A Tool-Kit for Document Image Processing

A project report submitted in partial fulfillment of the
requirements for the award of degree of

Master of Technology

in

Computer Science

BY

Y.Pradeep Kumar G. Koteswar Rao

Department of Computer and Information Sciences
University of Hyderabad
Hyderabad - 500 046

January 29, 2002

CERTIFICATE

This is to certify that the project work entitled "A Tool Kit for Document Image Processing " being submitted by *Y.Pradeep Kumar* bearing Reg. No. 2KMCMT19 and *G. Koteswar Rao* bearing Reg. No. 2KMCMT30 in partial **fulfillment** of the requirements for the award of *Master of Technology* in *Computer Science* by **the** University of Hyderabad, is a bonafide work carried **out** at **the** University of Hyderabad under my supervision.

The matter embodied in this **project** has not been submitted to any other university for the award of any degree or diploma.

Dr. Atul Negi
(Project Supervisor)
Reader,
Department of CIS,
University of Hyderabad,
Hyderabad - 500046.

Prof. H.Mohanty
Head,
Department of CIS,
University of Hyderabad,
Hyderabad - 500046.

Prof. **R.Tandon**
Dean,
School of MCIS,
University of Hyderabad,
Hyderabad - 500046.

ACKNOWLEDGEMENTS

It is indeed an honour to acknowledge our heartfelt gratitude and indebtedness to **Dr. Atul Negi for** his inspiring guidance, timely suggestions and constant encouragement without which the project would not have been this successful. It was a learning experience to be under his tutelage, which shall immensely help us in our future.

This project is a part of a major research work being conducted in the Department of Computer and Information Sciences (UoHyd) for the Ministry of Information Technology, India under the investigation of Prof. Arun Agarwal and Dr. Atul Negi. We extend our sincere thanks to Dr. Chakravarthy Bhagvati for his guidance.

We extend our thanks to **Prof. H.** Mohanty, Head, Department of CIS and **Prof.** R. Tan don , Dean, School of MCIS for providing us the necessary computing facilities.

We also thank the staff of A.I lab and TRCT lab for their timely co-operation.

We thank Peddi, Ganesh, Ambati, Guptha, **Raghava,** Ajay and Ramakrishna for helping us at various stages of the project.

We thank all our friends, who **have** made our stay at the campus a lively one.

Y Pradeep Kumar *G.Koteswar Rao*

Abstract

Document image analysis aims at the transformation of information presented on paper and addressed to human comprehension into an equivalent symbolic representation accessible to computer information processing. Document image analysis consists in the conversion of document's pixel representation into an equivalent knowledge network representation holding the document's content and layout.

The objective of our project is to develop a set of tools for document image processing. Document image processing involves several phases. In pre-processing phase skew detection and correction is done which is necessary for correct page segmentation and character recognition in post-processing phase. Our approach for skew detection and correction is based on an efficient and robust algorithm which involves computation of transition count variance at various orientations and finds the orientation for which the transition count variance is maximum. Skew correction is done using the scanning line model. Skew is corrected by computing the offsets along ***the*** *horizontal and vertical directions and shifting the gray values from the skewed image to the reconstructed image.*

A document has figures, tables and charts in addition to text. In layout analysis phase we segment the document image into its constituent objects, seperate the text from non-text areas and classify them into appropriate objects. For page segmentation, the run length smearing algorithm is applied first on the binarized document image followed by the connected component algorithm for extracting the connected components. After component extraction, component classification method is applied on the extracted components which classifies the components into text and non-text ones considering the gray level distribution of the pixels in the component. A prior knowledge of the document layout and structure is required for doing the various tasks in an efficient manner. We tested the various algorithms on a variety of English and Telugu documents and obtained satisfactory results.*

Contents

List of Figures

Chapter 1

INTRODUCTION

Printed paper documents are one of the primary information medium in the society used. They are used for transferring information over long distances in uniform way, reading, dissemination, mark-up and can also be kept as an evidence. The goal in the early 1980s of the *"Paperless office"* has now given way to the objective of handling the flow of electronic and paper documents in an efficient and integrated way.

The benefits and advantages by shifting to electronic media from paper media are effective handling, easy retrieval, effective search and transmission and this shift is expected to give a great boost to business efficiency in modern offices. In order to achieve this goal of paperless office, an efficient and integrated approach to derive maximum efficiency in handling both electronic and paper documents, it is imperative to devise an automated approach for document analysis and handling. Document analysis aims at the transformation of information presented on paper into an equivalent symbolic representation accessible to a computer information processing system.

1.1 Computerized document processing

The objective of computer document processing is to recognize text/graphics and continuous tone pictures in the image and to extract the intended information as a human would extract. Textual processing and graphical processing are two categories of document image analysis dealing with the processing of text components and graphical elements of a document image respectively (see Fig 1.1).

Textual processing includes

- Determining the skew (Due to a tilt in the document during scanning)
- Separating text from the graphics
- Isolating words, text lines, paragraphs, columns
- Buliding a logical structure of the document
- Perform optical character recognition

Graphical processing includes

- Raster to vector conversion
- Line fitting
- Symbol recognition.

Continuous tone pictures are another document component that need to be handled differently from text/graphics. Except for recognizing their location on a page, analysis of these components usually falls within the domain of machine vision.

Document image analysis systems reduce the megabytes of picture data into a more concise semantic description.

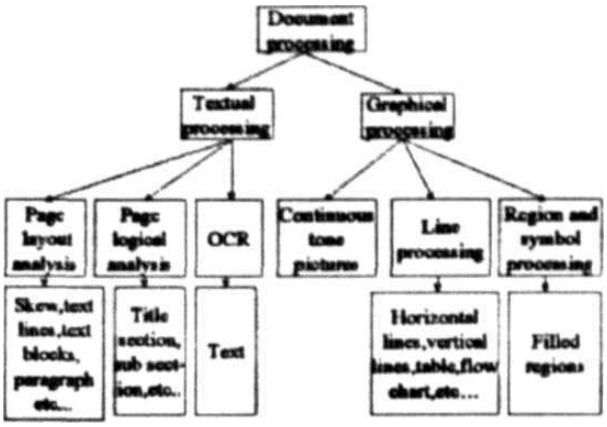

Figure 1.1: A hierarchical document processing structure

1.2 Overview of document image understanding

1.2.1 Functional architecture

A functional architecture specifies the major functional components without concerning itself with practical considerations such as shared resources[37] . The functional modules and interactions of a DIP (Document image processing) system are shown in Figure 1.2. The DIU(Document image understanding) task is divided into three conceptual levels: document image analysis, document image recognition and docurmmt understanding. Within these levels there are several processing modules (or tools): binarization, area-segmentation, area-labelling, OCR, photograph analysis, graphics analysis, picture understanding, natural language processing, and graphics understanding. The interaction between modules allows for the interpretation of individual sub-areas to be combined to form a higher level of representation e.g., the interpretation of a photograph caption by a natural language processing module and objects in a photograph located by a photograph analysis module can be used by a photograph understanding module to label the object's identity.

Input to the system is a high-resolution gray-scale image (e.g., 300ppi). Using gray-scale imagery increases image analysis capability e.g., gray-scale character recognition[34] and analysis of half-tone photographs. The output of the system is

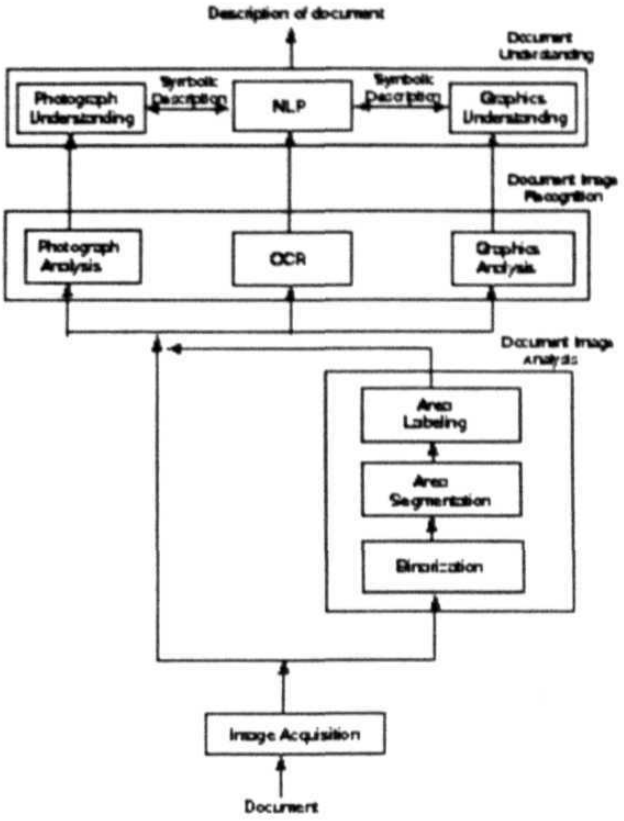

Figure 1.2: Functional architecture

a description of the contents of the document components. An editable description should contain the following entries: (i) component identities and locations on the document e.g., text, graphics, half-tones etc., (ii) spatial relationships between components (iii) layout attributes e.g., component size, number of lines in text blocks etc., and (iv) logical grouping of components.

Level of document understanding is determined by the level of representation that the DIU system can derive from document image. Three types of information can be derived from a document: (i) layout (geometric) structure is a physical description of document regions i.e., size, location, spatial relationships between regions (ii) logical structure is a grouping of layout components based on human interpretation of the content of the components and the spatial constraints between components and (iii) content interpretation contains coded data of a component which can be used to

derive logical structure or can be stored for later access. A system whose output contains only layout structure is a document layout analysis system and a system whose output contains all three types of information is a DIU system. From this perspective, layout analysis is an intermediate step of DIU [17].

Representation levels

In order to develop a system for DIU, it is necessary to establish the levels of data representation as data is passed between tools. Five levels of representation, each at an increasing level of abstraction are distinguished:

1. Pixel

   It is the most primitive representation of a document. Pixel information is mainly used in area-segmentation and area-labeling.

2. Connected component

   It is formed by a group of connected black pixels. It is more time-efficient to access a connected component than a group of pixels. It is an appropriate representation for character recognition and line-drawing analysis.

3. Symbol

   It is the output of the recognition subsystems e.g., text, graphic commands, line-drawing descriptions.

4. Frame

   It is the representation of a group of document components as formed by layout structure analysis and logical structure analysis.

5. Tree

   It is the arrangement of the logical units on a document.

Advantages of representation

This hierarchical representation of data has several advantages: It is space-efficient and allows different tools to use the same set of data.

Data at a particular level are linked bidirectionally to their immediate neighboring levels (except pixel and tree levels which have only one neighboring level). The links facilitate robust control and problem solving strategies which require selective data access to different levels.

This representation supports different control strategies:

1. Top-down

   It begins at the tree level and progressively moves downward to the low level pixel details. The detailed description of this method is given in section 1.3.1.

2. Bottom-up

   It begins at the pixel level and progressively moves upward to the higher level components of the document. The detailed description of this method is given in section 1.3.1.

3. Opportunistic

   It accesses data at different levels according to the problem state.

A tool can use data from more than one level by following data linkage.

Document descriptions

For a given document the output of a DIU system is an representation of its contents. In particular, editable descriptions are of interest. The Office Document Architecture (ODA) and the Standard Generalized Markup Language (SGML) are two international standards for the representation and interchange of structured documents[6). ODA is commonly used as the standard output format of a document understanding system. SGML is more closely linked to the publishing and printing communities where great flexibility of document design and layout are of prime importance. ODA provides most of the basic framework needed as the basis for an interactive editing system as well as for document storage and transfer.

1.3 Decomposition and structural analysis

A document image is a visual representation of a printed page such as a journal article page, a facsimile cover page, a technical document, an office letter etc.,. Typically, it consists of blocks of text i.e., letters, words, and sentences that are interspersed with tables and figures. The figures can be symbolic icons, gray-level images, line drawings or maps. A digital document image is a two-dimensional representation of a document image obtained by optically scanning and digitizing a hard copy document. It may also be an electronic version that was created for publishing or drawing applications available for computers.

The document decomposition and structural analysis task can be divided into three phases[32]. Phase 1 consists of block segmentation where the document is decomposed into several rectangular blocks. Each block is a homogeneous entity containing one of the following: text of a uniform font, a picture, a diagram or a table. The result of phase 1 is a set of blocks with the relevant properties. A textual block is associated with its font type, style and size; a table might be associated with the number of columns and rows etc., . Phase 2 consists of block classification. The result of phase 2 is an assignment of labels (title, regular text, picture, table, etc.,) to all the blocks using properties of individual blocks from phase 1 as well as spatial layout rules. Phase 3 consists of logical grouping and ordering of blocks. For OCR it is is necessary to order text blocks. Also the document blocks are grouped into items that "mean" something to the human reader (author, abstract, date, etc.,) and is more than just the physical decomposition of the document. The output of phase 3 is a hierarchical tree-of-frames, where the structure is defined by the shape of the tree and the content is stored entirely in the leaves. The tree-of-frames can be converted to the SGML representation to ensure portability, easy information retrieval, editability and efficient semantic indexing of the document.

1.3.1 Block segmentation

Approaches for segmenting document image components can be either top-down or bottom-up. Top-down techniques divide the document into major regions which are further divided into sub-regions based upon knowledge of the layout structure of the document. Bottom-up methods progressively refine the data by layered grouping operations.

Top-down methods

1. Smearing

   It is based on the Run-Length Smoothing Algorithm (RLSA)(24]. It merges any two black pixels which are less than a threshold apart, into a continuous stream of black pixels. The method is first applied row-by-row and then column-by-column, yielding two distinct bit maps. The two results are then combined by applying a logical OR to each pixel location. The output has a smear wherever printed material (text, pictures) appears on the original image.

2. The X-Y tree decomposition method

   It assumes that a document can be **represented** in the form of nested rectangular blocks. A local peak detector is applied to the horizontal and vertical projection profiles to detect local peaks (corresponding to thick black or white gaps) at which the cuts are placed; it is local in that **the** width is determined by **the** nesting level of recursion, e.g., gaps between paragraphs are thicker than those between lines.

3. The Hough transform approach

   It exploits the fact that documents have significant linearity. There exist straight lines in tables and diagrams. Centroids of connected components corresponding to text also line up. Columns of text are separated by straight rivers of white space. Text itself can be viewed as thick textured lines. The Hough transform is a technique for detecting parametrically re-presentable forms like

straight lines in noisy binary images. The Hough transform is truly a representation of the projection profiles of the document in every possible orientation. The accumulator array can be checked for the particular orientation which has the maximum number of transitions to and from a minimum value. Transitions corresponding to text are usually regular and uniform in width and thus are easy to identify. Maximum values are registered at the center line of the characters and slightly lower values correspond to the ascender and descender lines. The analysis of the accumulator array has an added advantage. It can provide the angle of skew in the document. Since the segmentation phase assumes that blocks align parallel to the page margins, skew detection and correction is an essential preprocessing step.

Bottom-up methods

The image is first processed to determine the individual connected components. At the lowest level of analysis, there would be individual characters and large figures. In the case of text, the characters are merged into words, words are merged into lines, lines into paragraphs etc., .

The application of different operators for bottom-up grouping can be co-ordinated by a rule based system. Most characters can be quickly identified by their component size. However, if characters touch a line, as is often the case in tables and annotated line-drawings, the characters have to be segmented from the lines. One technique for segmenting characters from line structures is to determine the high neighborhood line density (NLD) areas in the line structures. The following is an example of the type of rules that can be used.

1.4 Image analysis

Image analysis involves the examination of image and extract the constituent parts of the image.

It consists of two phases.

- Feature extraction

  It is the process of extracting geometric information describing the orientation and layout of characters present in the image.

- Pattern classification

  It is the process of identifying the classes into which the extracted patterns fit into and assigning suitable labels for the patterns describing meaningful entities.

1.5 Applications of OCR

A part from being used as a document storage system it also eliminates laborious and repetitive work needed to index documents.OCR systems have many other applications.

A few of them are

- Use in postal department for postal address reading and as reader for handwritten and printed codes.
- Use by blind people as a reading aid when synthesized with the speech systems.
- In automated cartography, information units and libraries for facsimile.
- For business applications - financial business applications like optimization strategy for cheque sorting .
- In law enforcement applications.
- In educational administration for examination assessment and for attendance record evaluation.
- For digital bar code reading and as hand writing analyzer for automatic recognition of hand writing and signature verification.
- Use in customer billing for telephone billing system.

- Use as a credit card scanner in credit personnel identification system.

1.6 Motivation

In this Internet age, we come across various types of text/multimedia documents rich in content and layout. We need to manage the storage, retrieval, processing and transmission of such documents in an easy and efficient manner. The latest developments include digital libraries where documents are scanned and stored in digital form and are ready for access by anyone from anywhere in the world. In several automated applications like bank cheque processing, bar code reading, credit card scanning, address reading on postal items we need to automate the processing in an efficient manner. The traditional text-only document can no longer satisfy the needs of the people. Due to the tremendous growth of Internet, multimedia documents are gaining importance. In a multimedia document, the video information revealed by the document consists of a mixture of text, graphics and image. We need to isolate text from graphics/image, extract text from paragraph, extract lines from paragraph, seperate words from a line and finally extract characters from the word. For doing the above task of extracting entities at the character level we need to have tools which can handle the processing of document images in an efficient manner. The Image Processing Tool-Kit[30] (IPTK) provides the basic functions for doing the initialization task, reading pixel values, storing pixel values, allocating memory and other basic functions. While document image processing is certainly an aspect of image processing, here we are adding functions which are of more common use mainly in document image processing. This would allow to build more sophisticated programs for Document Image Processing(DIP) by making use of functions from a callable library just like the Image Processing Tool Kit(IPTK).

1.7 Organization of thesis

CHAPTER 1 This chapter explains the general concepts of document image processing and the applications of OCR

CHAPTER 2 This chapter explains skew detection and correction in document images, document binarization and projection profiles construction.

CHAPTER 3 This chapter explains run length smearing, connected component extraction and text/image classification in document images.

CHAPTER 4 This chapter discusses the implementation details of the various functions for document image processing.

CHAPTER 5 This chapter gives guidelines for testing the various functions.

CHAPTER 6 This chapter gives the conclusion and the future work to be done in document image processing.

Chapter 2

PREPROCESSING

In this chapter, we describe the various preprocessing methods needed to be applied on a document image like potential text block identification, skew detection and correction and binarization. We explain features like projection profiles, crossing counts and circumscribed rectangles.

2.1 Introduction

Optical scanning is the commom method by which we capture data from a paper document. The scanned data is stored in a file of picture elements, called pixels, that are sampled in a grid pattern from each page of the document. These pixels have value of 0 or 1 for binary images and 0 to 255 for gray scale images.

The preprocessor takes a binary image as its input and produces the separated character and graphics MBR(Maximum Bounding Rectangle) as its output. The preprocessing on a scanned document is performed in three steps. The first step identifies the potential textual block in the scanned image. The second step estimates and corrects the skew present in the scanned document and the third step removes noise.

2.2 Basic features

When a document image is being processed, we first segment the document into different area types such as headlines, text lines and graphic areas using features which reflect the global properties of the document. The text lines are extracted using features like the detailed shape of the document components. For extracting features from the document components, three feature extraction methods exist namely projection profiles, crossing counts (both represent global features of the document) and position/size of circumscribed rectangles (which give precise information of each document component). The feature extraction methods are described in detail below.

2.2.1 Projection profiles

A projection profile[2] is obtained by determining the number of black pixels that fall on a projection axis. The horizontal projection profile *PPh* is obtained by doing a row-wise left to right scan, while the vertical projection profile *PPv* is obtained by doing a column-wise top to bottom scan. This feature plays a very important role in skew normalization, document component extraction and character segmentation are shown in figures 2.1 and 2.2 respectively.

A projection profile is expressed is as follows.

$$PPh(i) = \sum_{j} F(i,j) \qquad \textit{horizontal projection profile} \qquad \textbf{(2.1)}$$

$$PPv(j) = \sum_{i} F(i,j) \qquad \textit{vertical projcction profile} \qquad \textbf{(2.2)}$$

Where *F(i,j)* is the pixel value (0 for white, 1 for black) of the document image, and i, *j* denote the horizontal and vertical coordinates of the pixel respectively when the top left corner of the image is set to F(0,0).

2.2.2 Crossing counts

A crossing count[2] is obtained by counting the points at which the pixel value turns from 0 (white) to 1 (black) and vice-versa on horizontal *(CCh)* or vertical *(CCv)*

raster scan lines. Crossing counts are used to measure the gray level distribution of the document.

A crossing count is expressed is as follows.

$$CCh(i) = \sum_{j} \overline{F(i,j)} * F(i,j+1) \qquad \textit{Horizontal crossing counts} \qquad (2.3)$$

$$CCv(j) = \sum_{i} \overline{F(i,j)} * F(i+1,j) \qquad \textit{Vertical crossing counts} \qquad (2.4)$$

2.2.3 Circumscribed rectangles

A circumscribed rectangle[2] contains a connected area of black pixels in a document and is used to exactly represent the area's position and size. It can be obtained by contour line tracing. Each circumscribed rectangle is expressed by four values x, y, *h* and *w* i.e., the coordinates of the top left corner of the rectangle and its height and width.

2.3 Skew in Document Images

When the document image is scanned, it may be skew because of some reasons. Skew in a document image(see fig 2.3) can be a serious problem in document analysis, such as incorrectness of character segmentation and recognition. Hence skew detection and reconstruction should be performed before document analysis.

2.4 Previous work

2.4.1 Hough transform

Hough transform in a common technique to detect the skew angle of a document image. Nakano, Srihari, Govindaraju and Hindis[33] proposed skew detection techniques based on Hough transform. The Hough transform maps each point in the original (x,y) plane to all points in the (P,0) Hough space of lines through (x,y) with slope

Figure 2.1: Binarized document image and its horizontal projection profile

9 and distance P from the origin. The dominant lines are found from peaks in Hough space and thus the skew.

One limitation to the above method is that if the text is too sparse, it may be difficult to choose a peak in the Hough space. It is time-consuming and is sensitive to non-text areas and noises.

2.4.2 Projection profiles

A series of projection profiles[2] are obtained at a number of angles close to the expected orientation, and the variation is calculated for each of the profiles. The profile which gives the maximum variation corresponds to the projection with the best alignment to the text lines. This projection angle is the skew angle.

Baird[5] proposed an algorithm to determine the skew angle using an energy function on sets of projection counts of character locations. But this method is also sensitive to non-text areas.

2.4.3 Projection histogram

Ciardiello[10] has suggested a method using the projection histogram in which a sample region with the maximum average density of black pixels per row is rotated

Figure 2.2: Vertical projection profile

by pre-specified angles. The horizontal projection histogram of the region is evaluated for each angle. The skew angle corresponds to a rotation for which the mean square deviation of the histogram is maximised.

2.4.4 Connected component analysis

Hinds[16] has suggested a method for skew angle detection using connected component analysis and line fitting. It is time-consuming and sensitive to non-text areas and noise.

2.4.5 Gradient direction information

Skew angle[31] is obtained by searching for a peak in the histogram of the gradient orientation of the input gray level image. Skewness is corrected by rotation at such an angle.

2.5 Algorithm implementation details

Yi-Kai chen and Jhing-fa Wang[9] identified the skew in document image and corrected it based on maximization of variance of transition counts. The method is robust and efficient and is insensitive to non-text areas like graphics, images and

high noise present in gray scale images. The document may contain less text areas, high noises, tables, figures, flow-charts and photos. In addition the page orientation (column-wise or row-wise) can also be detected [9].

The method can be divided into two parts:

- ***Maximization of variance of transition counts for skew detection and page orientation determination***
- ***Skew reconstruction using scanning line model***

2.6 Algorithm

1. ***Maximization of variance of transition counts for skew detection and page orientation determination***

   The transition counts (TC) on each scanning line is defined as the number of transition phenomenon (pixel from black to white or white to black) on the scanning line.

   The basic idea of using transition-counts variance to detect the skew angle is the fact that peaks of the transition-counts histogram at the skew angle appear periodically and the transition-counts histogram at the other angles look flat. That is, the transition-counts variance will be the maximum at the skew angle and this is sufficient to detect the skew angle and the page orientation.

   Before proceeding to compute the *TCV*(Transition count variance), we need to identify a potential text block containing enough text so that we can apply our method to detect the skew angle in an easy and efficient manner.

2. ***Potential textual block identification***

   The scanned document image in general has many heterogeneous objects (i.e., non-textual objects like line drawing, continuous tone pictures etc.,) which adds to the inaccuracy while estimating the skew present in the scanned document.

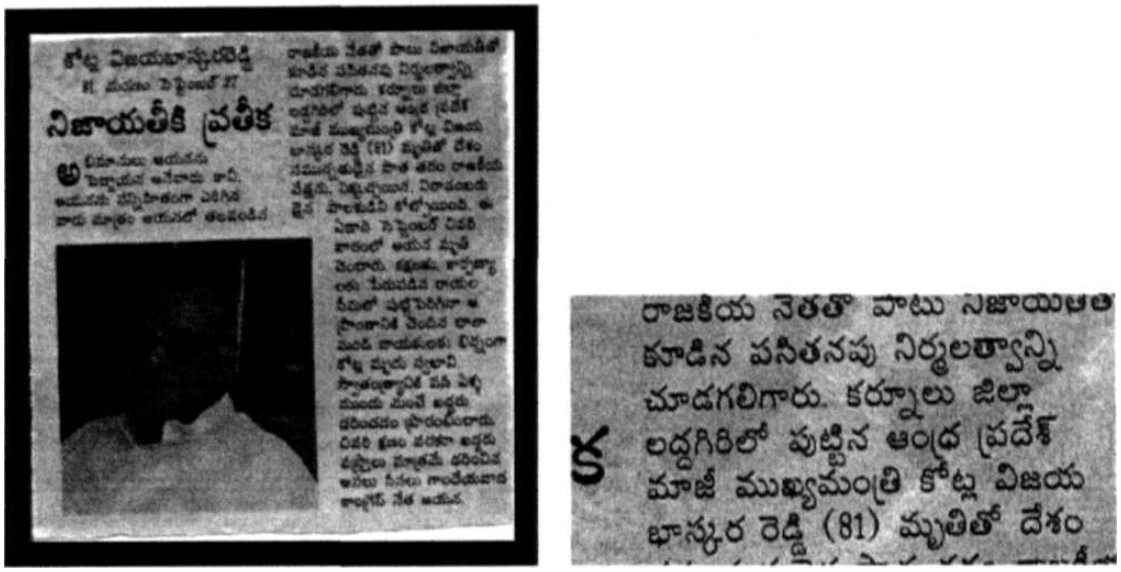

Figure 2.3: Potential textual block identification of a skewed document image

Thus it is necessary to identify a small potential text block, which improves the accuracy as well as increase **the** skew estimation speed.

The textual block is identified as follows.

We choose a strip of 256 pixels in horizontal direction at the middle of the document as well as a strip of the same width in vertical direction. Then we compute the total counts of transition in the strip. If the total counts of transition exceed the threshold from experimental results, then the strips are regarded as containing enough text and are used to compute transition-counts variance for skew detection. Otherwise the strips are shifted in horizontal and vertical direction until we confirm that the strips contain enough text.

2.7 Skew angle detection

After the potential text block is identified, we compute the transition-counts variance(TCV) at each angle from -45 degrees to +45 degrees by the following equations. Subscripts

Figure 2.4: A skewed document image

h and *v* stand for horizontal and vertical orientations respectively. Four equations need to be computed.

$$TCV_h[\theta] = \frac{\sum_{i=0}^{H-1}(TC_h[\theta][i] - TM_h[\theta])^2}{H} \tag{2.5}$$

$$TCV_v[\theta] = \frac{\sum_{j=0}^{W-1}(TC_h[\theta][j] - TM_h[\theta])^2}{W} \tag{2.6}$$

$$TM_h[\theta] = \frac{\sum_{i=0}^{H-1} TC_h[\theta][i]}{H} \tag{2.7}$$

$$TM_v[\theta] = \frac{\sum_{j=0}^{W-1} TC_v[\theta][j]}{W} \tag{2.8}$$

- *TCVh [0]* is the horizontal transition-counts variance at *9* degrees.
- TCV_V *[0]* is the vertical transition-counts variance at *9* degrees.
- *TCh [0] [i]* is the horizontal transition-counts of the i^{th} row at *0* degrees.
- TC_V *[0] [j]* is the vertical transition-counts of the j^{th} col at *0* degrees.
- *TMh [0]* is the horizontal transition-counts mean at *9* degrees.

Figure 2.5: Skew corrected document image

- TM_V ***[0]*** is the vertical trasition-counts mean at *0* degrees.

H is the image height and *W* is the image width. *0* varies from -45 degrees to +45 degrees. After the horizontal and vertical *TCV* at each angle are computed, the maximum *TCV* in horizontal and vertical directions are labelled as *TCVmaxh* and *TCVmaxv* respectively.

If *TCVmaxh* is larger than TCV_{maxv}, we can decide that the text is row-wise, and the skew angle is the angle with the *TCVmaxh*-

Otherwise, the text is column-wise and the skew angle is the angle with the *TCVmaxv*. In this way, the skew angle and page orientation can be determined at the same time.

2.8 Skew reconstruction using scanning line model

Scanning line model

Any digital straight line of slope *0* = arctan(A/B), can be expressed approximately by the chain code of the form *PP....QPP....Q* - $(P^mQ)^n$.

$$P = NINT\left(\frac{A}{B}\right). \tag{2.9}$$

$$Q = \begin{cases} P+1, & if A - B \times P > 0 \\ P-1, & if A - B \times P > 0 \end{cases} \tag{2.10}$$

$$m = NINT\left(\frac{B}{|A - B \times P|}\right) - 1. \tag{2.11}$$

$$n = |A - B \times P|. \tag{2.12}$$

where *NINT* stands for *nearest integer.*

2.9 Skew reconstruction

After the skew angle is detected, the scanning line model is used to calculate the vertical and horizontal offsets for skew reconstruction.

Using the chain code of the scanning line at the skew angle, with the parameters P, Q, m, n the vertical offset Y_shift_j of the jth column can be computed as follows.

$$Y\_shift_j = b \times (m+1) + \left[\frac{r}{P}\right], \quad if \left[\frac{r}{p}\right] < m \tag{2.13}$$

$$Y\_shift_j = b \times (m+1) + m, \quad if \left[\frac{r}{p}\right] \geq m \tag{2.14}$$

where b is $\frac{j-1}{m \times P+Q}$, r is $(j-1)\%(m \times P + Q)$

In the same manner,the horizontal offset X_shift_i of the ith row can be computec as follows.

$$X\_shift_i = b \times (m+1) + \left[\frac{r}{p}\right], \quad if \left[\frac{r}{p}\right] > m \tag{2.15}$$

$$X\_shift_i = b \times (m+1) + m, \quad if \left[\frac{r}{p}\right] \geq m \tag{2.16}$$

where b is $\frac{i-1}{m \times P+Q}$, r is $(i-1)\%(m \times P + Q)$

After skew reconstruction, the reconstructed image becomes larger than the original image.

The new width and height of the reconstructed image are calculated as follows.

$$newwidth \text{ — } oldwidth + Xshift_{oldheight;} \tag{2.17}$$

$$newheight \text{ — } oldheight + Y\ shift_{oldwidth;} \tag{2.18}$$

where the parameters $X\ shift_{oldheight}$ and $Y\ shift_{oldwidth}$ are the horizontal and vertical offsets at the positions of the oldheight and oldwidth respectively.

1. Then the pixel at the ith row and the jth column on the skew document with positive angle is mapped to (i + $Yshift_j$) *th* row and the *(j* — $Xshift_i$ + *X shift oldheight) th* column in reconstructed image.

2. In contrast,it is mapped to (i — $Yshift_j$ + $Yshift_{oldwith}$) *th* row and the *(j* — $Xshift_i$ + $X\ shift_{oldheight}$) *th* column in reconstructed image for the document with negative angle.

2.10 Another approach for skew detection

A more simpler approach for skew angle detection is to do a left margin search of the skew document and find out the points which constitute the left margin of the document. This approach is simpler compared to the above transition counts approach for skew angle detection.

2.11 Skew detection using left margin search

In this method we need to find the orientation of the leftmost margin with the vertical in the skew document. The method is described in detail as follows.

We do a column-wise scan from top extending to the bottom until we encounter a black to white transition which signifies a transition from the black background to the white foreground of the image. We store the positional information of that point. This point becomes one of the end points of the left margin of the skew document.

We compare the x value of this point with the x value of the center of the skew document. If the x value of the point is greater than that of the center, we confirm that it is the bottom end of the left margin, else it would be the top end.

If the point happens to be the top end, we can confirm that the document is skewed in the anti-clockwise direction , otherwise the document is skewed in the clockwise direction.

The second end of the left margin is identified as follows. If the first point is confirmed to be the top end of the left margin,we do a row-wise scan from the bottom extending to the top until we encounter a black (background) to white(foreground) transition which completes the search for left margin.

Otherwise, we do a row-wise scan from the top extending to the bottom until we get the point.

From the points information we compute the slope of the line joining the two points constituting the left margin, and estimate the skew as the angle made by the left margin with the vertical. We as well get the orientation of the skew document i.e., either clockwise or anti-clockwise.

2.12 Skew reconstruction by rotation through the identified skew

After the skew is identified through the left margin search, we rotate the skewed image to the correct orientation by considering the direction in which skewed image is present.

If the direction of the image is clockwise, we rotate the skewed image in Anti-clockwise direction through the measured skew angle and vice-versa.

We rotate the skewed image about its center by translating the center of the

skewed image to **the** origin, rotating the skewed image through the skew angle about the origin and finally translating the center of the image to its initial position.

During rotation, the background pixels might interfere with the foreground pixels resulting in a noisy image which is filtered by applying a median filter on the noisy deskewed image to get a noise-free deskewed image.

2.13 Comparison of two approaches for skew detection and correction

1. **First approach**

   In the first approach, the method is robust as it is efficient in reconstructing the skewed image even when it has a good amount of noise. Signal to noise ratio (*SNR*) is defined as the ratio of the transition counts of the original image and the transition counts of noises.

   SNR = (Transition counts of the original image)/(Transition counts of the noises)

   Since most noisy signals are random, the influence of noise to *TCV* at each angle is almost the same. So, the maximum *TCV* is invariant at the skew angle. If the *SNR* is grater than 1, the algorithm can give good result; but if the *SNR* is less than 1, some documents are successfully detected and some fail.

   Skew reconstruction is fast and efficient as we only need to find the horizontal and vertical offsets at the scanning line orientation. It has the advantages of dealing with large skew angles and is successful in handling documents with less text areas, high noises, tabular form and hand-written characters.

2. **Second approach**

   In the second approach, reconstructing the skewed image by rotating through the correct orientation is time consuming as it has to compute new pixel positions for all the pixels in the skewed image and are then transferred to the new

image. If there is more noise in the skewed image, the rotation might result in loss of clarity and it is directly related to the noise present in the document image.

Chapter 3

LAYOUT ANALYSIS

In this chapter, we explain document segmentation using run length smearing, component extraction using connected component analysis, paragraph seperation using stripe merging procedure and graphics/image classification using connectivity histogram procedure.

3.1 Introduction

The layout structure of a document considers the geometric properties of the document's primitive objects and the relations between them. Three different layout classes are generally distinguished i.e., text, graphics and photographic images. The layout analysis has the task of relating the primitive objects of the document to one of these three classes and to establish further structures within one class. Analyzing the layout structure is one of the first processing steps that interprets the primitive objects of the document as being textual components or part of a line drawing.

A feature-based document analysis system which utilizes domain knowledge to segment and classify mixed text/graphics/image documents is discussed. Due to the tremendous increase in storage, transmission and retrieval of multimedia documents which are composed of a mixture of text, graphics and image, a proper segmentation of such a document is a prerequisite to facilitate further processing like recognition of text,

vectorization of graphics and compression of image. After isolation of the constituent objects comprising a document is completed, it is advantageous to reproduce, transmit and store the document in the processed form as the processed results occupy less memory space.

3.2 Previous work

1. Wong etal[29]. proposed a document analysis system to process a technical document. He proposed a constrained run-length smearing algorithm which segments the document into several blocks and classifies them based on their geometric properties.

2. Nagy and Seth[20] proposed a technique called recursive x-y cut method to do segmentation and classification.

3. Srihari etal[ll] proposed a texture analysis based algorithm to classify text and image. The drawback in the above algorithm is that it fails to classify horizontal line and rule line.

4. Stephen and Srihari[37] presented a newspaper reading system by utilizing four filters to classify segmented blocks into different types of media.

3.3 Segmentation approach

The purpose of document segmentation [28] is to segment a document into several seperate blocks with each block representing one type of medium. We developed a document analysis system which would automatically segment the document image and classify text, graphics and image embedded in the document and analyze the structure of each segmented medium.

We perform a run-length smearing operation followed by the stripe merging procedure to segment the document. The advantage of our method over the previous ones is that it will generate an intact text block instead of several individual stripes.

It not only saves tremendous storage space but also relieves the burden in performing OCR(Optical Character Recognition).

Each segmented block is then fed to the block classification module to determine which type of medium it is. A feature-based block classification scheme is employed to classify the document into predetermined categories and extract useful information from each block. In the classification scheme, graphics and image are treated as two different media. It means that they should be processed using different methods.

Graphics should be processed using a vectorization procedure instead of a compression procedure to attain a good result in removing redundancy.

3.4 Document segmentation

The method we adopted in segmenting documents[23] is a combination of run length smearing algorithm and stripe merging procedure. We first perform the smearing operation in both the horizontal and vertical directions to generate blocks. Due to the selection of an improper threshold in the run-length smearing process, an intact paragraph might be segmented into several consecutive horizontal stripes with each stripe representing one horizontal text line.

The stripe merging procedure[23] then merges the text stripes that belong to the same paragraph. It will generate an intact text block instead of several smaller text stripes. By this it not only saves tremendous storage space but also relieves the burden in performing optical character recognition.

3.5 Run length smearing

Run length smearing[23] is an operation which connects two non-adjacent runs into one merged run if the distance between these two runs is smaller than a threshold. The smearing operation is first applied to a document row by row and then column by column to obtain two intermediate results. The two intermediate results are then combined by performing a logical OR operation to generate the final result. The

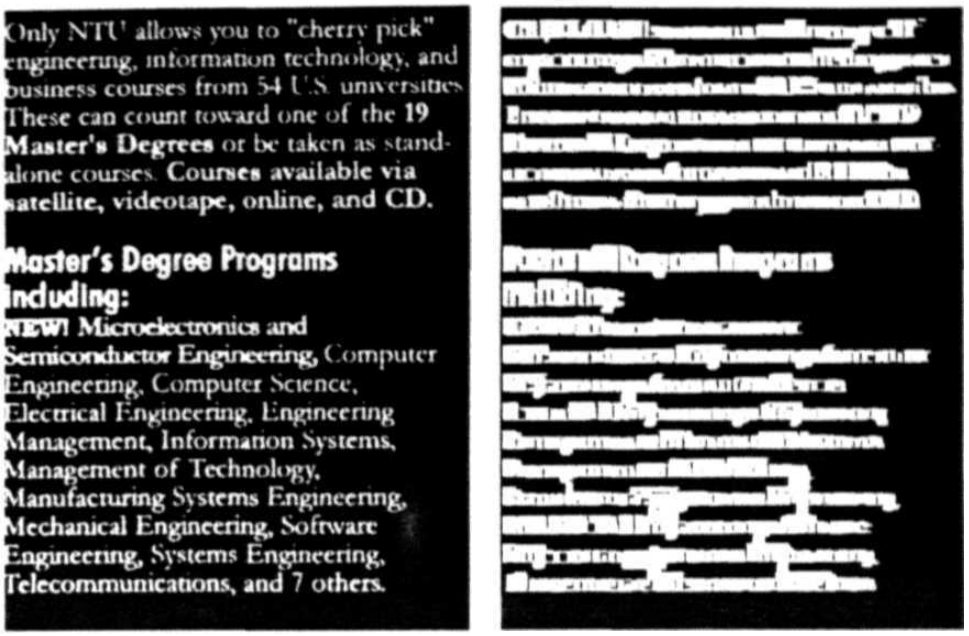

Figure 3.1: Smeared document image

connected component algorithm is applied to the final result to find the blocks. The detailed description of the run-length smearing algorithm is discussed in the following section.

3.5.1 Run length smearing algorithm

First, the smearing operation[23] is applied to the document in the horizontal direction row by row. Two runs having distance smaller than the threshold will be merged into one run.

Consider a string 5 $=(a_1\ldots,a_i,\ a_{i+1},\ldots,a_j\text{-}i,a_j,\ldots,a_n)$

with two 1-runs $r_1 = (a_1,\ldots,a_i)$ $r_3 = (a_j,\ldots.a_n)$

and one 0-run $r_2 = (a_{i+1},\ldots,a_{k-1})$.

The two 1-runs r1 and r_3 are merged if the value of j — i - 1 is smaller than t where j — i — 1 is the distance between r1 and r_3 and t is the preselected threshold. The contents of the 0-run r_2 are accordingly changed from 0 to 1 if runs r1 and *r3* satisfy the smearing condition.

To illustrate the smearing operation, we consider an example with threshold t being selected as 4. Two 1-runs are merged if the distance between these two runs is smaller than 4.

before smearing. 11111110000011111111100011

after smearing. 11111110000011111111111111

The result generated by the smearing operation is several 1-runs in each horizontal row. The smearing operation is then performed in the vertical direction column by column to connect two non-adjacent runs into one merged run if the distance between these two vertical runs is smaller than a threshold.

Two intermediate results are generated while the document is processed row by row and then column by column via the smearing operation. The two intermediate results are then combined by performing a logical OR operation to generate the final result.

Then, the connected component algorithm is applied to the final result to find the blocks.

3.6 Connected component

A connected component [21]is a grouping of contiguous pixels which symbolizes a logical entity (e.g., a character). Two pixels are said to be connected if there is a path of 1-pixels adjacent along the horizontal and vertical directions that links them. The connected component problem consists of assigning a label to each 1-pixel of a binary image so that two 1-pixels are assigned the same label if and only if they are connected.

3.6.1 Connected component extraction

The connected components is based on a two-pass algorithm. First the image is scanned in a top-bottom left-to-right fashion and a label is assigned to each 1-pixel based on the value of the adjacent 1-pixels already labelled. If there are no adjacent 1-pixels, a new label is assigned. Conflicting situations arise when a pixel is given two

different labels. Equivalent classes of labels are then constructed and stored to be used in the second pass of the algorithm to disambiguate such conflicts. During the second scan of the image, each label is replaced by the one selected as the representative of the corresponding equivalence class; for instance, the smallest one in the class if the labels are integers.

3.6.2 Connected component algorithm

- Step 1

  Scan the image top_down left_right
  for each row r of the image *do*
  Initialize the local equivalence table for row r
  for each 1_pixel of row r *do*
  if the upper and left adjacent pixels are all 0_ pixels
  then assign a new label
  if the upper and left adjacent pixels have the same label
  then assign this label
  if only one adjacent pixel has a label
  then assign this label
  otherwise assign the smaller label to the pixel and
  enter the two equivalent labels into the local table.
  Find equivalence classes of the local table.
  Choose a representative of each class.
  for each l_pixel of row r *do*
  replace its label with the smallest equivalent label.

- Step 2

  Scan the image bottom_up right_to_left
  for each row r of the image *do*
  Initialize the local equivalence table for row r.

for each 1_pixel of row r *do*
for each adjacent pixel with a different label *do*
enter the two equivalent labels into the local table.
Find equivalence classes of the local table.
Choose a representative of each class.
for each 1_pixel of row r *do*
replace its label with the smallest equivalent label.

If the considered block is an image or graphic block, then the result generated by the run-length smearing algorithm is the minimum enclosing rectangle covering the block. If the considered block is a text block (paragraph), then the result generated by the RLSA might be several consecutive stripes with each stripe representing one line of text. This will occur if the threshold value for the vertical smearing operation is smaller than the delimiting distance between two consecutive horizontal text lines.

3.7 Stripe merging procedure

Due to the RLSA, an intact text paragraph might be decomposed into several narrow stripes instead of an intact text block. It not only increases the storage demand but also wastes time in retrieval/storing data while performing OCR. The stripe merging procedure[23] is thus applied to merge long stripes of rows which belong to the same paragraph into a single stripe denoting an intact text paragraph.

The criterion for judging whether to apply the stripe merging procedure depends on the block size generated by the run-length smearing algorithm. If the block size of the smearing result is much larger than the preselected threshold, abandon the stripe merging procedure as the result generated is the final segmented result. Otherwise, perform the stripe merging procedure to merge those stripes belonging to the same paragraph into one block.

The principle of stripe merging is that two text stripes are merged if they belong to the same text block, i.e., belong to the same paragraph from the grammatical point

of view.

Assume the current stripe S_i is represented by two co-ordinates which are the upper-left co-ordinate (Xul_i, Yul_i) and the lower-right co-ordinate (Xlr_i, Ylr_i).

Accordingly, its previous stripe S_{i-1} and its next stripe S_{i+i} are characterized by the two coordinate sets which are $(Xul_{i-1}, Yul_{i-1}), Xul_{i-1}, Yul_{i-1})$ and (Xul_{i-1}, Yul_{i-1}), (Xlr_{i+1}, Ylr_{i+1})respectively. Stripes S_i and S_{i-1} or S_i and S_{i+1} are merged if they belong to the same text block.

There are three possible ways in delimiting two text blocks and this knowledge is utilized in deciding whether two text stripes can be merged or not.

- Case 1:

  Since two paragraphs are usually delimited by wider space, we can utilize this property to determine if two stripes S_i and S_{i-1} belong to the same text block.

  Stripes S_i and S_{i-1} do not belong to the same text block if the vertical distance between these two stripes is larger than a threshold.

$$|Yul_i - Ylr_{i-1}| > threshold \rightarrow S_i \otimes S_{i-1} \quad (3.1)$$

  where ® denotes "do not belong to the same text block".

- Case 2:

  Since indent usually appears at the beginning of the first line in a new paragraph, it can be utilized to decide if two stripes S_i, and S_{i-1} belong to the same text block.

  Stripes S_i and S_{i-1} do not belong to the same text block if indent occurs at the head of the stripe S_i.

$$|Xul_i - Xul_{i-1}| > threshold \rightarrow S_i \otimes S_{i-1} \quad \textbf{(3.2)}$$

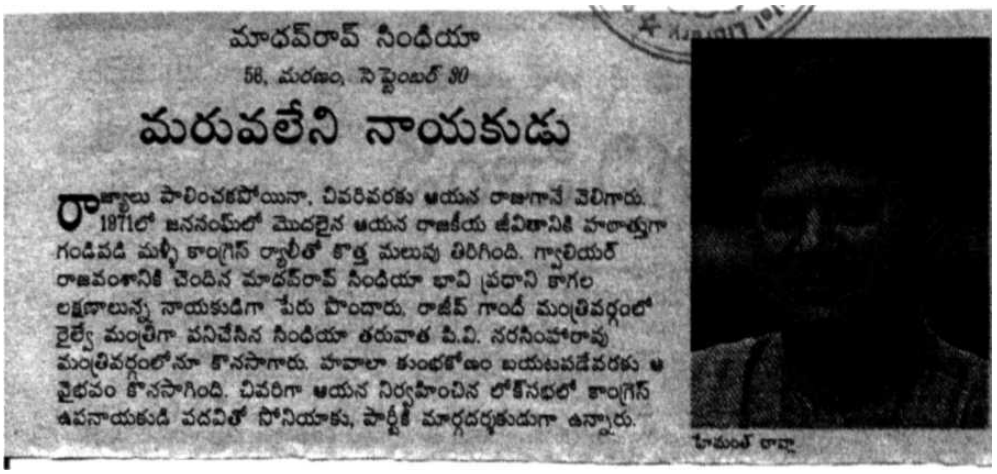

Figure 3.2: A document image for block classification

- Case 3:

  Since indent usually appears at the end of the last line in the current paragraph, it can be utilized to determine if two stripes S_i and S_{i+1} belong to the same text block.

  Stripes S_i and S_{i+1} do not belong to the same text block if indent occurs at the end of the stripe S_i.

$$|Xlr_i - Xlr_{i+1}| > threshold \text{ -+ } S_i \text{® } S_{i+1} \quad (3.3)$$

Two text stripes are merged if both of the following two conditions are satisfied.

1. The projection of these two stripes in the vertical direction overlap.
2. None of the above three cases occurs.

3.8 Block classification

After applying the document segmentation process,a document is segmented into several blocks with each block representing one type of medium. Each segmented

మరువలేని నాయకుడు

Figure 3.3: Text extracted document image

block is then fed to the block classification [23] module to determine which type of medium it is.

A feature-based block classification scheme is employed to classify the document into predetermined categories. Once the type of each block is determined, we can then determine which procedure needs to be triggered to process the block.

There exist different types of procedures to process different types of blocks.

1. For processing text blocks, we need to apply optical character recognition procedure.

2. For processing graphic blocks, we need to apply vectorization procedure.

3. For processing image blocks, we need to apply compression procedure.

3.8.1 Text classification

Classification of text[23] is done by evaluating the block size before applying the stripe merging procedure because the size of a text stripe is much smaller than that of a graphic or an image block. But some media rule line or small image/graphic block will be misclassified if the above rule is applied.

To get good classification result, we need to find a suitable feature of the text blocks and do the classification taking that feature into consideration. Text blocks

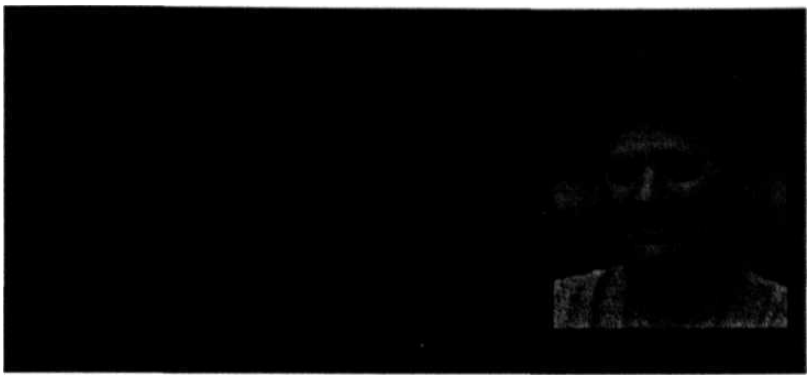

Figure 3.4: Image extracted document image

exhibit an important property, that is, the y-projection of a text block exhibits a certain periodic behaviour due to the limiting space that exists between two horizontal lines. The periodic pattern of text constitutes a feature, called projection feature, that exhibits itself in repeated form from some to none. The classification of text can thus be accomplished by utilizing the projection feature.

If the y-projection of a block exhibits periodic behaviour, then it is classified as a text block. Otherwise, it is classified as a non-text block which might be either an image or a graphic block.

3.8.2 Graphics and image classification

Earlier graphics and image[23] were treated as the same medium type, so they were processed using the compression technique to remove the redundant data. It is inadequate to process graphics using the compression technique as it will not attain the best result in removing redundancy. To get the optimal result, graphics should be processed using the vectorization technique and image should be processed using the compression technique.

We use the following notion to process graphic and image blocks. Graphics is a type of medium with sparsely distribution of black pixels, whereas image is a type of medium with densely distribution of black pixels.

We can define a graphic block as the block with sparsely distribution of value

"0" pixels and an image block as the block with densely distribution of value "1" pixels. We need to evaluate the black pixel (value "1" pixel) distribution in the block to do the classification.

The following steps need to be followed to do the classification of graphics and image.

- Step 1

  Build a 3 x 3 mask centered at each pixel in the considered block.

- Step 2

  Calculate the number of black pixels connecting to the considered pixel inside the mask. Each pixel will attain a value indicating the number of black pixels connecting to it. We call this value as the *connectivity value* of the pixel and denote it by *CV*. We find the *CV* of a pixel at position i and *j* as follows.

$$\begin{aligned} CV(i,j) = {} & P(i-1, j-1) + P(i,j) + \\ & P(i+1, j-1) + P(i-1, j) + P(i+1, j) + \\ & P(i-1, j+1) + P(t, j+1) + P(i+1, j+1) \end{aligned} \quad \textbf{(3.4)}$$

  where *P(i,j)* denotes the value of the pixel *{i,j).*

- Step 3

  Generate the *connectivity histogram* by summing the number of pixels with the same connectivity value.

- Step 4

  The connectivity histogram is used as the decision-making criteria on which the classification of graphics and image is done.

  The criteria for distinguishing graphics and image are detailed below.

  1. ***If the connectivity histogram of a non-text block is leftwards distributed, then the block is classified as a graphic block.***

2. *If the connectivity histogram of a non-text block is rightwards distributed,then the block is classified as an image block.*

To judge whether the connectivity histogram is leftwards distributed or rightwards distributed, a feature called *connectivity histogram value* numerizing the connectivity histogram is devised to do the classification task. It is obtained statistically from the connectivity histogram according to the following formula.

$$CHV = \frac{\sum(N(i) * i)}{\sum(N(i))} \qquad \textbf{(3.5)}$$

where *N(i)* denotes the number of pixels whose connectivity value equals i. The value of i ranges from 0 to 8. Since *connectivity homogeneous value* characterizes the black pixel distribution of the segmented block, it can be used to do the classification of graphics and image.

If the *connectivity homogeneous value* of the considered block is larger than the pre-selected threshold, then the block is classified as an image block. Otherwise, it is classified as a graphic block.

Chapter 4

IMPLEMENTATION DETAILS

In this chapter, we explain the working aspects of the various functions for document image processing.

4.1 Binarization

Since a document consists of different parts with varied gray level distribution, we adopted an adaptive thresholding approach which binarizes the document image suitably.

void HDAR-ThreshLimit(THEADER *it,int *limit);

Function description:

It takes the input image header which holds the entire information regarding the input image and writes the threshold limit to the "limit" argument passed as a parameter. The threshold limit is chosen based on the gray level distribution of the image being processed. If the gray level distribution is uniform then we choose the average of total gray values as the limit value. Otherwise, we choose the most frequently occuring gray value as the limit value.

void HDAR.Thresh(THEADER *it,int limit,int •thresh);

Function description:

It takes the input image header and the limit value and writes the threshold to the "thresh" argument passed as a parameter. The lower limit (threshold value) is obtained by computing the average of all the gray values which are greater than the limit.

void HDARJBinarize(THEADER *it,int thresh,THEADER *ot);

Function description:

It binarizes the document image by setting to 0 all gray values which lie between the lower limit (threshold value) and the higher limit. The higher limit is chosen to be 255. All the remaining gray values are set to 255.

4.2 Image resize

We need to resize the document image so that while applying image transformations on the resized image no loss of data occurs.

void HDAR_Resize(THEADER *it,THEADER *ot);

Function description:

It takes the input image header and extracts the number of rows and columns of the image. It computes the size of the output image as sqrt(2) times the input image if the input image is a squared one. If the input image is a non-squared one, we compute the output image size as 1.5 times the input image.

4.3 Image rotation

We need to rotate the image to the proper orientation to do skew correction after estimating the skew.

void HDAR_Rotate(THEADER *it,int skew, THEADER *ot);

Function description:

It takes the input image header and computes the center of the skewed image. It then rotates the input image about the center by applying the rotation transformation. The rotation about the center is carried as follows. First the center of the image is translated to the origin. Then rotation is performed about the origin by applying the affine transformation for rotation. The center of the image is then translated back to its original position.

4.4 Transition counts variance computation

We need to compute the transition count variance(TCV) along the horizontal and vertical directions so that it can be used in estimating the skew present in the document.

void HDAR_Compute TranCountVar(THEADER *it,int *htcvar,int *vtcvar);

Function description:

It takes the input image header holding the binarized image and computes the transition count variance along the horizontal and vertical directions.

4.5 Potential textual block identification

We need to identify a potential text block which can be used in the skew detection process to estimate the skew.

void HDAR_PotentTtextBlock(THEADER *it, THEADER *ot);

Function description:

It takes the input binarized image **and** identifies a potential text block whose transition count variance is greater than a given threshold.

4.6 Skew detection from the potential text block

We estimate the skew present in the document by computing transition count variance of the text block at various orientations along the horizontal and vertical directions and get the orientation which maximizes the transition count variance.

void HDAR_SkewDetect(THEADER *it, int *skew);

Function description:

It takes the input image header holding the potential text block and computes transition count variance at different angles along the horizontal and vertical directions and gets the orientation which maximizes the transition count variance along either of the directions. If the *TCV* is maximum along the horizontal direction, we confirm that the document is row-wise oriented, if the *TCV* is maximum along the vertical direction, we confirm that the document is column-wise oriented.

4.7 Chain code generation

We need to generate the chain code for a line which is at a given orientation with the horizontal/vertical. This chain code compactly describes the layout of the skewed line. It is used in skew reconstruction because the skewed line describes the orientation of the entire document and one can use the chain code parameters for skew reconstruction.

void HDAR_ChainParams(THEADER *it,Chaincode *cptr);

Function description:

It takes the input image header holding the skewed image and finds the end-points of the line which acts as a representative for the entire skewed document. The chain code parameters are constructed and these can then be used to compute the offsets for the reconstructed image.

4.8 Skew reconstruction

While reconstructing the skewed image we take the chain code parameters and compute the offsets for the deskewed image.

void HDAR_SkewConst(THEADER *it,Chaincode *cptr);

Function description:

It takes the input image header holding the skewed image and computes the horizontal and vertical offsets for each row and each column using the chain code parameters pointed to by cptr. It also computes the new height and new width of the reconstructed document image from the computed offsets.

4.9 Skew image orientation

We need to find the orientation of the skewed image by finding the angle made by the left margin with the vertical.

void HDAR_SkewOrient(THEADER *it,int *skew);

Function description:

It takes the input image header holding the skewed image and finds the end-points of the left margin. It then computes the skew of the document by evaluating the angle made by the left margin with the vertical.

4.10 Skew detection using left margin search

In this, we identify the skew of the document using left margin search and also the orientation of the document i.e., either clockwise or anti-clockwise.

void HDAR_SkewDetectLeftSearch(THEADER *it,int *skew,int *orient);

Function description:

It takes the input image header holding the skewed image and computes the angle made by the left margin with the vertical. It also finds the orientation of the skewed document by finding the boundary points of the skewed document.

4.11 Skew correction by rotation

We correct the skew in the document image by rotating the skewed document to the correct position using the skew and the orientation obtained from the left margin search.

void HDAR_SkewCorrect(THEADER *it,int skew,int orient,THEADER *ot);

Function description:

It takes the input image header holding the skewed image, the identified skew, the orientation and rotates the image about the center through the correct orientation (clockwise or anti-clockwise) by the identified skew.

4.12 Horizontal and vertical projection profiles

A projection profile is obtained by determining the number of black pixels that fall on a projection axis. The horizontal projection profile is obtained by doing a row-wise left to right scan whereas the vertical projection profile is obtained by doing a column-wise scan top to bottom on a binarized image.

void HDAR_HorizonProfiles(THEADER *it,THEADER *hpf)

Function description:

It takes the input binarized image as input and builds the horizontal projection profiles for the image.

void HDAR_VerticalProfiles(THEADER *it,THEADER *vpf)

Function description:

It takes the input binarized image as input and builds the vertical projection profiles for the image.

4.13 Run length smearing

Run length smearing is an operation where two non-adjacent white (gray value 255) runs are combined into a single white run if the distance between the two runs is less than some threshold. It is done row by row and then column by column to get two intermediate results. The final result is obtained by performing a logical OR operation on the two intermediate results.

void HDAR_HorizonSmear(THEADER *it,int rowthresh,THEADER *ot);

Function description:

It takes the input binarized image and the row threshold value as input **and** writes the resultant, smeared image into the output image file.

void HDAR_VerticalSmear(THEADER*it,int colthresh,THEADER *ot);

Function description:

It takes the input binarized image and the column threshold value as input and writes the resultant smeared image into the output image file.

void HDAR_CombinedSmear(THEADER ***ttl,THEADER *tt2,THEADER •at);**

Function description:

It takes the horizontal and vertical smeared images as input and writes the resultant smeared image into the output image file.

4.14 Text and Image seperation

Text and image are seperated by doing block classification from the binarized document image. The binarized document image is first smeared followed by connected component extraction which identifies the text and image parts. Then the Transition Count Variance(TCV) is evaluated for the text and image parts seperatly and based on the *TCV* the image parts are separated from the text parts.

void HDAR_TextImage(THEADER *it,struct Blocklnfo *Blockptr, THEADER *text,THEADER *image);

Function description:

It takes the smeared document image and the block information as input and writes the extracted text and image parts to "text" and "image" files respectively.

Chapter 5

TESTING and RESULTS

In this chapter, we explain the usage of various functions for different operations on a document image and the results obtained after applying them.

5.1 Preprocessing

5.1.1 Skew detection and correction

The scanning line model at the skew angle is used for skew reconstruction

1. The amount of skew present in the document image is obtained from the orientation made by the top margin with the horizontal.
2. We get the chain code parameters of the top margin.
3. Prom the chain code parameters, we compute the vertical and horizontal offsets of the pixel positions in the newly constructed image. We update the heights of the constructed image accordingly.
4. Then, the pixel values are transferred from the skewed image to the reconstructed image.

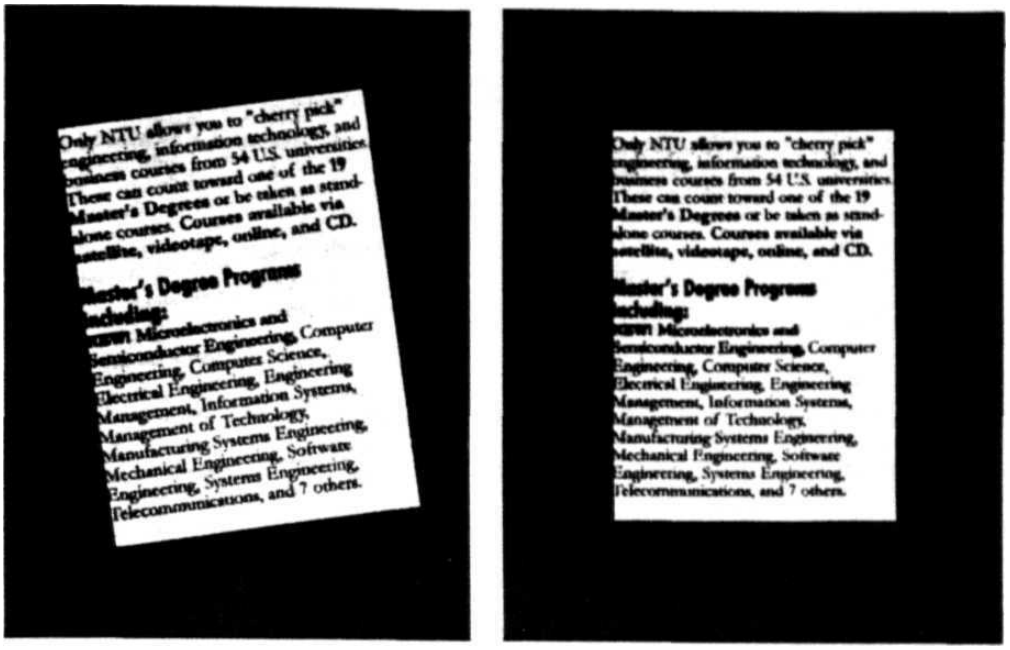

Figure 5.1: Skewed and skew corrected images

The following function calls need to be used for the above process.

```
struct ChainParams{
struct ChainPts points[2];
int ang;
int orient;
} *cptr;
HIPLJnitialize(it,inputfile,0,0,ID);
HIPL_AllocateMem(it);
HDAR_SkewConst(it,cptr,ot);
HIPL_StoreResult(outputfile,ot);
```

where it and ot are the input and output THEADERs,inputfile is the skewed document image file, cptr is the pointer to the ChainParams structure which holds the end points information of the top margin, "ang" in the above structure gives the angle made by the top margin with the horizontal, "orient" gives the orientation of the skewed document i.e., either clockwise or anti-clockwise.

5.1.2 Binarization

The document image is binarized using the adaptive thresholding technique. Function call sequence is as follows.

```
HDAR_ThreshLimit(it,limit);
HDAR_Thresh (it,limit,thresh);i
HDAR_Binarize(it,thresh,ot);
```

where it and ot are the input and output THEADERs, limit is the average of the pixel values in the image or the most frequently occuring value in the image.Either one of them is chosen depending on the type of the document being processed. Using this limit, we set the threshold for the entire document image by computing the average of all pixel values which are above the limit. This threshold value is used as the lower limit for the binarization process whereas the upper limit is chosen to be 255. All pixel values which are in the range of lower limit and upper limit are set to 0 and the others are set to 255.

5.1.3 Projection profiles

The horizontal and vertical projection profiles are obtained from the binarized image by computing the row-wise and column-wise white pixel counts and constructing profile images from the obtained counts.

```
HDAR_HorizonProfiles(it,hpf);
HDAR_VerticalProfiles(it,vpf);
```

where it, hpf and vpf are THEADERs hold information about the input image,horizontal and vertical profiles respectively.

5.1.4 Run length smearing

The document image is smeared by applying the smearing operation along the horizontal and vertical directions by choosing horizontal and vertical smearing thresholds

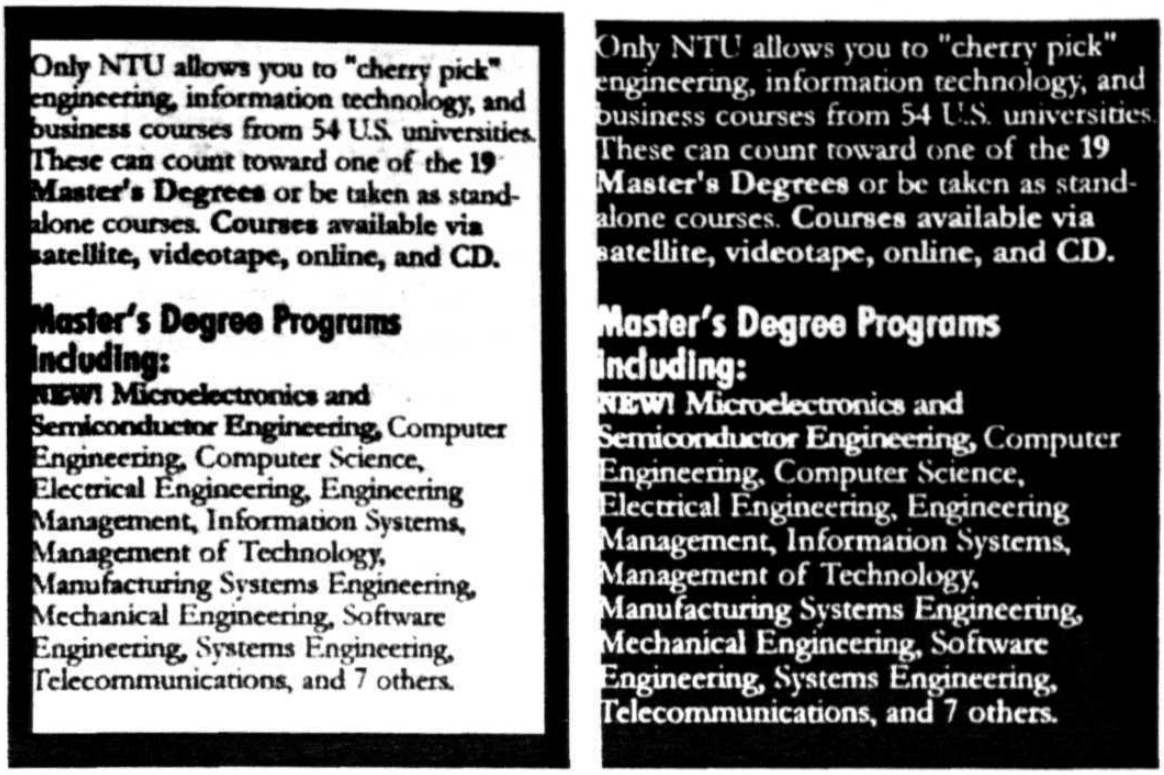

Figure 5.2: Binarized document image

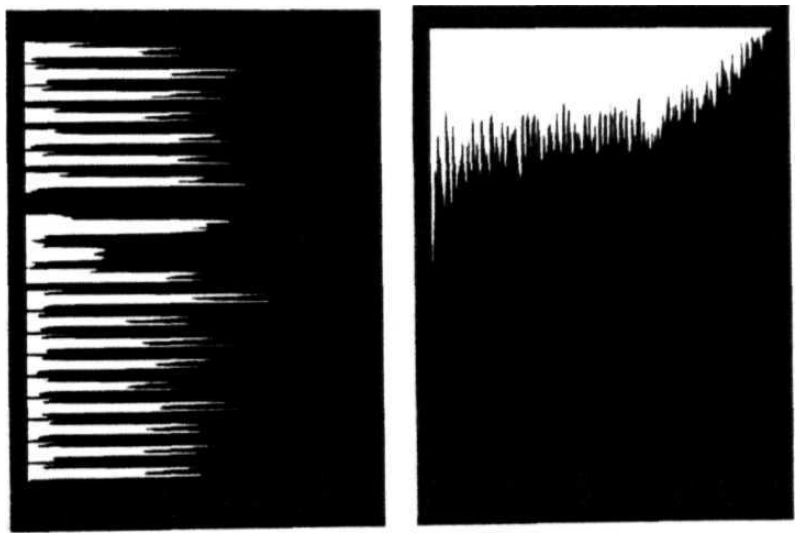

Figure 5.3: Horizontal and vertical projection profiles

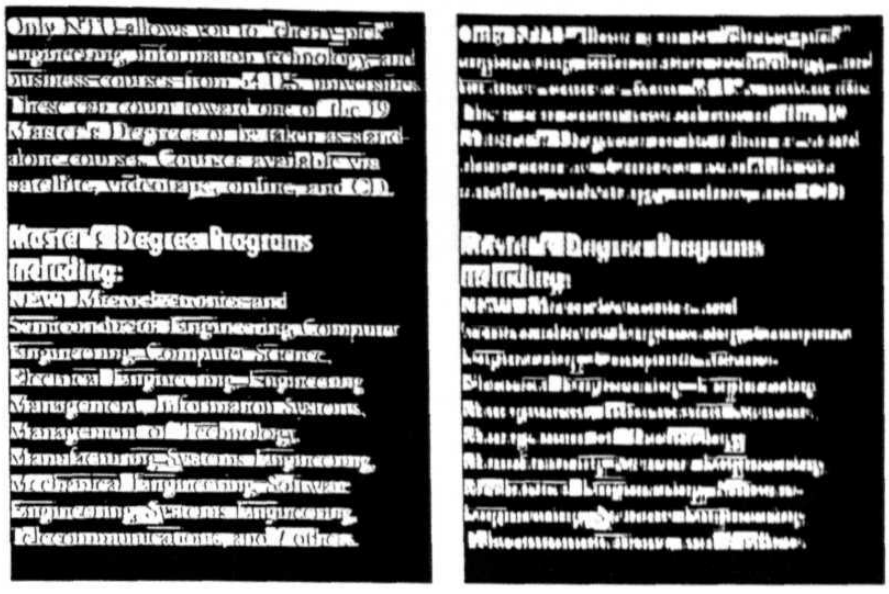

Figure 5.4: Horizontal and vertical smeared document images

appropriately.After obtaining the smearing results along the two directions, the two are logically combined to get a combined smeared image.

```
HDAR_HorizonSmear(it,rowthresh,otl);
HDAR_VerticalSmear(it,colthresh,ot2);
HDAR_CombinedSmear(otl ,ot2,ft);
```

where it is the input image,otl and ot2 are the two smeared images and ft is the combined smeared image.

5.2 Text and image separation

Text and image are separated by applying run length smearing followed by connected component extraction. From connected component information, we separate the text and image parts by computing the Transition Count Variance(TCV) **for** the component parts.

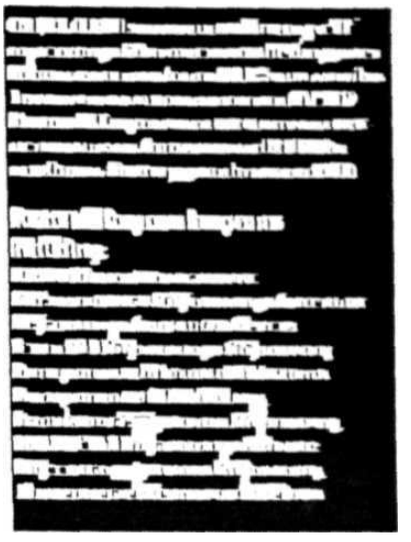

Figure 5.5: Combined smeared document image

HDAR_TextImage(it,blockinfo,text,image);

where it, text and image represent the original, text extracted and image extracted document files respectively.blockinfo is a pointer to the component block information.

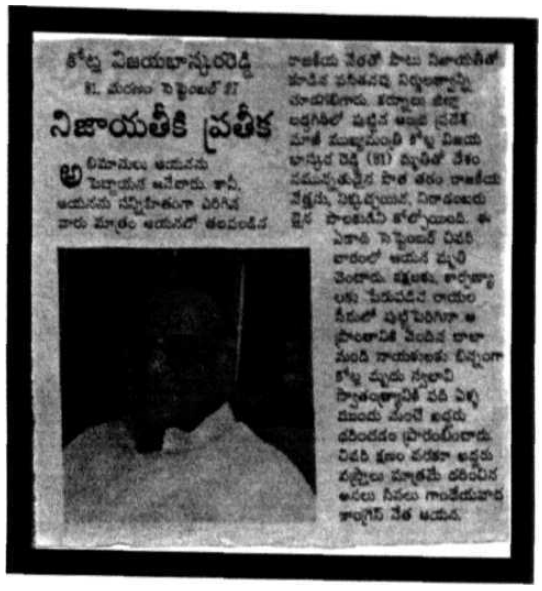

Figure 5.6: Original document image

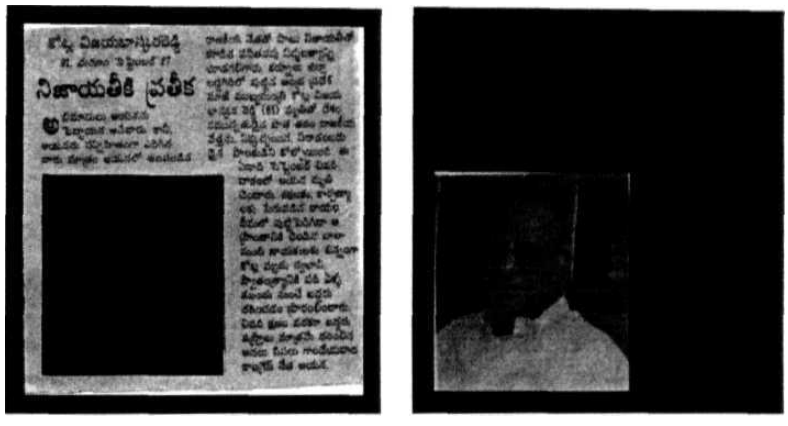

Figure 5.7: Text and image separated document images

Chapter 6

CONCLUSION and FUTURE WORK

In the pre-processing phase of document image processing we developed functions which perform skew detection and correction in an efficient and easy manner. Good results are obtained by processing a variety of printed and hand-written documents and the processing times are compared for the various types of documents. In the layout analysis phase we developed functions which do paragraph separation, line separation, and text/graphics seperation. Lines are seperated by applying run length smearing and paragraphs are seperated by applying stripe-merging procedure. Graphical parts of a document include line drawing, tables, flow-charts and continuous **tone** pictures (images). Graphical parts are separated from the text parts by applying the connectivity histogram procedure. In future the developed functions can be extended to do further processing on the extracted components of the document image. The operations which can be performed include line processing on the line drawings, tables, flow-charts and symbol/region processing on graphical objects to extract more information on the type of the objects. Document page layout analysis can be done which builds a logical structure of the document like title section, sub-section etc., for a page and group the different parts of a paragraph into a coherent entity. The extracted text can then be given as input to the OCR.

Appendix A

A PGM file format

The acronym PGM stands for Portable Gray Map. The PGM format is very simple and easy to deal with. It has a header which (such as *p2* or p5) signifies the type of the file (either ASCII encoded or binary encoded file). Generally a comment preceded by a # is present in the second line of the file. In the third line it has an integer specifying the number of gray levels in the image. In the next line it contains two integers, which specify the number of rows and number of columns present in the image. It is then followed by the gray level values for each pixel constituting the image. PGM file format is described in detail as follows.

Header /* P2 or *P5* */
Comment
No-of_Graylevels
No_Rows No_Cols
Gray level values

Appendix B

Image Processing Tool Kit

B.1 Introduction to IPTK

IPTK[30] (Image Processing Tool Kit) is an extensive collection of image processing functions organized in the form of a library. All library functions have HIPL as the prefix, and the first letter after this prefix is a capital letter. These functions take pointers to THEADER structure as input along with other parameters. They return 0 on success and a negative value on failure.

B.I.I Procedure for using library functions

Every function in the library takes atleast one THEADER as input along with other parameters. Most functions take input THEADERs and an output THEADER as arguments. The THEADERs must be properly initialized and memory should be allocated to the arrays in the THEADERs before they are passed as arguments to the functions. Data is taken from the input THEADERs for processing and result is placed in the output THEADER. The output THEADER can be used as an argument in other functions or it can be used to create an ouput image file.

The HIPL library contains all the basic image processing functions. This library

can be included in a C file **like** all other standard libraries. The header file **for** this library is *hipl.h.* For using the HIPL library one has to include the *hipl.h* header **file.** During compilation the user has to link the HIPL library.

B.2 Various functions available in the HIPL library

B.2.1 General functions

IPTK contains an extensive set of image processing functions, organized in the form of a library. All library functions have HIPL as prefix, and the first letter after this prefix is a capital letter. These functions take pointers to THEADER as input along with the other parameters. They return 0 on success and a negative value on failure.

B.2.2 Initialise

This is used to initialize a THEADER data structure. This function behaves differently for the input and the output THEADERs. If it is an input THEADER, it creates a raw image file from the input image file and gets the header information. If it is an output THEADER it fills the header information into it. THEADER initialized for the output operation can be used for an input operation if required.

Function prototype

int HIPL_Initialise(THEADER *T, char *filename,int LOflag,int 1_Fflag, DATA *ID);

The first parameter to the initialize function is the THEADER.Memory for the THEADER should be allocated after the initialize function is called. The second parameter is the name of the input image file and is used for extracting the rows and columns information from the image file. The raw file is created from this image file. When the THEADER is used for input, the file name of the input image is given as

the second parameter. While initializing the THEADER for output, a NULL string is an acceptable parameter. The I-Oflag specifies whether the THEADER is used for input or output. A value of 0 indicates input and a value of 1 indicates output. The type of the array to be allocated is given by the I_Fflag. This should be 0 if an integer array is required and 1 for a floating point array. The last parameter is a pointer to a structure called DATA which contains information about the rows and columns of the image. This structure is filled in while initializing the input THEADER whereas the data in this structure is used to calculate the part size, allocate memory for the arrays and initialize the counters while initializing the output THEADER. Instead of using this structure the file name could be used during output initialization. Rows and columns information can be obtained directly from the image file. But such an approach decreases the speed and increases the probability of a file I/O error. In higher level functions, this structure should not be modified between calls to initialize an input THEADER and its corresponding output THEADER. Initialize returns 0 on successful completion and a negative value in case of an error.

B.2.3 Memory allocation

This function is used to allocate memory for the arrays in a THEADER.Image data is read into this array from the raw input file for input operations. For output operations, this array contains data for writing into the raw file. Processing is done only on this raw data. This array may not be as big as the image dimensions. To save on the system memory, this array may only be a small fraction of the total image size for large image sizes.

The image is split into a number of parts for processing the image horizontally i.e. row-wise. The columns are not split. Each part contains a fixed number of rows and each row is of full length. The term "part-size" refers to the number of rows in the image that each part contains. The part-size is calculated as follows.

$$PARTSIZE = \frac{(MAXMEM \times 1024)}{Numberofcolumns}$$

MAXMEM is a system constant which represents the memory allocation capability of the system. Its value can be modified by the user using the Set memory function. The largest integer value less than or equal to the fraction is taken as the part size.

Function prototype

int HIPL _AllocateMem(THEADER *T);

This function allocates memory depending on the arrtype field in the THEADER, the image dimensions and MAXMEM value. The arrtype field specifies the type of the array to be allocated. This function also initializes the counters in the THEADER. These counters are utilized by the higher level functions to keep track of the number of rows that have been processed.

B.2.4 Getpart

Getpart reads a part of the image from the raw file into an array of the THEADER. This function is called by the higher level functions repeatedly until the entire image has been processed.

Function prototype

int HIPL_Getpart(THEADER *T);

Getpart takes the THEADER as the input and returns the number of rows read from the raw image file. In case of an error, Getpart returns a negative value. The number of rows in the image may not be a multiple of the part size.

$$Numberofparts = (\frac{rows}{partsize}) + 1$$

By returning the number of rows read, the higher level functions can consider each part as a separate image for processing. The number of rows which are retained is given by the following formula.

Rowvsretained — MaskRows — 1

B.2.5 Putpart

Putpart is an output function analogous to the Getpart. It writes the contents of the array into the raw image file. In other functions which create an output **image file** from the array contents, every call to Getpart is followed by a corresponding **call** to Putpart.

Function prototype

int HIPL_Putpart(THEADER *T);

Putpart takes THEADER as the input. It returns 0 on success and a negative value on failure. As in the case of Getpart, Putpart does not perform file I/O if the entire image fits into the array. The resultant image file is created directly if the entire image fits into the array. It is created directly from the array instead of using the raw file. Putpart takes care of the above problem by keeping track of the part number and the row number.

B.2.6 Store result

Store result creates the output image file from the THEADER. If the entire information is contained in the array, the data is taken from the array. Otherwise, data is taken form the raw image file. Store result calls a file format specific function to create the actual image file.

Function prototype

int HIPL_StoreResult (char *filename, THEADER *T);

This function takes the name of the output image file and the THEADER as parameters. It returns a 0 on success and a negative value on failure.

B.2.7 Freeing memory

This function frees the memory allocated to the arrays in the THEADER.

Function prototype

```
int HIPL_FreeMem(THEADER *T) ;
```

This function returns 0 on successful deallocation and returns a negative value on failure.

B.2.8 Closeall

This function removes the temporary file associated with the THEADER.

Function prototype

```
int HIPL_Closeall(THEADER *T);
```

This function returns 0 on successful removal of temporary file and a negative value on failure. The Closeall and Freemem functions act as destructors to THEADER. If both these functions are called for a THEADER no further references should be made to the THEADER.

B.2.9 Refresh

For certain operations, it may require that the same THEADER be used as input to many functions. In such cases, refresh resets the counters that have been modified by the Getpart/Putpart functions. Refresh is also useful in situations where the output THEADER of one function is used as input to other function. This type of refresh is known as a SOFT Refresh - where only the counters are reset. The hard refresh gives

the user the flexibility to modify the part size, which has been set by the initialize function. The part size is calculated on the basis of another user defined value and the array is reallocated accordingly

Function prototype

int HIPL_Refresh(THEADER *T, int flag) ;

The input to the refresh function is the THEADER and a flag indicating the SOFT or the HARD option. This function returns 0 on success and a negative number on failure.

B.3 Image acquisition and operation

The input to the application is a gray scale image in PGM format obtained by scanning a document using a flat-bed scanner. The scanning resolution is set at 150 dpi. The image file consists of two parts: the header and the actual image content. The header contains information regarding the image height,image width and the number of gray levels constituting the image. After the needed information is extracted processing is done accordingly using the HIPL library functions.

Mask initialisation

This function sets the mask parameters in the THEADER to the given values. This function must be called before calling any mask function.

Function prototype

int HIPL_Maskinit (THEADER *T, int rows, int cols, int orgx, int orgy);

T is the input THEADER, rows and cols define the mask dimensions. Orgx and Ofgy give the location of the mask origin. This function returns 0 on success and a negative number on failure.

Bibliography

[1) Arun Agarwal. Understanding paper documents, *sadhana,* 18,part 2:189 208, junel993.

[2] Terno Akiyama and Norihiro HAGITA. Automated entry system for printed documents. *Pattern Recognition,* 1990.

[3] Jean-Christophe Pret Anik Simon and A. Peter Johnson. A fast algorithm for bottom-up document layout analysis. *IEEE,* 19, No. 3, March 1997.

[4] Venkatesh Natrajan Atul Negi and D.Srinivas. A heuristic approach to layout analysis and extraction of handwriting fields from indian bank forms. *Dept of Computer and Information Sciences, University of Hyderabad,* 2001.

[5] Baird. The skew angle of printed documents. *Proc. of Society of Photographic Scientists and Engineers,* 40:21-24, 1987.

[6] H. Brown. Standards for structured documents. *The Computer Journal,* 32:505 514, 1989.

[7] R. Cao and C.L. Tan. Text/graphics separation in maps. *Graphics Recognition Workshop, GREC 2001, Kingston, Canada,* sept 2001.

[8] R. Cao and C.L. Tan. Separation of overlapping text from graphics. *International Conference on Document Analysis and Recognition, ICDAR 2001, Seattle, USA,* pages 44-48, sept2001.

[9] Yi-Kai chen and Jhing fa Wang. Skew detection and reconstruction based on maximization of variance of transistion-counts. *Pattern Rcognition,* 33:195-208, 2000.

[10] Ciardiello. An experimental system for office document handling and text recognition. ***In proceedings of international conference on pattern recognition,* 2:739-** 743, 1998.

[11] Sargur Dachen Wang and Srihari. Classification of newspaper image blocks using **texture analyis.** ***Computer vision, graphics and image processing,* 47:327-352,** 1989.

[12] G.R Thoma D.S.Le and H.Wechster. Automated page orientation and skew angle detection for binary document images. *Pattern Recognition,* 1994.

[13] C.L. Tan etal. News articles retrieval from microfilm images. *1JCAJ'99 Workshop:* ***Text Mining, Foundations, Techniques and Applications, Stockholm, Sweden,*** pages 110-116, 31 July - 6 Aug 1999.

[14] J. Schuman etal. Document analysis from pixels to contents. *IEEE,* 80:1101-1119, 1992.

[15] K. Kubotaetal. Document understanding system. *Proceedings 7th International* ***conference on pattern recognition,* pages 612-C17, 1984.**

[16] S.C Hinds etal. A document skew detection method using run-length encoding **and hough transform.** ***In Proceedings of International conference on pattern*** *recognition,* 1:464-468, 1990.

[17] Y.Y. Tang etal. Document analysis and understanding. *First International* ***Conference on Document Analysis and Recognition, Saint Malo, France,* pages** 17-31, 1991.

[18] L. Fan and C.L. Tan. Binarizing document image using coplanar prefilter. *International* ***Conference on Document Analysis and Recognition, ICDAR 2001,Seattle,*** *USA,* pages 34-38, Sept 2001.

[19] L.O. Forman and Kasturi. Document image analysis system. *Computer,* pages 5-8, July 1992.

[20] S.C. Seth G. Nagy and S.D. Stoddard. Document analysis with expert system. ***In Proceedings of Pattern Recognition in Practice II, Amsterdam,* June,1985.**

[21] Rafael C. Gonzalez and Richard E. Woods. *Digital Image Processing.* Addison Wesley Publication, 1993.

[22] H.Bunke H.S.Baird and K.Yamamoto. *Structured document analysis.* Springer-verlag, 1993.

[23] ChiJiwa liu Kuo-china Fan and Yuan kai Wang. Segmentation and classification of mixed text/graphics/image documents. *Pattern Recognition Letters,* 15:1201-1209, 1994.

[24] R.G. Casey K.Y. Wong and F.M. Wahl. Document analysis system. *IBM J. Res. Develop,* 26, No. 6:647-656, 1982.

[25] S.W. Lam. An adaptive approach to document classification and understanding. ***IAPR Workshop on Document Analysis Systems, Kaiserslautern Germany,* 1994.**

[26] S.W. Lam and S.N. Srihari. Multi-domain document layout understanding. *First* ***International Conference on Document Analysis and Recognition, Saint Mala,*** *France,* pages 112-120, 1991.

[27] Leslie Lamport. *A document preparation systemLATEX.* Low price edition, 2000.

[28] Q.Yuan and C.L. Tan. Page segmentation and text extraction from gray. scale images in microfilm. *SPIE Document Recognition and Retrieval VIII, San Jose, California,* pages 323-332, Jan 2001.

[29] Q. Wang R. Cao, C.L. Tan and P. Shen. Segmentation and analysis of double-sided handwritten archival documents, *4th IAPR International Workshop on Document Analysis Systems, DAS2000, Rio de Janeiro, Brazil,* pages 147-158, 10-13 December 2000.

[30] Reddy and Naidu. Image Processing Tool-Kit. Master's thesis, Dept of Computer and Information Sciences, University of Hyderabad, 1999.

[31] Changming sen and Deyi si. Skew and slant correction for document images using gradient direction. *Proceedings of the IEEE,* 1992.

[32] S.N. Srihari. Document image understanding. *IEEE Fall Joint Computer Conference, Dallas, Texas,* pages 87-95, 1986.

[33] S.N. Srihari and Govindaraju. Analyis of textual images using hough transform. ***Machine vision applications,* pages 144-153, 1989.**

[34] A.C. Girardin S.W. Lam and S.N. Srihari. Gray-scale character recognition using **boundary features.** ***SPIE/IS and TSymposium on Electronic Imaging Science and Technology, San Jose, California,* 1992.**

[35] C.L. Tan and B. Yuan. Document text segmentation using multi-band disc **model.** ***SPIE Document Recognition and Retrieval VIII, San Jose, California,*** *USA,* pages 212-221, 24-25 Jan 2001.

[36] C.L. Tan and Zheng Zhang. Text block segmentation using pyramid structure. ***SPIE Document Recognition and Retrieval VIII, San Jose, California,* pages** 297-306, 24-25 Jan 2001.

[37] S.N. Srihari V. Govindaraju and D. Wang. Newspaper image, understanding. ***Lecture Notes in Artificial Intelligence J. Siekmann (editor), Springer Verlag*** *NY,* 444:375-386, 1989.

[38] Y. Zhou and C.L. Tan. Chart analysis and recognition in document images. ***International Conference on Document Analysis and Recognition, ICDAR 2001,*** *Seattle, USA,* pages 1055-1058, 10-13 September 2001.