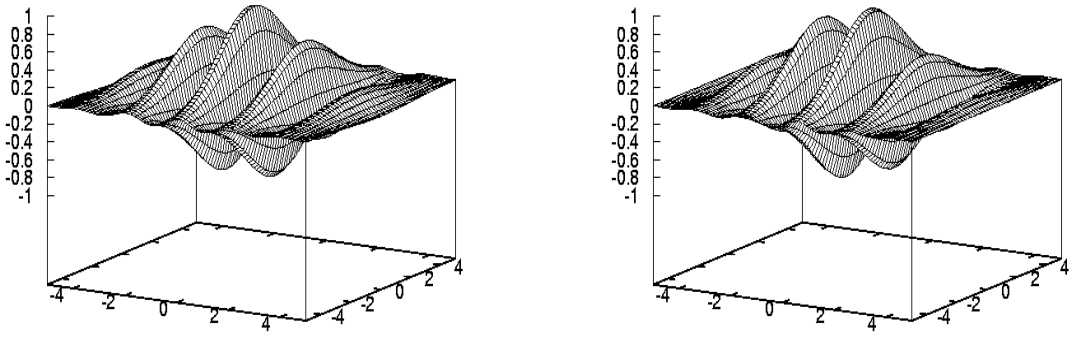
$$h(x,y) \;=\; g(x',y')\; exp[j2\pi\; Ux] \tag{2.7}$$

where x' and y' are the rotated components of the x and y co-ordinates in the rectangular co-ordinate system. U is the radial frequency in cycles/image width.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} \;=\; \begin{bmatrix} cos\theta & sin\theta \\ -sin\theta & cos\theta \end{bmatrix} \begin{pmatrix} x \\ y \end{pmatrix} \tag{2.8}$$

$$g(x,y) = \frac{1}{(2\pi\sigma_x\sigma_y)} exp\left(-\frac{1}{2}\left[\left(\frac{x}{\sigma_x}\right)^2 + \left(\frac{y}{\sigma_y}\right)^2\right]\right) \tag{2.9}$$

σ_x and σ_y account for the spatial spread and the bandwidth of the filter function $h(x,y)$. If B_r is the radial frequency bandwidth in octaves and B_θ is the angular bandwidth in degrees, then,



(a) (b)

Figure 2.6: Gabor Function: U = 0.5 CPI, $\theta = 0^0$ (a) Even Symmetric Gabor Function (b) Odd Symmetric Gabor Function.

$$\begin{aligned} \sigma_x &= \frac{\sqrt{2}}{2\pi U}\frac{2^{B_r}+1}{2^{B_r}-1} \\ \sigma_y &= \frac{\sqrt{2}}{2\pi U \tan\left(\frac{B_\theta}{2}\right)} \end{aligned} \tag{2.10}$$

The power of discrimination of the different filters are dependent on the values of B_r and B_θ. The aspect ratio of $g(x, y)$, defined as

$$\lambda = \frac{\sigma_y}{\sigma_x} \tag{2.11}$$

is the symmetry measure of the filter.

Any combination of B_r, B_θ and U involves two filters, one corresponding to the sine term and the other corresponding to the cosine term in the complex exponential in Eqn. 2.7. The cosine filter (refer figure 2.6(a)), also known as the real part of the filter function, is an even-symmetric filter and acts more like a low pass filter, while the sine part being odd-symmetric acts like a high pass filter (refer Fig. 2.6(b)). Gabor filters of $B_r = 1$ octave and $B_\theta = 45^o$ at four different orientations ($\theta = 0^o,\ 45^o,\ 90^o,\ 135^o$) have been used for the reported work. Three different radial frequency values ($U =\ 0.2,\ 0.3\ and\ 0.5$) have been chosen since they have been observed to be working well for all kinds of documents

mentioned in section 2.3. Thus, the combination of the three radial frequencies and four different angles, generate 12 cosine and 12 sine filters. These filters are pictorially presented in figures 2.7 (b) and (c), respectively. A block schematic of the proposed filterbank, using the above 24 filters, is shown in figure 2.7(a).

2.4.4 Block Energy Analysis (BEA)

A *space-frequency* filter bank, using Gabor filters, has been designed and implemented for separating the text and non-text areas in a gray level document image. Spatial convolution accomplishes the filtering of image, I by the filter h. The output, $\hat{I}$, is generated by this filtering.

$$\hat{I}(x,y) \;=\; I(x,y) \;*\; h(x,y) \tag{2.12}$$

The equivalent energy image, e, is evaluated by squaring of magnitude of each pixel value in the output image. We denote symbols e and o for the energy images equivalent to the cosine and sine filtered outputs, respectively.

$$e(x,y) \;=\; \hat{I}(x,y)^{\,2} \tag{2.13}$$

The energy response of a complex filter (refer Eqn. 2.7) at pixel location (x, y) is:

$$E_{(u_l,\ \theta_k)}(x,y) \;=\; \sqrt{e^2_{(u_l,\theta_k)}(x,y) + o^2_{(u_l,\theta_k)}(x,y)} \tag{2.14}$$

where $e_{(u_l,\theta_k)}(x,y)$ and $o_{(u_l,\theta_k)}(x,y)$ are the energy outputs of the cosine (even) and sine (odd) filters, with radial frequency u_l and angle θ_k, respectively, at pixel location (x, y). The total local energy is the sum of the outputs of all the complex filters as shown in figure 2.7.

$$E_T(x,y) \;=\; \sum_{l=1}^{3}\sum_{k=1}^{4} E_{(u_l,\ \theta_k)}(x,y) \tag{2.15}$$

We low pass filter the magnitude response image of the Gabor filter bank (the E_T image) using a 11×11 Gaussian mask ($\sigma_x = \sigma_y = 3$). It is seen that this removes artifacts and irregularities present and, thereby, helps in text detection. We evaluate

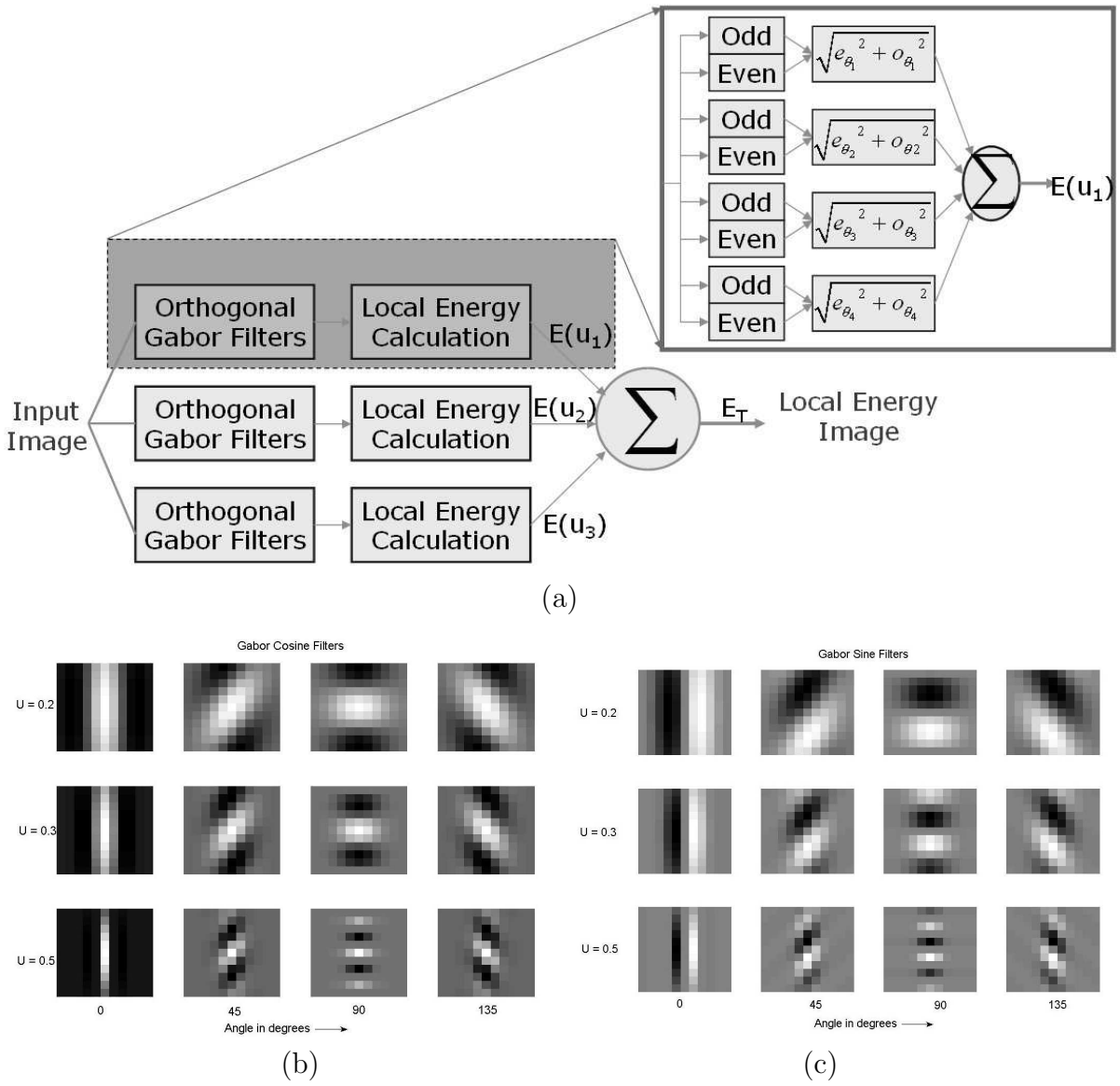


Figure 2.7: (a) Block schematic representation of the Multi-channel Gabor filter bank used for text/non-text separation. Pictorial view of all the Gabor cosine filters (b) and sine filters (c) in use for this algorithm. Here, the rows of images correspond to different radial frequencies, while the columns correspond to different orientations in degrees.

block energy at each pixel by considering blocks of 15×15 size. An average of the block energies is evaluated across all the blocks in the image. Twice this average energy is heuristically set as the threshold for block wise text and non-text separation: a block is considered to be text if the block energy is above the threshold.

2.4.5 Harris' Corner Points

Checking every pixel in an image for information is computationally heavy. Moreover, image information is better presented by a group of neighboring pixels rather than individual pixels. Depending on their location, some pixels contain richer information than their neighbors. In such a situation, it is appropriate to tap these pixels. **Corners** are referred to as interest points as they are rich in edge information. They are located using an interest point detector.

Harris and Stephens (49) proposed a corner detection operator in 1988. This has been widely used in applications, where corner points are of importance. The following is their algorithm for detecting corner points.

1. For each pixel $I(x, y)$, obtain the autocorrelation matrix M as follows:

$$M(x,y) \;=\; G_\sigma \;\otimes\; \begin{bmatrix} \left(\frac{\partial I}{\partial x}\right)^2 & \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) \\ \left(\frac{\partial I}{\partial x}\right)\left(\frac{\partial I}{\partial y}\right) & \left(\frac{\partial I}{\partial y}\right)^2 \end{bmatrix} \tag{2.16}$$

   G is a Gaussian blurring function with variance σ.

2. Construct the cornerness map by calculating the cornerness measure $C(x, y)$ for each pixel:

$$C(x,y) \;=\; det(M) - k \times trace(M) \tag{2.17}$$

   Here k is a constant.

3. Select a threshold T and set all values in $C(x, y)$ below this threshold to zero. We chose T to be 10% of the maximum value in C.

4. Perform non-maximal suppression to find the local maxima.

5. Declare the non-zero points in $C(x, y)$ as the corner points.

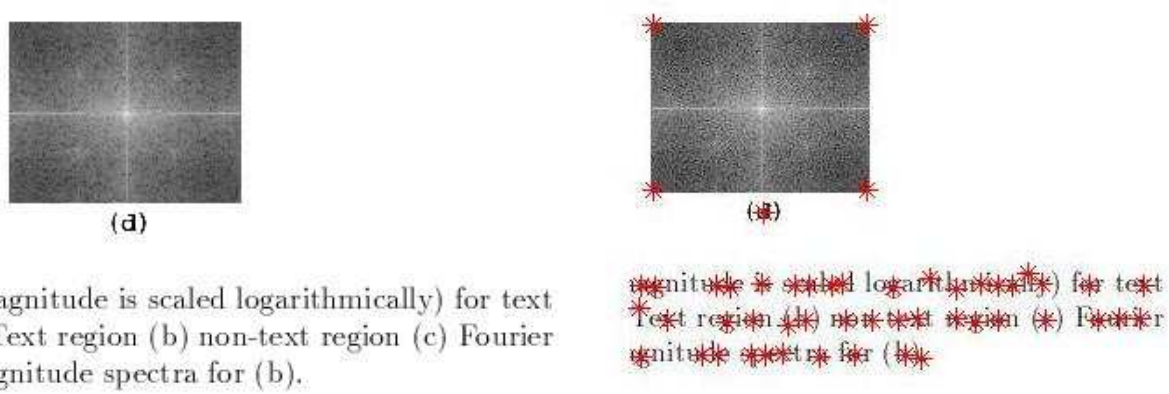


Figure 2.8: Harris' corner detection. (a) A sample image with both text and non-text. (b) The detected corner points are denoted by a star (*).

The operation to search for text areas is performed only on the corner points. This reduces the computational time. Figure 2.8(b) demonstrates the selection of the corner points from the image (shown in Fig. 2.8(a)) containing both text and non-text.

2.4.6 Delaunay Tessellation

Delaunay Triangles is a construction of computational geometry over a set of vertex points in a two dimensional space. It is a powerful way to represent the objects in a space with respect to their neighbors in a very convenient and robust way.

In a 2-D space, given any three non-collinear points, a triangle could be drawn. When a number of such points are given, however, a triangle needs to meet some constraints to be a Delaunay triangle. One important constraint is that the outer circle of a triplet should not contain any other data point inside it. This is clearly shown in figure 2.9, where $T1$ is a Delaunay triangle while $T2$ is not. This is because circumscribing circle $C2$, of triangle $T2$, contains the point p_j inside its circumference. Xiao and Yan (50) have successfully used Delaunay tessellation to extract the text regions in a document image. Here, they describe the various properties of Delaunay triangles and the constraints under which they could be employed for separating text areas in a document image. Figure 2.10 shows the formation of Delaunay triangles for the text regions of a document image.

Each document image is binarized and connected component analysis yields the connected blocks. The geometric centers of each of these components are considered as the points for Delaunay triangulation. A statistical analysis of the shapes and sizes of all the triangles is performed for the document. If all the triangles are of similar size, then the whole image is considered to be consisting of text elements. However, if some triangles are

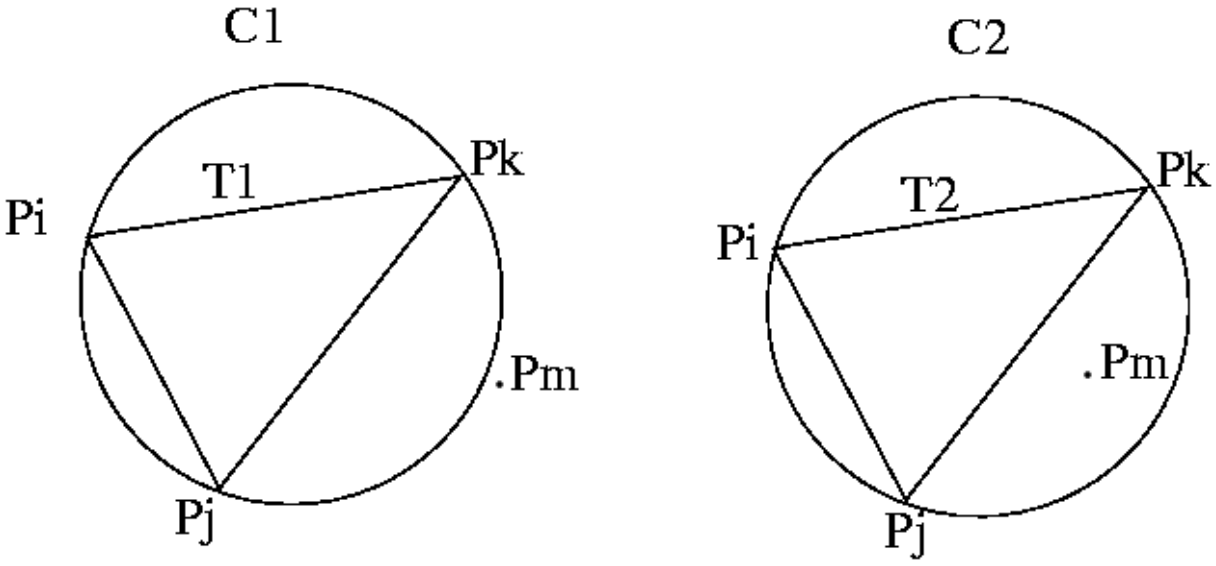


Figure 2.9: Definition of Delaunay triangulation between 3 points pi, pj, pk. Here, T1 is a Delaunay triangle. T2 is not one because pj lies inside the enclosing circle C2, which violates the Delaunay triangulation criteria.

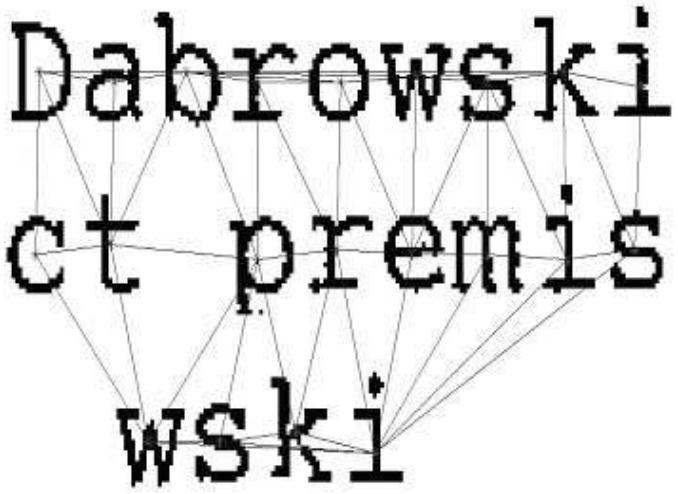


Figure 2.10: Delaunay triangulation on a text block.

larger than the others, based on a statistical measure, the smaller triangles are considered to consist of points derived from the text elements. If most of the triangles associated with a point satisfy the text criteria, the point is considered to be a text point.

2.5 Text Extraction from Complex Gray Images

The collected database contains document images with varieties of clutter. It contains text in both Manhattan and complex layout. As has been described in section 2.3, the scanned images are stored as color images. First these images are converted to gray scale images by replacing each pixel by the average of its R, G and B components.

In the first step of the algorithm, we statistically analyze the shapes and sizes of the connected components present. The input to this stage is an image obtained by binarizing the gray image, by the technique suggested by Pati (51). In this technique, the contrast of the image is first enhanced using a non-linear stretch. A global threshold is then determined based on which the image is converted from gray to binary. CCA, as described in section 2.4.2, is applied on the binary image to detect the non-text elements and replace them with background pixel value.

Most of the non-text elements get removed from the document image based on their shape and size information. However, this mechanism fails to remove the CC's of sizes comparable to characters. It also fails to remove portions of the surrounding areas in natural images. This is amply demonstrated in figure 2.11(a). This figure shows the output for the image shown in figure 2.1(a), after the pixels detected as non-text elements are changed to background. Even otherwise, some portions of the already removed CC's remain. This is because those portions of the CC's were forced to be background elements during binarization of the gray image. To remove these elements from the document, a multi-channel filterbank is employed. Here it is assumed that the text-areas are richer in high frequency components due to the rapid change in the gray values in regular intervals (refer figure 2.5). A detailed description of the multi-channel filterbank is provided in section 2.4.3 and a block schematic representation of the same is provided in figure 2.7. This generates the energy image as shown in figure 2.11(b) from the input gray document image.

Block energy analysis is performed on the energy image. The Gabor filters generate a high energy point even at spurious and noisy points and edges. The block energy analysis, described in section 2.4.4, removes such isolated energy rich points by smearing the energy image. A statistical analysis of the block energy analyzed image helps determine the text areas. Figure 2.12 shows a schematic representation of the proposed scheme (52).

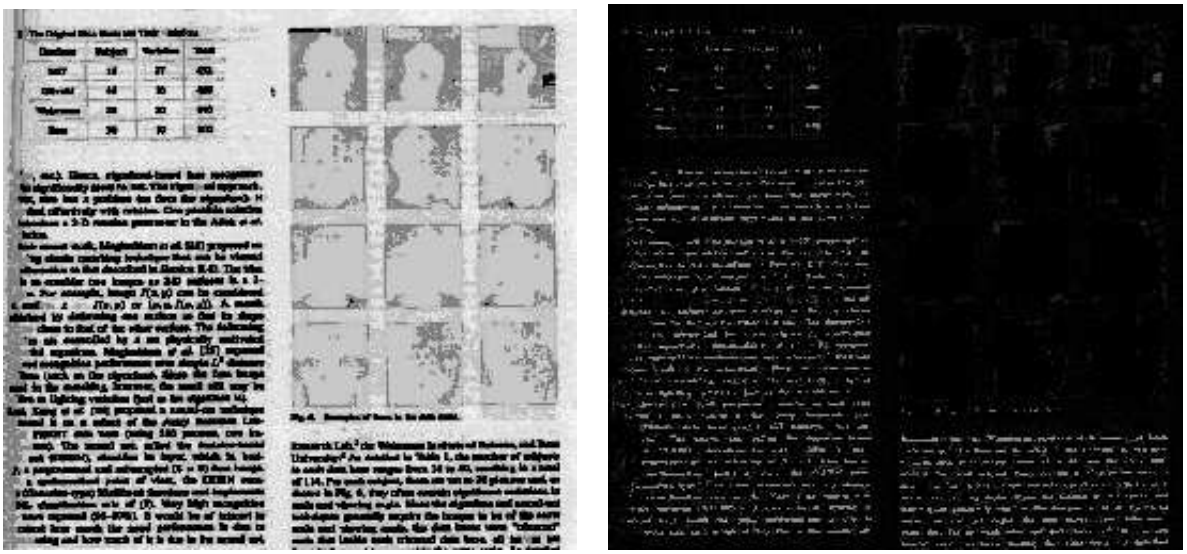


Figure 2.11: Example intermediate outputs in text segmentation. (a) the output of the CCA stage of the proposed algorithm. (b) The energy image (output of the Gabor Filterbank).

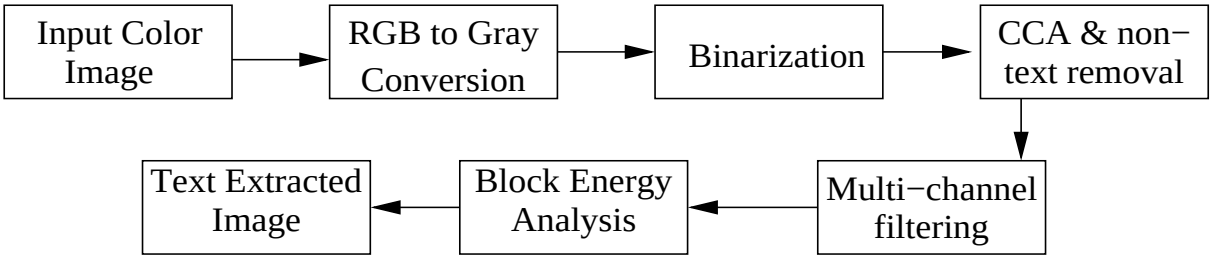


Figure 2.12: Block Schematic of the proposed System for Text Segmentation.

Thus, the proposed scheme can be summarized as an algorithm as follows:

1. Remove isolated noise from the image by anisotropic diffusion (refer Sec. 2.4.1).
2. Binarize the image after contrast enhancement.
3. Remove the line based clutter with CCA (See Sec. 2.4.2).
4. Generate the E_T image from the output of the Gabor filter bank (See Sec. 2.4.3 & 2.4.4).
5. Separate the text areas based on BEA (refer section 2.4.4).

2.6 Analysis of Computational Complexity

Timing analysis is an important aspect of any algorithm development. A theoretically elegant and/or technically sound algorithm is seldom of any use if it fails to generate results within the specified time. It is important to have algorithms which work under generic conditions and are computationally efficient. A timing analysis helps the users to decide about the amount of time and other resources an algorithm needs. When this timing is specified in terms of the number of computations, it becomes machine independent. In this section, an effort is made to specify the timing requirements of the proposed algorithm.

Some aspects of the algorithm are one-time job (*static computation*) while the computation involved with others is dependent on image size (*dynamic computation*). Here, we make an effort to consider only the dynamic time requirements of the algorithm. The various modules of the algorithm are: (i) Binarization, (ii) CCA, (iii) generation of the filter masks, (iv) filtering and energy image generation, and (v) block energy analysis and text extraction. It is assumed that the input image, $I(i, j)$, has M rows and N columns (MN pixels).

Binarization

The contrast stretching prior to the binarization is based on the histogram of the image. This requires MN comparisons for an image of $M \times N$. Contrast enhancement is not actually performed. The value of threshold for binarization in the contrast stretched domain is deduced from the histogram of the image. An inverse logic determines the threshold in the original image domain. Then, another MN comparisons are needed for converting the gray image to binary.

CCA

CCA needs $O(MN)$ logical comparisons.

Filter Generation

We have chosen the filter sizes to be 13×13 and there are 24 such filters. The filter coefficients are pre-computed, stored and read from memory during the filtering operations. Thus, no arithmetic computations are needed for the filter generation during actual processing of the image.

Filtering

Filtering of the image in space domain involves convolution of each of the filter masks with the input image. This requires 169 (mask size is 13×13) multiplications and 168 additions, a total 337 arithmetic operations at each pixel location. 24 such filters (12 odd and 12 even filters), thus, require 8088 arithmetic operations per pixel. Moreover, generation of the energy image needs sum of the individual filtered images (mentioned in equations 2.12 – 2.15). This adds another 12 sets of four arithmetic operations, namely, two multiplications, an addition and a root evaluation. It sums up to 48 arithmetic operations at each pixel, making a total of 8136 operations per pixel. Thus, the number of computations needed for spacial domain filtering is $8136*MN$ arithmetic operations, besides, some conditional operations to take care of the boundaries.

A convolution in the space domain implies a multiplication in the Fourier transformed domain. FFT algorithms are available which perform the Fourier transform of the input image, $I(i, j)$ of size $M \times N$, in $O(MN)$ time complexity. An algorithm downloaded from the National Research Council of Canada (53) is employed to Fourier transform the image and filters. Filtering by this technique would involve the following computations:

- Forward transform of the input image involves $O(MN)$ computations.
- Creating filter masks of size same as that of input image. This is performed by centering the mask on a zero matrix of size $M \times N$. However, the computational requirement of this part is negligible.
- Forward transform of the 24 filters: $24*O(MN)$.
- Complex multiplication in the transform domain involves $96*MN$ multiplications and $48*MN$ additions: a total of $144*MN$ arithmetic operations.
- Addition of the 24 filtered images in the transformed domain involves $24*MN$ operations.
- Inverse transform of the filtered image to space domain involves $O(MN)$ operations.

The total operations involved in filtering by the Fourier transform method is: $26*O(MN)$ along with another $168*MN$ arithmetic operations. When $O(MN)$ is small, this method is more time efficient than the spatial filtering technique.

BEA

For BEA, we use blocks of 11×11 centered on the pixel under consideration. The average energy of all the pixels under this window is evaluated and assigned as the energy of the center pixel. Thus, for each pixel, we perform 120 additions and one division. So a total of 121*MN operations are involved.

Once the block-averaged-energy image is obtained, an average of the energy values at each pixel location is evaluated. This needs MN operations. Based on the average value, each pixel is checked to decide if it contributes to a text element. This requires MN conditional operations. Thus, in this module a total of 122*MN arithmetic and MN logical operations are involved.

2.7 Enhancement of Computational Efficiency

Based on the analysis presented in the previous section, we observe that the total computation requirement of the algorithm is 3*MN logical operations, 290*MN arithmetic operations with another 27*$O(MN)$ arithmetic operations. It is clear that most of the computation is for the filtering operation. This is due to a brute force technique of energy evaluation at every pixel.

A substantial part of the filtering computation could be saved, if the energy is evaluated only on a selected few points. We employ the Harris' corner detector for selection of these information rich points. Then energy is evaluated at these points only. When the algorithm is modified thus, the block energy principle is no longer valid. So, the coefficients of the 24 filters are arranged in a vector form and used for information representation at that point. Such a vector is a feature vector for the point. A set of pre-inspected feature vectors are kept as a training set and each arriving point on the document is classified as text or non-text based on this set. A nearest neighbor classifier is used to make this decision. Based on this decision, the non-text elements are removed from the document. The text image, thus generated, is binarized.

To enhance the yield of the algorithm, a reinforce mechanism is employed. The output of the previously mentioned classification is analyzed to generate the connected components. Each CC is represented by its geometric center. Thus, all the connected components in the image, containing mostly text but some non-text elements, are represented by a set of points. Delaunay triangles are formed using these points and a triangulation based text/non-text separation algorithm (50) throws away the remaining non-text elements. This process not only reduces the computation of the algorithm but also enhances the efficacy of the text extraction process. Figure 2.13 shows a block schematic of the

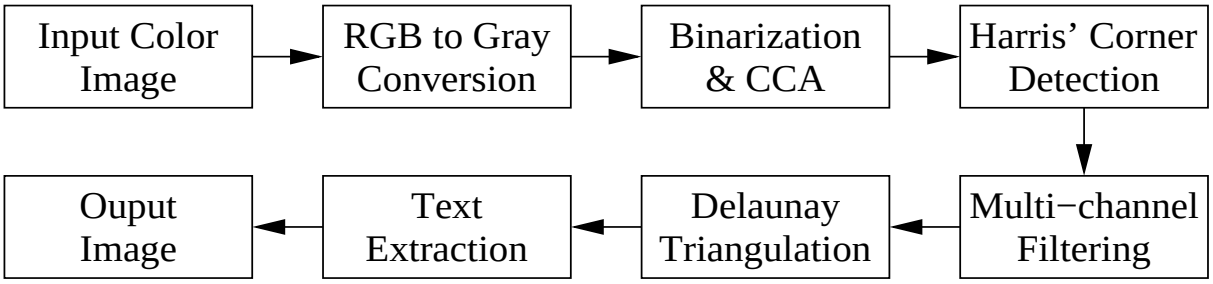


Figure 2.13: Block schematic representation of the time optimized algorithm for text extraction from complex gray images.

modified process.

2.8 Experimental Results

The proposed segmentation scheme is applied on the document images. The results of the time optimized version of the algorithm are also presented for a comparison. Henceforth, the term "proposed algorithm" would mean the version of the algorithm presented in section 2.5 and the term "time-optimized algorithm" would refer to the one described in section 2.7.

Figure 2.15 illustrates the various levels of output for the sample input image, with a Manhattan Layout, shown in figure 2.15(a). Figure 2.15(b) illustrates the result after connected component analysis on the binarized image of Fig. 2.15(a). It can be noted that the objects identified as non-text are homogeneously set to an intensity, almost similar to the background (refer section 2.4.2). Figure 2.15(c) shows the E_T image, the image comprising of pixel values corresponding to the E_T at that pixel location, while Fig. 2.15(d) represents the E_T image after a low pass filtering with a Gaussian mask. It can be noted that the E_T values in the edge locations in the text areas are significantly higher than these at other pixel locations. It may also be noted that while in the figure 2.15(c) image, both the text and non-text areas have similar energy values, the contrast between the text and non-text regions has grown significantly in the Gaussian smoothed image, Fig. 2.15(d). The final result, after BEA, is shown in Fig. 2.15(e) and the result of the time-optimized algorithm is presented in Fig. 2.15(f). Note that though there are slight differences visible between the outputs of the two algorithms the textual information content in both, are the same.

Figure 2.16 compares the results obtained by the two text separation schemes on a Kannada-English bi-lingual document page. Figure 2.16(a) shows the input newspaper document. The result of anisotropic diffusion filtering is shown in figure 2.16(b).

Fig. 2.16(c) presents the magnitude image of the Gabor filter bank outputs while Fig. 2.16(d) presents the results after the BEA. The latter is also the scheme proposed by Chan and Coghill (14). The final result of the proposed algorithm is presented in Fig. 2.16(e) while that of the time-optimized algorithm is shown in Fig. 2.16(f). It can be observed that while the result presented in Fig. 2.16(e) is better than the ones shown in either Fig. 2.16(c) or Fig. 2.16(d), the result of the time-optimized algorithm, figure 2.16(f), is also as good. We observe that a scheme without the CCA has not been able to produce the desired effect. This is because certain regions in natural images too contain frequency components in the same band as text regions. So, patches of natural images are detected as text regions. This has also proven the script independence of our proposed algorithm.

Figure 2.17 demonstrates the effect of anisotropic diffusion based filtering on noisy images. Figure 2.17(a) shows the original input image scanned from a newspaper. Newspaper printing is low-resolution printing. Hence, even the non-text portions of the document image are rich in high frequency components. So, the Gabor filterbank based mechanism fails here if the input image is fed directly to the algorithm. However, the diffusion filtering helps here. The results of the algorithm for the original input image is presented in Fig. 2.17(c) and the result for the filtered input image is shown in Fig. 2.17(b). It can be noted that the image in figure 2.17(c) has characters that are smeared, broken and even missing, in addition to non-text portions of the image having been detected as characters.

The ability of our scheme to detect even handwritten text areas in images of cards has been demonstrated with figure 2.18. In this figure, Fig. 2.18(a) shows the input image while Fig. 2.18(b) illustrates the hand-written text region, detected well by our proposed scheme. Fig. 2.18(c) presents the result of the time-optimized algorithm.

Figure 2.19(a) shows another input to the proposed scheme while Fig. 2.19(b) presents the output of our system and Fig. 2.19(c) presents the result of the time-optimized algorithm. We can see from Fig. 2.19(b) that the text present on the surface of the airplane image has been detected. This demonstrates that the proposed scheme is not only efficient in separating text and non-text regions but also capable of detecting regions containing textual information in natural images. However, this subtle detection of the text on the air plane surface is missed by the time-optimized algorithm since it is located far away from the other text regions. Besides, only 3 characters are present, in an isolated fashion. Thus, the Delaunay triangulation rejects these elements from being text.

The observation that the proposed scheme is capable of detecting text elements present in natural and scenic images, propels us to test the algorithm on camera captured image. Figure 2.20(a) shows a document image captured with the Minolta digital camera (mentioned in section 2.3). Figure 2.20(b) shows the E_T image for Fig. 2.20(a) while the

output for the same is presented in Fig. 2.20(c). Please note that the proposed scheme works fairly well for camera captured document images also and is independent of the skew of the document image. The results presented in this image not only prove that the proposed algorithm is capable of handling both scanned and camera captured images, but also that it is efficient in dealing with both Manhattan and non-Manhattan layouts.

2.9 Conclusion and Discussion

In the reported work, we have presented a scheme for detection and separation of text elements in a document image. The document image could be obtained either by a flatbed scanner or with a hand held device like camera. The scheme has been illustrated to be working well for different kinds of documents, irrespective of script, kind of clutter or mode of acquisition. This model mostly relies on separating the text and non-text elements based on the spatial frequencies of the locality, efficiently treated by a Gabor filter bank.

The line based clutters have analytically been found to be equally rich in high frequency components. The CCA algorithm works on the geometry of the elements of the image and has been shown to compliment the filterbank approach, enhancing the performance of the system. It may be noted from Fig. 2.18 that though handwritten lines are similar to line drawings, they have been classified as text elements. Anisotropic diffusion reduces additive noise efficiently. This is illustrated in Fig. 2.16(b). Generally, additive noise is found in old newspapers and books. Thus, the choice of anisotropic diffusion has paid off well.

Comparing the scheme with the work reported by Chan and Coghill (14), the BEA is seen to be enhancing the performance of the proposed system. This is mostly because, while our scheme takes into account the relationship of a given pixel with its neighboring pixels, such a concept is missing in previously reported Gabor filter based works. However, our experience tells that some post processing with morphological operators, if properly designed and chosen, would enhance the performance of the layout analysis system reported by Chan and Coghill.

A timing analysis reveals that the brute force evaluation of the energy at each pixel is demanding in terms of computation. Thus, a revised technique of evaluating energy at selected few points is proposed. These points are the information rich corner points and are easily detected with Harris' corner operator. A reinforce mechanism, Delaunay triangulation on the so detected text areas, helps to remove the leftover non-text elements. Thus, the algorithm is rendered time-efficient, while delivering quality output.

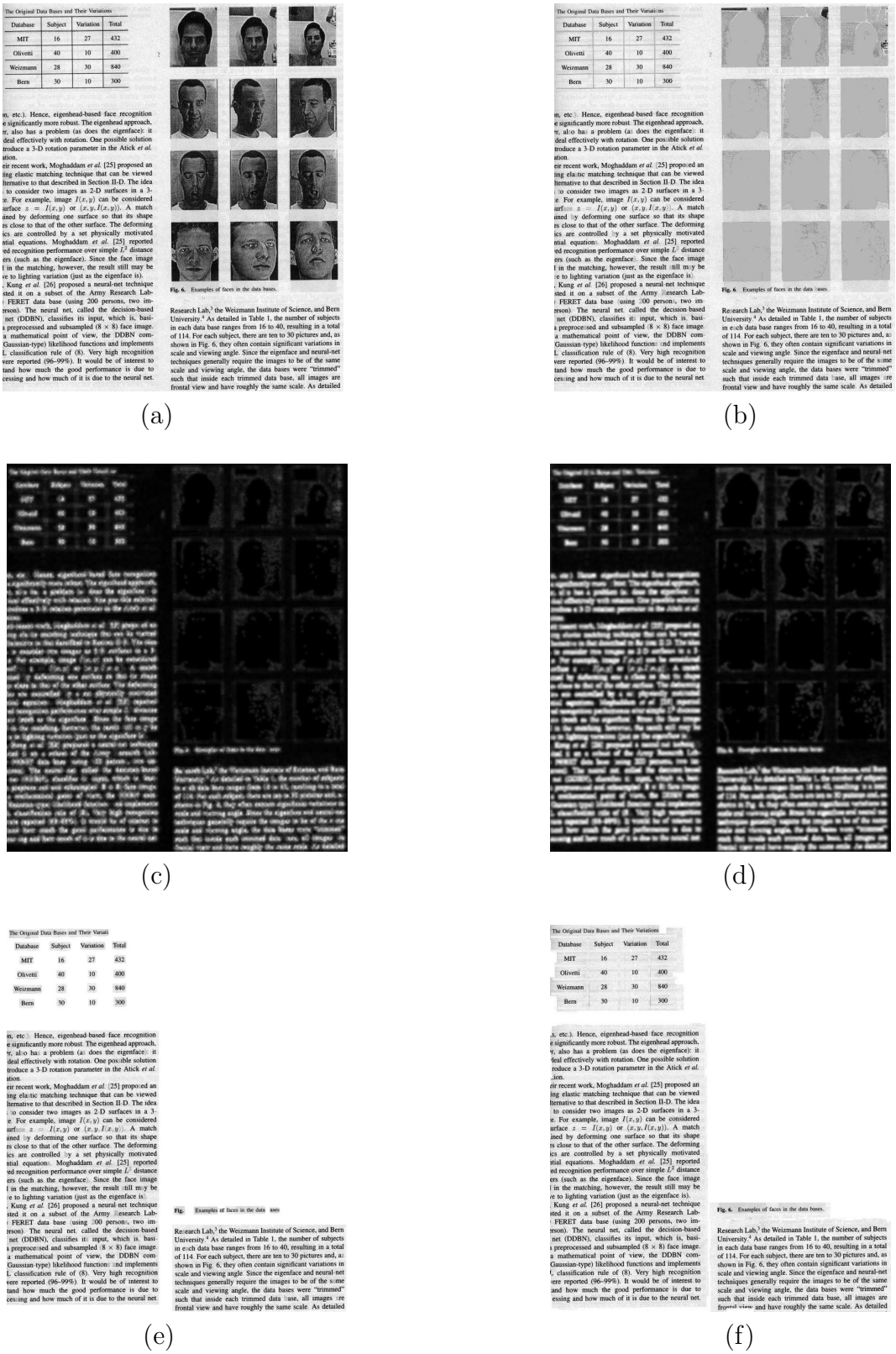
(a) (b)

Figure 2.14: (a) Sample Hindi input image. (b) Output of the proposed algorithm.

Chaudhuri *et al.* (25) have proposed a scheme for separating text areas in documents mainly consisting of north Indian scripts (*e.g.* Devanagari), taking into account the connectivity of characters in words. They have employed a scheme based on the horizontal projection profile of the image, where they look for the line connecting the characters, named as *shiro-rekha*. However, our scheme does not use structural features and, hence, is independent of the script characteristics. This is shown in figure 2.14(b) which is output of the proposed algorithm for Fig. 2.14(a). It may be observed here that the involved script is Devanagari.

The major advantages of our algorithm are:

1. handles multi-script documents,
2. invariant to skew,
3. works fairly well on camera captured digital document images too, and
4. accommodates complex layout, involving different kinds of clutter.

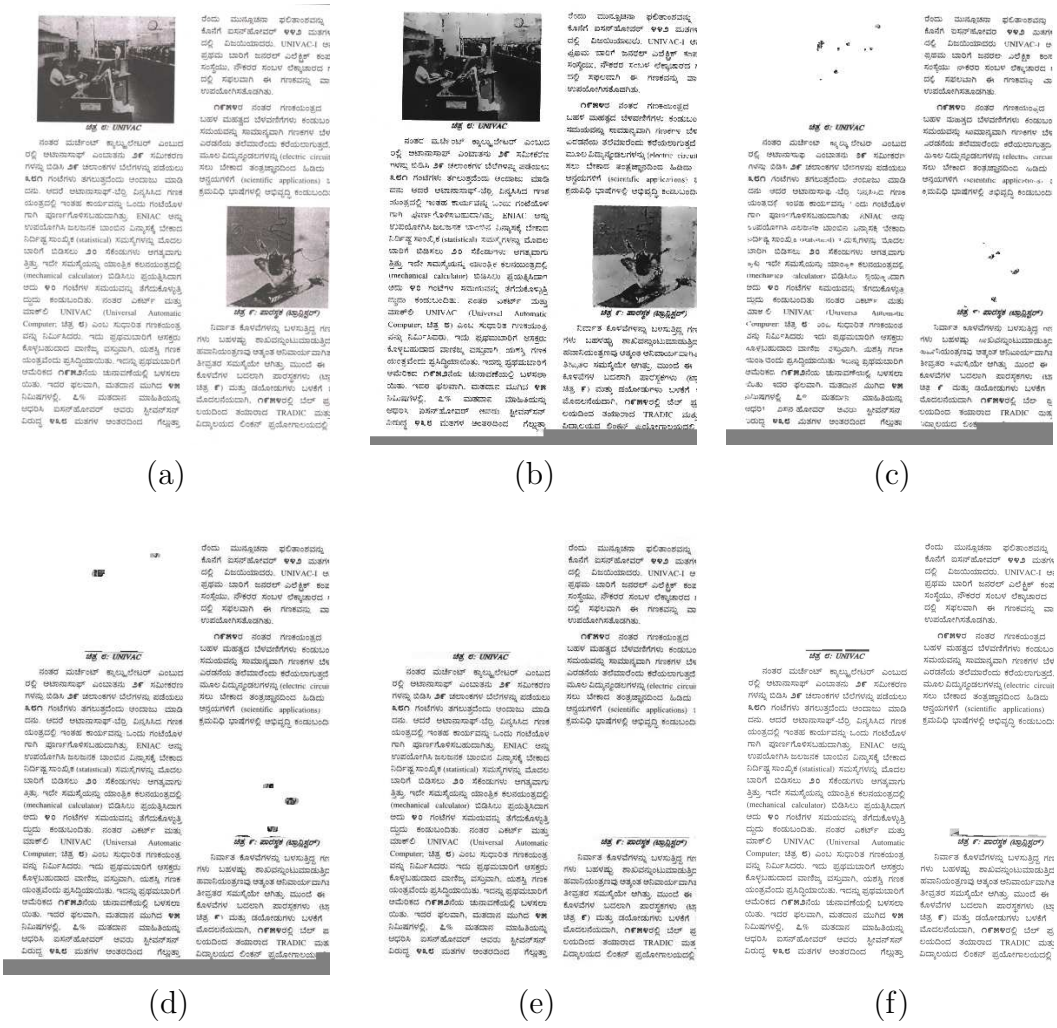


(a) (b)

(c) (d)

(e) (f)

Figure 2.15: Experimental results for (a) an ordinary document (1100×1000) with text/image/line-drawing, (b) objects identified as non-text region are homogeneous set, (c) magnitude response(local energy) of multichannel filter, (d) Gaussian smoothed output of the Gabor filter bank (block averaged energy image), (e) segmented text using block energy analysis and thresholding, and (f) output of the time optimized algorithm.



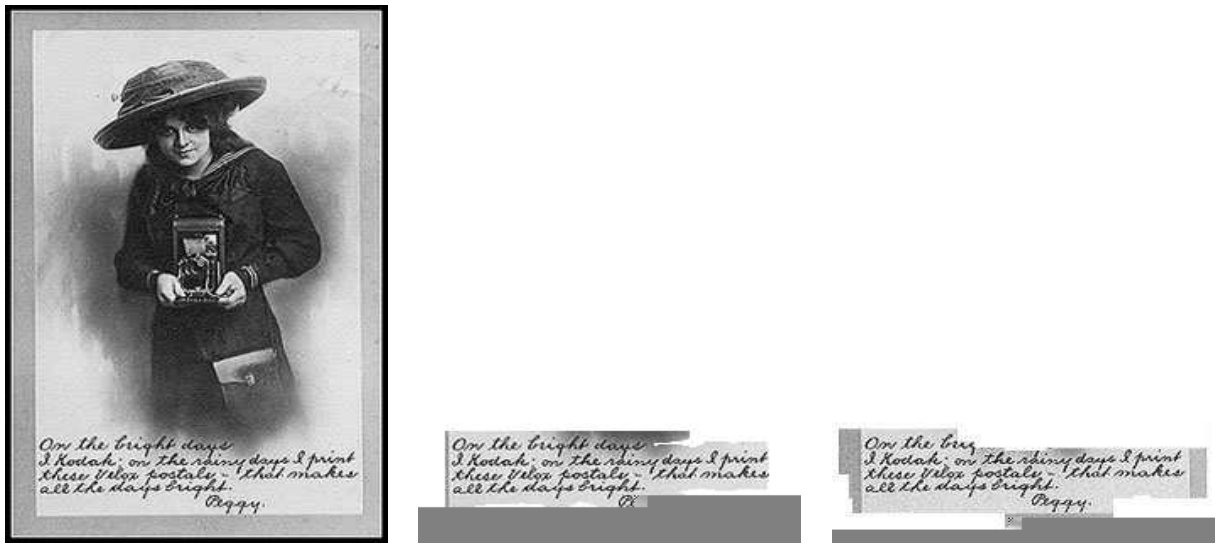
(a) (b) (c)

(d) (e) (f)

Figure 2.16: (a) Original bilingual document image, (b) Anisotropic diffusion filtered output, (c) Result of Gabor filter with local energy, (d) Result of Gabor filter + BEA, (e) result with proposed system (not time-optimized), and (f) result of the proposed time-optimized system.

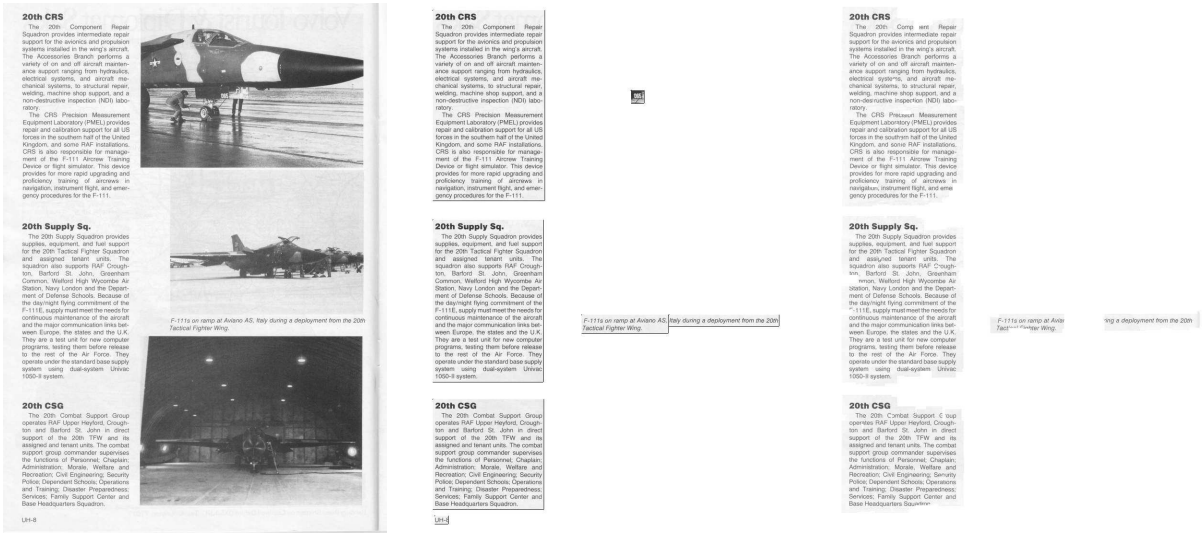
(a) (b) (c)

Figure 2.17: The results of the proposed algorithm with and without anisotropic diffusion filtering (ADF). (a) Original input image, (b) result of the proposed algorithm with ADF, and (c) result without ADF.



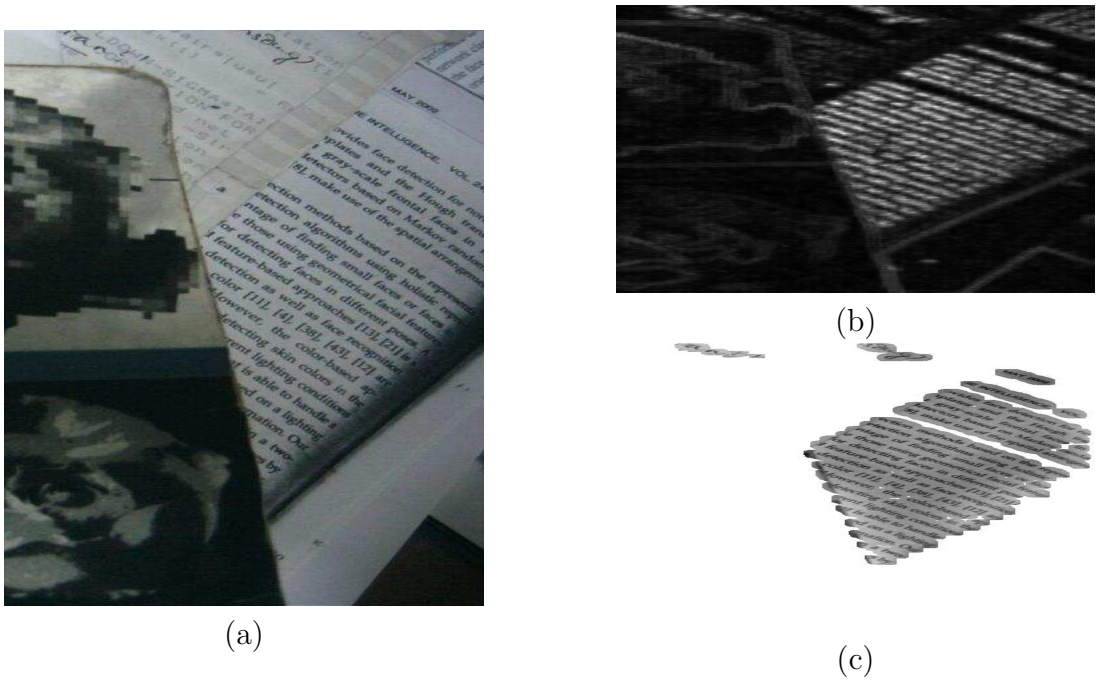
(a) (b) (b)

Figure 2.18: (a) Natural Image (250×250) with Hand-written text, and (b) extracted text region using the proposed algorithm using BEA, and (c) the output of the proposed time-optimized algorithm.



(a) (b) (c)

Figure 2.19: Experiment on Newspaper Document (800×1100) (a) Input Image with poor text regions (b) result of text segmentation with original proposed algorithm, and (c) result of the text extraction with time-optimized algorithm.



(a) (b) (c)

Figure 2.20: Experiment on camera capture image (380×836) (a) Image captured by Digital Camera at 1m height (b) Gabor response invariant to skew (c) resultant segmented text.

Chapter 3

Text Extraction from Complex Color Images

"Experimental confirmation of a prediction is merely a measurement. An experiment disproving a prediction is a discovery." – Enrico Fermi, Italian physicist, 1901-1954.

Summary:

Color document images have lot more information than their gray counterparts. Embedded text mostly forms a iso-color cluster and exploitation of this property yields better result. Here, we propose to exploit the color information alongside the intensity information for better text localization.

Availability of mobile and hand-held imaging devices, such as, cell phones, PDA's, still and video cameras have resulted in new applications, where the text present in the acquired images is extracted and interpreted for various purposes. In this chapter, we present a modification to the algorithm presented in the previous chapter for automated detection of text in color images. Proposed system involves Gabor function based multi-channel filtering on the intensity component of the image along with Graph-Theoretical clustering applied on the color space of the same image, thereby utilizing the advantages of both texture analysis and connected component for text detection. Our approach performs well on images with complex background.

3.1 Introduction

The previous chapter describes an algorithm for extraction of text from the intensity component of the document images or gray document images. This algorithm is observed to work better for documents that predominantly contain text. Examples of such documents are shown in figure 2.1. It may be recalled that the database, described in section 2.3, contains color images. These images are converted to their gray counterpart before processing by the technique described in the previous chapter. This was a choice made with processing of document images in mind.

However, color provides vital information about the nature of text elements which otherwise are missing from the gray image. This is amply demonstrated in figure 3.1. In Fig. 3.1(a), the text and the background are represented with different colors. This aids in easy localization of text. Fig. 3.1(b) shows the intensity equivalent of the image in Fig. 3.1(a). It can be observed here that the difference between pixel intensity values of the text body and the background is hard to notice. The bounding boxes are the only indicators by which it is possible to localize and read the text. Besides, in many situations, use of only the textural information fails to generate the desired output. This is amply demonstrated in figures 3.2 and 3.3. In these figures, a lot of false positives are detected leading to generation of unwanted information. Thus, exploitation of the color information along with the intensity, promises a better output for text localization.

With wide availability of digital cameras and handy-cams, acquisition of images is a easy and affordable game for anyone. Many of such acquired images contain textual information, which can be extracted and used for various purposes. Besides, these devices enable capturing textual information from hard-to-scan situations such as stone depictions. Moreover, such cameras are small in size, require less power, are easy to handle and shoot and most importantly do not need a computer to be carried for acquiring text images. Thus, it is possible to extract textual information from most situations. The challenge, however, is to extract text regions and bring them to a form such that OCR could be applied on them.

In scenic images, text could be naturally present (the text is photographed) or embedded into the image at a later time. The text in images and videos provide useful information for automated annotation and indexing. Such text eventually helps in mining the relevant images from the database. A scheme for efficient extraction of textual information from images is needed for this task. Images may have text of various styles and sizes in simple or complex background. Document analysis softwares, generally, assume high scanning resolution, high quality document images with fairly simple layout structure. These assumptions do not hold for camera captured scenic or document images.

(a) (b)

Figure 3.1: (a) A color image and its corresponding intensity image (b). This image depicts the importance of color information for text localization.

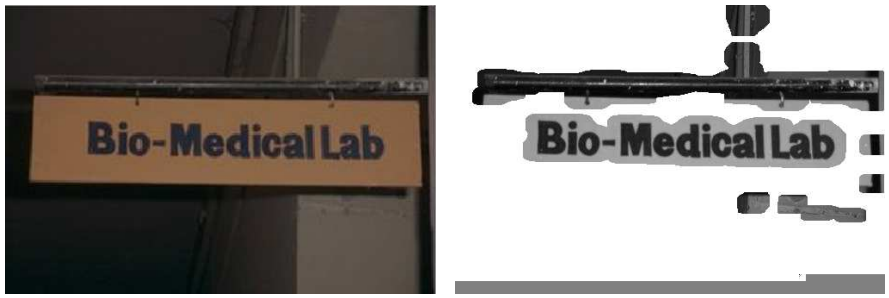
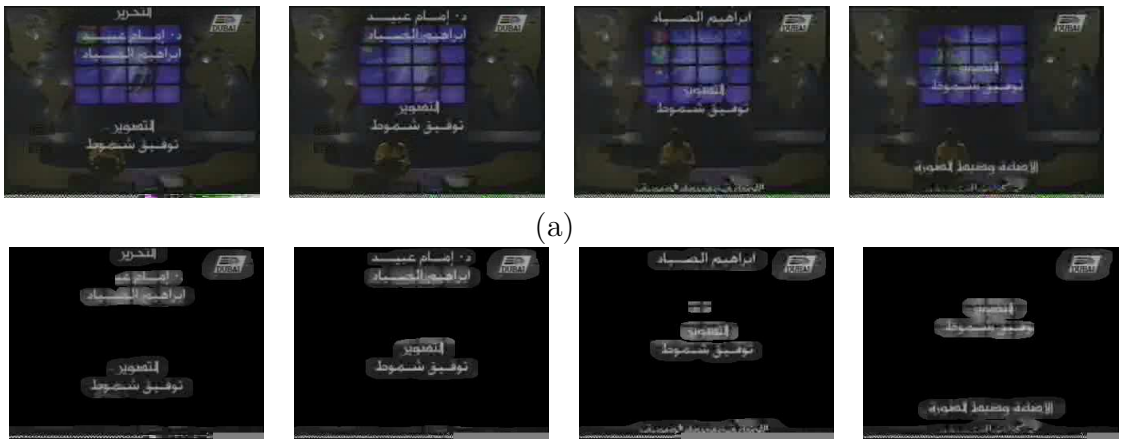


Figure 3.2: Text localization using Gabor filter showing false detected text.



(a)

(b)

Figure 3.3: Results of text localization on a video sequence with a dynamic background: (a) input sequence and (b) regions detected as text.

3.2 System Description

A color image has each of its pixels represented with the three primary color components, red (R), green (G) and blue (B), namely the RGB values. This could also be represented with any other equivalent system convertible to the RGB system. For an image represented with 24-bit color format, each of the red, green and blue components have values ranging from 0 to 255. Thus, there are a total of 16,581,375 different combinations, using one a pixel could be represented. An object is represented with a number of neighboring and connected pixels. Thus, in a real situation, the pixels forming an object have very little variation in terms of their color combination. So, it is safe to assume that an object forms a compact cluster in the color space. We name each cluster as an iso-color component.

A color image could be represented with its two different components: (i) intensity component (representing the texture of the objects in the image) and (ii) color component. Each of these components contains a lot of information about the image. In the previous chapter, we have exploited the intensity of the image as a texture and have attempted to separate text-blocks from the rest. In this chapter, we intend to maximize the efficiency of text extraction from images by fusing information contained in the intensity and the color components of the image. A block schematic diagram of the proposed system is presented in figure 3.4(a). This system has two parallel channels: (i) gray channel processing the texture information from the intensity component of the image, and (ii) color channel processing the structural information of the objects in the image. The gray channel used here for textural information processing is same as the system reported in the previous chapter. Figure 3.4(b) presents this system in a block schematic style and section 2.5 presents a detailed discussion on the operation of this system in general as well as the functionalities of each of the constituting modules. Hence, any discussion on this system is avoided here. The color channel is presented in figure 3.4(c). This system along with the information fusion part of the proposed system is described in the following subsections.

3.2.1 Color Component Generation and Analysis

The first step in segmenting a color image is to define components with homogeneous color. The color of an individual component is assumed to vary slowly across that component with sharp edges on its boundaries. This is amply demonstrated in figure 3.2. It can be noted in this figure that the text has a dark-blue color on a plate which is brown in color. The whole background consists of walls which are painted with pale yellow color with linearly varying shade. However, a color image that appears uniform to human observer

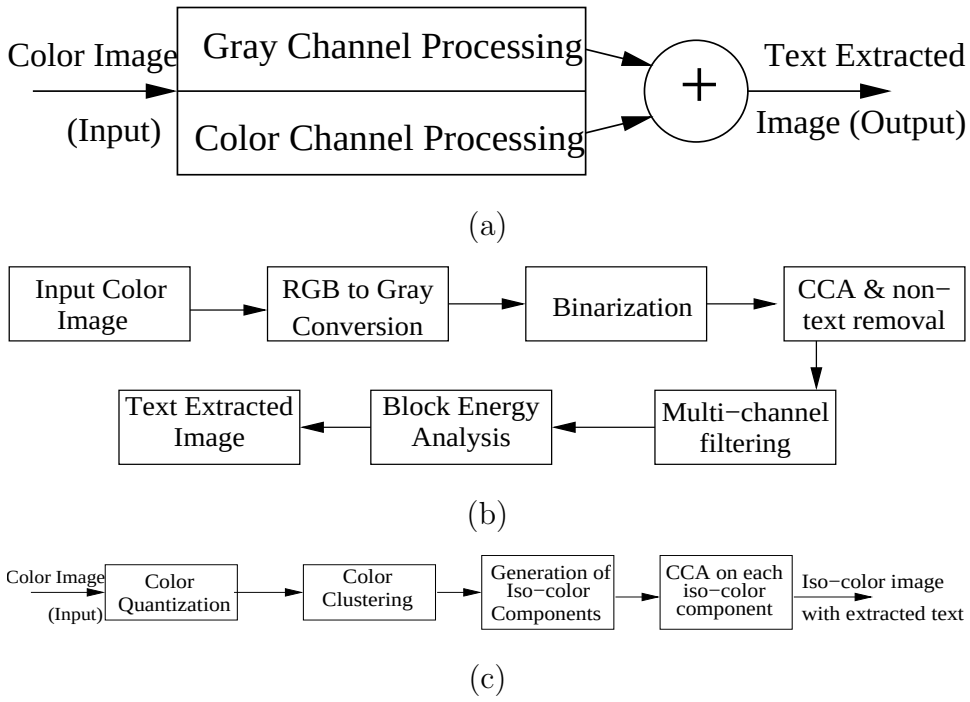


Figure 3.4: Overview of the proposed system is shown in a schematic diagram in (a). The gray and the color channels are presented in block schematic fashion in (b) and (c), respectively.

will show many small variations in color after digitization. These variations are hard to notice by humans and hardly provide any useful information. Hence, they may be eliminated by quantization of the number of available pallettes.

Let us say there are L pixels in the image, namely, $p(i_1, j_1)$, $p(i_2, j_2)$, $\cdots$, $p(i_L, j_L)$. These pixels are represented in the color space as $P_1(r_1, g_1, b_1)$, $\cdots$, $P_L(r_L, g_L, b_L)$. Here, each pixel is a point in the three dimensional space where the color components Red (R), Green (G) and Blue (B) form the axis system. Each of these primary color components vary from 0 to 255 generating 16,581,375 unique pallettes. Since the number of pixels in most images we use is much smaller than the number of unique color pallettes, the color space is sparse. So, we quantize each color into 16 different levels to generate only 4096 bins. A 3-D histogram is generated from this color quantized image. In this histogram, each bin represents an unique color pallette while the count in these bins corresponds to the number of pixels in the image with this pallette.

The 3-D color histogram is clustered using a Graph-Theoretical clustering algorithm. According to this algorithm, a relationship between a given bin, B_0 and its neighbors is established. We consider the 26 neighbors, B_1, $\cdots$, B_{26}, of the bin and their count. The bin, B_0, is linked to the neighboring bin with the highest count and this link is called a

branch. If the count of B_0 is the highest in the neighborhood, it is self linked and it is called a *root* node. Figure 3.5(a) demonstrates this concept in a two-dimensional scenario. In this figure, there are two roots; one is located on the second row and third column, while the other is at the fourth row and seventh column. All the other bins with a non-zero count have been linked to one of these two roots by branches, marked as arrows. A root with the nodes linked to it by the branches form a *tree* and each tree is a color cluster.

Let us say, such a clustering generates N different roots, *i.e.*, N clusters in the color space. An *iso-color component* image is formed when we selectively set to 1 all the pixels of a single cluster pallette and force other pixels to background. This generates N different *iso-color component* images. This principle of formation of iso-color component images, is explained in Fig. 3.5(b). In this image, the blocks C_1, C_2, $\cdots$, C_N represent the N clusters formed in the color space. The block C_1 is linked to n blocks namely, com_1, com_2, $\cdots$, com_n. Similarly, the block C_N is linked to m such *com* blocks. Each of these *com* blocks represent a quantized color pallette, which is a bin in the color space. Each *com* block has a count which tells us how many pixels in the image are represented by that color pallette. It may be observed that a number of pixels are linked to the com_1^1. It may be noted here that any given pixel will follow only a single link-path to one of the C-blocks. An iso-color component image is formed when all pixels connected to each of the *com* blocks associated with the C_1 cluster are considered as on-pixels and the others as background. An iso-color component image is marked as a binary image in figure 3.5(b). Each iso-color component image is then analyzed separately for their text content using the CCA, as shown in figure 3.4(c).

The color component analysis is summarized below as an algorithm.

1 Quantize the color space into prototype colors. The prototypes are found by masking the four least significant bits.

2 Compute the 3-D histogram (RGB color space with 16 levels) of the color image.

3 Construct 'lists' of pixels representing each histogram bin.

4 For every bin in the histogram, link to the bin which has maximum vote in its 26 neighborhood, forming non-overlapping unimodal clusters in tree structures. (The self-referencing bins are the local maximum and representative of the cluster, C_i, $i = 1,\ 2,\ \cdots,\ N$.)

5 Generate connected components from each cluster to construct binary images (see figure 3.5(b)).

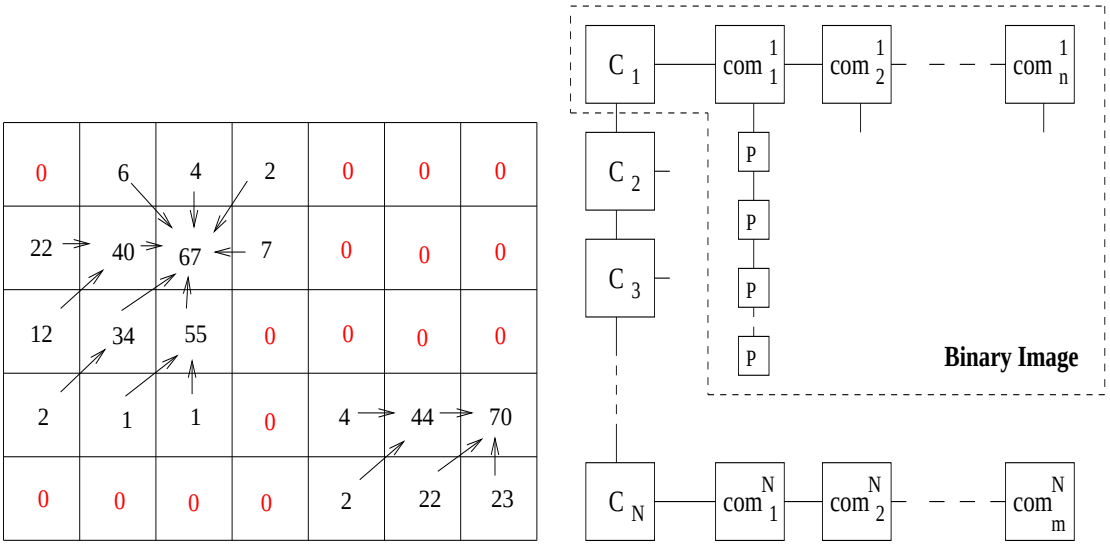


Figure 3.5: (a) Example for graph-theoretical clustering with 2D-color histogram of the image, (b) Cluster component structure.

6 Connected Component Analysis:

N_i is the no. of ON pixels in cluster i; $i = 1,\ 2,\ \cdots,\ N$.
$N_m\ =\ Max\ \{N_i\}$.
FOR each cluster i,

IF each N_i is greater than 5% of N_m.

FOR each connected component in the cluster:
Select text components based on geometrical and statistical information of the components (refer section 2.4.2).

END FOR each

END IF each

END FOR each

7 At the end of the CC analysis, final image is constructed by merging all components obtained from different color clusters.

3.3 Experimental Results and Discussion

The proposed algorithm is evaluated on the images present in the database, described in section 2.3. Two different quantitative measures are adopted for effective evaluation of

the algorithm. This is accomplished by counting the number of detected characters (54). These quantitative measures are formulated for the effectiveness of the algorithm.

$$Precision\ Rate\ =\ \frac{No.\ of\ correctly\ detected\ text\ characters}{No.\ of\ detected\ characters}\ X\ 100 \tag{3.1}$$

$$Recall\ Rate\ =\ \frac{No.\ of\ correctly\ detected\ text\ characters}{No.\ of\ characters\ present\ in\ original\ Image}\ X\ 100 \tag{3.2}$$

Thus, a low value of precision rate tells that the output has a lot of false positives, while a low value of recall rate informs that the algorithm missed a lot of actual characters. The accuracy of detection is evaluated on 70 representative images taken from the database described in section 2.3. This evaluation is carried out manually and by visual inspection of these images and their outputs. A precision rate of 91.7% is attained on an average while the average recall rate achieved is 89.2%.

Fig. 3.6 illustrates the comparison between general color clustering technique with CCA and the proposed algorithm. A camera-captured image is shown in Fig. 3.6(a) which has very little text. Fig. 3.6(b) shows the regions identified as text by color clustering and CC analysis and Fig. 3.6(c) shows the output using the Gabor filters. Fig. 3.6(d) shows the output of the proposed system, where the region of text is well detected in spite of the dominance of natural background.

A natural image with synthetic text is shown in figure 3.7(a). This image has 40098 unique colors. The interesting point of this example is the gradual change of the background colors. The application of the proposed text extraction technique, gives the results shown in Fig. 3.7(b). Fig. 3.7(c) shows a color image with complex background. The number of unique colors in this image is 88399. It can be observed that the majority of the text areas are correctly obtained and this is shown in Fig. 3.7(d). Test image 3.7(e) has background of two principal color tones. In addition, this document has white as text and background regions. Further, connected component analysis becomes complex here, because the color of the text in some regions is the same as the color of the background in some other region. The text areas are correctly obtained and are shown in figure 3.7(f).

Table 3.1 presents the time taken by each module of the algorithm for a typical camera captured image (520×640) and for a frame extracted from a video sequence of size 240×320. It may be noted here that the Gabor filtering is the most time consuming part of the proposed algorithm.

Table 3.1: Processing Time of different operations for Video Frame (240×320) and Camera Image (520×640) in seconds.

Technique	**For Video Frame**	**For Camera Image**
Gabor Filtering	3.0	7.0
Computing 3-D histogram	0.03	0.035
Constructing and Linking Tree	0.37	0.45
CC analysis in one cluster	0.07	0.08

3.4 Conclusion

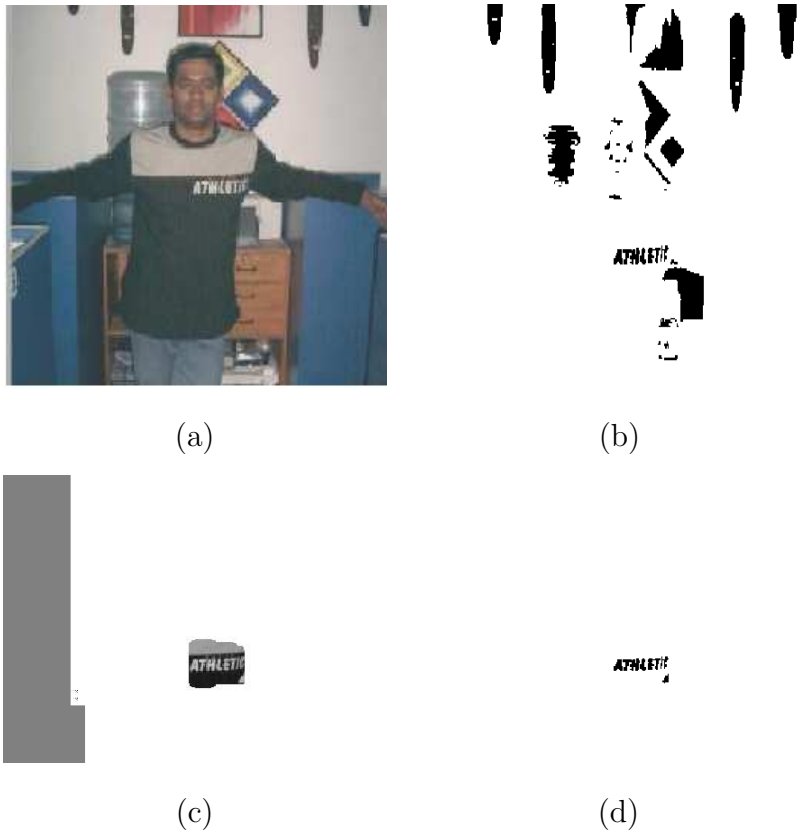
In this chapter, an approach to automatically localize and extract text from color document or scenic images, acquired through camera (still and mobile) or scanner, is proposed and implemented. The scheme has been illustrated to be working well for all such kinds of documents, irrespective of script, kind of clutter or mode of acquisition. The results obtained thus far are encouraging.

This model mostly relies on separating the text and non-text elements based on the spatial frequencies of the locality. This has been efficiently exploited by a Gabor filter bank. A bank of Gabor filters are employed in the proposed system because they are demonstrated to be simulating the human visual system (HVS). The hypothesis that a system designed on the working model of HVS would perform well is amply demonstrated.

Line based clutters are observed to be equally rich in higher frequency components. This necessitates the use of another method, based on the geometry of the elements of the image. A CCA algorithm is seen to be adequately complimenting the filterbank approach, enhancing the performance of the system.

The algorithm is able to detect and localize text in document images as well as in scenic images with complex background. It has the ability to deal with high resolution scanned images and camera captured images with equal efficiency. This is demonstrated with the high precision rates and recall rates recorded for the images. Not only is the proposed algorithm script invariant, it also is invariant to skew present in the document images. The disadvantage of our algorithm is the computational cost. The running of the algorithm on the mobile images shown in Fig. 3.6 takes about $3.5s$ and camera captured images shown in Fig. 3.7(c) takes $7.82s$ on a PC with Pentium-IV processor at 2.2 GHz with 512 MB of RAM. When text extraction performance is the criteria, our scheme performs consistently well.

Integrating an OCR which can perform well on low-resolution images with the above



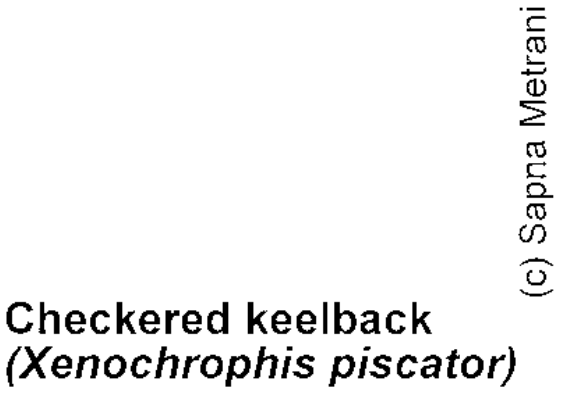
(a) (b)

(c) (d)

Figure 3.6: Text localization Results (a) Original Camera Image (b) Result only with Color Clustering and standard connected component analysis (c) Output of the Gabor filter (d) Result of proposed scheme.

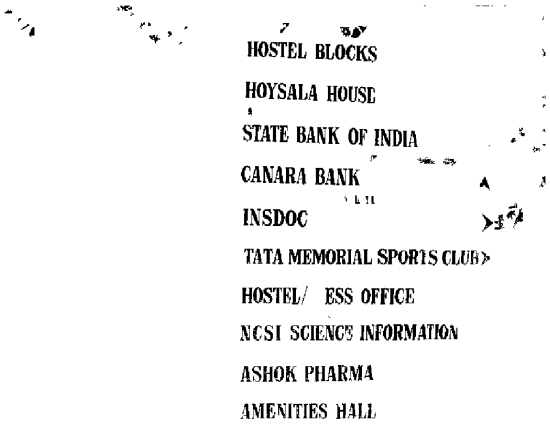
text extraction method is proposed as a scope for future investigations. The proposed approach could also be applied to handwritten documents.

(a)



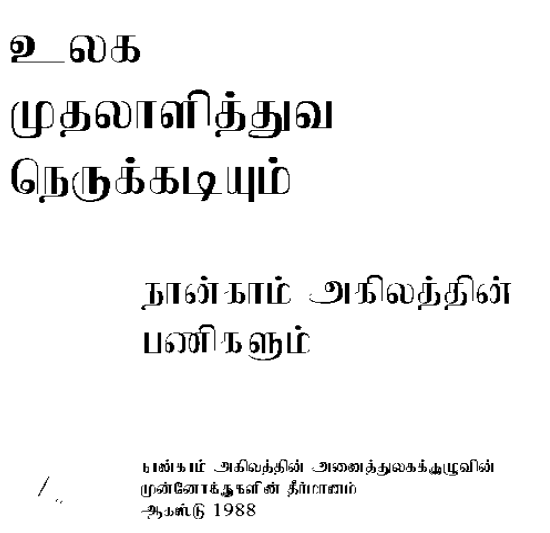
(b)

(c)



(d)

(e)



(f)

Figure 3.7: Some sample results are presented in the right column for the input images shown in the left column. (a) Scenic image with embedded text and its output (b), (c) camera captured image with natural text (text present in the scene) with the result in (d), and (e) scanned book cover image with result in (f).

Chapter 4

Word Level Script Identification

"It doesn't matter how beautiful your experiment is, it doesn't matter how carefully you collect your data – if it is based upon a faulty understanding of what is being tested, it's most likely useless." – L.D. Hosford

---

Summary:

Most of the multi-lingual documents have scripts changing with words in India. So, a script identification algorithm at the word level, is presented in this chapter. We start with bi-script cases, and then extend to tri-script and, subsequently, to eleven-script scenarios. A large database of 20,000 words, of different font styles and sizes, is collected for each script. This is so far the largest word dataset available for Indian scripts. Effectiveness of Gabor features is evaluated and compared against that of DCT using nearest neighbor, linear discriminant and support vector machine (SVM) classifiers. The combination of Gabor features and SVM classifier shows promising results, namely, over 98% for bi-script and tri-script cases and above 92% for the eleven-script scenario.

4.1 Introduction

Demand for tools with capability to recognize, search and retrieve documents from multi-script and multi-lingual environments, has increased many folds in the recent years. Thus, recognition of the script and language play an important part for automated processing and utilization of documents. Plenty of research has been carried out for accomplishing this task of script recognition at a paragraph/block or line level. While the former assumes that a full document page is of the same script, the latter imagines documents to contain text from multiple scripts but changing at the level of the line. Though the latter is a

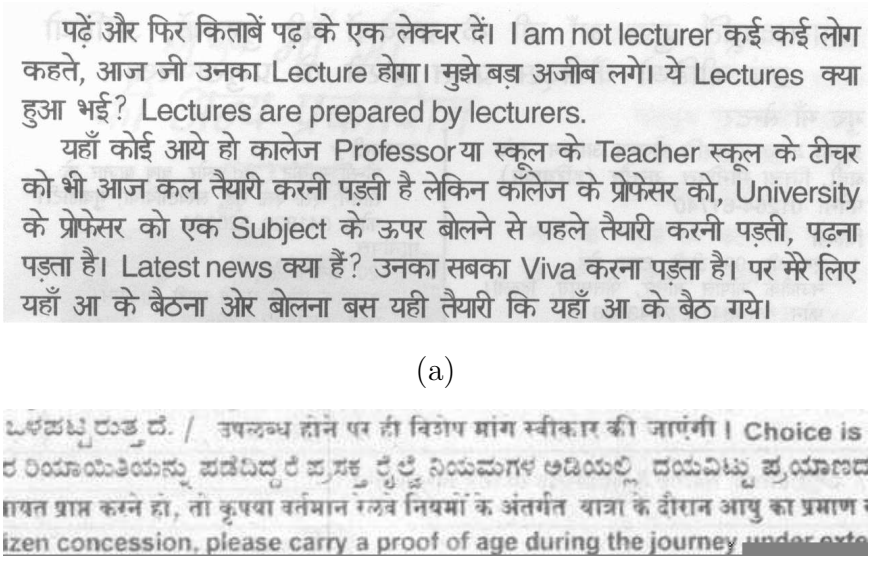


Figure 4.1: Sample documents to demonstrate the variation of script at the word level. (a) a bi-script document showing interspersed Hindi and English words. (b) a tri-lingual railway reservation form with words from Kannada, Devanagari and Roman scripts; the first line contains words from all the three scripts.

Figure 4.2: (a) An example image showing the word "Bharatiya" in different Indian scripts. Please note the shirorekha, the horizontal straight line joining the characters in the words, present in Assamese and Bengali (shown in the first and second columns of first row, respectively), Devanagari (second column of the second row), and Punjabi (first column of the fifth row) scripts. Example of two consonants combining to form a completely new symbol for (b) Odiya script, and (c) Devanagari script.

realistic assumption in some cases, most of the practical situations has the script changing with words. In figures 4.1 (a) & (b), we show two text images where the script changes at the word level. Fig. 4.1(a) shows a bi-script document where the presence of interspersed English words in a document of Devanagari script is clearly seen. Similarly, Fig. 4.1(b) shows the variation of script at both line and word level. It is important to mention that, many researchers assume that multi-script documents generally contain text from two scripts. With the figure 4.1(b), we emphasize the presence of three scripts in a document, which is a common occurrence in India.

Most of the OCR systems are designed using statistical pattern recognition techniques. It is generally observed that these systems produce good output for specific kinds of documents with a reasonable number of character classes. Including all the various symbols from all the various scripts from all over the world, together in the reference set will create a prohibitively large search space & hence, is not a feasible approach. Moreover, there are some symbols with a large visible similarity across different scripts, representing different characters. This would generate confusion about the script of the identified symbol.

India has thirteen different scripts representing its major languages. Figure 4.2(a) presents a single word "Bharatiya" (meaning Indian) in 12 different scripts. Most of these scripts have 12 vowels and about 37 consonants. In these scripts, a consonant may combine with another consonant or a vowel to generate a new symbol, unlike Roman script. Figures 4.2(b) & (c) show the combinations of two different consonants from Odiya and Devanagari scripts, respectively, to form new symbols. Thus, CV and CCV combinations, in many scripts, generate a huge set of frequently occurring graphemes. In Telugu script alone, Rajasekaran and Deekshatulu (55; 56) have identified some 2000 symbols which are used regularly. Kannada, with a very similar script & rules, has comparable number of symbols. Devanagari and similar scripts have close to 6000 such combinations each and other scripts have around 300 symbols.

Script identification acts as a preliminary level of filtering to reduce the search complexity. Moreover, for scripts such as Devanagari and Bengali, identification of script helps decide the further course of processing, such as, removal of the *shirorekha*[1] (the headline) from the word to separate the constituent symbols and recognize them. Thus, identification of the script is one of the necessary challenges before the designer of OCR systems, dealing with multi-script documents. Literature survey for script identification techniques, acting at different levels is presented in Sec. 1.3.2.

Most of the algorithms developed for identifying Indian scripts work on paragraph

[1]All the characters in a word are joined together by a headline in these scripts to form a word. Refer figure 4.1 for examples.

or line images. However, in general, the scripts in multi-lingual documents change with words. Figure 4.1(a) demonstrates this with an example bilingual document containing Devanagari and Roman scripts. In this chapter, efficacy of various combinations of two different features and three different classifiers is evaluated for the script recognition task. Various levels of script recognition scenarios (bi-script, tri-script & eleven-script), are explored. An initial design of a filterbank that generates good accuracy for various script identification tasks, followed by a feature selection algorithm to choose the minimal best subset of these features to deliver the best output is reported. Effectiveness of the word database, for the recognition exercise at higher levels (lines, paragraphs) is also explored.

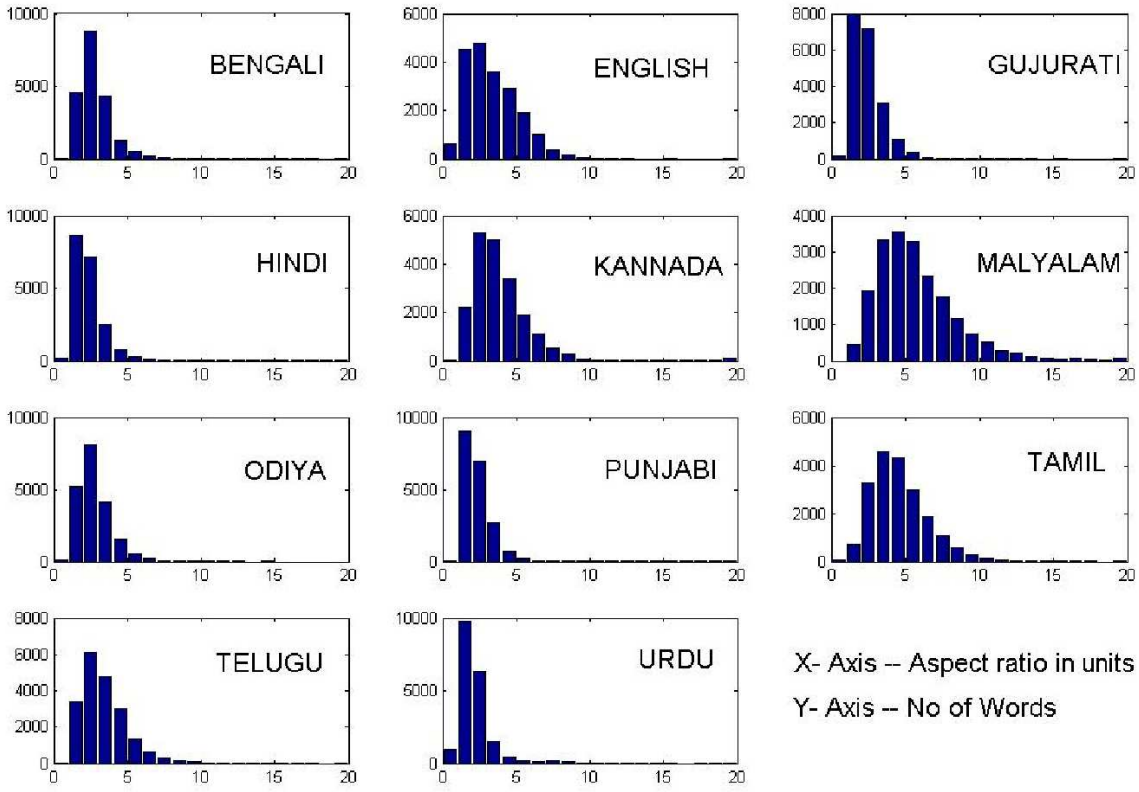
4.2 Data Description

Document images are scanned using: (i) Umax Astra 5400, and (ii) HP Scanjet 2200c, scanners at 300 dpi resolution and stored in 8-bit gray format. The images are scanned from magazines, newspapers, books and laser printed documents of eleven[2] different Indian scripts. Variations in printing style and sizes are ensured. About 100 pages are scanned from each of these scripts and are segmented into words by an automated process(51).

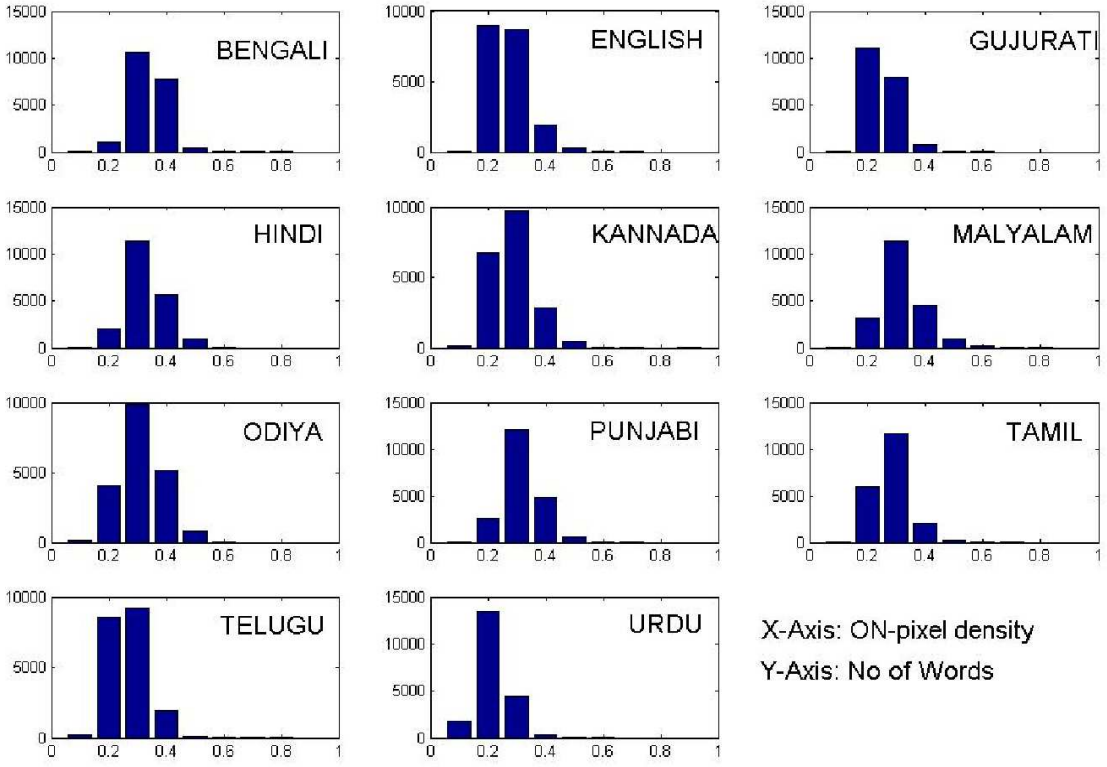
Each segmented word is visually inspected to make sure that at least a single base character is present. The blank rows and columns at the beginning & end of each word are removed. This makes sure that the word touches the boundaries of the matrix representing it. Each word matrix is available to us in binary form, where ON-pixels form the structural limbs of a character/pattern in the word. The sizes of the words are not normalized, *i.e.*, the words are retained in their original sizes as they appeared in the document image. Words of various lengths and ON-pixel densities are retained. Figure 4.3 presents the distribution of the aspect ratios and the ON-pixel densities for the word image dataset. It may be noted from Fig. 4.3(a) that the words have a wide range of aspect ratios in all the scripts. Similarly, figure 4.3(b) shows that ON-pixel densities of words across scripts also have a wide variability.

Once a large number of such words are collected, two-dimensional discrete cosine transform is performed on the size normalized version of these words and feature vectors are formed. The Euclidean distance is evaluated between all the combinations of two words from the same script. If the distance between any two words becomes zero, only one of the two is retained in the database. This ensures that no word is repeated in the

[2]The eleven Indian scripts are Bengali, Devanagari, Gujarati, Kannada, Malayalam, Odiya, Punjabi, Roman, Tamil, Telugu and Urdu.



(a)



(b)

Figure 4.3: Characteristics of the word database. (a) The distribution of the aspect ratio, (b) ON-pixel density.

feature space. At the end, 20,000 words are randomly selected from this large collection to form our word data set. 7000 of these are selected randomly from each script, to form the training set, while the rest 13,000 words form the test set. This set, so far the largest collection of word data for different Indian scripts, is available for public use from the world wide web network (57).

4.3 System Description

We studied the structural properties of eleven Indian scripts before designing an identifier for these scripts. Since, Assamese and Bengali scripts are nearly the same, we consider them as one script. An observation of these eleven scripts reveals the following properties:

- Bengali, Devanagari and Punjabi scripts have a *shirorekha* joining the individual symbols forming any word.
- These three scripts have a lot of vertical strokes. Besides, the structural limbs of Devanagari are more circular than those of Bengali and Punjabi.
- There are many limbs along 60^0 and 120^0 directions in Bengali script.
- Punjabi script has a lot of half-length vertical strokes.
- Kannada and Telugu scripts are very similar. There is a tick feature associated with Telugu script, which does not appear in Kannada.
- Tamil and Malayalam are somewhat similar to each other. Malayalam appears like Tamil script with smooth corners.
- Visually, Gujarati script looks like Devanagari but has more loops. In addition to high number of loops, Odiya script also has a lot of vertical strokes.
- Urdu looks quite unlike any other Indian script. It has a lot of connectivities and curvatures, aligned either along horizontal direction or at about 75^0 angle.

With these observations, we decided to employ features that are both frequency and direction sensitive. This, in our opinion, would be best able to discriminate between the scripts.

Two approaches can be pursued for script identification in a multi-script scenario. One of them extensively studies the similarities and differences in the structures between the co-occurring scripts (22; 25; 41), while the second method treats each script as a different

texture (34; 36; 38). The textural method is more robust because it deals with the script regardless of the size or style of the font. This claim of ours is supported by our earlier employment of a bank of Gabor filters for successful page layout analysis (52; 54) and script identification (34; 35; 36).

Thus, we employ a multi-channel filter bank, using Gabor functions. We have used a radial frequency bandwidth of one octave. This is because, the cortical cells in the visual cortex V1 of primates are observed to have that bandwidth and the coding of natural images is best attained at this bandwidth (58). An angular bandwidth of 30° is chosen for this experiment. This comes from our study of the properties of the scripts under consideration. We considered five different radial frequencies (0.0625, 0.125, 0.25, 0.5 and 1). The five radial frequencies combine with six angles (0, 30, 60, 90, 120 and 150 degrees) to form thirty different frequency-angle combinations. Each combination of an angle with a radial frequency has two filters (sine and cosine).

A Gabor filter is defined by its three independent parameters: (i) radial frequency, (ii) angle of orientation, and (iii) sine/cosine function. We denote such a filter by ${}_cH_\theta^u$ where the subscript 'c' to the left of H represents a cosine filter (this becomes 's' for a sine filter); 'u' & 'θ' denote the radial frequency and angle, respectively. The image (W) is filtered by spatial convolution with the filter (H).

$$\hat{W} \;=\; W \;*\; {}_cH_\theta^u \tag{4.1}$$

$${}_cf_\theta^u \;=\; \frac{\sum_{i=1}^{R}\sum_{j=1}^{C}\;\hat{W}^2(i,j)}{\sum_{i=1}^{R}\sum_{j=1}^{C}\;W^2(i,j)} \tag{4.2}$$

where $\hat{W}$ (in Eqn. 4.1) is the output image. Energy of the filtered image, f, is evaluated by adding the squares of pixel values in the output image. This energy is normalized by the total energy of the input word image to make a feature.

Each feature is associated with a specific combination of the three different filter parameters, namely, radial frequency, angle of orientation and function of the filter (sine or cosine). The sixty dimensional feature vector is arranged as shown in equation 4.3, with the thirty cosine coefficients placed before the sine coefficients.

$$F = \left[{}_cf_{\theta_1}^{u_1},\; {}_cf_{\theta_2}^{u_1},\; \cdots,{}_cf_{\theta_1}^{u_2},\; \cdots,\; {}_cf_{\theta_6}^{u_5},\; {}_sf_{\theta_1}^{u_1},\; \cdots,\; {}_sf_{\theta_6}^{u_5}\right] \tag{4.3}$$

We studied the efficacy of these frequencies for their separability with all bi-class problems involving Roman and one of the Indian scripts. This was accomplished by

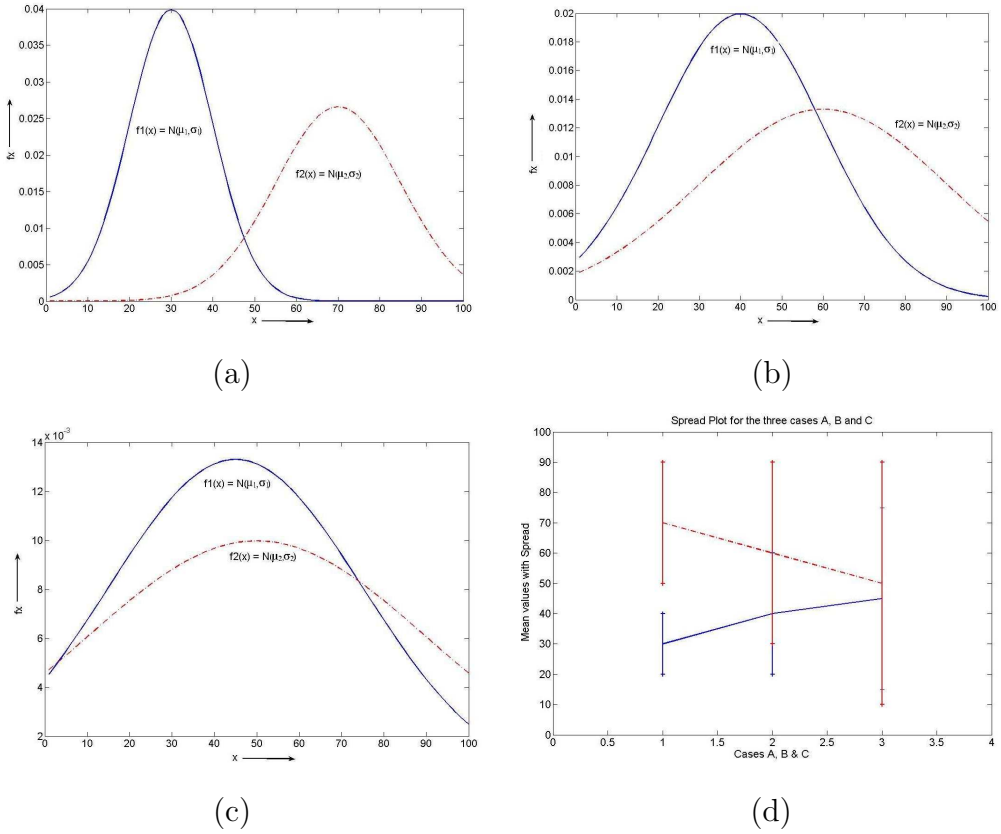


Figure 4.4: Demonstration of the discriminability of features and their spread plot. (a) a discriminable feature. (b) a partially discriminable feature. (c) a non-discriminable feature. (d) the spread plots of the features shown in (a), (b) and (c).

observing: (i) the spread and (ii) the divergence values of the features.

Each feature is assumed to follow a normal distribution. Thus, we evaluate the mean and the standard deviation of the individual features, from each of the classes. When we plot the Gaussian curves for the features from two classes, it generates curves similar to figures 4.4(a), (b) or (c). If the means are close and the standard deviations large, the two classes are said to have a large overlap. When the mean values are distinct with small standard deviations, the feature is discriminable. The spread plot for features with distributions shown in figures 4.4(a), (b) and (c), is shown in Fig. 4.4(d). We generate this plot to study the discriminability of each feature.

Divergence of a feature (59), in a bi-class scenario, is defined as:

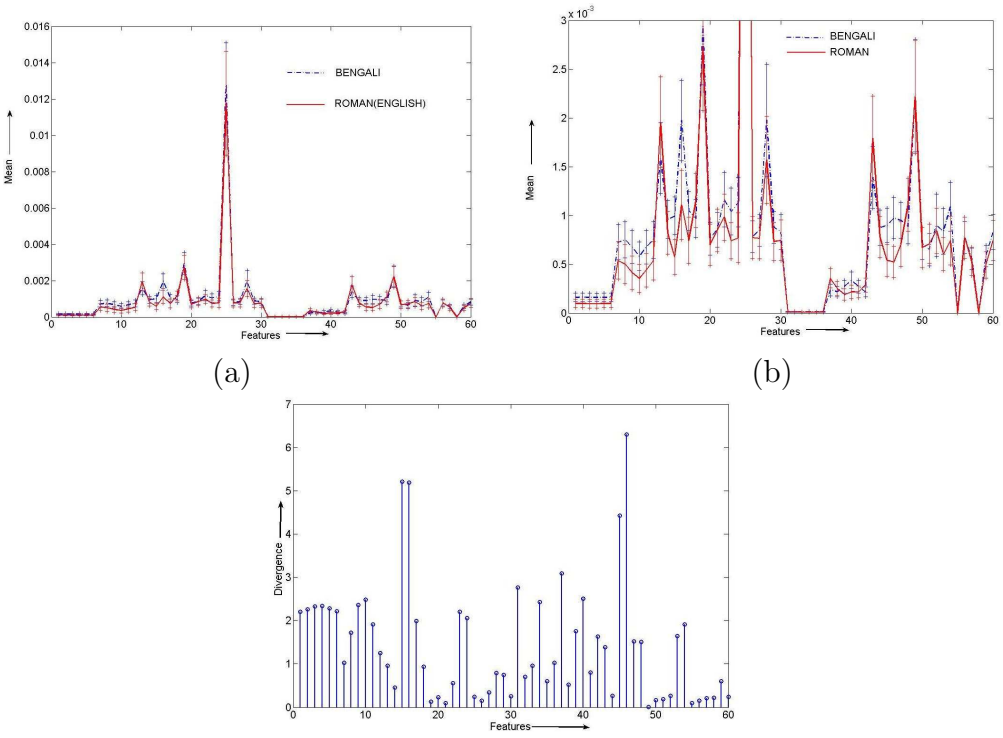
$$d = \frac{1}{2}\left(\frac{\sigma_1^2}{\sigma_2^2} + \frac{\sigma_2^2}{\sigma_1^2} - 2\right) + \frac{1}{2}(\mu_1 - \mu_2)^2\left(\frac{1}{\sigma_1^2} + \frac{1}{\sigma_2^2}\right) \tag{4.4}$$

where μ_1 and σ_1 are the mean and the standard deviation of the feature for class ω_1, assuming a normal distribution. $N(\mu_2, \sigma_2)$ represents the distribution of the same feature for class ω_2. Two classes are better discriminated with a bigger divergence. The divergence of each feature is evaluated to decide its separability.

Figure 4.5(a) shows a spread plot for the Gabor features generated by the filter bank, as mentioned above. Here, we use five radial frequencies (1, 0.5, 0.25, 0.125 and 0.0625) and six angles (0, 30, 60, 90, 120 and 150 degrees). Therefore, there are thirty cosine and an equal number of sine filters giving rise to a feature vector of dimension sixty. The first thirty features in the feature vector are the cosine coefficients and the next thirty features are the sine features. Similarly, the first six features correspond to the coefficients of cosine filter with radial frequency, $u = 0.0625$, and the increasing angles in the order mentioned above. Thus, the feature number 1 comes from the filter with radial frequency 0.0625 and angle 0 degrees while the 10^{th} feature corresponds to the filter with $u = 0.125$ and 90^0 angle. The 31^{st} to 60^{th} features are derived in the same order of filter parameters, but with sine filters. A magnified version of the spread plot in Fig. 4.5(a) is provided in Fig. 4.5(b) for better observability. The divergence plot for the same case is presented in Fig. 4.5(c). It is seen from figure 4.5(c) that the features numbered 15, 16, 45 and 46 have considerably larger values than others which means these features are better discriminable than the others. On a closer look at the spread plot shown in Fig. 4.5(b), it is these features which are clearly discriminable. Thus, it may be assumed that these features would generate a good accuracy, for this bi-script case of Bengali and English.

We observed the spread plot and divergence of all the bi-class cases involving separation of Roman script from each of other 10 Indian scripts. Based on our observation, we decide to use three different radial frequencies (0.125, 0.25 & 0.5) and all the six angles of orientation. The spatial spread of each filter along the $x-$ and $y-$coordinates are determined by the standard deviations of the Gaussian's, σ_x and σ_y, respectively. Both of them are functions of the radial frequency and angular bandwidth. The three radial frequencies with six θ's give a combination of eighteen odd and eighteen even filters. A given word image is filtered with these thirty-six filters to generate a thirty-six dimensional feature vector as mentioned in equation 4.3. These feature vectors are used for the identification of the script of the word. Figure 4.6(b) presents the output images for a sample Hindi word (Fig. 4.6(a)) when filtered by the eighteen cosine filters. Figure 4.7(a) shows a block diagram of the script identification system using the Gabor features.

Discrete Cosine Transform (DCT) concentrates the information content in a relatively few coefficients. For natural signals and images, the data compaction of DCT is close to that of the optimal KL transform. But unlike KLT, DCT is not dependent on the data. Its transform matrix exhibits symmetries which can be exploited to obtain efficient

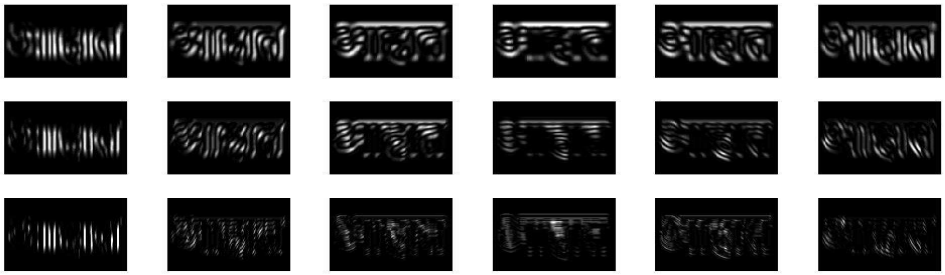


(a) (b)

(c)

Figure 4.5: Example case of Bengali and English bi-class discriminability analysis. (a) Spread plot using the 60-dimensional feature vectors from Bengali and English scripts, (b) a magnified version of the spread plot in (a), and (c) the divergence plot for the same case.

(a)



(b)

Figure 4.6: (a) A sample Devanagari word, and (b) the 18 cosine Gabor filtered output images. Here, each row corresponds to a radial frequency, in increasing order, and each column corresponds to an angle, in increasing order.

hardware and software implementations (60). Most image and video coders employ DCT. It has also been employed for other applications such as pitch modification for speech synthesis (61). Pati (51) has used DCT coefficients for machine recognition of printed Odiya characters. Salazar and Tran (62) have reported better quality image resizing in the DCT domain. DCT has also been used for motion estimation (63), image compression with morphological descriptor (64) and object boundary detection (65).

For an image $I(i, j)$, the DCT coefficient matrix $B(k, l)$ is given by,

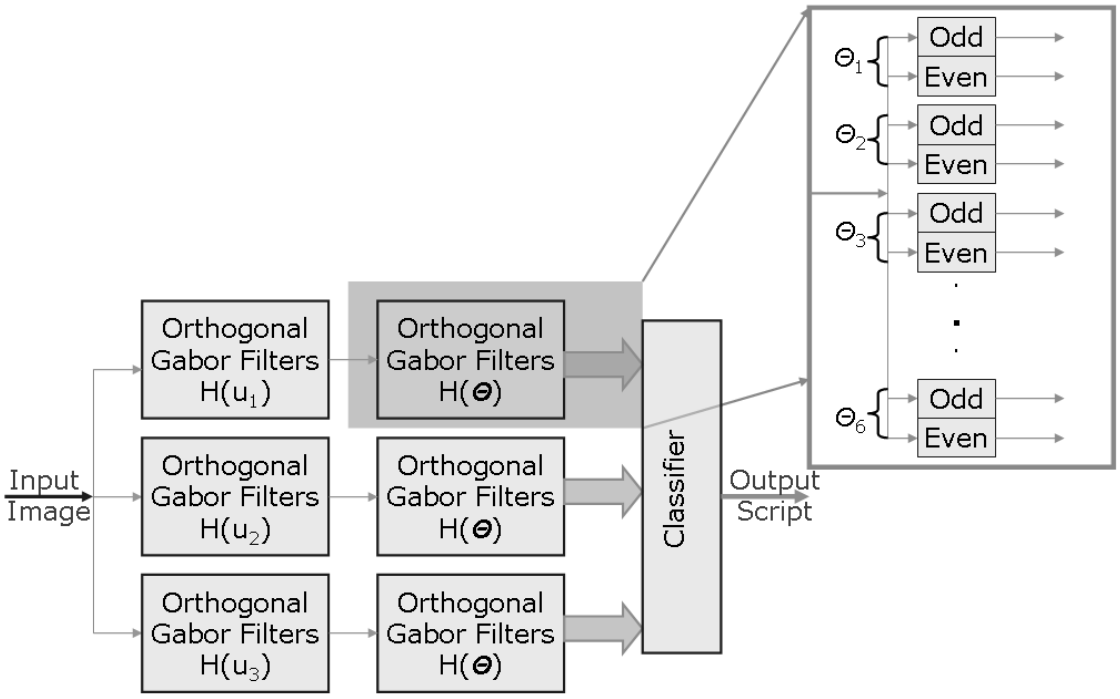
$$B(k,l) = \sum_{i=0}^{R-1} \sum_{j=0}^{C-1} I(i,j) \cos\left(\frac{\pi k(2i+1)}{2\,R}\right) \cos\left(\frac{\pi l(2j+1)}{2\,C}\right) \tag{4.5}$$

where R and C are the number of rows and columns of the image matrix; k and l are the frequency indices along the i and j directions, respectively.

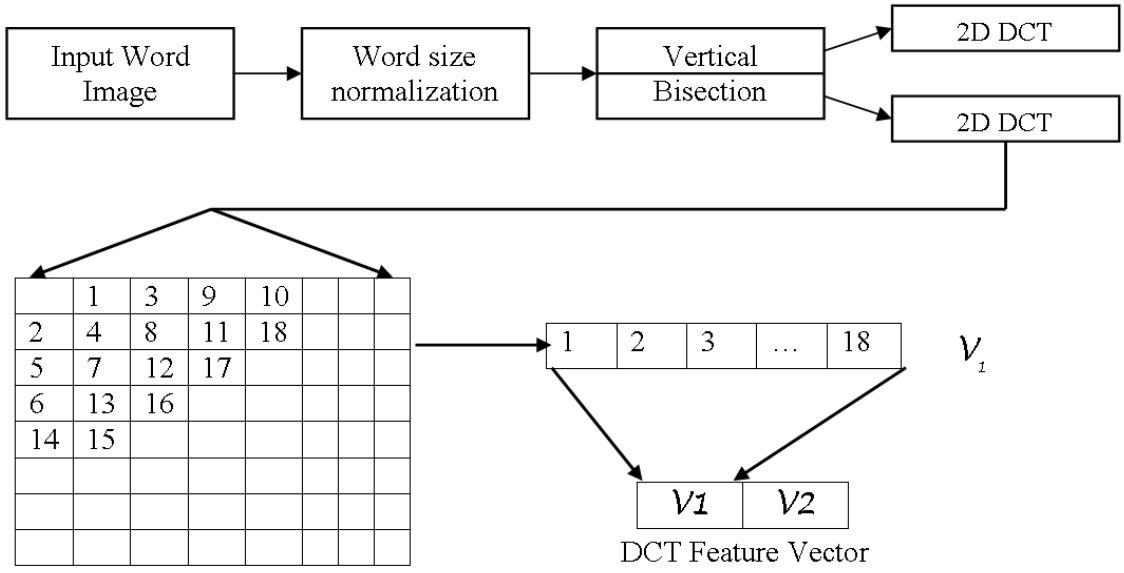
The energy compactness property of DCT justifies its use for script identification. Figure 4.7(b) diagrammatically presents the extraction of the DCT feature vector. Initially, the input word image is normalized to a standard size. It is then vertically divided into two equal blocks and 2-D DCT is performed on each of the block, independently. As shown in Fig. 4.7(b), eighteen low frequency coefficients are chosen in a zig-zag fashion from this DCT matrix of each half of the word image. The vectors are appended to form a thirtysix-dimensional feature vector, which is used for classification. We have taken 36 coefficients for a fair comparison with the Gabor filter based system.

We have used three different classifiers to decide about the script of the test words: (i) the nearest neighbor classifier (NNC), (ii) linear discriminant classifier (LDC), and (iii) the support vector machines (SVM's). Nearest neighbor has been a standard and time tested classifier. This classifier has proven to deliver good output, when we have a class representative training set. Here, Euclidean distance of the test pattern is evaluated in the feature space, with each of the training patterns. The class value of the nearest neighbor is assigned to the test pattern. A linear discriminant function (70) partitions the feature space using a hyper-plane. The two sides of this plane represent the two classes. The class value of the test pattern is decided based on which side of the plane it lies. A multi-class scenario could be handled as a number of bi-class scenarios.

Among the discriminant approaches for classification, the most recent is the Support Vector Machine (66), where the optimal hyper-plane decides the separation between individual classes of patterns. The creation of a unique model to represent a class, derived by training the model with prototypes of each class, aids in maximization of the correct classification rate. We have used the **SVMTorch − II**, a toolbox designed and developed



(a)



(b)

Figure 4.7: Block diagrams of (a) Gabor and (b) DCT feature extractors.

by Collobert and Bengio (67; 68). Of the various kernel functions this toolbox provides, we have used the Gaussian function with the total variance of the dataset, in a case by case basis, chosen as the standard deviation of the kernel function.

4.4 Experiments and Results

Most of the multilingual documents in India are bi-script in nature. So, an initial attempt to resolve the bi-script issues is made. Based on the success attained there, we decided to extend the experiments to tri- & multi-script cases. Since most of the official documents of national importance have three scripts, such an experiment is justified. We also explore the possibility of recognizing the script of the word without any prior information. This is a blind script recognizer, where the training set contains samples from all the classes. Thus, it is a 11-script scenario. In the sections below, we present each of these cases, separately. A neighborhood analysis, on the reference patterns of the scripts, provides us with an insight into their spread in the considered feature spaces. This helps us anticipate outcomes for each case.

Throughout this section, the Bengali, English, Gujarati, Hindi, Kannada, Malayalam, Odiya, Punjabi, Tamil, Telugu and Urdu scripts are abbreviated as BE, EN, GU, HI, KA, MA, OD, PU, TA, TE and UR, respectively, for ease of reference in the tabular columns.

4.4.1 Neighborhood Analysis of the Database

Each script has 7000 randomly chosen reference words. A set of 77,000 patterns is created by taking the reference words from all the eleven scripts. This database is analyzed for the spread and separability of each class, in Gabor and DCT feature spaces. The class values of the 5 nearest neighbors of each word, are observed and the word is labeled as: (i) an *insider* if the class values of the word and its 5 nearest neighbors are same, (ii) an *out-lier* if the class value of the word doesn't match with the class value of any of the five nearest neighbors, and (iii) a *border pattern* if the class value of the word matches with the class values of some of them. The %age of these labels from the reference set of each script is tabulated in Table 4.1.

There are 35,000 neighbors for the 7000 reference patterns of each script. They provided us with important information about the inter-mixing of various scripts & extent of their overlap, in the feature space, and the classes that are more likely to be confused among themselves.

Tables 4.2 (for Gabor features) and 4.3 (for DCT features) present the confusion (into scripts presented column wise) matrices for these 35,000 neighbors for each script

Table 4.1: Neighborhood analysis of the training patterns in the Gabor & DCT feature spaces using k-NN (k = 5) principle. The results are presented in %age.

		BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
Gabor	Inlier	71.9	94.0	73.0	53.9	61.0	62.8	49.6	54.4	76.2	57.2	81.3
	Border	25.7	4.5	23.8	41.7	37.1	34.6	47.2	42.5	21.2	38.7	16.5
	Outlier	2.4	1.5	3.2	4.4	1.9	2.6	3.2	3.1	2.6	4.1	2.2
DCT	Inlier	74.7	92.0	80.5	62.7	67.3	63.3	80.8	64.4	59.9	43.9	75.8
	Border	23.3	7.1	17.8	35.0	29.4	34.3	17.2	32.7	36.2	50.3	21.4
	Outlier	2.0	0.8	1.7	2.2	3.3	2.5	2.0	3.0	3.9	5.7	2.8

Table 4.2: Neighborhood analysis for Gabor features. Each row represents one script for which the neighbors are analyzed, while entry under each column shows the contribution of a particular script to the neighbors of the script of the row.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	30838	4	203	1528	198	469	410	616	465	246	23
EN	6	33734	70	82	12	363	306	238	131	13	45
GU	192	63	30302	164	491	501	2391	76	185	368	267
HI	2646	47	362	27400	141	294	218	3512	234	95	51
KA	119	22	167	79	29708	580	584	93	108	3205	335
MA	285	112	182	199	344	29426	2227	1013	1010	156	46
OD	366	91	1381	119	810	3089	27389	365	703	524	163
PU	690	131	80	3494	105	1152	365	28410	460	37	76
TA	388	109	122	177	103	1656	676	605	30924	68	172
TE	218	1	146	35	5368	356	586	53	144	27814	252
UR	39	41	391	57	1094	287	504	104	389	301	31793
SUM	35787	34355	33406	33334	38374	38173	35656	35085	34753	32827	33223

Table 4.3: Neighborhood analysis using DCT features. Each row represents the script for which the neighbors are analyzed, while each column shows the contribution of each of the eleven scripts to the neighbors of the script of row.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	30969	152	209	1829	96	357	317	672	280	90	29
EN	24	33668	64	17	111	590	264	30	128	87	17
GU	229	426	31921	52	56	330	927	66	654	173	166
HI	2433	75	114	29454	101	392	188	1946	255	28	14
KA	164	889	98	128	29645	1833	294	91	211	1612	35
MA	202	1149	231	81	396	29629	758	59	2028	453	14
OD	251	380	341	143	139	991	31782	256	546	113	58
PU	1010	176	195	2979	74	388	643	29187	276	62	10
TA	253	1443	402	125	95	2971	647	115	28610	211	128
TE	252	1515	463	59	3324	2048	496	125	798	25508	412
UR	72	172	907	49	156	241	356	50	764	1209	31024
SUM	35879	40045	34945	34916	34193	39770	36672	32597	34550	29546	31970

(presented row wise). The last row shows how many neighbors are contributed by a given script to various scripts including itself. Thus, if a large number of contributions come from a script other than itself, it means there is very high likelihood of the script of the line getting confused with the script which contributes this large number. In Table 4.2, 5368 neighbors for the reference patterns of Telugu script are from Kannada. Similarly, Kannada has 3205 neighbors from Telugu. This means that at all classification levels (bi-script or multi-script), Kannada and Telugu would be confused among themselves, with Gabor feature. Similar observation is made for Bengali, Punjabi and Devanagari (Hindi) scripts in both the tables 4.2 and 4.3.

4.4.2 Bi-script Recognition

The bi-script documents, such as books, newspapers and magazines have an Indian script as the major script with interspersed English words. Besides, in the border areas of the states, people know more than one language and the documents reflect that. Thus, a good recognition in bi-script scenario is very useful. With eleven scripts, we have 55 different bi-class problems. We report the accuracies of the bi-class script identification in Tables 4.4, 4.5 and 4.6, with NNC, LDC and SVM as classifiers. The performance is presented in %age, which gives the average recognition accuracy for both the involved scripts.

Table 4.4 presents the results for all bi-script combinations with a nearest neighbor classifier. In this table, the upper triangular part of the matrix presents the results with

Table 4.4: Bi-Script recognition accuracies with NNC. The lower triangle part of the matrix gives the results with Gabor features & the upper triangle, the DCT. The results presented are average bi-script accuracies in %. The script names are presented in their abbreviated forms as mentioned in section 4.4.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	**99.6**	99.1	95.1	99.2	98.9	99.0	96.9	98.9	99.0	99.5
EN	99.8	–	99.3	**99.6**	98.2	97.9	99.1	99.4	97.7	97.5	98.9
GU	99.3	99.7	–	99.4	99.4	98.9	98.4	99.3	98.3	98.4	98.2
HI	95.2	99.6	99.2	–	99.3	99.0	99.2	94.0	99.1	99.4	**99.6**
KA	99.3	**99.9**	99.3	99.5	–	96.8	98.9	99.3	98.9	*93.4*	99.3
MA	98.7	99.1	99.1	98.7	98.5	–	97.7	98.8	93.8	95.9	99.1
OD	98.5	99.2	96.1	99.1	97.6	93.4	–	98.3	97.8	98.3	99.0
PU	98.2	99.2	99.3	91.5	99.5	96.6	98.6	–	98.9	99.2	**99.6**
TA	98.5	99.4	99.5	99.0	99.5	96.1	97.4	97.9	–	97.6	98.3
TE	99.0	99.8	99.3	99.6	*89.2*	98.9	97.8	99.6	99.5	–	97.5
UR	99.6	99.8	99.2	99.6	98.1	99.2	98.8	99.6	99.2	98.7	–

the DCT based feature vector while the lower triangular part shows the Gabor feature results. Equivalently, the tables 4.5 and 4.6, present results with linear discriminant classifier and support vector machines, respectively.

Close inspection of Table 4.4 reveals that though both the features have fared well with NNC, the Gabor (mean (μ) = 98.4, standard deviation (σ) = 2.0) has performed slightly better than the DCT features ($\mu = 98.3,\ \sigma = 1.5$). The same trend has become much more dominant with linear discriminant classifier. In fact, in Table 4.5, the gap between Gabor ($\mu = 98.3,\ \sigma = 1.5$) and DCT ($\mu = 88.5,\ \sigma = 4.4$) features is widened. With SVM as the classifier (see Table 4.6), the Gabor ($\mu = 98.4,\ \sigma = 1.7$) again leads the performance (for DCT: $\mu = 97.8,\ \sigma = 1.4$). Thus, it may be established that for bi-script recognition, Gabor may be preferred over DCT, though for some specific scripts, DCT has outperformed the Gabor. Similarly, a comparison of Tables 4.4, 4.5 and 4.6, shows that SVM and NNC have performed comparably and with consistency. Combinations of LDC, for both features, yield the least.

The most frequent scenario that warrants script recognition involves the official documents by State Governments and the text books in state languages. These documents involve the script of the official language of the state and English. Table 4.7 presents the results separating Roman from the other ten scripts with different feature-classifier combinations. The recognition rate we have achieved for this important application exceeds 99% for all the biscript combinations, using the NN classifier. The lowest performance is for Malayalam, which is 99.1%, whereas the best performance is for Kannada, which is

Table 4.5: Bi-Script recognition accuracies with linear discriminant classifier. The lower triangle part of the matrix represent the results with Gabor features while upper triangle shows those using DCT. The results presented are average bi-script accuracies in %. The script names are presented in their abbreviated forms as mentioned in section 4.4.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	**96.6**	92.3	80.3	91.9	88.4	92.1	87.6	94.0	89.0	95.1
EN	99.4	–	88.8	95.3	90.3	87.7	89.4	91.3	88.3	87.0	91.4
GU	99.3	98.5	–	91.9	90.7	85.7	88.6	90.3	85.1	86.9	87.5
HI	94.6	98.7	99.3	–	91.2	88.6	90.1	78.9	92.4	90.1	93.9
KA	98.6	99.0	98.6	98.6	–	84.8	90.0	90.5	91.9	*74.8*	91.5
MA	98.0	98.3	98.4	98.3	98.8	–	80.7	88.0	83.4	76.9	88.6
OD	98.8	98.8	93.5	99.2	98.0	97.4	–	87.1	88.4	85.5	91.9
PU	99.0	98.8	99.3	93.2	99.6	97.7	99.0	–	89.4	88.7	92.4
TA	98.7	98.0	98.7	98.2	99.1	96.0	97.7	97.8	–	86.4	83.9
TE	99.2	99.5	98.8	99.2	*92.9*	99.3	98.4	**99.7**	99.6	–	85.9
UR	99.3	99.5	98.2	99.4	97.7	99.3	98.6	**99.7**	99.3	99.0	–

99.9%. From this table, it may be noted that Gabor features score a clear margin over DCT with all three classifiers.

As may be noted from the above tables, the level of mis-recognition is quite a script dependent phenomenon. An investigation is carried out to expose the existence of any relationship between the rate of mis-recognition and the aspect ratio or the ON-pixel density of the word. An average of the histograms of the mis-recognized words, for all bi-script cases, is obtained with respect to their aspect ratio and ON-pixel densities. Fig. 4.8(a) shows the ratio of the number of mis-recognized words for a particular aspect ratio with the total number of words available with that aspect ratio in the test database. Similarly, Fig. 4.8(b) presents the ratio of the number of mis-recognized words to the total words as a function of ON-pixel densities of the words. These two graphs show the ratios to be relatively flat. Hence, mis-recognition of a word may not have any significant relationship with its length or ON-pixel density.

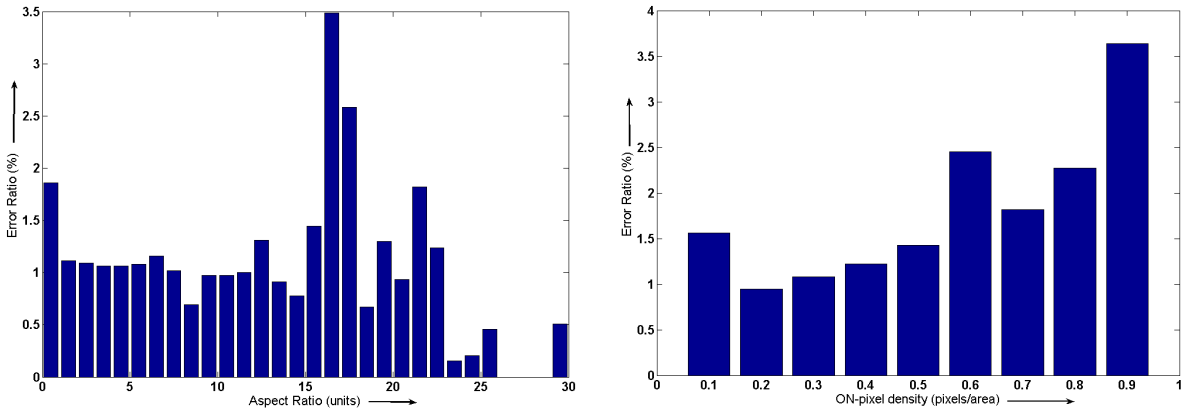
An inspection of the mean and standard deviation values for the features reveals a larger dynamic range for some features (refer figures 4.5(a) & (b)). In an Euclidean metric based classifier, this will bias the decision. Centering of features leads to efficient linear discriminant function evaluation and dynamic range normalization ensures equal contribution to decision making from all features. Results of this exercise are presented in Tables 4.8 and 4.9. It may be observed from comparison of these tables with Tables 4.4 and 4.5, respectively, that the centering and normalization of the features has very little effect for NNC and adversely affects the LDC. Since in the previous experiments, SVM

Table 4.6: Bi-Script recognition accuracies with support vector machines. The lower triangle part of the matrix represent the results with Gabor features while upper triangle shows the DCT feature results. The results presented are average bi-script accuracies in %. The script names are presented in their abbreviated forms as mentioned in section 4.4.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	**99.5**	98.1	95.6	99.1	98.7	98.1	97.2	98.0	98.8	**99.5**
EN	**99.8**	–	98.9	98.8	97.5	97.1	99.2	96.4	97.2	98.3	99.6
GU	99.1	98.7	–	99.2	98.7	98.8	97.9	98.5	98.4	98.6	96.1
HI	95.7	**99.8**	98.9	–	98.6	97.8	99.2	95.2	97.6	98.6	97.2
KA	99.0	99.7	99.0	99.5	–	96.8	97.9	98.2	99.2	95.7	97.8
MA	98.7	99.1	98.2	98.9	99.2	–	96.2	98.2	*93.9*	94.6	98.0
OD	**99.8**	99.3	98.5	99.7	98.2	97.8	–	97.4	98.3	96.9	98.5
PU	98.5	99.2	98.7	94.2	99.5	97.4	98.2	–	99.1	98.6	99.0
TA	98.9	95.7	98.8	99.2	99.6	96.9	98.6	98.7	–	97.1	96.4
TE	99.2	**99.8**	97.9	99.5	*93.6*	97.4	97.2	99.1	98.6	–	97.0
UR	99.7	99.7	99.3	**99.8**	98.7	99.6	99.6	93.7	96.2	98.5	–

Table 4.7: The recognition accuracies of the various feature-classifier combinations for the bi-script cases involving English words with one of the ten Indian scripts. Codes denoting the Indian scripts are described in Sec. 4.4. The last two columns present the mean (μ) and standard deviation (σ) for the bi-script results shown in the same row.

	BE	GU	HI	KA	MA	OD	PU	TA	TE	UR	μ	σ
GT-NNC	99.8	99.7	99.6	99.9	99.1	99.2	99.2	99.4	99.8	99.7	99.6	0.3
GT-LDC	99.4	98.5	98.7	99.0	98.3	98.8	98.8	98.0	99.5	99.4	98.8	0.5
GT-SVM	99.8	98.7	99.8	99.7	99.1	99.3	99.2	95.7	99.8	99.8	99.1	1.3
DCT-NNC	99.6	99.3	99.6	98.2	97.9	99.1	99.4	97.7	97.5	99.4	98.8	0.8
DCT-LDC	96.6	88.8	95.3	90.3	87.7	89.4	91.3	88.3	87.0	91.4	90.6	3.2
DCT-SVM	99.5	98.9	98.8	97.5	97.1	99.2	96.4	97.2	98.3	99.0	98.2	1.1



(a) (b)

Figure 4.8: The average error across the aspect ratios and ON-pixel densities of the words. (a) Histogram of the ratios of the mis-recognized words with respect to the aspect ratio (normalized width) of the words, and (b) plot of the ratios with respect to the ON-pixel density.

Table 4.8: The results of bi-script recognition using centered and normalized Gabor & DCT feature vectors. The upper triangular matrix tabulates the results of DCT while the lower triangular part, those of Gabor features, with NNC in both cases.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	99.0	97.4	95.2	97.8	97.0	96.8	97.4	97.8	97.5	98.4
EN	94.1	–	98.2	98.9	98.6	98.4	98.7	98.6	98.2	96.4	98.9
GU	92.0	97.2	–	96.3	98.1	98.0	97.8	97.1	97.4	97.1	98.0
HI	90.1	98.2	94.7	–	98.2	97.8	96.8	94.2	97.6	97.8	98.6
KA	93.0	96.0	95.3	95.1	–	93.5	97.6	98.7	96.5	88.4	97.4
MA	89.7	96.8	94.1	92.7	94.3	–	97.2	98.4	92.7	92.7	97.0
OD	86.5	94.8	92.3	92.2	93.0	85.1	–	97.1	97.9	97.2	98.4
PU	84.7	96.3	93.3	89.9	95.5	89.5	87.2	–	98.0	97.8	98.7
TA	87.5	95.1	93.6	93.1	94.5	90.2	89.6	90.1	–	94.1	95.8
TE	86.1	94.8	93.2	90.4	88.0	85.6	82.7	87.7	89.4	–	96.0
UR	95.2	97.1	96.3	97.8	95.6	97.6	96.5	97.1	95.0	95.2	–

Table 4.9: The results of centering and normalizing the features in bi-script recognition experiments with Gabor & DCT features. The upper and lower triangular matrices show results for Gabor and DCT features, respectively with the LDC.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	50.3	52.8	49.7	48.5	52.2	49.1	48.9	54.0	51.4	49.4
EN	47.9	–	48.3	48.3	50.1	49.1	48.1	53.1	48.5	46.3	49.2
GU	50.4	50.9	–	48.4	47.5	48.1	50.3	49.8	49.8	50.0	46.1
HI	49.7	50.5	49.3	–	50.4	50.9	49.0	50.3	51.4	49.2	51.4
KA	49.0	50.7	51.4	50.7	–	52.1	51.8	49.0	49.3	51.4	50.7
MA	49.0	48.0	52.0	49.5	49.2	–	52.1	47.8	49.1	47.6	49.3
OD	48.9	48.7	50.9	49.8	53.5	49.2	–	50.9	48.7	46.9	49.0
PU	49.4	49.9	51.4	50.9	49.0	47.7	49.5	–	51.5	51.3	52.0
TA	47.7	50.5	50.2	47.5	50.4	50.0	47.6	50.0	–	49.7	49.1
TE	48.7	51.3	50.2	47.5	51.2	50.7	48.6	49.8	51.1	–	52.3
UR	49.9	51.7	51.3	49.8	52.9	50.8	50.1	53.4	50.1	49.9	–

performed comparably with NNC, we ignored SVM classifier in this experiment.

Out of the 36 features, a few have better discriminating power than others for specific bi-script cases. This is already demonstrated in Fig. 4.5(c), where the divergence values for some features are higher than the rest, indicating possible better discrimination with those features. Contribution of other features may be insignificant or adverse (*i.e.*, they may confuse rather than contribute). The features with insignificant contribution add to the computation. So, removal of both these kinds of features improves the accuracy and computation time. We have employed a sequential floating feature selection algorithm, proposed by Pudil *et al.* (69), to choose the best-minimal subset of features that delivers the highest accuracy.

In bi-script recognition, the optimal subset of features is always case specific. In other words, for different bi-script cases, different filters need to be chosen for the best performance. We define a accuracy-feature ratio, F_c, which represents the average contribution of each of the features to the accuracy of each individual bi-script case.

$$F_c = \frac{Highest\ attainable\ accuracy}{Number\ of\ features\ required\ to\ attain\ this\ accuracy} \tag{4.6}$$

Let's say, we need 4 features to obtain the highest attainable accuracy of 96% for a sample bi-script case (for a given feature-classifier combination), the F_c for this case would be 24. A higher F_c value implies that it is possible to attain the maximum accuracy for that case with minimal number of features. Thus, higher this value, better it is. Table

Table 4.10: The results of classification using NNC with a selected subset of features. The results with Gabor and DCT features are tabulated in the lower and upper triangular matrices, respectively.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	99.6	99.5	98.5	99.4	99.0	99.2	98.6	99.2	99.2	99.5
EN	99.8	–	99.4	99.8	99.5	98.7	99.2	99.5	99.2	98.4	99.4
GU	99.4	99.8	–	99.5	99.7	99.2	98.9	99.5	98.4	98.6	98.6
HI	99.0	99.8	99.4	–	99.6	99.4	99.5	97.7	99.5	99.6	99.6
KA	99.3	99.7	99.5	99.5	–	97.6	99.6	99.5	99.2	98.1	99.5
MA	99.2	99.4	99.2	99.3	99.3	–	98.2	99.4	96.4	97.3	99.6
OD	99.8	99.4	99.0	99.7	98.4	99.1	–	98.9	98.5	98.7	99.2
PU	98.9	99.3	98.7	97.5	99.5	97.9	98.9	–	99.2	99.5	99.6
TA	99.1	99.7	98.8	99.4	99.6	98.2	98.8	98.8	–	97.9	99.0
TE	99.8	99.8	99.5	99.7	97.3	99.2	98.3	99.7	99.7	–	98.8
UR	99.7	99.8	99.5	99.8	99.1	99.7	99.0	99.6	99.5	99.2	–

Table 4.11: The number of features required on a case by case basis for delivering the best output accuracy. The maximum possible output accuracies are presented in Tab. 4.10.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	25	24	24	25	23	22	22	25	23	22
EN	25	–	26	25	26	23	24	25	25	24	25
GU	25	26	–	25	25	23	24	27	25	24	23
HI	26	26	25	–	25	25	24	24	23	21	23
KA	25	25	24	24	–	26	23	24	28	24	24
MA	24	23	23	24	25	–	24	26	27	25	23
OD	23	23	25	25	23	24	–	25	25	24	23
PU	22	24	26	24	23	25	25	–	25	26	24
TA	24	25	26	22	27	27	24	25	–	26	25
TE	23	24	24	22	25	25	24	25	25	–	25
UR	22	25	24	23	24	23	24	25	26	25	–

4.10 presents the maximum obtained accuracies, with the selected best minimal subset of features, for various bi-script cases. The combinations of NNC with Gabor and DCT yield the accuracies presented in the lower and upper triangular parts of this matrix, respectively. The minimum number of features required to obtain these accuracies, is listed in Table 4.11. Using Tables 4.10 and 4.11, the best F_c value is 4.53 for Bengali-Urdu and Hindi-Telugu cases for the Gabor features and NNC combination. Similarly, for DCT, 4.74 is the maximum F_c for Hindi-Telugu case. It may also be noted from the comparison of Tables 4.4 and 4.10 that in the case of the Kannada-Telugu bi-script, all the 36 features yield only 89.2% while a subset of 25 features improves this accuracy to 97.3% (Gabor-NNC).

4.4.3 Tri-script Recognition

A number of official documents in India contain three languages, namely, the state's official language, Hindi and English. We refer to such a combination of three scripts as the triplet of the state. Figure 4.1(b) presents a train reservation form containing the Kannada, Devanagari and Roman scripts. In the second series of experiments, we discriminate the scripts in all such triplets. Here, Devanagari & Roman are common to all the triplets while the script of the state varies. The results of such experiments are presented in Table 4.12. In this table, the μ and σ columns report the average and standard deviation of the recognition rates for all the nine triplets presented in the same row.

Table 4.12 shows that the maximum average accuracy obtained is 98.2%. This result occurs for Gabor features with NN and SVM classifiers. However, the Gabor-NNC combination is more consistent and results in a lower standard deviation ($\sigma_{NNC} = 1.9$ while $\sigma_{SVM} = 2.1$). So, the combination of the Gabor feature with the NNC has a slight edge. A prior knowledge of the triplet, however, may decide in favour of a different feature-classifier combination for a triplet-specific optimal result.

We have compared our results with the earlier reported results. Pal & Chaudhuri (39) have reported an accuracy of 97.2% for recognition of Devanagari words from the triplet of Telugu. Padma & Nagabhushan (41) have reported 97% for the same from the triplet involving Kannada script. We achieve an average of 99.4% for the recognition involving these triplets using the Gabor-NNC combination, with a much larger dataset, while the Gabor-SVM combination has yielded 99.6%.

Table 4.12: Tri-Script recognition accuracies (common scripts in all experiments are Hindi and English).

		BE	GU	KA	MA	OD	PU	TA	TE	UR	Mean	STD
GABOR	NNC	96.5	99.0	99.4	98.4	98.8	93.8	98.8	99.4	99.4	98.2	1.9
	LDC	94.5	97.5	97.9	96.9	98.0	93.4	97.3	98.3	98.3	96.9	1.8
	SVM	97.0	99.2	99.6	98.6	99.0	92.9	98.8	99.2	99.7	98.2	2.1
DCT	NNC	96.5	99.1	98.3	97.9	98.7	95.6	97.8	97.9	99.2	97.9	1.2
	LDC	82.2	86.1	85.5	82.4	83.6	77.6	85.5	83.0	87.2	83.7	2.9
	SVM	97.1	99.5	98.7	97.6	99.0	96.3	98.0	98.2	99.1	98.1	1.0

4.4.4 Multi-script Recognition

Since our identification scheme depends on statistical rather than structural features, we have an advantage that we could consider any number of scripts together and identify them. On the observation that our feature – classifier combinations are delivering us very good recognition accuracy, we tried to identify the scripts in a multi-script scenario, involving all the 11 earlier mentioned Indian scripts. Here, every test sample is compared with the reference samples from all the classes. We compare the efficacies of the Gabor and the DCT features, with the three classifiers, in Tables 4.13 and 4.14, respectively. In these tables, each row corresponds to one of the scripts under test. The columns lists the percentage of words that are assigned to the scripts represented by them. The diagonal entries in the matrices present the correct recognition accuracies for the scripts, for individual feature-classifier combinations.

The average recognition accuracy is evaluated by considering the correct recognition accuracy of each of the presented 11 scripts. The average and standard deviation of these accuracies for all combinations are presented in Table 4.15, for easier comparison. This table shows that the Gabor-SVM combination has an edge, with an average accuracy of 94.8% followed by the DCT-SVM accuracy of 93.8%. Similarly, the Gabor feature leads with a performance of of 83.1% using the LDC over the DCT (49.8%). On the contrary, it is the DCT which better combines with the NNC to generate 99.1% while the Gabor with the same classifier results in a slightly lower performance of 89.4%. This slight average improvement of the DCT over the Gabor features comes from the improvements in the classification accuracies of Odiya (Gabor – 84.5%; DCT – 94.2%), Devanagari (Gabor – 85.6%; DCT – 90.6%) and Gujarati (Gabor – 90.2%; DCT – 94.7%) scripts. This is justified from the study of the neighborhood analysis matrices presented in Tables 4.2 and 4.3. However, it may still be advisable to use the Gabor feature vector for multi-script classification and avoid linear discriminant classifier.

Table 4.13: Multi-Class recognition accuracies for Gabor features with the NNC, LDC & SVM classifiers (11 scripts are used). Each row corresponds to one of the scripts under test. The numbers in a row, under each column corresponds to percentage of words of this script that have been assigned to each of the scripts represented by the columns.

		BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
NNC	BE	92.1	0.0	0.4	3.0	0.4	0.7	0.9	0.9	1.0	0.5	0.0
	EN	0.0	97.8	0.1	0.2	0.0	0.7	0.5	0.4	0.2	0.0	0.1
	GU	0.4	0.1	90.2	0.3	0.8	0.8	5.5	0.2	0.4	0.6	0.6
	HI	4.7	0.0	0.6	85.6	0.3	0.5	0.3	7.3	0.4	0.1	0.1
	KA	0.2	0.0	0.3	0.2	88.7	1.1	1.3	0.2	0.1	7.1	0.8
	MA	0.6	0.2	0.4	0.4	0.8	88.2	4.5	2.3	2.3	0.3	0.1
	OD	0.7	0.2	2.8	0.2	1.8	6.3	84.5	0.7	1.3	1.3	0.3
	PU	1.1	0.2	0.1	7.7	0.2	2.5	0.7	86.2	1.1	0.1	0.1
	TA	0.7	0.4	0.2	0.4	0.2	3.3	1.7	1.3	91.5	0.1	0.3
	TE	0.5	0.0	0.3	0.0	12.7	0.5	1.2	0.2	0.2	83.8	0.6
	UR	0.1	0.2	0.5	0.1	2.2	0.5	0.8	0.1	0.5	0.5	94.5
LDC	BE	89.4	0.1	0.8	3.8	0.9	1.0	0.6	1.1	1.4	0.4	0.6
	EN	0.0	95.2	0.8	1.1	0.0	1.0	0.4	0.3	0.7	0.0	0.5
	GU	0.8	2.1	85.7	0.1	0.3	0.7	4.8	0.0	1.6	0.7	3.2
	HI	8.2	2.7	1.0	66.0	0.2	0.9	0.4	17.4	1.8	0.5	0.8
	KA	0.1	0.4	0.8	0.9	71.5	0.8	3.6	0.7	0.7	14.6	5.7
	MA	0.1	3.2	2.2	1.0	0.7	82.4	1.5	3.1	4.3	1.0	0.4
	OD	1.7	0.8	10.0	0.2	0.6	1.8	77.6	0.3	2.6	1.9	2.4
	PU	1.8	0.8	0.7	7.4	0.2	1.6	0.5	84.3	2.0	0.2	0.4
	TA	0.5	0.6	0.3	1.8	0.3	3.8	4.7	0.5	85.9	0.3	1.2
	TE	0.1	0.2	0.6	0.3	12.2	0.5	1.0	0.2	0.7	82.8	1.5
	UR	0.0	0.2	0.8	0.2	2.5	0.3	0.3	0.2	0.5	1.4	93.6
SVM	BE	96.2	0.0	0.2	1.7	0.2	0.3	0.4	0.3	0.4	0.3	0.0
	EN	0.0	98.2	0.0	0.5	0.0	0.4	0.4	0.3	0.1	0.0	0.0
	GU	0.2	0.1	95.5	0.2	0.3	0.2	2.5	0.1	0.2	0.4	0.3
	HI	1.6	0.0	0.3	93.3	0.1	0.2	0.2	4.0	0.2	0.1	0.0
	KA	0.1	0.0	0.1	0.4	93.3	0.4	0.5	0.2	0.2	4.3	0.5
	MA	0.2	0.2	0.2	1.0	0.4	93.6	1.5	1.2	1.3	0.4	0.1
	OD	0.2	0.1	1.4	0.5	0.6	1.3	94.0	0.3	0.5	0.8	0.2
	PU	0.4	0.1	0.1	3.4	0.1	1.1	0.3	93.8	0.6	0.0	0.1
	TA	0.2	0.2	0.0	0.8	0.1	1.5	0.9	0.8	95.2	0.1	0.1
	TE	0.2	0.0	0.2	0.3	5.3	0.4	0.7	0.1	0.1	92.3	0.4
	UR	0.0	0.1	0.2	0.2	0.6	0.1	0.4	0.0	0.2	0.4	97.9

Table 4.14: Multi-Class recognition accuracy using 36 DCT coefficients with the NNC, LDC & SVM classifiers (11 scripts are used). Each row corresponds to one of the scripts under test. The numbers in a row, under each column corresponds to percentage of words of this script that have been assigned to each of the scripts represented by the columns.

		BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
NNC	BE	92.8	0.2	0.3	3.6	0.2	0.6	0.4	1.2	0.5	0.1	0.1
	EN	0.1	97.7	0.1	0.0	0.2	0.9	0.4	0.1	0.3	0.2	0.0
	GU	0.4	0.7	94.7	0.2	0.1	0.6	1.4	0.1	1.1	0.3	0.5
	HI	4.2	0.2	0.1	90.6	0.2	0.5	0.3	3.5	0.5	0.1	0.0
	KA	0.2	1.9	0.1	0.2	89.6	3.4	0.5	0.1	0.5	3.5	0.0
	MA	0.4	2.1	0.4	0.2	0.6	90.0	1.3	0.1	4.1	0.7	0.0
	OD	0.4	0.7	0.5	0.2	0.3	1.7	94.2	0.5	1.2	0.2	0.1
	PU	2.0	0.3	0.3	5.6	0.2	0.7	1.1	89.2	0.4	0.1	0.0
	TA	0.5	2.9	0.8	0.3	0.2	5.7	1.1	0.2	87.7	0.4	0.3
	TE	0.4	2.6	1.0	0.1	6.1	3.8	1.0	0.2	1.6	82.2	0.8
	UR	0.1	0.3	0.5	0.1	0.3	0.4	0.5	0.1	1.6	2.2	93.0
LDC	BE	51.3	0.2	1.1	26.0	4.8	1.9	1.1	8.9	3.7	0.1	1.0
	EN	1.1	68.9	2.7	3.2	5.0	1.0	3.1	1.0	12.5	0.3	1.1
	GU	7.8	7.9	62.2	1.9	2.7	2.0	1.1	1.0	8.2	0.6	4.6
	HI	6.8	0.3	2.0	49.8	10.0	2.5	1.0	22.5	4.2	0.1	0.7
	KA	5.0	5.3	2.8	0.9	70.6	2.7	1.7	3.2	2.8	3.0	2.0
	MA	13.9	11.2	4.8	3.2	14.1	23.5	5.2	2.8	14.2	1.6	5.6
	OD	7.1	10.3	7.6	6.3	11.6	4.3	32.6	7.6	7.4	0.5	4.7
	PU	5.9	3.2	3.3	16.3	5.5	1.4	2.3	56.2	4.4	0.2	1.1
	TA	2.7	9.4	5.5	2.1	1.8	6.9	2.3	1.8	56.7	0.4	10.5
	TE	10.5	12.1	7.1	1.2	27.0	6.7	2.2	2.2	8.0	15.4	7.7
	UR	3.0	5.9	7.9	0.9	2.0	1.8	2.3	1.1	12.9	1.1	61.0
SVM	BE	92.6	0.0	0.3	3.7	0.1	0.2	0.2	1.0	0.5	0.3	1.0
	EN	0.0	96.6	0.1	0.0	0.2	0.4	0.2	0.0	0.3	0.5	1.6
	GU	0.2	0.1	95.4	0.1	0.1	0.3	0.7	0.1	0.7	0.4	2.0
	HI	1.0	0.0	0.1	94.3	0.1	0.1	0.1	2.3	0.4	0.1	1.5
	KA	0.1	0.6	0.1	0.1	90.3	0.7	0.1	0.1	0.4	4.2	3.4
	MA	0.2	1.0	0.3	0.5	1.0	84.4	0.6	0.3	7.7	1.1	3.0
	OD	0.1	0.2	0.6	0.1	0.2	0.8	94.3	0.4	1.1	0.4	2.0
	PU	0.6	0.0	0.3	4.0	0.1	0.1	0.4	92.1	0.5	0.2	1.6
	TA	0.1	0.3	0.4	0.2	0.1	1.5	0.4	0.1	93.6	0.4	2.9
	TE	0.1	0.5	0.4	0.0	2.0	0.9	0.2	0.1	0.9	91.0	3.8
	UR	0.0	0.0	0.4	0.0	0.1	0.0	0.1	0.0	0.4	0.7	98.4

Table 4.15: The statistics of the multi-script recognition accuracies. These statistics are generated from the accuracy of classification for each class, *i.e.*, considering only the diagonal elements of the matrices in tables 4.13 and 4.14.

	Gabor			DCT		
	NNC	LDC	SVM	NNC	LDC	SVM
Mean(μ)	89.4	83.1	94.8	91.1	49.8	93.8
STD(σ)	4.4	8.8	1.9	4.1	18.3	3.9

Using Gabor features, the worst performance of 83.8% is for Telugu, and the best performance is for English, namely, 97.8%. Similar performances are repeated with the SVM as the classifier. Thus, in a multi-script scenario, it makes better sense to separate the English words first, and the remaining scripts in a hierarchical manner, subsequently.

4.4.5 A Blind Script Identifier

It is observed from Tables 4.13 and 4.14 that scripts tend to group among themselves. Close observation of Tables 4.2 and 4.13, reveal that the scripts in any such group, share a large number of neighbors. Similar observation is made from Tables 4.3 and 4.14. Such scripts should be considered as a group at an initial level and then discriminated among themselves. So, a multi-level identification strategy is proposed. It is also possible to employ different feature-classifier combinations at different nodes of such a hierarchical classification system. Then, the best-minimal subset of features can be used at each node for maximizing the efficiency of this system.

A two level strategy is employed towards development of such a blind script identifier. A combined study of Tables 4.2, 4.3, 4.13 and 4.14 helps to devise a robust grouping of the scripts. Four groups are formed from the eleven scripts based on their visual similarity and class inter-mixing in the feature spaces. The four groups are: (i) G1 – Devanagari, Bengali and Punjabi, (ii) G2 – Gujarati, Malayalam, Odiya & Tamil, (iii) G3 – Kannada and Telugu, and (iv) G4 – Roman and Urdu. A test pattern is initially assigned to one of these groups and gets a script identity, subsequently. Table 4.16 tabulates the efficacy of such a grouping strategy, with various feature-classifier combinations. The upper and lower halves of this table give the grouping accuracy with the Gabor and DCT features, respectively. The μ and σ represent the average and standard deviation of the correct grouping (diagonal elements) for all the four groups. It is observed that the SVM has marginally out-performed the NNC and the accuracies with LDC are the lowest for both Gabor and DCT features. The Gabor–SVM combination (result shown in bold) yields the maximum average grouping accuracy as well as the correct classification for each of

Table 4.16: Grouping accuracy of the eleven scripts with the various features & classifiers. The upper part of the table gives the results with Gabor and the lower part with the DCT features.

	NNC				LDC				SVM			
	G1	G2	G3	G4	G1	G2	G3	G4	G1	G2	G3	G4
G1	96.2	3.1	0.5	0.2	93.4	5.3	0.5	0.8	98.1	1.6	0.2	0.1
G2	2.0	96.0	1.5	0.5	2.3	93.7	1.6	2.4	1.5	97.6	0.7	0.2
G3	0.6	2.5	96.2	0.7	1.0	9.6	88.0	1.4	0.7	1.5	97.4	0.4
G4	0.5	1.9	1.4	96.3	0.8	8.3	5.4	85.5	0.8	1.0	0.4	97.8
(μ, σ)	(96.2, 0.1)				(90.2, 4.1)				**(97.7, 0.3)**			
G1	97.5	1.9	0.3	0.3	81.5	10.9	7.1	0.5	98.3	1.3	0.2	0.2
G2	0.9	96.6	0.7	1.8	14.5	75.5	4.5	5.5	0.7	97.5	0.9	0.9
G3	0.6	6.0	90.7	2.7	11.8	33.5	49.8	4.9	0.5	3.0	95.2	1.3
G4	0.2	2.8	1.5	95.5	6.8	42.2	6.4	44.6	0.1	2.2	1.2	96.5
(μ, σ)	(95.1, 3.0)				(62.9, 18.4)				(96.9, 1.3)			

the groups.

On similar lines, the intra-group classification accuracies for the four groups with various feature-classifier combinations are presented in Tables 4.17, 4.18. 4.19 & 4.20 for G1, G2, G3 and G4 groups, respectively. Again, the upper and lower parts of these tables present accuracies with Gabor and DCT features, respectively. The notations μ and σ are consistent with the ones in the previous paragraph. It is observed that the SVM classifier has, for both Gabor and DCT features, produced comparable performance with NNC for all the groups. However, the best performances are group specific. Gabor–SVM combination leads in G1 and G2 while Gabor in combination with NNC has the best results for G4. DCT–SVM is the winner for G3. This performance of DCT over the Gabor features is expected, as DCT has better accuracies for the bi-script scenario involving members of this group. This can easily be located in Tables 4.4, 4.5 and 4.6. With these results, a 2-layered hierarchical blind script identifier is designed. In this tree, the initial grouping is accomplished with Gabor-NNC combination while the second level has four nodes, with each node using the best feature-classifier combination for that group. This tree structured blind recognizer, with the selected feature-classifier combination and associated accuracies at various levels, is shown in figure 4.9.

When the results of the various nodes of this tree are compared with the results presented in Table 4.13, it appears that, the individual accuracies presented here are lower than the mono-level classification system. This is because the results presented here are statistical in nature. Moreover, a proper choice of the different parameters for a SVM classifier plays a vital role for its accuracy. As is already mentioned in section

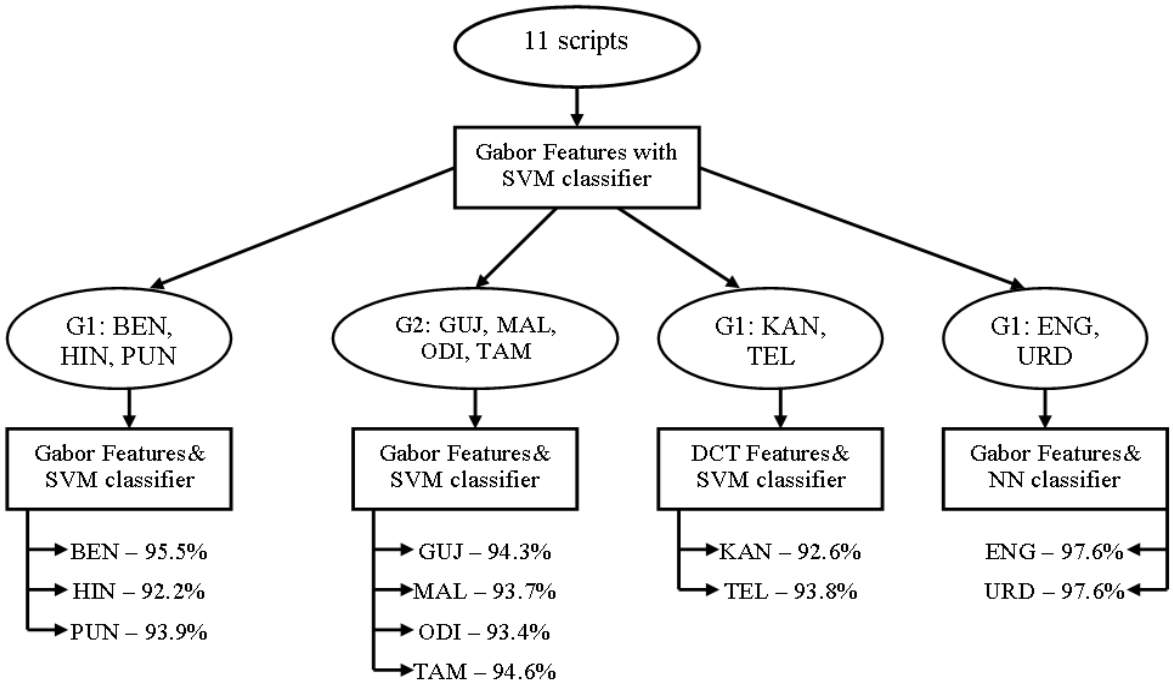


Figure 4.9: A hierarchical blind script classification system. The numbers shown with the script names are the final recognition accuracies in percentages.

Table 4.17: Intra-group accuracies of classification for group G1 using various classifiers. (Gabor features – upper half; DCT features – lower half.)

	NNC			LDC			SVM		
	BE	HI	PU	BE	HI	PU	BE	HI	PU
BE	95.5	3.3	1.2	95.1	3.6	1.3	97.3	2.1	0.6
HI	5.2	87.0	7.8	6.4	81.5	12.1	1.8	94.0	4.2
PU	1.5	8.2	90.3	0.6	5.5	93.9	0.5	3.8	95.7
(μ, σ)	(90.9, 4.3)			(90.2, 7.5)			(95.7, 1.7)		
BE	94.5	4.0	1.5	75.2	19.0	5.8	93.5	4.7	1.8
HI	4.7	91.6	3.7	18.6	61.1	20.3	1.2	95.6	3.2
PU	2.7	6.3	91.0	10.0	15.4	74.6	0.8	4.7	94.5
(μ, σ)	(92.4, 1.9)			(70.3, 8.0)			(94.5, 1.1)		

Table 4.18: Intra-group accuracies of classification for group G2 using the Gabor features (upper part of the table) with various classifiers. The efficacy of DCT features is presented in the lower part for easy comparison.

	NNC				LDC				SVM			
	GU	MA	OD	TA	GU	MA	OD	TA	GU	MA	OD	TA
GU	92.4	1.0	6.1	0.5	90.3	1.7	6.9	1.1	96.6	0.3	2.8	0.3
MA	0.4	91.9	5.2	2.5	1.2	93.0	2.4	3.4	0.2	96.0	0.8	2.0
OD	3.2	6.9	88.4	1.5	7.4	3.1	87.4	2.1	1.6	1.7	95.7	1.0
TA	0.3	3.8	2.1	93.8	0.7	5.0	3.5	90.8	0.1	1.9	1.1	96.9
(μ, σ)	(91.6, 2.3)				(90.4, 2.3)				(96.3, 0.5)			
GU	96.2	0.8	1.7	1.3	70.0	12.1	4.0	13.9	97.7	0.4	0.8	1.1
MA	0.5	93.2	1.7	4.6	6.8	62.1	14.6	16.5	0.6	87.2	1.1	11.1
OD	0.7	2.0	95.8	1.4	7.5	13.8	69.9	8.8	0.9	1.1	96.3	1.7
TA	1.1	6.6	1.4	90.9	9.0	13.4	6.4	71.2	0.8	1.5	0.5	97.2
(μ, σ)	(94.0, 2.5)				(68.3, 4.2)				(94.6, 5.0)			

Table 4.19: Accuracies of classification using the Gabor (upper part) and DCT (lower part) features with the 3 different classifiers for the scripts within group G3.

	NNC		LDC		SVM	
	KA	TE	KA	TE	KA	TE
KA	92.3	7.7	96.7	3.3	93.1	6.9
TE	13.8	86.2	10.9	89.1	6.0	94.0
(μ, σ)	(89.2, 4.3)		(92.9, 5.3)		(93.6, 0.6)	
KA	95.1	4.9	76.5	23.5	95.1	4.9
TE	8.3	91.7	26.9	73.1	3.7	96.3
(μ, σ)	(93.4, 2.4)		(74.8, 2.4)		(95.7, 0.8)	

Table 4.20: Efficacy evaluation for combinations of Gabor and DCT features with various classifiers for member scripts of group G4. Upper half of the table shows the results for Gabor while the lower part presents the efficacy of DCT features.

	NNC		LDC		SVM	
	EN	UR	EN	UR	EN	UR
EN	99.8	0.2	99.2	0.8	99.9	0.1
UR	0.2	99.8	0.3	99.7	0.5	99.5
(μ, σ)	(99.8, 0.0)		(99.5, 0.4)		(99.7, 0.3)	
EN	99.0	0.1	96.4	3.6	99.2	0.8
UR	1.2	98.8	13.6	86.3	0.1	99.9
(μ, σ)	(98.9, 0.1)		(91.4, 7.1)		(99.6, 0.5)	

4.3, the standard deviation for the Gaussian kernel of the network is taken from the standard deviation of the involved training dataset. This, though broadly acceptable, may not be the best one. Since extensive fine tuning of these parameters could not be done, there is always a possibility for improvement of the obtained results. Besides, a selected subset of features may deliver a better accuracy at the nodal points than what is presented. This is amply demonstrated by a comparison of the results of intra-group classification to resolve the issues of Kannada and Telugu presented in Table 4.19 and the feature selected results, presented in Table 4.10. The selected feature subset improves the accuracy to 97.3% (table 4.10) from 89.2% (in Table 4.4 & Table 4.19), for the Gabor-NN combination. The average accuracy obtained by the proposed blind identifier is 94.1%.

4.4.6 Line & Block Level Script Recognition

Script recognition algorithms that consider mono-script paragraph/block or line images are already in existence, for Indian scripts. A paragraph consists of may lines and, similarly, a line of text consists of many words. Assumption of a mono-script paragraph makes it necessary for all its constituent lines and words to be from the same script. Thus, script identification at the level of paragraph is sufficient for lines and words. Besides, it reduces the search space of the algorithm, as only one search decides the script at all levels of text existence. For example, let a paragraph consist of five lines and forty words. This implies one search for paragraph, five searches for lines and forty searches for words, for the script to be identified at the respective levels. Thus, it is advantageous to identify the script at the highest level of its existence as text-structure[3].

On the contrary, the recognition of the script at higher levels may adversely affect the performance of the system. In the paragraph with five lines and forty words, a mis-recognition at a paragraph level translates to five and forty mis-recognition at the line and word levels, respectively. If recognition rates at all these different levels of text existence are comparable, (Let's say we have 98% accuracy of recognition for all the three levels, paragraphs, lines and words.), then more mistakes are made if identification is at higher levels. Moreover, with prior partial information indicating the constancy of the script for lines or paragraphs, it is possible to make probabilistic decisions. This is why we go for recognition of the script at the lowest level, *i.e.*, words. Besides, it considers cases where the script changes with words where script identification at other levels is irrelevant.

The proposed script recognition model is already observed to deliver good accuracy

[3]The different levels of text structure are characters, words, lines, paragraphs, pages and documents. The lowest level of variation of script in this hierarchy is word. The script cannot vary at a sub-word level, *i.e.*, the characters in a given word can't be from different scripts.

Table 4.21: Script recognition accuracy at the line level, using the LineSet, is presented in the upper triangular part of the matrix, using the Gabor-NNC combination. The lower triangular part shows the word recognition rates using the WordSet.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	99.5	98.7	97.2	99.8	98.7	99.2	98.7	99.0	99.4	99.7
EN	99.8	–	99.2	99.7	99.8	99.3	99.4	99.2	99.3	99.4	99.5
GU	98.9	99.7	–	98.4	98.6	97.7	95.6	99.0	98.9	97.8	98.5
HI	95.2	99.6	99.2	–	99.5	99.0	98.9	95.7	98.8	99.0	99.4
KA	99.3	99.9	98.8	99.5	–	99.3	99.4	99.3	99.6	94.3	99.0
MA	98.7	99.1	98.7	98.7	98.5	–	95.8	98.5	97.4	97.9	98.6
OD	98.5	99.2	94.7	99.1	97.6	93.4	–	98.7	97.7	97.9	98.9
PU	98.2	99.2	99.4	91.5	99.5	96.6	98.6	–	98.3	98.9	99.2
TA	98.5	99.4	99.1	99.0	99.5	96.1	97.4	97.9	–	99.7	99.4
TE	99.0	99.8	99.0	99.6	89.2	98.9	97.8	99.6	99.5	–	98.1
UR	99.6	99.7	98.8	99.6	98.1	99.2	98.8	99.6	99.2	98.7	–

of recognition at the word level for various configurations of its existence, namely, bi-script, tri-script and multi(eleven)-script. We want to test the efficacy of this model for recognition accuracy of lines and paragraphs. This is accomplished in two stages: (i) accuracy evaluation using both the training and test sets at the same paragraph or line level, and (ii) accuracy evaluation using the word training set and test sets from lines and paragraphs. While, the former proves the robustness of the proposed system at every individual level of text structure (words, lines, paragraphs), the latter one makes the script identification system independent of such levels. We considered the Gabor features for this experiment based on its consistent performance. The positioning of the feature vectors of words, lines and paragraphs is examined in the feature space by NNC. Thus, it is sufficient to observe results of this combination, for the proof of concept.

A database of 300 paragraph & 2000 lines images is created from the scanned pages. A line consists of at least one word while a paragraph contains a minimum of 2 lines. Two new training sets are created with features from lines and paragraphs. The three training sets are named as: (i) WordSet – training set of 7000 words, (ii) LineSet – training set of 1000 line images and (iii) ParaSet – 150 paragraphs from every script. When the accuracy for lines images is evaluated, another 1000 lines are tested against the LineSet. Similar strategy is followed for paragraph images.

Table 4.21 compares the results of line and word level script recognition. It is observed that the word level accuracy ($\mu = 98.4\%$, $\sigma = 2.0$) is comparable with accuracy for the line images ($\mu = 98.6\%$, $\sigma = 1.1$). This is in spite of the fact that a line contains more script related textural information than words. However, for Kannada-Telugu bi-script

Table 4.22: Script recognition accuracy at the line level, using the WordSet, for the Gabor-NNC combination. The upper and lower triangular parts of the table show the accuracies of the line images without and with blank columns present, respectively.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	99.1	97.4	91.9	99.4	98.6	98.1	98.4	98.5	99.3	99.8
EN	93.4	–	99.4	99.6	99.7	94.5	97.2	98.9	98.3	99.5	99.6
GU	83.9	95.6	–	98.7	98.9	97.8	92.5	98.9	97.2	98.0	98.2
HI	73.0	92.8	89.7	–	98.6	98.8	98.9	84.3	97.9	98.0	99.2
KA	88.5	98.6	92.6	91.0	–	95.0	91.7	99.5	98.6	87.7	99.6
MA	88.3	93.0	91.1	92.9	90.5	–	90.2	95.1	97.0	98.2	99.2
OD	86.8	91.8	74.2	92.9	87.2	85.0	–	99.4	98.1	95.4	97.8
PU	83.5	92.2	88.7	67.5	89.9	91.6	92.1	–	98.9	98.7	99.0
TA	89.7	96.6	85.3	90.6	92.7	88.9	88.1	91.2	–	98.8	97.9
TE	91.3	98.0	93.5	91.4	78.8	91.9	90.4	90.6	93.5	–	97.6
UR	88.1	97.2	88.0	89.6	94.2	91.2	81.9	84.8	89.6	92.2	–

case, lines (94.3%) provide better accuracy than words (89.2%). Similar patterns are observed for Hindi-Punjabi and Bengali-Hindi cases, as well.

The upper part of Table 4.21 shows the accuracies achieved when line images are tested against the LineSet. Evaluating the accuracy of these lines against the WordSet will take this exercise to the next level. A comparable accuracy with this experiment would mean that one training set is enough for both words or lines. The lower part in Table 4.22 gives these results, which show a decline in accuracy ($\mu = 89.4$, $\sigma = 5.8$) for line images tested against the WordSet as compared to the LineSet ($\mu = 98.6\%$, $\sigma = 1.1$).

Investigation of the line images reveals the existence of large gaps between some words in a line. A high correlation exists between such gaps and the recognition accuracy, which propels us to explore the effect of removing the blank columns from a line image. Presence of a large gap disturbs the textural structure of a script and hence leads to greater misrecognition. The results of this experiment are presented in the upper part of Table 4.22. It is observed that the removal of blank columns has improved the accuracy ($\mu = 97.4$, $\sigma = 3.1$) which is comparable with the accuracies obtained with testing of lines and words with the LineSet and WordSet, respectively. This suggests removal of the blank columns from lines of text before recognizing the script of these lines against the WordSet. This avoids the formation of separate training sets for two different levels of script recognition.

Similar studies for paragraph text was carried out and the results are listed in Tables 4.23 and 4.24. Table 4.23 presents the paragraph level script recognition accuracies with ParaSet and WordSet in its upper & lower triangular parts, respectively. The result of testing the paragraphs using the WordSet ($\mu = 79.0$, $\sigma = 11.8$) is worse than testing with

Table 4.23: Script recognition accuracy for paragraph images. The upper and lower triangular parts of the table use the ParaSet and WordSet, respectively.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	99.7	97.7	91.7	99.0	97.3	96.3	96.7	98.0	98.7	98.0
EN	90.8	–	98.7	99.0	99.7	99.3	99.3	99.0	98.0	99.0	99.7
GU	72.0	84.2	–	97.7	99.7	96.7	91.3	99.7	99.3	99.0	98.7
HI	71.3	90.3	83.7	–	99.0	96.3	95.3	95.3	98.3	97.7	98.7
KA	78.8	97.5	89.6	85.5	–	99.0	98.0	99.0	100	92.3	98.0
MA	84.0	84.8	84.7	89.5	82.2	–	91.0	97.7	96.7	98.3	98.0
OD	78.7	76.2	65.7	89.3	74.3	73.5	–	97.7	97.3	98.0	97.0
PU	56.0	70.2	60.5	49.8	60.0	84.5	70.2	–	98.7	98.7	97.7
TA	73.9	83.3	81.3	80.5	83.5	77.0	58.7	57.2	–	99.3	99.3
TE	88.0	98.2	96.2	90.2	81.5	84.3	78.3	62.8	90.2	–	99.0
UR	75.3	95.2	86.3	84.0	94.3	83.3	70.7	53.8	68.8	89.4	–

Table 4.24: Script recognition accuracy for paragraph images with the Gabor-NNC combination using the WordSet. The upper and lower triangular parts of the table are the results with and without the removal of the gaps between lines of text.

	BE	EN	GU	HI	KA	MA	OD	PU	TA	TE	UR
BE	–	97.8	84.7	76.7	93.8	89.7	92.0	79.8	85.8	96.0	89.3
EN	90.8	–	96.0	94.7	98.7	93.3	95.2	94.3	94.8	98.3	97.2
GU	72.0	84.2	–	93.0	94.8	93.8	81.2	87.3	93.0	97.0	95.2
HI	71.3	90.3	83.7	–	94.7	94.3	94.3	61.5	91.8	95.3	93.2
KA	78.8	97.5	89.6	85.5	–	94.7	94.3	91.7	96.5	85.0	97.2
MA	84.0	84.8	84.7	89.5	82.2	–	81.3	87.3	89.5	95.0	94.2
OD	78.7	76.2	65.7	89.3	74.3	73.5	–	91.7	87.2	94.3	88.3
PU	56.0	70.2	60.5	49.8	60.0	84.5	70.2	–	89.8	92.8	82.2
TA	73.9	83.3	81.3	80.5	83.5	77.0	58.7	57.2	–	97.3	93.5
TE	88.0	98.2	96.2	90.2	81.5	84.3	78.3	62.8	90.2	–	94.5
UR	75.3	95.2	86.3	84.0	94.3	83.3	70.7	53.8	68.8	89.4	–

ParaSet ($\mu = 97.8$, $\sigma = 2.1$). However, removal of the blank rows from these paragraphs has improved their accuracy of recognition against the WordSet ($\mu = 91.4$, $\sigma = 6.5$) (see upper triangular part of Table 4.24). It may be noted here that the removal of blank rows from the paragraphs has only reduced the gap between lines and leaves the gap between the words untouched. An effort to remove these gaps may lead to further improvement in accuracies. In spite of all this, we can claim that the proposed system is robust to recognize script at any level of text existence with acceptably good accuracies.

4.5 Discussion & Conclusion

Script identification plays a vital role for the development of next generation OCR's. Besides reducing the search space, they help to exploit the language and/or script specific rules to enhance the OCR output. It is observed that word is the fundamental unit of text structure and scripts change with words.

Development of a robust system requires a strong database. A very large database of words is created, involving eleven Indian scripts. Lot of effort has been put in for the collection of paper manuscripts with variability in printing style and size, for all the scripts, acquisition of the images of these manuscripts and segmentation to generate the paragraph, line and word images. Such images are visually inspected and subsequently examined in the feature space, to avoid the repetition. A large variation in the aspect ratios and ON-pixel densities of the words is ensured. This collection has 20000 words from each script, which makes it the largest collection of words from Indian scripts. This database is made available online for its use by the research community.

A recognizer capable of recognizing Indian scripts with reasonable accuracy is designed and developed. This system is seen to work with different kinds of variations, such as font sizes and styles, for any script. This, also, is observed to efficiently perform at bi-script, tri-script and eleven-script scenarios.

The efficacies are evaluated for all the different combinations of two prominent features and three standard classifiers. The combination of Gabor filter bank with either SVM or NN classifier handles the issue of script recognition at the word level, reasonably well. NNC and SVM classifiers perform comparably and outperform LDC in most cases. However, the actual performance is highly script dependent. For example, using the Gabor-SVM combination, the overall classification performance for the tri-script combination involving Kannada, Devanagari and English is 99.6%, whereas the average correct recognition is only 93.6% for the bi-script combination of Kannada with Telugu, and it is only 94.2% when Punjabi is recognized against Hindi. Roman script has a recognition

accuracy of 97.8%, using NNC, against a training set consisting of all the eleven scripts, which is the best in the eleven-class scenario.

A comparison between the Gabor and DCT features, shows Gabor to be leading in many of the bi-script and tri-script cases. However, in the 11-script scenario using DCT, there is an improvement of the overall recognition accuracy along with that of Hindi, Odiya and Punjabi scripts. In bi-script cases, the average recognition accuracy of Kannada and Telugu duo has also improved from 89.2% using Gabor features to 93.4% using the DCT features. Interestingly, with Gabor features, when LDC is used for the above case, the recognition accuracy increases to 92.9% from 89.2% with NNC. However, NNC performs better (93.1%) with DCT, than LDC which has a recognition accuracy of 74.8%. Thus, it is immediately not very clear whether it is the feature or the classifier or the specific combination that is responsible for the peculiar behavior observed in this case.

An investigation on the effect of the length and ON-pixel density of the word on the recognition rate. It may be claimed that a longer word has more script specific information and, hence, is more likely to be correctly recognized. Similar assumption may be made for the ON-pixel density. However, the investigation results refute any such claim.

Design of a blind script identifier using a tree structure has not yielded desired results. However, employing feature reduction techniques with fine tuning of the SVM kernel parameters is likely to enhance the accuracy at each nodal point, finally contributing to the overall increase in performance.

The efficacy of the developed system to identify scripts of line and paragraph images is evaluated. It is observed that the removal of the inter-word and interline gaps enhances the output of the system for images at these levels even when the training set is from word images. This shows that words, lines and paragraphs co-exist in the Gabor feature space. Moreover, it is inferred that large gap between words and lines are detrimental to the performance of this system. It may be interesting to study the efficacy of the system after removing the inter-character gaps in a word, wherever that is possible. Despite all this, the results substantiate our assumption that the HVS inspired system is well suited for script identification in multi-script documents.

The size of 7000 training patterns per script is large. This consumes a lot of time while NNC is at work. Moreover, LDC and SVM require a selected few boundary patterns per class. Thus it is worth trying some of the prototype selection techniques to reduce the training set.

Study of the divergences shows that some features have better discriminating property than others. Thus, a selected subset of features helps in reducing the computation while enhancing the efficacy. Such subset is uniquely selected for each case separately. This strategy is shown to be effective for bi-script cases, which suffices as a proof of the concept.

However, such a study needs to be carried out for different feature-classifier combinations, at all levels of script identification. Normalizing the dynamic ranges of the various features after centering of the data sets has been seen not to have any significant effect on the NNC and SVM and cause large detrimental effect on the LDC, for both features.

The major contributions of this chapter may be summarized as follows:

- creating a large word dataset for eleven Indian scripts,
- design and development of a script recognizer to deal with bi-script, tri-script and multi-script cases, separately, all by the same system,
- study of the effect of aspect ratio and ON-pixel density on the recognition rates of the script at the word level,
- study of the effect of dynamic range normalization and feature set reduction in improving the output of the proposed system,
- design and implementation of a hierarchical blind-multi-script identifier, and
- evaluation of the efficacy of the proposed system at higher levels of textual existence, namely lines and paragraphs.

Chapter 5

Conclusion and Future Scope

"Furious activity is no substitute for understanding" – H. H. Williams.

"I didn't think; I experimented." – Wilhelm Roentgen.

5.1 Conclusion

The thesis has addressed two issues meant to enhance the features of the present day OCR technologies. They are: (i) localization of text areas in documents with complex layout & content, and (ii) identification of the script of a word in multi-script documents. The state of the art OCR technology is able to recognize even poor quality, but text only documents. However, the analysis and recognition of the documents containing non-text elements, such as pictures and graphics, is difficult for most of the OCR technologies. This is because of their inability to efficiently detect and isolate the text areas in these documents.

Detecting and localizing text in scenic images has interesting implications. Detection and interpretation of text on the road side or the traffic signals, might facilitate the design of automatic navigation system for driver-less cars. It may be used for retrieval of images based on textual rather than the textural information. However, a camera captured image, unlike a scanned one, has varieties of challenges such as perspective distortion, illumination variation, environmental as well as instrument noise. While these may complicate the detection of text in such images, the global properties of text such as geometry, contrast and regularity in spacing aid the system designer. Further, color information, whenever available, may help in better accomplishment of the job.

The text layout analysis and separation system proposed for gray images is observed to work fine for document images with variation in types of layout and clutter. This system generates the desired output for images captured using the camera, besides demonstrating a good performance for scanned images. Since the textural properties of text blocks are considered to distinguish the between text and non-text elements, the system is script independent. The system is evaluated on a large set of about 450 images, containing a wide variation in the nature of text present. This evaluation includes comparison of the results generated by the proposed system against the output from some of the established systems. It is found that the proposed system performs either at par or better than the existing ones. An analysis of the computational aspects of the algorithm has led to improvements, resulting in a system with better performance at a lower computational cost.

A system has also been developed for page layout analysis on color images. The contrast between the text and its background might exist in color space and may be lost in gray space. So, the use of color information along with the textural information is proposed. Such a strategy has paid off well, when the results of this system are evaluated against our own system, based on gray-converted images and some other systems. However, the price paid for such improvements in accuracy, is the increase in computational complexity.

The identification of the script at the level of words reduces the search space and enhances the computational aspects of the associated OCR system. Script and language specific processing and/or corrective mechanisms may then be employed for enhancing the accuracy of recognition. In many Indian languages, a pure vowel cannot occur in the middle of a word and this rule may be employed to enhance the accuracy of the system.

A script recognizer has been designed and developed based on the textural properties of word images. All the eleven widely used Indian scripts are handled here. A very large dataset of words is collected and is made available over the web to fellow researchers. Bi-script, tri-script and multi-script documents are addressed for evaluating the efficacy of the developed system. One of the two features, Gabor transform coefficients and DCT coefficients, is used in combination with one of the three classifiers (Nearest neighbor, linear discriminant and SVM). The effectiveness of various such combinations is evaluated. A hierarchical and blind script recognizer has been developed for systematically & efficiently handling script recognition, when no prior information is known about the nature of script of a test word. A proof of this concept is presented. We do not claim it to be the optimal blind script identifier. Many more experiments need to be performed before we actually develop the optimal blind script identifier. Some pointers towards this are discussed.

The developed script identifier works even at higher levels of textual existence, namely, the lines and paragraphs. This claim is validated with results and it is shown that no matter what the contents of the input image are, with minor preprocessing steps, one can achieve the desired accuracy for recognition of the script. The best part of this system is that creation of separate databases for different levels of these text structures is avoided.

Thus, the major contributions of the thesis are:

- design and development of a text extractor for gray images with complex layout and content, irrespective of the mode of acquisition of such images.
- a system to process the color information along with the illumination information to better isolate text elements from the rest, wherever such color information is available.
- a system to recognize bi-script, tri-script cases for Indian scripts. However, the concept is general and can easily be extended to combinations of bi-script and tri-script cases, involving non-Indian scripts too.
- a hierarchical system to recognize the script, in the absence of any prior information about the script, again for Indian scripts.

5.2 Scope for Future Work

Two of the important issues confronted by the document image researchers are addressed in this thesis. The contributions of the thesis advances the understanding of the associated problems and makes effort to provide at least a partial solution for them. However, the proposed solutions are far from optimal for the said problems. Though a large number of explorations have been made, due to constraints of time and resources, not all possible experiments could be tried. Thus, there exist plenty of opportunities and scope in this research area before we can have the final solutions to these problems. Few such studies, which may be performed in future, are listed below.

- The proposed text layout analysis system works good when the text content of the documents is high and fares reasonably on images where text content is low. However, for images which are predominantly non-text, the failure rate is high. Thus, systematic and analytical studies need to be made to make the algorithm independent of such constraints.

- The proposed text localization algorithm works on the principle of thresholds at multiple junctions. These thresholds are evaluated based on mere observation and experience. They are currently hard coded values but may be made dependent on the image statistics and independent of human intervention.
- When the size of the characters in the camera captured scenic images is large, the algorithm fails. This is because the text elements are eliminated based on their geometry information. The next level of investigation should make an effort to get rid of this problem.
- A systematic and rigorous analytical design of the filterbank would eliminate the unnecessary computations involved in the algorithm for text detection. This may be similar in its approach to the feature set reduction techniques employed for finding the best subset of features that optimizes the bi-script recognition accuracies.
- Efforts should be made to introduce flexibility in automatically selecting between different algorithmic routes, depending on the nature and statistics of the input image.
- Feature selection has been attempted for the Gabor-NNC combination in the bi-script recognition. Though this is sufficient to prove the concept of feature reduction on a case by case basis, it needs to be performed for each case separately & for all levels of script recognition.
- 7000 words from a script form a large training set. The neighborhood analysis proves this point and there is a huge potential to reduce this set, using an efficient prototype selection mechanism. This should be explored, on a case by case basis.
- Though for script recognition experiments, SVM have given good accuracy at all levels, fine tuning of the parameters with rigorous experimentation might be able to extend these results on the higher side.
- Soft-computing approaches to design and implement artificial neural networks with adjustable network topology may deliver better accuracy for some cases. These need to be explored for enhancing the efficiency of the proposed systems.
- The efficiency of the blind script recognizer may be enhanced by judicious selection of the feature set and the feature-classifier combination. This may happen at each individual node of the classification tree or the entire tree as a whole. Studies to make such selections may be carried out to arrive at the *optimal-tree-blind-script recognizer*

Bibliography

[1] "Wikipedia: The Free Encyclopedia," http://www.wikipedia.org.

[2] K.-C. Fan, L.-S. Wang, and Y. K. Wang, "Page segmentation and identification for intelligent signal processing," *Signal Processing*, vol. 45, no. 3, pp. 329–346, 1995.

[3] A. Simon, J. Pret, and A. Johnson, "A fast algorithm for bottom-up document layout analysis," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, pp. 273–276, 1997.

[4] H. Fujisawa and Y. Nakano, "A top-down approach for the analysis of documents," in *Proc. 10th Intl. Conf. on Pattern Recognition (ICPR'90)*, pp. 113–122, 1990.

[5] F. Wahl, K. Wong, and R. Casey, "Block segmentation and text extraction in mixed text/image documents," *Computer Vision Graphics and Image Processing*, vol. 20, pp. 375–390, 1982.

[6] A. Yamashita, T. Amano, Y. Hirayama, N. Itoh, T. M. S. Katoh, and K. Toyokawa, "A document recognition system and its applications," *IBM Journal of Research and Development*, vol. 40, pp. 341–352, 1996.

[7] Y. Y. Tang, S. W. Lee, and C. Y. Suen, "Automatic document processing: A survey," *Pattern Recognition*, vol. 29, no. 12, pp. 1931–1952, 1996.

[8] S. Messelodi and C. M. Modena, "Automatic identification and skew estimation of text lines in real scene images," *Pattern Recognition*, vol. 32, no. 5, pp. 791–810, 1999.

[9] Y. Zhong, K. Karu, and A. K. Jain, "Locating text in complex color images," *Pattern Recognition*, vol. 28, no. 10, pp. 1523–1535, 1995.

[10] A. K. Jain and B. Yu, "Automatic text location in images and video frames," *Pattern Recognition*, vol. 31, no. 12, pp. 2055–2076, 1998.

[11] C. Strouthpoulos, N. Papamarkos, and A. E. Atsalakis, "Text extraction in complex color document," *Pattern Recognition*, vol. 35, no. 8, pp. 1743–1758, 2002.

[12] D. Wang and S. N. Srihari, "Classification of newspaper image blocks using texture analysis," *Computer Vision, Graphics, and Image Processing*, vol. 47, pp. 327–352, 1989.

[13] A. K. Jain and S. Bhattacharjee, "Text segmentation using Gabor filters for automatic document processing," *Machine Vision and Applications*, vol. 5, pp. 169–184, 1992.

[14] W. Chan and G. Coghill, "Text analysis using local energy," *Pattern Recognition*, vol. 34, pp. 2523–2532, 2001.

[15] A. Antonacopoulos and R. T. Ritching, "Representation and classification of complex shaped printed regions using white tiles," in *Proc. Third Intl. Conf. on Document Analysis and Recognition*, (Montral, Canada), pp. 1132–1135, 1995.

[16] A. K. Jain and Y. Zhong, "Page segmentation using texture analysis," *Pattern Recognition*, vol. 29, no. 5, pp. 743–770, 1996.

[17] M. Chen and X. Ding, "A robust skew detection algorithm for gray scale document image," in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 617 – 620, 1999.

[18] L. Lo, "Ancientscripts.com: a compendium of world-wide writing systems from prehistory to today," http://www.ancientscripts.com, 2003.

[19] "Scripts and sounds," http://indiansaga.com/.

[20] S. Ager, "Omniglot: writing systems and languages of the world," http://www.omniglot.com.

[21] Y. K. Malaiya, "Languages and scripts of India," http://www.cs.colostate.edu/ malaiya/scripts.html.

[22] A. L. Spitz, "Determination of Script and Language Content of Document Images," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 3, pp. 235–245, 1997.

[23] J. Hochberg, P. Kelly, T. Thomas, and L. Kerns, "Automatic script identification from document images using cluster based templates," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 2, pp. 176–181, 1997.

[24] S. L. Wood, X. Yao, K. Krishnamurthi, and L. Dang, "Language identification for printed text independent of segmentation," in *Proc. Intl. Conf. on Image Processing*, pp. 428–431, 1995.

[25] A. R. Chaudhuri, A. K. Mandal, and B. B. Chaudhuri, "Page layout analyser for multilingual Indian documents," in *Proc. the Language Engineering Conference*, pp. 24–32, 2002.

[26] U. Pal and B. B. Chaudhuri, "Script line separation from Indian multi-script document," in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 406–409, 1999.

[27] T. N. Tan, "Rotation invariant texture features and their use in automatic script identification," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 20, no. 7, pp. 751–756, 1998.

[28] S. Chaudhury and R. Sheth, "Trainable Script Identification Strategies for Indian languages," in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 657–660, 1999.

[29] W. Chan and J. Sivaswamy, "Local energy analysis for text script classification," in *Proc. of Image and Vision Computing New Zealand (1999)*, (New Zealand), 1999.

[30] G. D. Joshi, S. Garg, and J. Sivaswamy, "Script identification from Indian documents," in *Seventh IAPR Workshop on Document Analysis Systems 2006, LNCS-3872*, (New Zealand), pp. 255–267, February 2006.

[31] R. Manthalkar and P. K. Biswas, "An automatic script identification scheme for Indian languages," in *Proc. Eighth Nat. Conf. on Comm., 2002*, (Mumbai, INDIA), pp. 31–34, 2002.

[32] V. Ablavsky and M. R. Stevens, "Automatic feature selection with applications to script identification of degraded documents," in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 750–754, 2003.

[33] J. Gllavata and B. Freisleben, "Script recognition in images with complex backgrounds," in *Proc. of the Fifth IEEE Intl. Sym. on Sig. Proc. & Info. Tech., 2005*, pp. 589–594, 2005.

[34] D. Dhanya, A. G. Ramakrishnan, and P. B. Pati, "Script identification in printed bilingual documents," *Sadhana*, vol. 27, no. 1, pp. 73–82, 2002.

[35] P. B. Pati, S. S. Raju, N. K. Pati, and A. G. Ramakrishnan, "Gabor filters for document analysis in Indian bilingual documents," in *Proc. Int. Conf. on Intelligent Sensing and Information Processing*, pp. 123–126, 2004.

[36] P. B. Pati and A. G. Ramakrishnan, "HVS inspired system for script identification in Indian multi-script documents," in *Seventh IAPR Workshop on Document Analysis Systems 2006, LNCS-3872*, (New Zealand), pp. 380–389, February 2006.

[37] H. Ma and D. Doermann, "Gabor filter based multi-class classifier for scanned document images," in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 968–972, 2003.

[38] S. Jaeger, H. Ma, and D. Doermann, "Identifying Script on Word-level with Informational Confidence," in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 416–420, 2005.

[39] U. Pal, S. Sinha, and B. B. Chaudhury, "Word-wise script identification from a document containing English, Devanagari and Telugu text," in *Proc. National Conf. on Document Analysis and Recognition*, pp. 213–220, 2003.

[40] U. Pal, S. Sinha, and B. B. Chaudhuri, "Multi-script line identification from Indian documents," in *Proceedings of the International Conference on Document Analysis and Recognition*, pp. 880–884, 2003.

[41] M. C. Padma and P. Nagabhushana, "Identification and separation of text words of Kannada, Hindi and English languages through discriminating features," in *Proc. National Conf. on Document Analysis and Recognition*, pp. 252–260, 2003.

[42] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 12, 1990.

[43] F. W. Campbell and J. G. Robson, "Application of Fourier analysis to the visibility of gratings," *J. Physiol.*, vol. 197, pp. 551–566, 1968.

[44] R. L. De Valois, D. G. Albrecht, and L. G. Thorell, "Spatial-frequency selectivity of cells in macaque visual cortex," *Vision Research*, vol. 22, pp. 545–559, 1982.

[45] M. Porat and Y. Y. Zeevi, "The generalized Gabor scheme of image representation in biological and machine vision," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 10, no. 4, pp. 452–467, 1988.

[46] M. C. Morrone and D. C. Burr, "Feature detection in human vision: a phase dependent energy model," *Proc. Roy. Soc. Lon.(B)*, vol. 235, pp. 221–245, 1988.

[47] S. Marcelja, "Mathematical description of the response of simple cortical cells," *J. Opt. Soc. Am.*, vol. 70, no. 11, pp. 1297–1300, 1980.

[48] J. Daugman, "Uncertainty relation for resolution in space, spatial frequency and orientation optimized by two-dimensional visual cortical filters," *J. Opt. Soc. Am. A*, vol. 2, no. 7, pp. 1160–1169, 1985.

[49] C. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. of fourth Alvey Vision Conference*, pp. 147–151, 1988.

[50] Y. Xiao and H. Yan, "Text region extraction in a document image based on the Delaunay tesselation," *Pattern Recognition*, vol. 36, pp. 799–809, 2003.

[51] P. B. Pati, "Machine recognition of printed Odiya text documents," Master's thesis, Department of Electrical Engineering, Indian Institute of Science, Bangalore, 2001.

[52] S. S. Raju, P. B. Pati, and A. G. Ramakrishnan, "Gabor filter based block energy analysis for text extraction from digital document images," in *Proc. First Int. Workshop on Document Image Analysis for Libraries (DIAL'04)*, pp. 233–243, 2004.

[53] "National Research Council of Canada," http://www.nrc-cnrc.gc.ca/, 2004.

[54] S. S. Raju, P. B. Pati, and A. G. Ramakrishnan, "Text localization and extraction from complex color images," in *Proc. Int. Sym. on Visual Computing, LNCS – 3804*, pp. 486–493, 2005.

[55] S. N. S. Rajasekaran and B. L. Deekshatulu, "Recogntion of printed Telugu characters," *Computer Graphics and Image Processing*, vol. 6, pp. 335–360, 1977.

[56] P. B. Pati and A. G. Ramakrishnan, "OCR in Indian Scripts: A Survey," *IETE Technical Review*, vol. 22, no. 3, pp. 217–227, 2000.

[57] P. B. Pati and A. G. Ramakrishnan, "Indian Script Word Image Dataset," www.ee.iisc.ernet.in/new/people/students/phd/pati/, 2005.

[58] D. J. Field, "Relation between the statistics of natural images and the response properties of cortical cells," *Journal of Opt. Soc. of Am. A*, vol. 4, no. 12, pp. 2379–2394, 1987.

[59] S. Theodoridis and K. Koutroumbas, *Pattern Recognition*. London: Academic Press, 1999.

[60] K. R. Rao and P. Yip, *Discrete Cosine Transform : Algorithms, Advantages, Applications*. New York: Academic Press, 1990.

[61] R. Muralishankar, A. G. Ramakrishnan, and P. Prathibha, "Modification of pitch using DCT in the source domain," *Speech Communication*, vol. 42, pp. 143–154, 2004.

[62] C. L. Salazar and T. D. Tran, "On resizing images in the DCT domain," in *Proc. Int. Conf. on Image Processing*, pp. 2797–2800, 2004.

[63] U.-V. Koc and K. J. R. Liu, "DCT based motion estimation," *IEEE Tr. on Image Processing*, vol. 7, no. 7, pp. 948–965, 1998.

[64] D. Zhao, W. Gao and Y. K. Chan, "Morphological representation of DCT coefficients for image compression," *IEEE Tr. CSVT*, vol. 12, no. 9, pp. 819–823, 2002.

[65] J. Tang and S. T. Acton, "A DCT based gradient vector flow snake for object boundary detection," in *Southwest04*, pp. 157–161, 2004.

[66] C. J. C. Burges, "A tutorial on support vector machines for pattern recognition," *Data Mining and Knowledge Discovery*, vol. 2, no. 2, pp. 955–974, 1998.

[67] R. Collobert and S. Bengio, "On The convergence of SVMTorch, an Algorithm for Large Scale Regression Problems," Tech. Rep., Dalle Molle Institute for Perceptual Artificial Intelligence, Martigny, Switzerland, 2000.

[68] "SVMTorch – II," IDIAP Research Institute, Martigny, Switzerland, www.idiap.ch/learning/SVMTorch.html, last visited: Oct, 2006.

[69] P. Pudil, J. Novovicova, and J. Kittler, "Floating search methods in feature selection," *Pattern Recognition Letters*, vol. 15, pp. 1119–1125, 1994.

[70] R. O. Duda, P. E. Hart and D. G. Stork, *Pattern Classification*. Singapore: John WIley and Sons, Inc., 2001.

Publications Related to the Thesis

Conferences:

1. P B Pati, Farshad Nourbakhsh, Bhavna A and A G Ramakrishnan, "Automatic Seal Information Reader," accepted in International Conference on Computing: Theory and Applications, March 2007, Kolkata, India.

2. Bhavna A, Peeta Basa Pati and A G Ramakrishnan, "Binarization And Localization of Text Images Captured On A Mobile Phone Camera," accepted in 4th Intl. Conf. on Intelligent Sensing and Information Processing, 2006, Bangalore, India.

3. Farshad Nourbakhsh, P B Pati and A G Ramakrishnan, "Efficient Document Page Layout Analysis using Harris' Corner points," accepted in 4th Intl. Conf. on Intelligent Sensing and Information Processing, 2006, Bangalore, India.

4. Arvind K R, P B Pati and A G Ramakrishnan, "Automatic Text Block Separation In Document Images," accepted in 4th Intl. Conf. on Intelligent Sensing and Information Processing, 2006, Bangalore, India.

5. Arvind K R, P B Pati and A G Ramakrishnan, "Horizontal Projection Profiles for extraction of Text Paragraphs from Document Images," accepted in IEEE Intl. Conf. on Signal and Image Processing, 2006, Hubli, India.

6. Farshad Nourbakhsh, P B Pati and A G Ramakrishnan, "Text Localization and Extraction from Complex Gray Imagess," accepted in Indian Conf. on Vision, Graphics and Image Processing, 2006, Madurai, India.

7. P B Pati and A G Ramakrishnan, "HVS inspired System for Script Identification in Indian Multi-script Documents," *7th IAPR Workshop on Document Analysis Systems 2006*, Nelson, New Zealand, LNCS-3872, pp:380 – 389.

8. S S Raju, P B Pati and A G Ramakrishnan, "Text Localization and Extraction from Complex Color Images," *Proc. of the Intl. Sym. on Visual Computing (ISVC05)*, Nevada, USA, Dec. 2005, LNCS - 3804, pp:486 – 493.

9. S S Raju, P B Pati & A G Ramakrishnan, "Gabor filter based block energy analysis for text extraction from digital document images," *Proc. First Intl. Workshop on Doc. Im. Ana. for Libraries(DIAL' 04)*, pp:233-243, IEEE Publications, Palo Alto, USA, Jan. 2004.

10. P B Pati, S S Raju, N K Pati and A G Ramakrishnan, "Gabor filters for document analysis in Indian Bilingual Documents," *Proc. of First Intl. Conf. on Intelligent Sensing and Information Processing (ICISIP-04)*, pp:123 – 126, 2004, IEEE Publications.

Journals:

1. D Dhanya, A G Ramakrishnan and P B Pati, "Script Identification in printed bilingual documents," *Sadhana*, 27(1):73-82, Feb'02.

2. Peeta Basa Pati and A G Ramakrishnan, "Word Level Multi-script Identification," revised manuscript communicated to *Pattern Recognition Letters*.

3. Peeta Basa Pati, S S Raju and A G Ramakrishnan, "Text Localization and Extraction from Color Scenic Images," manuscript under preparation to be communicated to *Machine Vision Applications*.