Document analysis using image processing techniques.

Yi. Lin
University of Windsor

Follow this and additional works at: https://scholar.uwindsor.ca/etd

Recommended Citation
https://scholar.uwindsor.ca/etd/1771

This online database contains the full-text of PhD dissertations and Masters' theses of University of Windsor students from 1954 forward. These documents are made available for personal study and research purposes only, in accordance with the Canadian Copyright Act and the Creative Commons license—CC BY-NC-ND (Attribution, Non-Commercial, No Derivative Works). Under this license, works must always be attributed to the copyright holder (original author), cannot be used for any commercial purposes, and may not be altered. Any other use would require the permission of the copyright holder. Students may inquire about withdrawing their dissertation and/or thesis from this database. For additional inquiries, please contact the repository administrator via email (scholarship@uwindsor.ca) or by telephone at 519-253-3000ext. 3208.
DOCUMENT ANALYSIS USING IMAGE PROCESSING TECHNIQUES

By

Yi Lin

A Thesis
Submitted to the Faculty of Graduate Studies and Research Through the Department of Electrical and Computer Engineering in Partial Fulfillment of the Requirements for the Degree of Master of Applied Science at the University of Windsor

Windsor, Ontario, Canada
2003
The author has granted a non-exclusive licence allowing the National Library of Canada to reproduce, loan, distribute or sell copies of this thesis in microform, paper or electronic formats.

The author retains ownership of the copyright in this thesis. Neither the thesis nor substantial extracts from it may be printed or otherwise reproduced without the author's permission.

L'auteur a accordé une licence non exclusive permettant à la Bibliothèque nationale du Canada de reproduire, prêter, distribuer ou vendre des copies de cette thèse sous la forme de microfiche/film, de reproduction sur papier ou sur format électronique.

L'auteur conserve la propriété du droit d'auteur qui protège cette thèse. Ni la thèse ni des extraits substantiels de celle-ci ne doivent être imprimés ou autrement reproduits sans son autorisation.
Abstract

Image thresholding and page segmentation are necessary components of any image understanding and recognition system. In order for an OCR to function properly, texts in a document image has to be isolated and then fed to the OCR for recognition. This requires development of a robust and accurate page segmentation technique. In any page segmentation technique, a preprocessing step in terms of image restoration and thresholding is needed. This thesis therefore concentrates on the development of efficient and robust image thresholding and page segmentation algorithms.

In this thesis, three efficient contrast enhancement techniques are proposed that in conjunction with the thresholding techniques of Ridler and Calvard constitute the preprocessing step for the image segmentation algorithm. This thesis also provides a survey of the pertinent page segmentation techniques in the literature, and proposes a new block labeling technique based on smearing algorithm. An exhaustive experimentation is conducted in this thesis to demonstrate the efficiency of the proposed techniques.
To my husband Tao Dai for his love and support, and to my parents for their encouragement.
Acknowledgments

I would like to express my sincere gratitude to both my supervisor Dr. M. A. Sid-Ahmed and my co-supervisor Dr. M. Ahmadi for their encouragement, support and guidance during the course of this research. I would also like to thank my committee members, Dr. X. Chen, and Dr. Gasparr for their critical comments, suggestions and evaluations.

I would also like to recognize the following individuals and corporations for their contribution: Yasser Alginahi for his help and time on the page segmentation part of this thesis; Idirs El-Feghi and SongTao Huang for their comments on my thesis; Canadian Bank Note Company for their critical comments and also financial support.

Finally my warmest acknowledgments to my husband, my parents for their constant love and support.
Table of Contents

Abstract .................................................................................................................. IV
Dedication .............................................................................................................. V
Acknowledgement ................................................................................................. VI
List of Figures ....................................................................................................... IX
List of Tables ......................................................................................................... X
List of Symbols .................................................................................................... XI

Chapter 1  Introduction......................................................................................... 1
  1.1 General Introduction ...................................................................................... 1
  1.2 Related Work ................................................................................................. 2
    1.2.1 Preprocessing .......................................................................................... 2
    1.2.2 Page Segmentation ............................................................................... 6
      1.2.2.1 Block Segmentation ......................................................................... 7
      1.2.2.2 Block Labeling ................................................................................. 12
      1.2.2.3 Block Classification ......................................................................... 14
  1.3 Applications ................................................................................................. 17
  1.4 Thesis Organization ...................................................................................... 18

Chapter 2  Preprocessing...................................................................................... 19
  2.1 General Introduction ...................................................................................... 19
  2.2 Proposed Methods ....................................................................................... 25
    2.2.1 Global Linear Contrast Enhancement .................................................. 25
    2.2.2 Global Nonlinear Contrast Enhancement ............................................. 28
    2.2.3 Local Nonlinear Contrast Enhancement .............................................. 31
  2.3 Ridler and Calvard Thresholding .................................................................... 32
  2.4 Simulation Results ....................................................................................... 34
    2.4.1 Results from Ridler and Calvard Thresholding without preprocessing . 36
    2.4.2 Results from Using Global Linear Contrast Enhancement plus
        Ridler and Calvard Thresholding ............................................................... 38
    2.4.3 Results from Using Global Nonlinear Contrast Enhancement plus
        Ridler and Calvard Thresholding ............................................................... 40
    2.4.4 Results from Using Local Nonlinear Contrast Enhancement plus
        Ridler and Calvard Thresholding ............................................................... 42
    2.4.5 Comparison of Proposed Methods ....................................................... 43

VII
List of Figures

<table>
<thead>
<tr>
<th>Figure</th>
<th>Description</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Figure 2.1</td>
<td>Cross Section of a filter function for use in homomorphic filtering.</td>
<td>22</td>
</tr>
<tr>
<td>Figure 2.2</td>
<td>Difference between Linear and Nonlinear Contrast Enhancement.</td>
<td>28</td>
</tr>
<tr>
<td>Figure 2.3</td>
<td>Gamma Correction.</td>
<td>29</td>
</tr>
<tr>
<td>Figure 2.4</td>
<td>Image Examples.</td>
<td>35</td>
</tr>
<tr>
<td>Figure 2.5</td>
<td>Results from Using Ridler and Calvard Thresholding Only.</td>
<td>37</td>
</tr>
<tr>
<td>Figure 2.6</td>
<td>Results from Using Global Linear Contrast Enhancement + Ridler and Calvard Thresholding.</td>
<td>39</td>
</tr>
<tr>
<td>Figure 2.7</td>
<td>Results from Using Global Nonlinear Contrast Enhancement + Ridler and Calvard Thresholding.</td>
<td>41</td>
</tr>
<tr>
<td>Figure 2.8</td>
<td>Results from Using Local Nonlinear Contrast Enhancement + Ridler and Calvard Thresholding.</td>
<td>43</td>
</tr>
<tr>
<td>Figure 2.9</td>
<td>Comparison of three Methods in Preprocessing.</td>
<td>45</td>
</tr>
<tr>
<td>Figure 2.10</td>
<td>Comparison of Results Using Different N.</td>
<td>49</td>
</tr>
<tr>
<td>Figure 2.11</td>
<td>Simulation Result of Using Different Parameter p.</td>
<td>56</td>
</tr>
<tr>
<td>Figure 3.1</td>
<td>Binary sequence got from RLSA.</td>
<td>66</td>
</tr>
<tr>
<td>Figure 3.2</td>
<td>block labeling after step 1.</td>
<td>66</td>
</tr>
<tr>
<td>Figure 3.3</td>
<td>block labeling after step 2.</td>
<td>67</td>
</tr>
<tr>
<td>Figure 3.4</td>
<td>block labeling after step 3.</td>
<td>67</td>
</tr>
<tr>
<td>Figure 3.5</td>
<td>Detail Steps of Page Segmentation</td>
<td>77</td>
</tr>
<tr>
<td>Figure 3.6</td>
<td>Final Results of some images.</td>
<td>87</td>
</tr>
</tbody>
</table>
List of Tables

Table 2.1  Performance Comparison of three methods in preprocessing .................. 46
Table 2.2  Comparison of performance using different N .................................. 49
Table 3.1  Comparison between three page segmentation methods ...................... 60
List of Symbols

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MIN</td>
<td>Minimum intensity value</td>
</tr>
<tr>
<td>MAX</td>
<td>Maximum intensity value</td>
</tr>
<tr>
<td>AVE</td>
<td>Average intensity value</td>
</tr>
<tr>
<td>y₁</td>
<td>Input image sequence</td>
</tr>
<tr>
<td>y₂</td>
<td>Output image sequence</td>
</tr>
<tr>
<td>r</td>
<td>Nonlinear parameter</td>
</tr>
<tr>
<td>p</td>
<td>Power parameter</td>
</tr>
<tr>
<td>N</td>
<td>Image is divided into N by N sub regions</td>
</tr>
<tr>
<td>T_{f_{max}}</td>
<td>Maximum intensity value of foreground</td>
</tr>
<tr>
<td>T_{b_{max}}</td>
<td>Maximum intensity value of background</td>
</tr>
<tr>
<td>T_{b_{min}}</td>
<td>Minimum intensity value of background</td>
</tr>
<tr>
<td>Th</td>
<td>Threshold of horizontal smearing</td>
</tr>
<tr>
<td>Tv</td>
<td>Threshold of vertical smearing</td>
</tr>
<tr>
<td>Tf</td>
<td>Threshold of final smearing</td>
</tr>
<tr>
<td>BC</td>
<td>Total number of black pixels in a block</td>
</tr>
<tr>
<td>DC</td>
<td>Total number of black pixels in original data in the same block</td>
</tr>
<tr>
<td>TC</td>
<td>Total number of white-black transitions of original data</td>
</tr>
<tr>
<td>H</td>
<td>Height of a block</td>
</tr>
<tr>
<td>E</td>
<td>Eccentricity of rectangle surrounding the block</td>
</tr>
<tr>
<td>S</td>
<td>The ratio of the number of block pixels to the area of surrounding rectangle</td>
</tr>
<tr>
<td>R</td>
<td>The mean horizontal length of black runs of original data in each block</td>
</tr>
<tr>
<td>σ</td>
<td>Standard deviation</td>
</tr>
<tr>
<td>C</td>
<td>Parameters used in block classification, and it is end up with numbers, for example: C₁, C₁₁, etc.</td>
</tr>
</tbody>
</table>
Chapter 1: Introduction

1.1 General Introduction

Due to weakness of computer systems and printing devices, we have documents with complex, poorly illuminated and non-uniformly distributed background. Systems for document understanding require a process that extracts text characters and other features, like photo, from such background, so that pattern recognition techniques for text and face can be applied.

Even if a document has a white, uniform background and contains only black text, the document image input by an image scanner often has uneven brightness due to uneven illumination. Even if the illumination becomes uniform at document scanning, it may still suffer from shading, noise, and a complex background. An uneven force applied during scanning may also result in complex and non-uniformly distributed background. In order to design a robust document understanding system, the document analysis algorithm has to be tolerant to these kinds of problems.

Here are the main problems we need to mitigate in order to extract texts, images and other objects:

- Complex backgrounds such as illustrations, photographs, and computer graphics;
- Poorly illuminated background due to the uneven illumination, shading, etc;
- Non-uniform character brightness due to the uneven illumination, shading, force applied during scanning, etc.

There are two stages in a document analysis system:
- Preprocessing: eliminate the background and generate the binary sequence.
- Page segmentation: including block segmentation, block labeling, block classification, text and image extraction.

1.2 Related Work

1.2.1 Preprocessing

In [1], intensity gradient-based thresholding is used for poorly illuminated document images. It initially locates object pixels, and then grows regions around these by generating a threshold value for each pixel. It can be assumed with a high degree of probability that pixels at locations having a high gray level gradient form a part of an object, and that nearby pixels with similar gray levels will also be part of the same object. If regions are grown carefully, a thresholded image will result. This method is good for images with a background that gradually change in gray levels. However, it cannot segment text characters in documents with complex background.

In [2], the Ridler and Calvard thresholding [3] was proved and a new derivation for the iterative scheme was proposed. By maximizing interclass variance between dark and bright regions, one can find single and also multiple thresholds. The method was
examined on some images with background. The method performs well on those images for which the contrast of background and foreground differs substantially. If the contrast of the background and foreground in an image are similar, then it may fail. For multilevel thresholding, there is the need to select one threshold out of several threshold levels to obtain the binary image for page segmentation. In different kinds of images, the threshold one select from the multilevel thresholds will make a difference.

Some methods have been developed for character pattern extraction from grayscale/color document images with complex background. Local adaptive binarization is used in Ohya’s method [4]. An original image is firstly divided into sub-regions, for each of which the gray-level threshold is determined. If the original image is thresholded with the determined threshold in each sub-region, there will be discontinuity between sub-regions. Therefore, the threshold of each pixel is determined by linearly interpolating thresholds between the central pixels of sub-regions. Using these interpolated thresholds, smooth bi-level images can be obtained. As long as the background is very light compared to character patterns, then the characters can be separated from such images. However, these methods are not capable of extracting text from complex background with many gray levels.

In [5], Liang, Ahmadi, and Shridhar developed and implemented a morphological approach to character string extraction of a regular periodic overlapping text/background images that minimizes the shape distortion of characters. Mathematical morphology, because of its ability to grasp the geometry and structure of images, was adopted to
realize the new scheme. The underlying strategy of the algorithm is to maximize background component removal while minimizing the shape distortion of text characters by using appropriate morphological operations. This algorithm although is effective for regular periodic background, there are some requirements on the images:

1. The graphic symbols in the background are periodically distributed;
2. The width ratio of the minimal stroke of the character strings to the background symbols is approximately 1;
3. The resolution of digitizer is not such that the topological property of each character is eliminated by low resolution.

These requirements make the algorithm only works on certain kinds of images, as a result, it is not suitable as a general preprocessing tool.

In Zhong’s method [6], the bounding boxes around text components are first located using the spatial variance of the image. The color segmentation is applied within each box to segment character patterns from colorful documents. Although the representative colors are determined locally in this method, it cannot handle those text blocks in which the color of text characters changes gradually. Another disadvantage of the method is that the text layout is strongly limited. It can only handle horizontal text lines with almost zero skew.

In [7], a method dealing with color documents is proposed. The color document image is transformed into binary image using an edge-detection technique. First the gradient of R, G, and B value on each pixel is calculated and then based on a predefined threshold
value, a pixel is determined whether it is an edge pixel or not. To calculate the gradient, two Sobel operators are used. This method works for images with light and uniform background. It fails for images with complex and non-uniformly distributed background.

Another algorithm for the extraction of character strings from a color document is proposed in [8]. First the set of representative colors of a document is determined. Then potential character strings are extracted from each representative color image using multi-stage relaxation. Since the extracted candidate strings may contain false results, corresponding to a part of the background or a picture, the likelihood of a character string is computed in a subsequent step using features derived from the density of lines and spaces. When all extracted elements of all representative color images are superimposed, some elements overlap each other. The reason is that the elements are not always character strings. Therefore, conflict resolution is done to eventually select proper character strings.

In [9], Goto and Aso presented a new method by which small character patterns can be extracted from document images with complex backgrounds. The method is based on local multilevel thresholding, pixel labeling and region growing. Since the proposed method is independent of the text line extraction process, it can handle text lines with various shapes and layout. The new method consists of the following stages:

1. Local multilevel thresholding and initial pixel labeling;
2. Edge compensation in labeled images;
3. Creation of merging inhibition tables;
4. Region growing based on label merging between neighboring sub images;

5. Creation of decomposed images.

But in this algorithm, there are many parameters that need to be set. For example, in stage 1, the window size, window width and a parameter $\alpha$. In stage 2, the length of every step and the total length of every step. In stage 4, the parameter $\theta$ and the threshold for fusion, etc. All these values are obtained from experiments, and no optimum value exists for $\theta$.

1.2.2 Page Segmentation

Automatic page segmentation of a digitized document image is a necessary element of a document analysis system capable of understanding a document consisting of a mixture of text and graphic images. In the past three decades, several approaches for text block separation from mixed text/graphics images have been proposed. The block segmentation is a procedure that subdivides the area of digitized documents into blocks in order to process the document images systematically. Each of the blocks ideally is required to contain only one type of image data. Then the segmented blocks will be classified as predefined data such as text, graphics, halftone images, etc.

There are 3 steps involved in page segmentation: block segmentation, block labeling and block classification.
1.2.2.1 Block Segmentation

The document image can be segmented into blocks by two different methods, top-down and bottom-up approaches. In the top-down approach, certain global operations are performed on the entire image. In the bottom-up approach, on the other hand, all the components in the document images are individually detected and then merged together into larger blocks. The most well known techniques for segmentation (top-down approach) are the Run-Length-Smearing Algorithm (RLSA) [10] and Projection Profile cuts or Recursive X-Y Cuts (RXYC) [11].

The RLSA was first proposed by Wahl, Wong and Casey [10]. It is used to obtain a bit-map of white and black areas representing blocks containing the various types of data. The basic RLSA is applied to a binary sequence in which white pixels are represented by 0's and black pixels by 1's. When applied to pattern arrays, the RLSA has the effect of linking together neighboring black areas that are separated by less than a pre-selected number of pixels t. The RLSA is applied row-by-row as well as column-by-column to a document, yielding two distinct bit-maps. Because spacing of document components tend to differ horizontally and vertically, different values of t can be used for row (t_{h} =300) and column (t_{v} =500) processing. The two bit-maps are then combined in a logical AND operation. Additional horizontal smoothing using the RLSA (t_{h} =30) produces the final segmentation result. This technique is quite fast and gives good results for some documents but was found to be too dependant on the threshold values (t_{h}, t_{v} and t_{s}) and fails if the input document is skewed.
Nagy and Stodard [11] described one of the top-down segmentation strategies called
RXYC (Recursive X-Y Cuts). This approach is known as the projection profile cuts. Printed pages are conventionally made up of rectangular blocks, and a page can be recursively cut into rectangular blocks. Thus the document is represented in the form of a tree of nested rectangular blocks. The root represents the entire page and the succeeding nodes are obtained by a process of alternating horizontal and vertical cuts. At each step of the recursive process, the projection profile, computed along both horizontal and vertical direction is simply the sum of all the pixel values along that line. Then division along the two directions is accomplished by making cuts corresponding to deep valleys in the projection profile corresponding to the distance between paragraphs, with the width larger than a predetermined threshold. A local peak detector is applied to horizontal and vertical profiles to detect local peaks (corresponding to white gaps) at which the cuts are placed; it is local in that the width is determined by the nesting level of the recursion, e.g., gapes between paragraphs are thicker than those between lines.

Although the recursive x-y cuts could decompose a document image recursively into a set of rectangular blocks, they did not give details to define cuts. Later Jaekyu, Haralick, and Phillips [12] improved this method by using bounding boxes. They first applied a connected component-labeling algorithm to obtain the bounding boxes of connected components. Then created a root node and calculated the horizontal and vertical projection profiles within the region of interest. Finally made divisions at large gaps in the projection profiles whose widths exceeded a certain threshold value. Whenever divisions were made, it created a new child node, at each recursion level, horizontal and

Chapter 1: Introduction 8
vertical divisions alternated. This improved method is also used in [13], in which OCR is performed after the page segmentation.

In [14], The Hough Transform approach 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 representable forms like straight lines in noisy binary images. In fact, it is shown that 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 corresponding 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. This method is invariant to skew; however, it requires much computation time for the preliminary steps and extensive usage of memory resources. It is also highly CPU intensive and consequently it is too slow to be applied for document analysis without support of special hardware.

Kim [15] developed an algorithm for block segmentation that divides the page by white
spaces, which separate the block despite the skew of the document. Then these blocks can be classified independently of the block rotation. This approach connects each component to generate larger connected components with appropriate size; then labels are assigned to different blocks. Some measurements (features) such as the total number of black pixels and the number of the horizontal black-white transitions in the original image block are considered. A ratio of the total number of black pixels to the black-white transitions is obtained for each block and can represent a feature for block classification. Then all the line segments are removed and the total number of black pixels is measured. If this value is almost the same as above value (measured before removing the line segments), the block is a symbol, otherwise it is a picture. Pure line segments are removed from the original image of the block and then the black-white transitions are measured. If the ratio of the total number of black pixels to the number of black-white transitions is the same as the original value, then the block is text; otherwise it is line drawing.

In [16], a new method of page segmentation based on analysis of background (white areas) is presented. The proposed method is capable of segmenting pages with non-rectangular layout; as well as various angles of skew. Most of the previous methods have developed under the assumptions that (1) all printed areas can be circumscribed by rectangles (or a page has rectangular layout) and (2) the skew of a page image is normalized beforehand. In order to analyze background without any assumptions, it is necessary to represent white areas of any shape as potential borders of printed areas, as well as to select right borders effectively and efficiently from the represented white areas.
As the representation of white areas, they employ thin lines obtained by thinning of background. These thin lines are suitable for the representation of white areas since thinning preserves the connectivity of white areas which are originally connected to enclose printed area. Based on the representation, the selection is simply defined as to find loops which enclose printed areas. By eliminating unnecessary thin lines using not only features of white areas but also features of black areas, they can achieve some good results.

But this method is very time-consuming, therefore in [17], Yip and Chi proposed an improvement to the technique by reducing the image size to 1/16. By doing this, they can significantly reduce the processing time for page segmentation and content classification. To achieve a better segmentation result, they take one more processing step to remove the chains generated inside a picture region. To do this, they first identify the boundary of each region. Then they delete all the chains inside a boundary.

When document layout analysis is performed in a top down manner, a page may be split into one or more column blocks of text, each column is split into paragraph blocks, analysis may be performed in a bottom-up manner, where connected components are merged into characters, then words, then text lines, etc. Although the bottom-up clustering technique is insensitive to a document’s skew angle, it is slower and unreliable due to decision made with low statistical confidence in small neighborhoods.
In [18], a bottom up method is proposed. It first creates connected components. Connected components are rectangular boxes bounding together regions of connected black pixels. The objective of the connected component stage is to form rectangles around distinct components on the page, whether they are characters or images. These bounding rectangles then form the skeleton for all future analysis on the page. After the connected components have been determined, the next step is to group neighboring connected components of similar dimensions. All the connected components of documents fall into one of three categories: small, medium or large depending on their size. The connected components are then merged in accordance to the category they fall under. Merging requires the use of a prefixed threshold (different for each category) so as to provide a means of determining the neighborhood of a group. After recursive grouping, a page is partitioned into coherent blocks.

### 1.2.2.2 Block Labeling

Extracting and labeling of various disjoint and connected components in an image is central to many automated image analysis applications. Traditional connected components labeling scans an image and groups its pixels into components based on pixel connectivity, \textit{i.e.} all pixels in a connected component share similar pixel intensity values and are in some way connected with each other. Once all groups have been determined, each pixel is labeled with a gray level or a color (color labeling) according to the component it was assigned to.
In [19], several labeling methods are introduced. Basically labeling algorithms can be divided into two large classes:

- Local neighborhood algorithms (performing iterative local operations)
- Divide-and-conquer algorithms

A local neighborhood algorithm is described below:

- The image is scanned in a row-wise manner until the first pixel at the boundary of a binary image object is hit.
- A "fire" is set at this pixel that propagates to all pixels belonging to the 8-neighborhood of the current pixel.

This operation is continued recursively until all pixels of the image object are "burnt" and the fire is extinguished. After the end of this operation, all pixels belonging to this object have value 1 and cannot be distinguished from the background. A by-product of the algorithm is the area of the object. This procedure is repeated until all objects in the image are counted.

Divide-and-conquer algorithm for connected component labeling uses the split and merge algorithm:

- Inhomogeneous regions consisting of 0s and 1s are split recursively until we reach homogeneous regions consisting only of 1s;
- These regions are assigned a unique label. This is the split step.
• Label equivalences can be established by checking the borders of all homogeneous regions.

• Those regions having equivalent labels are merged to a single connected component.

But the traditional labeling methods are used for those rectangular blocks. It is hard to deal with blocks with convex or concave shape. In such a case, some problems will occur, for example, pixels in the same block may have different labels.

1.2.2.3 Block Classification

After dividing the documents into different regions, we need to classify them as predefined data i.e. classify the blocks as text or graphic or halftone image.

Scherl, Wahl, Fuchsberger [20] described a simple method for obtaining characteristic features for text, graphic and halftone image segments. It subdivides the document into small, overlapping windows to generate a histogram. Text consists of white background with black characters on it. The characters consist of many transitions between black and white. Therefore, a small sharp peak at a bright gray level and a lot of darker gray levels are typical of text. Meanwhile, the histogram of a halftone image has no similar sharp characteristics. The shape of the histogram strongly depends on the content of the image. In some cases, it might be possible that the histogram of a graphic looks like the histogram of text, especially if a graphic consists only of lines. In this case, the histogram is not suitable for discrimination of text and graphics. The shape of the histogram is largely dependent upon the size of the window. A larger window results in a
weaker dependence of the shape of the histogram on the position of the window within the text. However, larger windows decrease the accuracy.

Another method for block classification was proposed by Wahl [10]. They use the block height and the block mean black pixel run length as basic features. Several measurements such as the total number of black pixels in the segmented image block, BC (Black Count); the minimum x-y coordinates of a block and its x-y lengths (xmin, Δx, ymin, Δy); the total number of black pixels in the original image for the block, DC (Original Black Count); and the number of horizontal white-black transitions in the original image block, TC (Transition Count) are taken to classify text line blocks and graphic or images block. It exploits the face that text lines have approximately a constant and small height. Classification of each block is done by computing several features for each block from the above measurements such as block height, its eccentricity (Δx/Δy), the ratio (BC/ΔxΔy) and the mean horizontal length of the black runs of the original data for each block (DC/TC). These features are used to classify different types of data using a linear pattern classifier.

While this technique was reported to perform well on a variety of printed documents, no information was provided on the size of the experiments and performance statistics were lacking. Then in [21], Le, Thoma and Wechsler proposed a new approach for the classification of a binary document image using connectionist models. Four connectionist models were considered and they include the back propagation, the radial basis function, the probabilistic connectionist, and the Kohonen’s self-organizing feature
map [22]. All four connectionists showed good classification results. The back propagation connectionist takes more time to train, but it requires less memory. The probabilistic connectionist requires that the entire training data must be stored and used for each classification of an unknown pattern. The Kohonen’s self-organizing feature map requires a large memory size for the output neuron array. On the other hand, the Radial Basis Function (RBF) connectionist has the highest classification accuracy, but it requires intermediate amounts of memory and training time.

In [17], a hierarchical approach that combines a cross-correlation method, the Kolmogorov complexity measure, and a neural network classifier for classifying sub-images into one or two categories: text or picture. However, the results of the neural network classifier are very sensitive to the background of the text region. Also it is very dependent on the training samples. For the cross-correlation method and Kolmogorov complexity measure [23], the classification results are very dependent on the similarity and the complexity of document images.

In [24], a new class of distance mapping for binary images was proposed. This new class of distance mapping for binary images is based on a border-to-border distance rather than on the distance between an individual point and the border of it. These mappings can be used to calculate efficiently meaningful features which reflect geometrical properties of objects within binary images. Using the properties that text consists of characters with limited size, graphics are usually composed of long lines and images mainly consist of several compact black and white regions, a discrimination is possible.
Another robust and computationally efficient algorithm was presented in [25]. It uses a
simple mask that makes use of the different correlation properties between the text and
halftone region, and has comparable or better performance than more sophisticated and
computationally intensive spectral analysis techniques. Comparing this approach to that
of Pavlidis and Zhou [26], they make use of the fact that, for halftone images the
correlation between neighboring scan lines remains fairly constant as the distance
increases, while for text it drops rather quickly. The new approach uses of the same fact,
but in both the horizontal and vertical directions, and also, they only look at black pixel
correlations that make sense for black on white images. But the results are too dependent
on the size of the masks. For example, small fonts don't survive through either large or
small mask; big fonts survive through small mask, but not large mask; and halftone
images survive through both small mask and large mask. So this approach is only
suitable for extracting halftone images.

1.3 Applications

Here are some of the applications of document analysis system:

- Document authentication;

- Document entry for information processing and retrieval systems;

- Preprocessing of Documents for OCR;

- Developing a system for document content analysis for visually handicapped.
1.4 Thesis Organization

Chapter 1 introduces a general overview of the concept of document analysis, various methods in preprocessing and page segmentation. Chapter 2 covers three preprocessing methods: global linear contrast enhancement, global nonlinear contrast enhancement and local nonlinear contrast enhancement, together with an iterative threshold selection algorithm for converting a grayscale image to a binary image. Simulation and comparison results are also given in Chapter 2. Page segmentation stage is then discussed in considerable detail in Chapter 3. It is divided into three sub stages: block segmentation, block labeling and block classification. Simulation results are also shown in Chapter 3. Finally the thesis will end with some closing remarks in Chapter 4.
Chapter 2: Preprocessing

2.1 General Introduction

There are printed documents where text characters are printed on colored backgrounds, as well as complex backgrounds such as illustrations, photographs, and computer graphics, which has poorly illuminated and non-uniformly distributed background. Systems for document analysis and recognition require a process that separates text characters from colored and/or complex backgrounds so that the subsequent stages can perform.

Even if a document contains a white, uniform background with only black text, the document image input by an image scanner or a camera often produce uneven brightness due to uneven illumination. Even if the illumination is almost even at document scanning, digitized documents often suffer from shading, noise and complex background. Moreover, when a document is scanned, the uneven force applied during scanning can have non-uniform effects on the resultant digitized document. These are the some of the main problems that could arise with documents:

- Complex backgrounds such as illustrations, photographs, and computer graphics;
- Non-uniform illuminated background due to uneven illumination, shading, etc;
• Non-uniform character brightness due to uneven illumination, shading, force applied during scanning, etc.

In order to design a robust, general document recognition system, the character extraction processes have to be tolerant to these problems.

We have surveyed several methods dealing with image preprocessing in chapter 1. Some of these are too complicated, because of the number of stages required for its implementation as well as many parameters need to be initialized. Some of these methods are not efficient to deal with complex and non-uniformly distributed background. For example, the Ridler and Calvard thresholding [2], which works well on images for which background is much lighter than the foreground. If the contrast of the background and foreground in an image is very similar, or the background is non-uniformly distributed, then it can fail. If we can adjust the contrast of digitized documents, which could fulfill the requirement that the background is much lighter than the foreground, then this method may work. Based on this assumption, we here proposed three methods as preprocessing steps prior to the Ridler and Calvard thresholding [2] in order to obtain acceptable results.

The simplest way to adjust image contrast is contrast enhancement. The objective of contrast enhancement is to adjust an image to emphasize features of interest. In most cases, these features are small and have small intensity variations, so methods that improve the intensity variations in an image are useful for contrast enhancement. The following methods are commonly used for contrast enhancement:
Intensity Windowing [27]

One of the simplest ways to enhance an image is to focus only on the intensity range of the image that contains features of interest. This can be accomplished by windowing the image - setting all pixels below a threshold value to the minimum intensity value, and all pixels above another threshold value to the maximum intensity value. When the image is displayed, the new intensity range will be stretched to monitor display range, thereby enhancing detail in the specified intensity window.

High Pass Filtering [28]

In high pass filtering the objective is to get rid of the low frequency or slowly changing areas of the image and to bring out the high frequency or fast changing details in the image.

There are various methods of implementing a high pass filter. The simplest way is to take a pixel and subtract it from its neighbors. In this way we stress the difference of the pixel from its neighbors. If the pixel is in an area of little change, then the difference between the pixel and its neighbors will be zero. However if the pixel is on an edge, then the difference will be large.

Homomorphic Filtering [29]

An image is formed through the mapping of a three-dimensional scene onto a two-dimensional surface or plane. The light intensity distribution, f (n1, n2), over that plane
characterizes the image and is considered to be dependent on two distributions. The first distribution is the reflectivity of the various objects in the scene and will be denoted by reflect \((n_1, n_2)\). The second distribution is the amount of illumination received by the various objects; this will be denoted by illumination \((n_1, n_2)\). Since the objects carry the details in the image, that is, borders between objects, edges within an object, etc., then reflect \((n_1, n_2)\) will contain higher frequencies (sharp transition in gray levels) than illumination \((n_1, n_2)\), which characterizes the illumination and which usually has a more gradual variation over the scene.

One method of providing enhancement to the various objects in a given scene is by de-emphasizing the illumination effect. This is carried out by first separating the two components reflect \((n_1, n_2)\), the light intensity distribution, and illumination \((n_1, n_2)\), the illumination; and then emphasizing the high frequencies. This can be achieved by taking the log of the intensity function \(f(n_1, n_2)\). The result is then filtered through a 2-D filter having the characteristic shown in figure 1.

![Diagram](attachment:image.png)

Figure 2.1, Cross Section of a filter function for use in homomorphic filtering. \(D(w_1, w_2) = \sqrt{w_1^2 + w_2^2}\)
The antilog of the output produces the final result. This procedure can be described by the following equations:

\[
\begin{align*}
\ln(f(n_1, n_2)) &= \ln(\text{reflect}(n_1, n_2)) + \ln(\text{illumination}(n_1, n_2)) \\
o(n_1, n_2) &= \ln(f(n_1, n_2)) \ast h(n_1, n_2); \\
g(n_1, n_2) &= e^{o(n_1, n_2)}
\end{align*}
\]  

(2.1)

\(n_1\) is from 1 to image width; \(n_2\) is from 1 to image length.

Here * denotes convolution of 2 matrices. This procedure should decrease the effect of the non-uniform illumination in the image and enhance the details of the scene.

**Unsharp Masking [27]**

There are a number of linear methods for image enhancement. One common method is unsharp masking [27], where a blurred version of an image is subtracted from the original to emphasize fine details. We can use multiple iterations of smoothing to smooth the image and subtract a specified fraction of the smooth image from the original. For example, if the original image signal is \(F\), with width \(W\) and length \(L\). And the image after multiple iterations of smoothing is \(F_1\). The final image \(F_2\) is given by the following equation:

\[
F_2(i, j) = F(i, j) - \frac{n}{m} \times F_1(i, j)
\]  

(2.2)

where \(i\) is from 1 to \(L\), \(j\) is from 1 to \(W\), and \(n < m\).

**Histogram Equalization [30]**

Histogram modeling techniques (e.g. histogram equalization) provide a sophisticated
method for modifying the dynamic range and contrast of an image by altering that image such that its intensity histogram has a desired shape. Unlike contrast stretching, histogram modeling operators may employ non-linear and non-monotonic transfer functions to map between pixel intensity values in the input and output images. Histogram equalization employs a monotonic, non-linear mapping which re-assigns the intensity values of pixels in the input image such that the output image contains a uniform distribution of intensities (i.e. a flat histogram). This technique is used in image comparison processes (because it is effective in detail enhancement) and in the correction of non-linear effects introduced by, say, a digitizer or display system.

**Weighted Region Ranking [27]**

In a variation of so-called weighted region ranking, a circular neighborhood about each point is searched to count the number of points lighter and darker than the central point. This yields the rank of each pixel that we display as an output image. The size of the search region and the amount of histogram smoothing prior to ranking control the amount of enhancement performed.

**Constant Variance Enhancement [27]**

In many applications, the local image variance is a good measure of local contrast. Areas with low variance are essentially flat and have low contrast, while areas with high variance are often near edges and have high contrast. Constant variance enhancement
[27] is an approach that utilizes this fact to increase image contrast. For each point in the image, the output pixel is defined to be: \( \text{out} = \text{GM} + (\text{in} - \text{LM}) \times (\text{GS} / \text{LS}) \), where \( \text{GM} \) and \( \text{GS} \) are the global mean and standard deviation for the image, and \( \text{LM} \) and \( \text{LS} \) are the local mean and standard deviation in an \( N \) by \( N \) neighborhood of the output pixel.

These methods are generally used to capture a gray image with good contrast, but they can not be used for background elimination in the preprocessing stage. In this chapter, we will propose 3 methods based on contrast enhancement, and then apply the Ridler and Calvard thresholding [2].

### 2.2 Proposed Methods

#### 2.2.1 Global Linear Contrast Enhancement

In our application, the contrast enhancement is not only used for highlighting the details, but also for eliminating most of the background. Now let us look at some intensity parameters of the image, and based on this, we can find a new way of contrast enhancement for those images with non-uniformly distributed background.

Let \( \text{AVE} \) be the average intensity of the whole image, \( T_{\text{max}} \) be the maximum intensity of the foreground, and \( T_{\text{max}}, T_{\text{min}} \) be the maximum and minimum intensity value of the background respectively.

Because for those images with complex, poorly-illuminated and non-uniformly distributed background:
(1) The background is always lighter than the foreground (in this thesis, only these classes of images are considered; actually if the foreground is lighter than the background, the analysis is the same);

(2) The background occupied a larger area than the foreground.

We can find the relationships between these parameters as:

\[ T_{f_{\text{max}}} < T_{b_{\text{min}}} < AVE < T_{b_{\text{max}}} \]  \hspace{1cm} (2.3)

For example: in an image, the minimal and maximal intensity value of foreground is 0 and 50 respectively. The minimal and maximum intensity value of background is 150 and 200 respectively. The total number of foreground and background pixels is 500 and 10,000 respectively. The average intensity value of this image is calculated approximately by:

\[ AVE = \frac{(50/2) \times 500 + [(150 + 200)/2] \times 10000}{500 + 10000} = 176.25 \]

Which is between the minimal and maximum intensity value of the background.

Although equation 2.3 is designed for images with complex, poorly illuminated and non-uniformly distributed background, it can also be applied to the normal documents, with uniform background. For these kinds of images, we only need to consider a certain intensity value for the background. For example for images with uniform background, the background can be taken as \( T_b \). In this case, equation 2.3 can be simplified to equation 2.4.

\[ T_{f_{\text{max}}} < AVE < T_b \]  \hspace{1cm} (2.4)
And after applying equation 2.4 to images with uniform background, all the background could be eliminated. Experimental results for these types of images will be shown in the simulation results part.

As a result, the average intensity of the whole image should be greater than the minimum intensity of the background, which is also greater than the maximum intensity of the foreground. Therefore all pixels whose intensity values are above the average intensity of the whole image could be eliminated as background. And then we linearly extend the intensity of remaining pixels to the whole range [0, 255], which could increase the contrast between the foreground and the background. The following is the equation for the new contrast enhancement method:

\[
y2(i, j) = \begin{cases} 
255; & \text{if } (y1(i, j) \geq AVE) \\
\frac{(y1(i, j) - \text{MIN}) \times 255}{AVE - \text{MIN}}; & \text{if } (\text{MIN} \leq y1(i, j) < AVE) 
\end{cases} \tag{2.5}
\]

Where \(y2(i, j)\) is the pixel intensity value of output image of the preprocessing stage, and \(y1(i, j)\) is pixel intensity after contrast enhancement. MIN is the minimum luminance value of the input image, and AVE is the average luminance value of the input image.

In this way, we can eliminate part of the background whose intensity is between \([\text{AVE}, T_{\text{max}}]\), for those background between \([T_{\text{min}}, \text{AVE}]\), they still exist. But the intensity difference between the foreground and the remaining background has been greatly increased. So after this, we can apply thresholding methods to get the final binary sequence.
Since in equation 2.5, the contrast enhancement is applied to the entire image, and also the relationship between the output and input images is linear, we call this method global linear contrast enhancement.

### 2.2.2 Global Nonlinear Contrast Enhancement

In 2.2.1, the contrast is enhanced linearly. However, the contrast is not enhanced enough when the gray levels of many pixels are distributed near the minimum gray level and the maximum gray level in the image. Especially in our case, the purpose of contrast enhancement is to eliminate background, which means we should make the mapping weighted toward higher (brighter) values to eliminate more background. In order to achieve this, the linear contrast enhancement can be upgrade to the nonlinear contrast enhancement.

The difference of linear and nonlinear contrast enhancement is shown in figure 2.2. Line 1 is the curve for linear contrast enhancement, in which the pixels between [MIN, AVE] are linearly mapped into [0, 255]. Line 2 is the curve for nonlinear contrast enhancement, in which the pixels between [MIN, AVE] are nonlinearly mapped into [0,255] in a way that weighted towards higher intensity value. In this way, the mapping of pixels whose intensity is close to AVE is shrunk, while the mapping of pixels whose intensity is close to MIN is expanded.
This is also called Gamma correction [31] in computer graphics (figure 2.3). Gamma correction controls the overall brightness of an image. Images that are not properly corrected can look either bleached out, or too dark. Trying to reproduce colors accurately also requires some knowledge of gamma. Gamma can be any value between 0 and infinity. If gamma is 1 (the default), the mapping is linear. If gamma is greater than 1, the mapping is weighted toward higher (brighter) output values. If gamma is less than 1, the mapping is weighted toward lower (darker) output values. In our application, we are using $p$ greater than 1 to map the pixel toward higher intensity value. And from figure 2.2, one can figure out that when $p$ is increased, the contrast enhancement is getting more towards the higher intensity value.
Figure 2.3, Gamma Correction

This nonlinear contrast enhancement can be realized using the following equations:

\[
y2(i, j) = \begin{cases} 
255; & \text{if } (y1(i, j) \geq AVE) \\
\left[ - \frac{(y1(i, j) - AVE)^p}{r} \right] + 255; & \text{if } (MIN \leq y1(i, j) < AVE) 
\end{cases}
\]  \hspace{1cm} (2.6)

with

\[
r = \frac{(AVE - MIN)^p}{255}
\]  \hspace{1cm} (2.7)

Where \(y2(i, j)\) is the pixel intensity of output image of the preprocessing stage, and \(y1(i, j)\) is pixel intensity before contrast enhancement. \(MIN\) is the minimum luminance of the input image, and \(AVE\) is the average luminance of the input image. \(r\) is the nonlinear parameter and \(p\) is the power parameter and is greater than 1.

Compared with linear contrast enhancement, the nonlinear contrast enhancement makes
the background lighter, and the foreground darker, which makes the result much better prepared for the thresholding.

2.2.3 Local Nonlinear Contrast Enhancement

The method introduced in 2.2.2 is used on the whole image, so we call it global contrast enhancement. But in some cases, the global intensity range of the background is very large, while in some locations, the intensity range is very small. Or the global minimum intensity of the background is very close to the maximum intensity of the foreground, while in some locations, they are not. Then after the global contrast enhancement, the following problems may occur:

1. Most of the background is still there, because the global intensity range of the background is too large;
2. The difference between the foreground and the background can not be greatly increased;

One method that can be used to solve these problems is local contrast enhancement. Although in some cases, the intensity range of the background is very large in the whole image, or the minimum intensity of the background is very close to the maximum intensity of the foreground. If we divide the whole image into sub regions, in each sub image, these kinds of problems should not exist, or be improved a lot.

Therefore, we can improve the global contrast enhancement by partitioning the image
into N-by-N sub regions. And apply contrast enhancement introduce in 2.2.3 on each sub region.

\[
y_{2}(i, j) = \begin{cases} 
255; & \text{if } y_{1}(i, j) \geq AVE(k) \\
\frac{(y_{1}(i, j) - AVE(k))^{p}}{r(k)} + 255; & \text{if } MIN(k) \leq y_{1}(i, j) < AVE(k) 
\end{cases}
\]

(2.8)

\[
r(k) = \frac{(AVE(k) - MIN(k))^{p}}{255}
\]

(2.9)

Where \( y_{2}(i, j) \) is the luminance of pixel in sub region \( k \) after contrast enhancement, and \( y_{1}(i, j) \) is luminance of pixel in sub region \( k \) of original image. \( MIN(k) \) is the minimum luminance of sub region \( k \), and \( AVE(k) \) is the average luminance of sub region \( k \). \( r(k) \) is the nonlinear parameter of sub region \( k \) and \( p \) is the power parameter in the range \([1, \infty)\). Selection of parameter \( p \) and \( N \) will be discussed in details in the simulation results of this chapter.

2.3 Ridler and Calvard thresholding [2] [3]

After local nonlinear contrast enhancement, the image with complex, non-uniformly distributed background is improved. Most of the background has been eliminated and the contrast between the foreground and the remaining background has been greatly increased, which makes the thresholding much easier. We now can apply an efficient thresholding method such as that of Ridler and Calvard [2] [3]. The single level threshold selection is given by the following equations:
\[ T_{k+1} = \frac{1}{2} \left( \frac{\sum_{i=0}^{T_k} n(i) \times i}{\sum_{i=0}^{T_k} n(i)} + \frac{\sum_{i=T_{k+1}}^{N} n(i) \times i}{\sum_{i=T_{k+1}}^{N} n(i)} \right) \] (2.10)

Until \[ T_{k+1} = T_k \] (2.11)

In equation 2.10, \( n(i) \) is the total number of gray-levels having value \( i \) (from 0 to 255), and \( T_{k+1}, T_k \) are the thresholds at iterations \( k+1 \) and \( k \), respectively. Since the right-hand side of equation 2.10 could yield a real number, it is rounded to the nearest integer value after every iteration. Convergence is obtained when equation 2.11 is satisfied.

We can extend the single level thresholding to the selection of multi-level thresholds [2], shown in equation 2.12 and 2.13.

\[ T_{i,k+1} = \frac{1}{2} \left( \frac{\sum_{i=T_{i-1,k}}^{T_{i,k}} n(i) \times i}{\sum_{i=T_{i-1,k}}^{T_{i,k}} n(i)} + \frac{\sum_{i=T_{i,k+1}}^{T_{i+1,k}} n(i) \times i}{\sum_{i=T_{i,k+1}}^{T_{i+1,k}} n(i)} \right) \] (2.12)

Until \[ T_{i,k+1} = T_{i,k} \] (2.13)

Where \( i = 1, 2, \ldots, M; M = \text{number of thresholds.} \) The first subscript in T represents the \( i \)th threshold level, and the second subscript represents the iteration number. After multi-level thresholding, there will be \( M \) regions in the image stand for \( M \) gray levels. In
our application, we need to generate binary sequence, which is needed in the page segmentation. So single level thresholding is used in our simulation.

2.4 Simulation Results

So far we have proposed 3 methods in the preprocessing stage: global linear contrast enhancement, global nonlinear contrast enhancement and local nonlinear contrast enhancement. Now let us look at some simulation results.

Figure 2.4 lists several image examples. Image example 1 is a document with uniform background. And image 2, 3 are documents with poorly illuminated, non-uniformly distributed background.

PAGE SEGMENTATION OF DIGITIZED COMPOSITE DOCUMENTS HAS BEEN STUDIED EXTENSIVELY IN THE LITERATURE. MOST OF THE METHODS PRESENTED ARE BASED ON THE ASSUMPTION THAT COMPOSITE DOCUMENTS HAVE A PLAIN BACKGROUND. THEREFORE, IT IS NECESSARY TO PERFORM A PREPROCESSING PRECEDURE TO EXTRACT THE TEXT BEFORE A RECOGNITION ALGORITHM IS APPLIED.

(2.4a) Image Example 1
Figure 2.4 Image Examples
2.4.1 Results from Ridler and Calvard Thresholding [2] without preprocessing

First we only apply Ridler and Calvard thresholding without any preprocessing stage, and the results are given in figure 2.5.

PAGE SEGMENTATION OF DIGITIZED COMPOSITE DOCUMENTS HAS BEEN STUDIED EXTENSIVELY IN THE LITERATURE. MOST OF THE METHODS PRESENTED ARE BASED ON THE ASSUMPTION THAT COMPOSITE DOCUMENTS HAVE A PLAIN BACKGROUND. THEREFORE, IT IS NECESSARY TO PERFORM A PREPROCESSING PRECEDURE TO EXTRACT THE TEXT BEFORE A RECOGNITION ALGORITHM IS APPLIED.

(2.5a) Result for Image Example 1
From the results, we can see Ridler and Calvard Thresholding is efficient for images with light and uniform background, but not for those images with poorly illuminated, non-uniformly distributed background.
2.4.2 Results from Using Global Linear Contrast Enhancement + Ridler and Calvard Thresholding

PAGE SEGMENTATION OF DIGITIZED COMPOSITE DOCUMENTS HAS BEEN STUDIED EXTENSIVELY IN THE LITERATURE. MOST OF THE METHODS PRESENTED ARE BASED ON THE ASSUMPTION THAT COMPOSITE DOCUMENTS HAVE A PLAIN BACKGROUND. THEREFORE, IT IS NECESSARY TO PERFORM A PREPROCESSING PRECEDURE TO EXTRACT THE TEXT BEFORE A RECOGNITION ALGORITHM IS APPLIED.

(2.6a) Result for Image Example 1

Document Example

Surname LINYITAO
First Name JANETTE
Nationality CHINA
Sex F
Date of Birth 01.01.1976
Date of Expiry 01.01.2005

(2.6b) Result for Image Example 2
<table>
<thead>
<tr>
<th>Surname</th>
<th>LINYITAO</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>JANETTE</td>
</tr>
<tr>
<td>Nationality</td>
<td>CHINA</td>
</tr>
<tr>
<td>Sex</td>
<td>F</td>
</tr>
<tr>
<td>Date of Birth</td>
<td>01.01.1976</td>
</tr>
<tr>
<td>Date of Expiry</td>
<td>01.01.2005</td>
</tr>
</tbody>
</table>

(2.6c) Result for Image Example 3

Figure 2.6, Results from Using Global Linear Contrast Enhancement + Ridler and Calvard Thresholding

From figure 2.6, it is obvious that after using global linear contrast enhancement prior to thresholding, the performance is much better than using only the thresholding.
2.4.3 Results from Using Global Nonlinear Contrast Enhancement +
Ridler and Calvard Thresholding

PAGE SEGMENTATION OF DIGITIZED COMPOSITE DOCUMENTS HAS BEEN STUDIED EXTENSIVELY IN THE LITERATURE. MOST OF THE METHODS PRESENTED ARE BASED ON THE ASSUMPTION THAT COMPOSITE DOCUMENTS HAVE A PLAIN BACKGROUND. THEREFORE, IT IS NECESSARY TO PERFORM A PREPROCESSING PRECEDURE TO EXTRACT THE TEXT BEFORE A RECOGNITION ALGORITHM IS APPLIED.

(2.7a) Result for Image Example 1

Document Example 2

Surname  LINYITAO
First Name  JANETTE
Nationality  CHINA
Sex  F
Date of Birth  01.01.1976
Date of Expiry  01.01.2005

(2.7b) Result for Image Example 2
Document Example3

Surname  LINYITAO
First Name  JANETTE
Nationality  CHINA
Sex  F
Date of Birth  01.01.1976
Date of Expiry  01.01.2005

(2.7c) Result for Image Example3

Figure 2.7 Results from Using Global Nonlinear Contrast Enhancement + Ridler and Calvard Thresholding
2.4.4 Results from Using Local Nonlinear Contrast Enhancement +

Ridler and Calvard Thresholding

PAGE SEGMENTATION OF DIGITIZED COMPOSITE DOCUMENTS HAS BEEN STUDIED EXTENSIVELY IN THE LITERATURE. MOST OF THE METHODS PRESENTED ARE BASED ON THE ASSUMPTION THAT COMPOSITE DOCUMENTS HAVE A PLAIN BACKGROUND. THEREFORE, IT IS NECESSARY TO PERFORM A PREPROCESSING PRECEDURE TO EXTRACT THE TEXT BEFORE A RECOGNITION ALGORITHM IS APPLIED.

(2.8a) Image Example 1

Document Example 2

Surname LINYITAO
First Name JANETTE
Nationality CHINA
Sex F
Date of Birth 01.01.1976
Date of Expiry 01.01.2005

(2.8b) Image Example 2
2.4.5 Comparison of Proposed Methods

Now let us take a closer look at one more example here to make a performance comparison of three proposed methods:
(2.9b) Preprocessing Result Using Global Linear Contrast Enhancement

(2.9c) Preprocessing Result Using Global Nonlinear Contrast Enhancement
The image in 2.9a is in bad illumination and with complex, non-uniformly distributed background. The results from using those 3 proposed methods are in 2.9b, 2.9c and 2.9d for global linear contrast enhancement, global nonlinear contrast enhancement and local nonlinear contrast enhancement respectively. Comparing the results from three methods, we may find that the result using global linear contrast enhancement (2.9b) is very bad, it has many background left because of the bad illuminated and non-uniform background. In figure 2.9c, the image is much better using global nonlinear contrast enhancement, since the nonlinear operation removes almost all the background. But the strokes are not uniform. That is because in the bad illuminated image, some foreground is paler and has almost the same intensity as the background. When applying nonlinear mapping, some foreground text is mapped into a much higher intensity, which is eliminated as background. Finally in 2.9d, when we use local nonlinear contrast enhancement, not only
the background is removed, but also the strokes are uniform, which gives the best result.

Here is a table comparing these three methods:

<table>
<thead>
<tr>
<th>Method</th>
<th>Dealing with Different Backgrounds</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Complex</td>
</tr>
<tr>
<td>Global Linear Contrast Enhancement</td>
<td>Good</td>
</tr>
<tr>
<td>Global Nonlinear Contrast</td>
<td>Good</td>
</tr>
<tr>
<td>Enhancement</td>
<td></td>
</tr>
<tr>
<td>Local Nonlinear Contrast</td>
<td>Good</td>
</tr>
<tr>
<td>Enhancement</td>
<td></td>
</tr>
</tbody>
</table>

From the table, the local nonlinear contrast enhancement has the best performance. Because for those images with poorly illuminated and non-uniform background, we can’t get good results using global methods. Only if we use the local contrast enhancement, in each sub region, the luminance could be regarded as uniform, thus generate good result.

2.4.6 Discussion of Parameter N

When we use the local nonlinear contrast enhancement, it is very important to divide the image into reasonable N-by-N sub regions. If N is too big, some sub regions don’t have any foreground in it, as a result the darker part of the background will be regarded as foreground. If N is too small, in some sub regions, the intensity range of the background
is still very large, or the minimum intensity of the background is still very close to the maximum intensity of the foreground. One example of results from different N is given in figure 2.10.

Document Example 7

Surname          LINYITAO
First Name        JANETTE
Nationality      CHINA
Sex               F
Date of Birth    01.01.1976
Date of Expiry   01.01.2005

(2.10a) N=2

Document Example 7

Surname          LINYITAO
First Name        JANETTE
Nationality      CHINA
Sex               F
Date of Birth    01.01.1976
Date of Expiry   01.01.2005

(2.10b) N=3
Document Example 7

Surname: LINVITAO
First Name: JANETTE
Nationality: CHINA
Sex: F
Date of Birth: 01.01.1976
Date of Expiry: 01.01.2005

(2.10c) N=4

Document Example 7

Surname: LINVITAO
First Name: JANETTE
Nationality: CHINA
Sex: F
Date of Birth: 01.01.1976
Date of Expiry: 01.01.2005

(2.10d) N=5
In figure 2.10, there are 5 images from using different N. Table 2.2 shows the performance comparison using different N.

<table>
<thead>
<tr>
<th>N</th>
<th>Remaining Background</th>
<th>Non-uniform stroke</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>3</td>
<td>No</td>
<td>Yes</td>
</tr>
<tr>
<td>4</td>
<td>No</td>
<td>No</td>
</tr>
<tr>
<td>5</td>
<td>Yes</td>
<td>No</td>
</tr>
<tr>
<td>6</td>
<td>Yes</td>
<td>No</td>
</tr>
</tbody>
</table>

When N=2 (figure 2.10a), the image quality is poor, there are some remaining background; and the strokes of characters are non-uniform, some are thick, while some
are thin. When N=3 (figure 2.10b), the image is still not acceptable, although there is almost no background, but the strokes are not uniform. When N=4 (figure 2.10c), the image is good, not only the background is almost removed, but also all the characters are in uniform stroke. When N=5 (figure 2.10d), the image is not good, because there is some background introduced. When N=6 (figure 2.10e), although the strokes of characters are uniform, more background is introduced. We have tested the methods on many images, and from the experiments, N=4 give the best result.

2.4.7 Discussion of Power Parameter p

Another important parameter used here is the power parameter p. If p is too large, in some images, pixels are mapped into higher intensity; as a result, some foreground could also be regarded as background. Here is an example of results from using different p.

Document Example 7

Surname    LINYITAO
First Name  JANETTE
Nationality CHINA
Sex         F
Date of Birth 01.01.1975
Date of Expiry 01.01.2005

(2.11a) p=1.0
<table>
<thead>
<tr>
<th>Surname</th>
<th>LINYITAO</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>JANETTE</td>
</tr>
<tr>
<td>Nationality</td>
<td>CHINA</td>
</tr>
<tr>
<td>Sex</td>
<td>F</td>
</tr>
<tr>
<td>Date of Birth</td>
<td>01.01.1976</td>
</tr>
<tr>
<td>Date of Expiry</td>
<td>01.01.2005</td>
</tr>
</tbody>
</table>

(2.11b) \( p=1.3 \)

<table>
<thead>
<tr>
<th>Surname</th>
<th>LINYITAO</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>JANETTE</td>
</tr>
<tr>
<td>Nationality</td>
<td>CHINA</td>
</tr>
<tr>
<td>Sex</td>
<td>F</td>
</tr>
<tr>
<td>Date of Birth</td>
<td>01.01.1976</td>
</tr>
<tr>
<td>Date of Expiry</td>
<td>01.01.2005</td>
</tr>
</tbody>
</table>

(2.11c) \( p=1.5 \)
<table>
<thead>
<tr>
<th>Surname</th>
<th>LINYITAO</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>JANETTE</td>
</tr>
<tr>
<td>Nationality</td>
<td>CHINA</td>
</tr>
<tr>
<td>Sex</td>
<td>F</td>
</tr>
<tr>
<td>Date of Birth</td>
<td>01.01.1976</td>
</tr>
<tr>
<td>Date of Expiry</td>
<td>01.01.2005</td>
</tr>
</tbody>
</table>

(2.11d) p=1.8

<table>
<thead>
<tr>
<th>Surname</th>
<th>LINYITAO</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>JANETTE</td>
</tr>
<tr>
<td>Nationality</td>
<td>CHINA</td>
</tr>
<tr>
<td>Sex</td>
<td>F</td>
</tr>
<tr>
<td>Date of Birth</td>
<td>01.01.1976</td>
</tr>
<tr>
<td>Date of Expiry</td>
<td>01.01.2005</td>
</tr>
</tbody>
</table>

(2.11e) p=2.0
Document Example 7

Surname: LINVITAO
First Name: JANETTE
Nationality: CHINA
Sex: F
Date of Birth: 01.01.1976
Date of Expiry: 01.01.2005

(2.11b) p=2.8

Document Example 7

Surname: LINVITAO
First Name: JANETTE
Nationality: CHINA
Sex: F
Date of Birth: 01.01.1976
Date of Expiry: 01.01.2005

(2.11i) p=3
Document Example 7

Surname: LIN YITAO
First Name: JANETTE
Nationality: CHINA
Sex: F
Date of Birth: 01.01.1976
Date of Expiry: 01.01.2005

(2.11) p=4

Figure 2.11, Simulation Result of Using Different Parameter p

P is a very important parameter in the nonlinear contrast enhancement. If p is too small, it will not be able to eliminate the background; if p is too large, the pixels are mapped to a much higher intensity, which makes some characters blurry and broken. So we should be very careful to choose p. Figure 2.11 gives some result using p between 1 and 4. From the results, we can say, p around 2 gives the best result.

So in the preprocessing simulations, we use the local nonlinear contrast enhancement with p=2, N=4, which gives the best performance.

2.5 Summary and Conclusion

In this chapter the fundamental concepts of document analysis have been presented. In order to provide the binary sequence to the page segmentation stage, three new methods: global linear contrast enhancement, global nonlinear contrast enhancement, local...
nonlinear contrast enhancement are proposed to eliminate most of the background. After that, Ridler and Calvard thresholding is used to get the binary sequence.

These algorithms are tested on many document samples and security documents. Based on the simulation results, local contrast enhancement has the best performance. Meanwhile, two important parameters $p$ and $N$ are discussed in more details. And $p=2$, $N=4$ give the best result.

The computational speed of proposed methods is very fast. For an image with size of $512\times768$, the methods only cost less than one second.

After the preprocessing, we should end up with a foreground on a uniform (or white) background, and proceed to the page segmentation stage.
Chapter 3: Page Segmentation

3.1 General Introduction

Automatic block segmentation, labeling and classification of a digitized document image are necessary elements of a document analysis system. While it is easy for human visual system to distinguish between text, graphs, background and Images in an image document, unfortunately this is not an easy task for machine to make such distinguish. Therefore before any document recognition or analysis algorithm to work, a page segmentation algorithm is normally utilized. A page segmentation algorithm is consisting of three stages:

- Block segmentation: documents are divided into blocks;
- Block labeling: labels are assigned to each block;
- Block classification: blocks are classified as different type of data based on different features.

In this chapter, we will discuss these sub stages in more details. The Run Length Smearing Algorithm is used for segmentation of the document. Then a new labeling method will be proposed. And finally, a linear classifier for block classification will be introduce.
3.2 Block Segmentation

In this section, we discuss the block segmentation stage [32], which is a procedure that subdivides the area of a digitized image into blocks in order to process the document images systematically. Each of which ideally is required to contain only one type of data.

Basically the document image can be segmented into blocks by two different methods: bottom-up and top-down.

- **Bottom-up Approach**

In the bottom-up approach, all the components in the document images are individually detected and then merged together into larger blocks. The image is first processed to determine the individual connected components. In the case of text, the characters are merged into words; words are merged into lines, lines into paragraphs and paragraphs into even larger blocks, if such a merging is possible. In the bottom-up approach, it is usually necessary to determine whether a connected component is a part of: text, or image. Possible features for performing this classification are: size, branching structure, topology and shape measures. Although the bottom-up clustering is insensitive to the document’s skew angle, it is slower and unreliable due to decisions made with low statistical confidence in small neighborhoods.

- **Top-down Approach**

In the top-down approach, certain global operations are performed on the entire image to divide it into major regions that are further divided into sub-regions. This approach is fast
and very effective for processing the documents.

The most well known techniques of top-down approach are Run Length Smearing Algorithm (RLSA) [10] and projection profile cuts or Recursive X-Y Cuts (RXYC) [11]. If large blocks corresponding to paragraphs are needed, then the RXYC is better than the RLSA, which will need a merging algorithm to merge blocks corresponding to single text lines into blocks. On the other hand, the RLSA is preferred to obtain small blocks of text line.

Table 3.1 lists the comparison between the three methods: Bottom-up, RLSA and RXYC.

<table>
<thead>
<tr>
<th>Method</th>
<th>Computational Speed</th>
<th>Complexity</th>
<th>Other Shortcomings</th>
</tr>
</thead>
<tbody>
<tr>
<td>Run Length Smearing Algorithm (Top-Down)</td>
<td>Fast</td>
<td>Low</td>
<td>The results are dependent on the thresholds.</td>
</tr>
</tbody>
</table>
| Recursive X-Y Cut (Top-Down)          | Medium              | Medium     | (1) The computational time depends on the number of the components.  
                                           |                     |            | (2) There is still some thresholds involved            |
| Bottom-up Method                      | Slow                | High       | (1) This approach requires extensive usage of memory resources.  
                                           |                     |            | (2) It also depends on some thresholds when making merging. |

From the table, the Run Length Smearing Algorithm has the best performance in the page segmentation. Because:
• Its computational speed is fast.

• For normally document spacing, the thresholds are proportional to the width and height of the document.

**Run Length Smearing Algorithm [10] [32]**

The Run Length Smearing Algorithm (RLSA) has been used by many document analysis systems for block segmentation. It is used to obtain a bit-map of white and black areas representing blocks containing the various types of data. The basic RLSA is applied to a binary sequence for which white pixels are represented by 0’s and black pixels by 1’s. The algorithm transforms a binary sequence x into an output sequence y according to the following rules:

1. 0’s in x are changed to 1’s in y if the number of adjacent 0’s is less than or equal to a predefined threshold t;
2. 1’s in x are unchanged in y.

For example, for t=2;

Data= 001000110100001

Smearred data = 111000111100001

This operation is applied row by row as well as column by column to the binary image from preprocessing stage, yielding two distinct bit-maps for horizontal and vertical directions. When applied to pattern arrays, the RLSA has the effect of linking together neighboring black areas that are separated by less than t pixels. Because spacings of document components tend to differ horizontally and vertically, different values of t are
used for the horizontal and vertical directions. After that, the two bit maps are combined by applying a logic AND operation to each pixel. This combination gives almost the desired final result. However, some black regions representing text lines are interrupted by small gaps. Therefore, an additional horizontal smearing is applied again with smaller threshold $t$ to produce the final segmentation result. This is especially necessary when documents with uniform character spacings are to be processed.

Here are the detail steps of the algorithm:

1. Apply the horizontal smearing, which means row-by-row;
2. Apply the vertical smearing, which means column-by-column;
3. Apply a logical AND operation on each pixel;
4. Apply a final horizontal smearing with small threshold to generate the final result.

There are 3 threshold values involved here: threshold for the horizontal smearing ($Th$), threshold for the vertical smearing ($Tv$) and the threshold for the final smoothing smear ($Tf$). These smearing threshold values are very important and have a large influence on the results of the RLSA. For example, a very small value of the horizontal threshold $Th$ simply closes individual characters in a word. A slightly larger value of the threshold smears together individual characters in a word. But if the threshold is not large enough, it will not cover the inter-word space. On the other hand, larger value of $Th$ often causes sentences to be joined with non-text regions, or to be connected with adjacent columns. Similar comments hold for $Tv$. Too large value of $Tv$ may link different text lines into one block and also may join text lines with non-text regions. In these cases, segmented
blocks may lead to error in classification.

In our application, we assume they are always in normal spacing. The normal spacing can be described as:

- Horizontally, 1 pixel between characters and 5 pixels between words.
- Vertically, 10 pixels between lines.

After an exhaustive simulation, I found out that in most cases, these thresholds are proportional to the size of the image. So I set these values are follows:

- $Th$ (threshold of horizontal smearing) equals to $1/5$ of the image width;
- $Tv$ (threshold of vertical smearing) equals to $1/5$ of the image length;
- $Tf$ (threshold of final smearing) equals to $1/50$ of the image width.

These values were tested on many documents, and were found to be adequate for those images with normal spacing. Although for almost all the documents, they are with normal spacing. In some special cases, for example, wide spacing or narrow spacing, we may need to change threshold values linearly. For example, if a document has wider spacing with 10 pixels between words, which is two times of normal horizontal spacing. Then all the horizontal threshold values should be changed to two times as much as default value. On the other hand, if another document has narrow spacing with 5 pixels between lines, which is half of normal vertical spacing. Then all the vertical threshold values should be change to half of default value.
3.3 Block Labeling

After block segmentation, in order to identify each block separately for subsequent feature extraction and classification, labels are assigned to different blocks. There are two criteria for the labeling:

- Different blocks should have different labels;
- Pixels in the same block should have the same label;

Traditional connected components labeling scans an image and groups its pixels into components based on pixel connectivity, i.e. all pixels in a connected component share similar pixel intensity values and are in some way connected with each other. Once all groups have been determined, each pixel is labeled with a gray level or a color (color labeling) according to the component it was assigned to.

In a traditional labeling method, it is assumed that all blocks have rectangular shape. But when there is any concave shape in the block, there will be some problems with these traditional methods. For example, pixels in the same block will have different labels. In this section, we propose an efficient block labeling approach. Assuming image width and image length correspond to the width and length of the whole image. TYPE is a variable represents the total number of labels in the document. And for each pixel, we only consider its 8 neighbors and it has its own label. Here are the 3 steps of the approach:

**Step1:** Do the labeling from top to bottom, from left to right.

1. Set the labels of all the pixels to be 0.
2. Scan image starting from [0, 0] to [image length, image width]; TYPE (a variable stands for the total number of labels in the document) starts from 1.

3. If a pixel value is 1, and its black neighbors (pixel value is 1) are not labeled (which means their labels are all 0), label it as the current label TYPE, and increase TYPE by one (TYPE +1).

4. If a pixel value is 1, and it has black neighbors (pixel value is 1) whose label are not 0, then take the smallest (or largest) label of its neighbors. (If you take the smallest or largest label, then take the same direction in step 2 and 3. Doing this, aims to eliminate the redundant labels in either direction.)

5. If the pixel value is 0, ignore it.

After step 1, all the black pixels have their labels, but some pixels in the same block may have different labels.

**Step2:** Do the labeling from bottom to top, from left to right.

1. Using the result from step1, and start from [image length, 0] to [0, image width].

2. If the pixel value is 1, then check if the labels of its black neighbors are less than (or larger than) its label. If so, change its label to the smallest (or largest) label among its black neighbors.

**Step3:** Do the labeling from top to bottom, from right to left.

1. Using the result from step2, and start from [0, image width] to [image length, 0].

2. If the pixel value is 1, then check if the labels of its black neighbors are less than
(or larger than) its label. If so, change its label to the smallest (or largest) label among its black neighbors.

Here is an example to illustrate this approach; figure 3.1-3.4 gives the detail steps of the algorithm. In figure 3.1, it shows the binary sequence we get from RLSA. 0 stands for the white pixel, and 1 stands for the black pixel. As we can see, there is concave and convex shape in it.

```
0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ...```

Figure 3.1, binary sequence got from RLSA

In step 1, we do the labeling from the top left point to the bottom right point. After step 1, we can get:

```
```

Figure 3.2, block labeling after step 1
As we can see, there are 4 labels in this case, but actually only 2 labels are needed. This problem is because of the convex and concave shape of the block. So in step 2, we do the labeling from the top right point to the bottom left point, and we get the following:

![Figure 3.3, block labeling after step 2](image)

We may see, there are 2 improvements: one is that block 2 is combined with block 1; the other is that one row in block 4 is combined with block 3. But the left lines of block 4 are still there which should also be combined into block 3. Then we do step 3, and here is the result:

![Figure 3.4, block labeling after step 3](image)
After this step, the redundant 2 labels are eliminated; only leave the necessary labels for
the subsequent processing.

3.4 Block Classification

A method for block classification was proposed by Wahl [10] [32]. They have used
block height (the height of block) and block mean black pixel run length (mean of black
pixel run length in a block) as the basic features. A threshold is set with respect to the
height of the block, and this threshold is one parameter to classify text line blocks and
graphic images. It exploits the fact that text lines have approximately a constant and
small height. To calculate features for the block classification, useful measurements can
be performed simultaneously with labeling; for example, when assigning a label to a
pixel, a counter for this particular label, BC, can be increased by 1. When the labeling
procedure is finished, the states of these counters represent the areas of the corresponding
block segments. Similarly, coordinate of the surrounding rectangle of each block can be
measured by a simple comparison of the coordinates \(x, y\) of the currently labeled pixel
with four coordinate registers \(x_{\text{min}}, x_{\text{max}}, y_{\text{min}}, y_{\text{max}}\), associated with each label. They are
initialized as 255, 0, 255, and 0 respectively. With each label assignment, the coordinate
of the corresponding label are updated:

\[
x_{\text{min}} = \min (x, x_{\text{min}})
\]

\[
x_{\text{max}} = \max (x, x_{\text{max}})
\]

\[
y_{\text{min}} = \min (y, y_{\text{min}})
\]

\[
y_{\text{max}} = \max (y, y_{\text{max}})
\]
Measurement applied directly to the original data are useful as well as measurements performed on the bitmap from block segmentation. For example, the number of black pixels of the original data within each block can be calculated by using an additional counter DC for each pixel. Whenever a label assignment occurs, the associated DC is increased by 1 if and only if the pixel of the original input bitmap at this location is black. Another useful quantity is the number of horizontal white-black transitions TC of the original data within each block segment, which can be counted in a similar way. Furthermore, the measurements derived from adjacent regions have been merged simultaneously with updating labels; no additional measurements are needed in order to do this.

After labeling procedure is done, the final states of the coordinate registers provide the following measurements:

- Total number of black pixels in a segmented block (BC).
- Minimum x-y coordinates of a block and its x, y lengths (\(x_{\text{min}}, \Delta x, y_{\text{min}}, \Delta y\)).
- Total number of black pixels in original data from the same block (DC).
- Number of horizontal white-black transitions of original data (TC).

Classifying each block is done by computing several features for each block from the above measurements and then using a linear pattern classifier. The features are:

1. The height of each block segment:

\[ H = \Delta y \]
2. The eccentricity of the rectangle surrounding the block:

\[ E = \frac{\Delta x}{\Delta y} \]

3. The ratio of the number of block pixels to the area of the surrounding rectangle:

\[ S = \frac{BC}{(\Delta x \times \Delta y)} \]

4. The mean horizontal length of the black runs of the original data from each block:

\[ R = DC/TC \]

These features are used to classify the blocks into different classes. Text is considered to be the predominating data type in documents; it can be regarded as textured stripes of approximately constant height \( H \) and mean length of black runs \( R \); the text blocks should cluster with respect to these features.

The mean values \( H_m \) and \( R_m \) for the text cluster may vary within a certain range for different types of documents depending on character size and font. Furthermore, the standard deviation \( \sigma(H) \) and \( \sigma(R) \) of the text cluster may vary depending on whether a document is printed in a single font or with discrimination scheme for text, which adjusts the decision boundaries within a certain range according to the properties of the predominant text on a particular document. A three-step, self-adjusting classifier for text is proposed here.

**Step 1:**

To estimate the means \( H_m \) and \( R_m \) and the standard deviations \( \sigma(H) \) and \( \sigma(R) \) of a supposed text block, blocks satisfying the following intuitive constraints are selected:
H/R > C1
H < C2
E > C3
S > C4

Blocks satisfying these constraints are highly likely to be text lines. Where C1 = 4, C2 = 100, C3 = 10 and C4 = 0.5 based on Wahl's experimentations [10] [32]. The average text height $H_m = 30.6$, $R_m = 3.6$, $\sigma(H) = 2.33$ and $\sigma(R) = 0.66$.

Step 2:

Using the estimate text cluster properties H, R, $\sigma(H)$ and $\sigma(R)$, the following tests ascertain whether the data constitute a reasonable cluster at all:

Number N of blocks in the cluster > C11

Ratio N to total number of blocks > C12

$R_m < C13$

$H_m < C14$

$\sigma(H) < C15$

$\sigma(R) < C16$

$\sigma(H)/H < C17$

$\sigma(R)/R < C18$
If all these conditions are met by the properties of the hypothesized cluster, then it is decided that a text cluster exists and the block discrimination is performed in the next step. Wahl [10] [32] has determined these parameters by experimentation and found their value are: $C_{11}=10$, $C_{12}=0.5$, $C_{13}=8$, $C_{14}=60$, $C_{15}=5$, $C_{16}=2$, $C_{17}=0.5$, $C_{18}=0.5$; the cluster generated by the features of the document meets these conditions.

Step3:

A variable, linear, separable classification scheme is used to assign the following four classes to the blocks:

Class 1: Text

\[ R < C_{21} \times R_m \]

\[ H < C_{22} \times H_m \]

Class 2: Horizontal solid black lines:

\[ R > C_{21} \times R_m \]

\[ H < C_{22} \times H_m \]

Class 3: Non-text (graphic and halftone images)

\[ E > 1/C_{23} \]

\[ H > C_{22} \times H_m \]

Class 4: Vertical solid black lines:
E<1/C23
H>C22× H_m

Noting that values have been assigned to the parameters Cij based on several training documents in Wahl’s experiments [10] [32], C21=3, C22=3 and C23=5.

In our application dealing with general documents, generally there are only two types of data: text and image. So the classification becomes much easier.

3.5 Simulation Results

3.5.1 Simulation Results of Page Segmentation Procedure

Based on the results from the preprocessing stage, we apply the RLSA on the image; then do the labeling; and finally make the classification. Figure 3.5 shows the whole procedure of the page segmentation, including horizontal smearing, vertical smearing, logical AND, final smoothing smearing, block labeling and classification.

And we use the thresholds as follows:

- Th (threshold of horizontal smearing) equals to 1/5 of the image width;
- Tv (threshold of vertical smearing) equals to 1/5 of the image length;
- Tf (threshold of final smearing) equals to 1/50 of the image width.
Document Example 4

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surname</td>
<td>LINYITAO</td>
</tr>
<tr>
<td>First Name</td>
<td>JANETTE</td>
</tr>
<tr>
<td>Nationality</td>
<td>CHINA</td>
</tr>
<tr>
<td>Sex</td>
<td>F</td>
</tr>
<tr>
<td>Date of Birth</td>
<td>01.01.1976</td>
</tr>
<tr>
<td>Date of Expiry</td>
<td>01.01.2005</td>
</tr>
</tbody>
</table>

(3.5a) Binary Image getting from Preprocessing Stage

(3.5b) Horizontal Smearing
(3.5c) Vertical Smearing

(3.5d) Logical AND
(3.5e) Final Smoothing Smearing

(3.5f) Block Labeling
Document Example 4

Surname: LINYITAO
First Name: JANETTE
Nationality: CHINA
Sex: F
Date of Birth: 01.01.1976
Date of Expiry: 01.01.2005

(3.5g) Text Extracted

(3.5h) Image Extracted

Figure 3.5, Detail Steps of Page Segmentation

Chapter 3: Page Segmentation
3.5.2 Final Results

The proposed algorithm is tested on a lot of real documents. Due to the confidential information they may contain, those images are not presented in the thesis. I generated some document examples with bad illuminated, complex and non-uniformly distributed background to be used, and figure 3.6 shows some of the final results. There are 9 examples here. The first seven images are with complex, bad-illuminated and non-uniformly distributed background, and then another 2 images with no background or with light and uniform background.
Document Example1

Surname: LINYITAO
First Name: JANETTE
Nationality: CHINA
Sex: F
Date of Birth: 01.01.1976
Date of Expiry: 01.01.2005

(3.6b) Extracted Text of Document Example 1
Document Example 2

Surname: LINYITAO
First Name: JANETTE
Nationality: CHINA
Sex: F
Date of Birth: 01.01.1976
Date of Expiry: 01.01.2005

(3.6d) Extracted Text of Document Example 2
### Document Example 3

<table>
<thead>
<tr>
<th>Surname</th>
<th>LINYITAO</th>
</tr>
</thead>
<tbody>
<tr>
<td>First Name</td>
<td>JANETTE</td>
</tr>
<tr>
<td>Nationality</td>
<td>CHINA</td>
</tr>
<tr>
<td>Sex</td>
<td>F</td>
</tr>
<tr>
<td>Date of Birth</td>
<td>01.01.1976</td>
</tr>
<tr>
<td>Date of Expiry</td>
<td>01.01.2005</td>
</tr>
</tbody>
</table>

(3.6f) Extracted Text of Document Example 3
<table>
<thead>
<tr>
<th>Field</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Surname</td>
<td>LINYITAO</td>
</tr>
<tr>
<td>First Name</td>
<td>JANETTE</td>
</tr>
<tr>
<td>Nationality</td>
<td>CHINA</td>
</tr>
<tr>
<td>Sex</td>
<td>F</td>
</tr>
<tr>
<td>Date of Birth</td>
<td>01.01.1976</td>
</tr>
<tr>
<td>Date of Expiry</td>
<td>01.01.2005</td>
</tr>
</tbody>
</table>

(3.6h) Extracted Text of Document Example 4
Document Example 5

Surname: LINYITAO
First Name: JANETTE
Nationality: CHINA
Sex: F
Date of Birth: 01.01.1976
Date of Expiry: 01.01.2005

(3.6j) Extracted Text of Document Example 5
(3.6k) Document Example 6

Surname        LINYITAO
First Name     JANETTE
Nationality    CHINA
Sex            F
Date of Birth  01.01.1976
Date of Expiry 01.01.2005

(3.6l) Extracted Text of Document Example 6
Document Example 7

Surname: LINYITAO
First Name: JANETTE
Nationality: CHINA
Sex: F
Date of Birth: 01.01.1976
Date of Expiry: 01.01.2005

(3.6n) Extracted Text of Document Example 7
A digitized image that consists of text strings and uniformly distributed background symbols must be segmented if the characters in the text string are to be recognized. This paper describes the development and implementation of a morphological approach to character string extraction from regular periodic overlapping text/background image that minimizes the shape distortion of characters. The effectiveness of this algorithm is demonstrated on several text images.

(3.6o) Document Example 8

A digitized image that consists of text strings and uniformly distributed background symbols must be segmented if the characters in the text string are to be recognized. This paper describes the development and implementation of a morphological approach to character string extraction from regular periodic overlapping text/background image that minimizes the shape distortion of characters. The effectiveness of this algorithm is demonstrated on several text images.

(3.6p) Extracted Text of Example 8
PAGE SEGMENTATION OF DIGITIZED COMPOSITE DOCUMENTS HAS BEEN STUDIED EXTENSIVELY IN THE LITERATURE. MOST OF THE METHODS PRESENTED ARE BASED ON THE ASSUMPTION THAT COMPOSITE DOCUMENTS HAVE A PLAIN BACKGROUND. THEREFORE, IT IS NECESSARY TO PERFORM A PREPROCESSING PRECEDURE TO EXTRACT THE TEXT BEFORE A RECOGNITION ALGORITHM IS APPLIED.

(3.6q) Document Example 9

PAGE SEGMENTATION OF DIGITIZED COMPOSITE DOCUMENTS HAS BEEN STUDIED EXTENSIVELY IN THE LITERATURE. MOST OF THE METHODS PRESENTED ARE BASED ON THE ASSUMPTION THAT COMPOSITE DOCUMENTS HAVE A PLAIN BACKGROUND. THEREFORE, IT IS NECESSARY TO PERFORM A PREPROCESSING PRECEDURE TO EXTRACT THE TEXT BEFORE A RECOGNITION ALGORITHM IS APPLIED.

(3.6r) Extracted Text of Document Example 9

Figure 3.6, Final Results of some images
3.6 Summary

In this chapter, page segmentation stage is discussed in considerable details. There is three sub stages in the page segmentation: block segmentation, block labeling and block classification. Run Length Smearing Algorithm (RLSA) is used in the block segmentation, and optimized smearing thresholds are discussed for our application. Then a new, efficient labeling method is proposed in the block labeling. Finally a three step linear classifier is used to make the classification.

The page segmentation algorithm is computational efficient. For an image with size 512*768, it only costs around 2 seconds to finish all 3 stages using a PIII computer.

The whole document analysis system proposed here is not only efficient for those images with poor illumination and non-uniformly distributed background, but also for images with no background or uniform background. Therefore, it can be used as a general document analysis system.
Chapter 4: Summary, Conclusions and Future Work

4.1 Summary

This thesis has explored applications of contrast enhancement and thresholding methods for implementation of a document analysis system. Systems for document understanding require a process that extracts text characters and other features, like photo, from such backgrounds that are often poorly and non-uniformly illuminated, so that the OCR system or the face recognition stage could properly function. In this thesis, preprocessing stage is used to eliminate background and create a binary sequence; then page segmentation stage is used to segment the document and extract the text.

In chapter 1, several methods dealing with documents with complex, poorly illuminated and non-uniformly distributed background are discussed and criticized. In chapter 2, preprocessing stage is introduced, in which 3 new contrast enhancement approaches: global linear contrast enhancement, global nonlinear contrast enhancement and local nonlinear contrast enhancement are proposed. Furthermore, Ridler and Calvard thresholding is used to generate the binary sequence. Then simulation results are given; comparison between three proposed methods is made and some parameters are discussed. In chapter 3, page segmentation stage is introduced, in which three sub stages are
discussed: block segmentation, block labeling and block classification. In block segmentation, Run Length Smearing Algorithm is used to separate different blocks. In block labeling, a new and efficient labeling method is proposed. In block classification, a linear classifier is used to classify different types of data and extract the text. The simulation results of final images are shown.

4.2 My Contributions

In this thesis, several methods dealing with text extraction from background are discussed and criticized. Three contrast enhancement methods in the preprocessing stage are proposed and compared. The Ridler and Calvard threshold selection method is applied to generate the binary sequence. Another new and efficient block labeling algorithm is proposed in page segmentation stage. The proposed methods are tested on a large number of real documents, though for the rules of privacy, only a few examples are shown.

4.3 Conclusions

Compared with other image preprocessing techniques [1]-[26], three proposed algorithms are efficient for not only images with poorly illuminated, non-uniformly distributed backgrounds, but also images with no background or uniform background. Therefore, the proposed document analysis system can be used as a general document analysis system. Also, the computational speed of the document analysis system is quite fast, which makes it a good candidate for real applications.
4.4 Future Work

The document analysis system proposed in this thesis is quite efficient, and it could be used in some real document analysis systems. In order to realize the authentication, OCR and face recognition are needed to form the document understanding system. We leave these as future work.
References


[28] G. Simon, M, Chatterjee, "Introduction to high pass filtering", Multidimensional Image Processing Laboratory, Penn State University


//-------------------------------------------------------------------------------
#include <vel.h>
#pragma hdrstop
#include <math.h>
#include <algorithm>
#include <stdio.h>
#include <time.h>
#include <vector>
#include <iostream>
#include <functional>

#define PI 3.141592654
#define MAX1 6548.45;
#define MAX2 478.26;
#define pi 3.1415192653
#define RAD (float)(pi/180.0)
#define MAX_THETA 90
#define DELTA_THETA 1 //0.2
#define RANGE_THETA 10 //15
#define min(a, b) (((a) < (b)) ? (a) : (b))
#define max(a, b) (((a) > (b)) ? (a) : (b))
#define NN 4
#define N2 16

#include "Unit1.h"
//-------------------------------------------------------------------------------
#pragma package(smart_init)
#pragma resource "*.dfm"
 TForm1 *Form1;
void hrlsa(int **Bi,int **ho,int th);
void vrlsa(int **Bi,int **ve,int tv);

int image_width,count,image_length,i,j,n1,n2,PEL,R,G,B,r,g,b,W,L;
float **w,**w2;
float skew,tt;
int counter;
int **Bi,**ho,**ve,**and,**final,**original,**original2;
int th,tt,ts,hh,hmax;
int have1,ttt;
int xmax[1000]={0},xmin[1000]={1000},ymax[1000]={0},ymin[1000]={1000},deltax[100]
0]={0},H[1000]={0},Area[1000]={0};
clock_t start,end;
__fastcall TForm1::TForm1(TComponent* Owner)  
   : TForm(Owner)
{
}

void __fastcall TForm1::LoadImage1Click(TObject *Sender)
{
   Image1->AutoSize=true;
   OpenPictureDialog1->Execute();
   if(!OpenPictureDialog1->FileName.IsEmpty())
   {
      Image1->Picture->LoadFromFile(OpenPictureDialog1->FileName);
   }
   W=Image1->Width;
   L=Image1->Height;
   image_width=W;
   image_length=L;
   Image1->Show();
   Image2->Hide();
   Image3->Hide();
}

void __fastcall TForm1::Exit1Click(TObject *Sender)
{
   Close();
}

void __fastcall TForm1::SaveImage1Click(TObject *Sender)
{
   SavePictureDialog1->Execute();
   if(SavePictureDialog1->FileName!="")
      Image1->Picture->SaveToFile(SavePictureDialog1->FileName);
}

void __fastcall TForm1::Preprocessing1Click(TObject *Sender)
{
   if(Image1->Width==105)
      Application->MessageBox("Please Load the Image First!","Error!",MB_OK);
   else
```c
{ Graphics::TBitmap *DestBitmap=new Graphics::TBitmap;

DestBitmap->Height=image_length;
DestBitmap->Width=image_width;
Image1->AutoSize=false;
Image1->Width=image_width;
Image1->Height=image_length;

original=(int **)malloc(image_length*sizeof(int *));
for(n1=0;n1<image_length;n1++)
  *(original+n1)=(int *)calloc(image_width,sizeof(int));

start=clock();
for(i=0;i<image_length;i++)
{
  for(j=0;j<image_width;j++)
  {
    PEL=Form1->Image1->Canvas->Pixels[j][i];
    B=(PEL&0x00FF0000)>>16;
    G=(PEL&0x0000FF00)>>8;
    R=(PEL&0x000000FF);
    original[i][j]=(int)(0.30*double(R)+0.59*double(G)+0.11*double(B));
  }
  ProgressBar1->Position=ProgressBar1->Position+1;
}

Bi=(int **)malloc(image_length*sizeof(int *));
for(n1=0;n1<image_length;n1++)
  *(Bi+n1)=(int *)calloc(image_width,sizeof(int));

original2=(int **)malloc(image_length*sizeof(int *));
for(n1=0;n1<image_length;n1++)
  *(original2+n1)=(int *)calloc(image_width,sizeof(int));

int l,w,length,width;
w=(int)(image_width/NN);
l=(int)(image_length/NN);
int sum1[N2],min[N2],p[N2],have[N2];
float r[N2];

count=0;
for(i=0;i<NN;i++)
{
  for(j=0;j<NN;j++)
  {
```
min[count]=255;
p[count]=0;
sum1[count]=0;
if(i==(NN-1)) {length=image_length;}
else {length=(i+1)*l; }

if(j==(NN-1)) {width=image_width;}
else {width=(j+1)*w;}

for(n1=i*1;n1<length;n1++)
for(n2=j*w;n2<width;n2++)
{
    sum1[count]+=original[n1][n2];
    if(original[n1][n2]<min[count]) {min[count]=original[n1][n2];}
    p[count]++;
}
count++;
ProgressBar1->Position=ProgressBar1->Position+1;
}

int k;
double tem,e;
//e=Edit5->Text.ToDouble();
e=2;
for(i=0;i<count;i++)
{
    have[i]=(int)(sum1[i]/p[i]);
    tem=(double)(have[i]-min[i]);
    r[i]=pow(tem,e)/255.0;
}

count=0;
float summ=0.0;
int h[N2][256]={0},temp,tk1,tk,ni,nii,niik,nik,ave[N2];

for(i=0;i<NN;i++)
{
    for(j=0;j<NN;j++)
    {
        sum1[count]=0;
        if(i==(NN-1)) {length=image_length;}
        else {length=(i+1)*l; }

        if(j==(NN-1)) {width=image_width;}
        else {width=(j+1)*w;}

}
for(n1=i*n1<n1<length;n1++)
for(n2=j*n2<n2<width;n2++)
{
    hh=original[n1][n2];
    tem=abs(hh-have[count]);
    if(hh>=have[count]) {hh=255;R=G=B=hh;}
    //else {hh=(int)((hh-min[count])*255.0/(have[count]-min[count]));R=G=B=hh;}
    else {hh=(int)(-pow(tem,e)/r[count]+255);R=G=B=hh;}
    original2[n1][n2]=hh;
    summ+==hh;
    h[count][hh]++;
    //DestBitmap->Canvas->Pixels[n2][n1]=RGB(B,G,R);
}
    count++;
    ProgressBar1->Position=ProgressBar1->Position+1;
}

FILE *fptr;
fptr=fopen("th.dat","w");
for(j=0;j<N2;j++)
{
    tk=0;
    tk1=100;
    do
    {
        tk=tk1;
        ni=0;
        nii=0;
        nik=0;
        niik=0;
        for(i=0;i<=tk;i++)
        {
            ni+=h[j][i];
            nii+=h[j][i]*i;
        }
        for(i=tk+1;i<256;i++)
        {
            nik+=h[j][i];
            niik+=h[j][i]*i;
        }
        if((ni!=0)&&(nik!=0)) tk1=(int)(0.5*(nii/ni+niik/nik));
    }while(tk1!=tk);
    ave[j]=tk;
fprintf(fp,"%d ",tk);
}
fclose(fp);

count=0;
for(i=0;i<NN;i++)
{
    for(j=0;j<NN;j++)
    {
        if(i==NN) {length=image_length;}
        else {length=(i+1)*l; }

        if(j==NN) {width=image_width;}
        else {width=(j+1)*w;}

        for(n1=i*l;n1<length;n1++)
        for(n2=j*w;n2<width;n2++)
        {
            hh=original2[n1][n2];
            if(hh>=ave[count]-10) {Bi[n1][n2]=0;R=G=B=255;}
            else {Bi[n1][n2]=1;R=G=B=0;}
            DestBitmap->Canvas->Pixels[n2][n1]=RGB(B,G,R);
            count++;
        }
    }
}

//Single level thresholding

/*for(i=0;i<image_length;i++)
{
    for(j=0;j<image_width;j++)
    {
        temp=original2[i][j];
        h[temp]++;
    }
}

tk=0;
tk1=100;
do
{
    tk=tk1;
ni=0;
nii=0;
nik=0;
niik=0;
for(i=0;i<=tk;i++)
{
    ni+=h[i];
    niik+=h[i]*i;
}
for(i=tk+1;i<256;i++)
{
    nik+=h[i];
    niik+=h[i]*i;
}
tk1=(int)(0.5*(ni/nİ+niik/nik));
}while(tk1!=tk);

Edit1->Text=tk;  /*

//Multilevel Thresholding
*/
int h1[256]={0},temp,tk1[5],tk[5],ni,nii,niik,nik;

ProgressBar1->Max=image_length;
for(i=0;i<image_length;i++)
{
    for(j=0;j<image_width;j++)
    {
        temp=original2[i][j];
        h1[temp]++;
    }
}
tk[0]=tk1[0]=0;
tk[1]=90;
tk1[1]=80;
tk[2]=150;
tk1[2]=140;
tk[3]=200;
tk1[3]=190;

for(j=1;j<4;j++)
{
do
{
tk[j]=tk1[j];
ni=0;
nii=0;
nik=0;
niik=0;
for(i=tk[j-1];i<=tk[j];i++)
{
    ni+=h1[i];
    nii+=h1[i]*i;
}
for(i=tk[j]+1;i<tk[j+1];i++)
{
    nik+=h1[i];
    niik+=h1[i]*i;
}
tk1[j]=(int)(0.5*(nii/ni+niik/nik));
}while(tk1[j]!=tk[j]);
}

Edit2->Text=tk[1];  /*

/*for(i=0;i<image_length;i++)
{
    for(j=0;j<image_width;j++)
    {
        //if(original2[i][j]>=have1-stt-10) {Bi[i][j]=0;R=G=B=255;}
        //if(original2[i][j]>=tk) {Bi[i][j]=0;R=G=B=255;}
        //else {Bi[i][j]=1;R=G=B=0;}
        DestBitmap->Canvas->Pixels[j][i]=RGB(B,G,R);
    }
    ProgressBar1->Position=ProgressBar1->Position+1;
}  */

end=clock();
tt=((end - start) / CLK_TCK);
Edit1->Text=tt;
Edit2->Text=image_width;
Edit3->Text=image_length;

Image1->Picture->Bitmap=DestBitmap;
delete DestBitmap;

ProgressBar1->Position=0;
void __fastcall TForm1::PageSegmentation2Click(TObject *Sender)
{
if(Image1->Width==105)
Application->MessageBox("Please Do the Preprocessing First!","Error!",MB_OK);
else
{
Graphics::TBitmap *DestBitmap=new Graphics::TBitmap;

DestBitmap->Height=(int)(Image1->Height);
DestBitmap->Width=(int)(Image1->Width);

ho=(int **)malloc(image_length*sizeof(int));
for(n1=0;n1<image_length;n1++)
    *(ho+n1)=(int *)calloc(image_width,sizeof(int));

ve=(int **)malloc(image_length*sizeof(int));
for(n1=0;n1<image_length;n1++)
    *(ve+n1)=(int *)calloc(image_width,sizeof(int));

and=(int **)malloc(image_length*sizeof(int));
for(n1=0;n1<image_length;n1++)
    *(and+n1)=(int *)calloc(image_width,sizeof(int));

final=(int **)malloc(image_length*sizeof(int));
for(n1=0;n1<image_length;n1++)
    *(final+n1)=(int *)calloc(image_width,sizeof(int));

int
int **type;
int temp;

type=(int **)malloc(image_length*sizeof(int));
for(n1=0;n1<image_length;n1++)
    *(type+n1)=(int *)calloc(image_width,sizeof(int));

for(i=0;i<image_length;i++)
    for(j=0;j<image_width;j++)
        type[i][j]=0;

ProgressBar1->Max=6*image_length;

//Horizontal Smearing
th=Image1->Width/5;
hrlsa(Bi,ho,th);

//Vertical Smearing
tv = Image1->Height/5;
vrlsa(Bi,ve,tv);

//Logical AND
for(i=0;i<image_length;i++)
{
    for(j=0;j<image_width;j++)
    {
        and[i][j]=ho[i][j]*ve[i][j];
    }
    ProgressBar1->Position= ProgressBar1->Position+1;
}

//Smoothing Smearing
ts=(ceil)(0.005*Image1->Width);  //For GRDVI,UTO2VI,ts=0
hrlsa(and,final,ts);

for (int i=0;i<(int)(Image1->Width) ; i++)
{
    for (int j=0;j<(int)(Image1->Height); j++)
    {
        if (final[j][i]==0)
            PEL = 255;
        else
            PEL = 0;

        DestBitmap->Canvas->Pixels[i][j]=RGB(PEL,PEL,PEL);
    }
    ProgressBar1->Position= ProgressBar1->Position+1;
}

} Image1->Show();
Image1->Picture->Bitmap=DestBitmap;

delete DestBitmap;

//Feature Extraction and Classification
for(n1=0;n1<1000;n1++)
    { xmin[n1]=image_width;
      ymin[n1]=image_length; }

count=1;
for (i=0; i<image_length; i++)
{
    for (j=0; j<image_width; j++)
    {
        if (final[i][j]==1)
        {
            for (n1=-1; n1<2; n1++)
            {
                for (n2=-1; n2<2; n2++)
                {
                    if ((n1!=0) && (n2!=0))
                    {
                    }
                }
            }
        }
    }
}

if ((i+1) >= 0) && ((i+1) < image_length) && ((j+1) >= 0) && ((j+1) < image_width)
{
    if (final[i+n1][j+n2]==1)
    {
        if (type[i+n1][j+n2]!=0)
        {
            if (type[i][j]==0)
            {
                type[i][j]=type[i+n1][j+n2];
            }
            else
            {
                if (type[i+n1][j+n2]<type[i][j])
                {
                    type[i][j]=type[i+n1][j+n2];
                }
            }
        }
    }
}

}

if (type[i][j]==0) {type[i][j]=count; count++;}

ProgressBar1->Position=ProgressBar1->Position+1;

}

for (i=image_length-1; i>=0; i--)
{
    for (j=0; j<image_width; j++)
    {
        if (final[i][j]==1)
        {
            hh=type[i][j];
            for (n1=-1; n1<2; n1++)
{ 
    for(n2=-1;n2<2;n2++)
    { 
        if((n1!=0)|(n2!=0))
        {

            if(((n1+i)>=0)&&(n1+i)<image_length)&&(n2+j)>=0)&&(n2+j)<image_width)
            {
                final[i+n1][j+n2]=1
                {
                    if(type[i+n1][j+n2]<hh)
                    {
                        type[i][j]=type[i+n1][j+n2];
                        hh=type[i][j];
                    }
                }
            }
    }
    ProgressBar1->Position=ProgressBar1->Position+1;
}

for(i=0;i<image_length;i++)
{ 
    for(j=image_width-1;j>=0;j--)
    { 
        if(final[i][j]==1)
        {
            hh=type[i][j];
            for(n1=-1;n1<2;n1++)
            {
                for(n2=-1;n2<2;n2++)
                { 
                    if((n1!=0)|(n2!=0))
                    {

                        if(((n1+i)>=0)&&(n1+i)<image_length)&&(n2+j)>=0)&&(n2+j)<image_width)
                        {
                            final[i+n1][j+n2]=1
                            {
                                if(type[i+n1][j+n2]<hh)
                                {
                                    type[i][j]=type[i+n1][j+n2];
                                }
                            }
                        }
                    }
                }
            }
        }
    }
}
hh=type[i][j];

}

}

}

}

ProgressBar1->Position=ProgressBar1->Position+1;

}

for(i=0;i<image_length;i++)
{
    for(j=0;j<image_width;j++)
    {
        if(final[i][j]==1)
        {
            temp=type[i][j];
            if(i>ymin[temp]) ymin[temp]=i;
            if(i<ymin[temp]) ymin[temp]=i;
            if(j>xmax[temp]) xmin[temp]=j;
            if(j<xmin[temp]) xmin[temp]=j;
        }
    }
    ProgressBar1->Position=ProgressBar1->Position+1;
}

//int tt;
FILE * File1;
File1 = fopen("parameters.txt","w");
//int AVE[1000],STD[1000]={0},sum,count1;

tt=(ceil)(0.022*image_length);
for(n1=1;n1<count;n1++)
{
    deltax[n1]=xmax[n1]-xmin[n1]+1;
    H[n1]=ymax[n1]-ymin[n1]+1;
    Area[n1]=deltax[n1]*H[n1];
    if(ymin[n1]<Image1->Height-1&&ymin[n1]>0&&deltax[n1]>3&&H[n1]>10))
    {

fprintf(File1,"%d %d %d %d %d \n",
ymax[n1],ymin[n1],xmax[n1],xmin[n1],H[n1]);
}
}
fclose(File1);

ProgressBar1->Position=0;
free(ho);
free(ve);
free(and);
free(final);
free(type);
free(original);
}
}

//**************************************************************************
void __fastcall TForm1::TextExtraction2Click(TObject *Sender)
{
    Graphics::TBitmap *DestBitmap=new Graphics::TBitmap;
    DestBitmap -> Height = Image1 -> Height;
    DestBitmap -> Width = Image1 -> Width;
    //Image1->AutoSize=false;
    //Image1->Height = DestBitmap-> Height;
    //Image1->Width = DestBitmap-> Width;

    int PELz;
    FILE *File1;
    File1 = fopen("parameters.txt","r");

    for(int k=1; k<count; k++)
    {
        fscanf (File1,"%d %d %d %d %d \n",
        &ymax[k],&ymin[k],&xmax[k],&xmin[k],&H[k]);
    }

    for(int k=1; k<count; k++)
    {
        //if((Ru[k]<par3) && (H[k]<par1) && (!(AngleX[k]<1.0 && & E[k]>=50 )))
        if((H[k]<Image1->Height/8))
        {
            for(int x=xmin[k]; x<=xmax[k]; x++)
            for(int y=ymin[k]; y<=ymax[k]; y++)
            {
                //T_Buffer[x][y] = Buffer11[x][y];
            }
        }
    }
}
if (Bi[y][x] == 1)
    PELz = 0;
else
    PELz = 255;

//PELz=T_Buffer[x][y];
DestBitmap->Canvas->Pixels[x][y]=RGB(PELz,PELz,PELz);

ProgressBar1->Max=2*count;

ProgressBar1->Position=0;
Image1->Picture->Bitmap=DestBitmap;
delete DestBitmap;
fclose(File1);
//free(original2);

void hrlsa(int **Bi, int **ho, int th)
{
    for(i=0;i<image_length;i++)
    {
        for(j=0;j<image_width;j++)
        {
            ho[i][j]=Bi[i][j];
        }
    }

    for(i=0;i<image_length;i++)
    {
        count=0;
        for(j=0;j<image_width;j++)
        {
            if(Bi[i][j]==0) {count++;}
            else
            {
                if((count<=th)&&(count>0))
                {
                    for(n1=0;n1<count;n1++)
                    {
                        ho[i][j-n1-1]=1;
                    }
                }
            }
        }
    }
}
count=0;

}
}
void vrlsa(int **Bi, int **ve, int tv)
{
for(i=0;i<image_length;i++)
{
for(j=0;j<image_width;j++)
{
    ve[i][j]=Bi[i][j];
}
}
for(j=0;j<image_width;j++)
{
    count=0;
    for(i=0;i<image_length;i++)
    {
        if(Bi[i][j]==0) {count++;}
    else
    {
        if((count<=th)&&(count>0))
        {
            for(n1=0;n1<count;n1++)
            {
                ve[i-n1-1][j]=1;
            }
            count=0;
        }
    }
}
}
<table>
<thead>
<tr>
<th>NAME:</th>
<th>YI LIN</th>
</tr>
</thead>
<tbody>
<tr>
<td>PLACE OF BIRTH</td>
<td>SHENYANG, CHINA</td>
</tr>
<tr>
<td>YEAR OF BIRTH</td>
<td>1974</td>
</tr>
<tr>
<td>EDUCATION</td>
<td>XUEJUN HIGH SCHOOL, HANGZHOU, CHINA</td>
</tr>
<tr>
<td></td>
<td>1986-1992</td>
</tr>
<tr>
<td></td>
<td>ZHEJIANG UNIVERSITY, HANGZHOU, CHINA</td>
</tr>
<tr>
<td></td>
<td>1992-1996 B.ENG</td>
</tr>
<tr>
<td></td>
<td>UNIVERSITY OF WINDSOR</td>
</tr>
<tr>
<td></td>
<td>2001-2003 M.A.Sc</td>
</tr>
</tbody>
</table>