University of Windsor

Scholarship at UWindsor

Electronic Theses and Dissertations

Theses, Dissertations, and Major Papers

2014

Optical Character Recognition of Printed Persian/Arabic Documents

Mahnaz Shafii
University of Windsor

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

Recommended Citation

Shafii, Mahnaz, "Optical Character Recognition of Printed Persian/Arabic Documents" (2014). *Electronic Theses and Dissertations*. 5179.
https://scholar.uwindsor.ca/etd/5179

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.

Optical Character Recognition of Printed Persian/Arabic Documents

By

Mahnaz Shafii

A Dissertation
Submitted to the Faculty of Graduate Studies
through the department of Electrical and Computer Engineering
in Partial Fulfillment of the Requirements for
the Degree of Doctor of Philosophy
at the University of Windsor

Windsor, Ontario, Canada

2014

© 2014 Mahnaz Shafii

Optical Character Recognition of Printed Persian/Arabic Documents

by

Mahnaz Shafii

APPROVED BY:

Dr. M. O. Ahmad (External Examiner)
Concordia University

Dr. B. Boufama
School of Computer Science

Dr. J. Wu
Department of Electrical and Computer Engineering

Dr. H. Wu
Department of Electrical and Computer Engineering

Dr. M. Sid-Ahmed, Advisor
Department of Electrical and Computer Engineering

September 18, 2014

DECLARATION OF ORIGINALITY

I hereby certify that I am the sole author of this dissertation and that no part of this dissertation has been published or submitted for publication.

I certify that, to the best of my knowledge, my dissertation does not infringe upon anyone's copyright nor violate any proprietary rights and that any ideas, techniques, quotations, or any other material from the work of other people included in my dissertation, published or otherwise, are fully acknowledged in accordance with the standard referencing practices. Furthermore, to the extent that I have included copyrighted material that surpasses the bounds of fair dealing within the meaning of the Canada Copyright Act, I certify that I have obtained a written permission from the copyright owner(s) to include such material(s) in my dissertation and have included copies of such copyright clearances to my appendix.

I declare that this is a true copy of my dissertation, including any final revisions, as approved by my dissertation committee and the Graduate Studies office, and that this dissertation has not been submitted for a higher degree to any other University or Institution.

ABSTRACT

Texts are an important representation of language. Due to the volume of texts generated and the historical value of some documents, it is imperative to use computers to read generated texts, and make them editable and searchable. This task, however, is not trivial. Recreating human perception capabilities in artificial systems like documents is one of the major goals of pattern recognition research. After decades of research and improvements in computing capabilities, humans' ability to read typed or handwritten text is hardly matched by machine intelligence. Although, classical applications of Optical Character Recognition (OCR) like reading machine-printed addresses in a mail sorting machine is considered solved, more complex scripts or handwritten texts push the limits of the existing technology. Moreover, many of the existing OCR systems are language dependent. Therefore, improvements in OCR technologies have been uneven across different languages. Especially, for Persian, there has been limited research. Despite the need to process many Persian historical documents or use of OCR in variety of applications, few Persian OCR systems work with good recognition rate.

Consequently, the task of automatically reading Persian typed documents with close-to-human performance is still an open problem and the main focus of this dissertation.

In this dissertation, after a literature survey of the existing technology, we propose new techniques in the two important preprocessing steps in any OCR system: Skew detection and Page segmentation. Then, rather than the usual practice of character segmentation, we propose segmentation of Persian documents into sub-words. The choice of sub-word segmentation is to avoid the challenges of segmenting highly cursive Persian texts to

isolated characters. For feature extraction, we will propose a hybrid scheme between three commonly used methods and finally use a nonparametric classification method.

A large number of papers and patents advertise recognition rates near 100%. Such claims give the impression that automation problems seem to have been solved. Although OCR is widely used, its accuracy today is still far from a child's reading skills. Failure of some real applications show that performance problems still exist on composite and degraded documents and that there is still room for progress.

DEDICATION

I dedicate this dissertation to my family, especially…

> to my husband for his patience and understanding;
>
> to Dad and Mom for teaching me the importance of hard work and higher education;
>
> and finally to my lovely son, who inspired me to achieve a big milestone in my life while being a mom!

ACKNOWLEDGEMENTS

First and foremost I want to thank my advisor Dr. Maher Sid-Ahmed. It has been an honor to be his last Ph.D. student. He has taught me, both consciously and unconsciously, to become an independent researcher. I appreciate all his contributions of time, ideas, and funding to make my Ph.D. experience productive and stimulating. The joy and enthusiasm he has for his research was contagious and motivational for me, even during tough times in the Ph.D. pursuit.

For this dissertation, I would like to thank my reading committee members: Professors J. Wu, H. Wu, B. Boufama and the chair of defense professor H. Hu. I would also like to thank my external examiner professor Omair Ahmad from Concordia University.

I am also very grateful to Professor Majid Ahmadi for his guidance and support at the beginning of my Ph.D. work. My appreciation extends to Andria Ballo from the Department of Electrical Engineering for her support, reminders, and advice. She truly helped me to minimize frequency of my long commute to school for 4 years.

I gratefully acknowledge the funding sources that made my Ph.D. work possible. I was supported by research and teaching assistantships by the University of Windsor and NSERC funding.

Lastly, I would like to thank my family for all their love and encouragement. For my parents, who raised me with a love of science and supported me in all my pursuits. For my sisters’ and my brother’s encouragements and supports during tough times. And most of all for my loving, supportive, encouraging, and patient husband Amir whose selfless support during my Ph.D. study is so appreciated. Thank you!

TABLE OF CONTENTS TABLE OF CONTENTS

DECLARATION OF ORIGINALITY iii

ABSTRACT iv

ACKNOWLEDGEMENTS vii

LIST OF TABLES xi

LIST OF APPENDICES xv

CHAPTER 1 BACKGROUND ON TEXT RECOGNITION 1

1.1. Introduction 1

1.2. OCR Classifications 1

    1.2.1. Online text recognition 2

        1.2.1.1. Handwriting recognition development 3

    1.2.2. Offline text recognition 4

        1.2.2.1. Evolution of offline text recognition 5

1.3. OCR Systems Classical Flow 6

    1.3.1. Preprocessing Phase 6

    1.3.2. Recognition Phase 11

CHAPTER 2 SKEW DETECTION AND CORRECTION OF SCANNED DOCUMENTS 15

2.1. Introduction 15

2.2. Skew Angle Detection Methods 15

    2.2.1. Projection Profile 16

    2.2.2. Hough Transform [HT] 18

    2.2.3. Nearest Neighbor [N-N] 22

    2.2.4. Fourier Transform 25

2.3. Skew correction mechanism 29
2.4. Skew Detection based on an axes-parallel bounding rectangle 29
  2.4.1. The proposed algorithm and its implementation 30
2.5. Our method compared to existing methods 34
  2.5.1. Typical Samples 34
  2.5.2. Extreme Samples 35
  2.5.1. Experimental Results 38
2.6. Conclusions 41
CHAPTER 3 PAGE SEGMENTATION 42
3.1. Introduction 42
3.2. Introduction 42
  3.2.1. Top-Down Analysis Methods 43
  3.2.2. Bottom-Up Analysis Methods 46
3.3. Page Segmentation Using Resampling Technique 48
  3.3.1. Image Resampling 48
  3.3.2. Sampling Rate 50
  3.3.3. Determination of Sampling Rate for Page Segmentation 51
  3.3.4. Work Flow of Our Resampling Method 52
3.4. Validations and Examples 53
3.5. Text/Non-text Classification 56
  3.5.1. Proposed method for text/non-text classification 57
3.6. Conclusions 59
CHAPTER 4 PRINTED PERSIAN/ARABIC TEXT RECOGNITION 60
4.1. Introduction 60

4.2. Characteristics of Persian Texts 60
4.3. Segmentation of Persian Scripts 61
    4.3.1. Proposed Algorithm 63
    4.3.2. Label Settings 65
    4.3.3. Experimental Results 66
4.4. Algorithm for Segmentation of a Text Page 67
4.5. Feature extraction....................70
4.6. Classification....................71
4.7. Hybrid labeling decision 72
4.8. Conclusions....................73
CHAPTER 5 CONCLUSION AND FUTURE WORK 74
5.1. Overview of the Dissertation and Conclusions....................74
5.2. Future Work 76
REFERENCES/BIBLIOGRAPHY 77
Appendix A. Projection Profile skew detection MATLAB source code 86
Appendix B. Hough transform skew detection MATLAB source code 87
Appendix C. Nearest-Neighbor skew detection MATLAB source code 88
Appendix D. Fourier method skew detection MATLAB source code 89
Appendix E. Our skew detection method MATLAB source code....................91
Appendix F. Our Page Segmentation MATLAB source code....................96
VITA AUCTORIS 105

LIST OF TABLES

Table 1: Comparison of commonly used OCR's feature extraction methods 13

Table 2: Success rate and average skew correction time 35

Table 3: Skew detection success rate comparison 37

Table 4: Error rate and processing time for five page segmentation algorithms [75] 56

LIST OF FIGURES

Figure 1: Optical Character Recognition Classifications 2

Figure 2: Offline text containing just special information (left), online text containing temporal sequence of points traced out by the pen (right) 2

Figure 3: Timeline of handwriting recognition 4

Figure 4: Offline Optical Character Recognition Classic process flow 6

Figure 5: Hierarchy of Offline segmentation step of Optical Character Recognition 9

Figure 6: A skewed text line 15

Figure 7: Histograms of a skewed document (top) and a non-skewed document (bottom) 16

Figure 8: A skewed image of a page (left), Hough Transform lines with the highest accumulator values (right) 19

Figure 9: Hough Parameter space for the skewed page in Figure 8 20

Figure 10: A skewed image of a page (left), HT lines with highest accumulator values (right) 21

Figure 11: A skewed image of a page (left), connected components used for N-N skew detection method (right) 23

Figure 12: Histogram of the image in Figure 11 23

Figure 13: A skewed image of a page (left), connected components used for N-N skew detection method (middle), incorrect skew correction (right) 24

Figure 14: Histogram of the image in Figure 13 25

Figure 15: A skewed image of a page (left), original image sliced into 4 equal blocks (right) 26

Figure 16: Fourier spectrum of the 4 blocks in Figure 15 27

Figure 17: Histogram of the integrals of normalized value of the peaks in the Fourier spectrum (Left), corrected skew of the image in Figure 15 using Fourier transform 27

Figure 18: A skewed image of a Japanese text (top-left), outcome of the skew correction algorithm (top-right), Fourier spectrum of one of the image blocks (bottom-left), and histogram of the normalized value of the peaks in the Fourier spectrum (bottom-right) . 28

Figure 19: Left: An angled rectangle embedded in an axes-parallel rectangle. Right: A rectangle at zero angle 30

Figure 20: Corner pixels detection of a text.. 31

Figure 21: Comparison of success rate and average skew correction time 35

Figure 22: Sample images in the extreme samples category 36

Figure 23: Skew correction of a page of an English technical paper................................ 38

Figure 24: Skew correction of a page of an English book with graphical images........... 38

Figure 25: Skew correction of a page of an English book with a table at 209°................ 39

Figure 26: Skew correction of a Japanese text.. 39

Figure 27: Skew correction of a Japanese text.. 40

Figure 28: Skew correction of an English text.. 41

Figure 29: Illustration of image upsampling and required interpolation for missing information... 49

Figure 30: Illustration of image degrading in a sequence of downsampling-upsampling 50

Figure 31: Segmentation of an image with different sampling rates 50

Figure 32: White space transition peaks of a vertical scan of an image 51

Figure 33: Workflow of our resampling algorithm.. 52

Figure 34: Examples of complex page segmentation using our algorithm....................... 55

Figure 35: Examples of complex page segmentation and labeling.................................. 58

Figure 36: Our proposed algorithm for segmentation of a word to its sub-words........... 63

Figure 37: An example of labeling change when two components are the same 66

Figure 38: Examples of Persian word segmentation to its sub-words............................. 66

Figure 39: Examples of overlapping sub-words 67

Figure 40: Our proposed algorithm for segmentation of a text page 67

Figure 41: Initial labeling of a sample Persian text 68

Figure 42: An example for sorting sub-words based on their maximum row values 68

Figure 43: Examples for δ calculation for different components 69

Figure 44: An example for sorting sub-words based on their maximum column values . 69

Figure 45: Extracted feature vectors of labeled sub-words training images.................... 70

Figure 46: Extracted feature vectors of connected components of the document image . 71

Figure 47: A selection of our library of sub-words used in evaluation of feature extraction methods. Highlighted yellow cells indicate incorrect classification for each feature extraction method... 73

LIST OF APPENDICES

Appendix A: Projection Profile skew detection MATLAB source code 86

Appendix B: Hough transform skew detection MATLAB source code 87

Appendix C: Nearest-Neighbor skew detection MATLAB source code 88

Appendix D: Fourier method skew detection MATLAB source code 89

Appendix E: Our skew detection method MATLAB source code 91

Appendix F: Our Page Segmentation MATLAB source code 96

CHAPTER 1

BACKGROUND ON TEXT RECOGNITION

1.1. Introduction

Optical Character Recognition (OCR) is a field of research in artificial intelligence, pattern recognition, and computer vision. OCR is a common method of digitizing pictures of printed or handwritten texts so that they can be electronically edited, searched, and stored more compactly and efficiently. Despite a century long research and development in this field, machines are still nowhere near human's reading capabilities. The goal of an OCR system is recognition of text (same as humans) in a complex document.

In this chapter, we look into OCR classifications and their classic process flow. For each step we give a brief overview of the historical background and common methods used.

1.2. OCR Classifications

OCR systems are mainly classified into online text recognition and offline text recognition. Subsequently, offline OCR is classified into two subcategories of handwritten and typed text recognition (Figure 1).

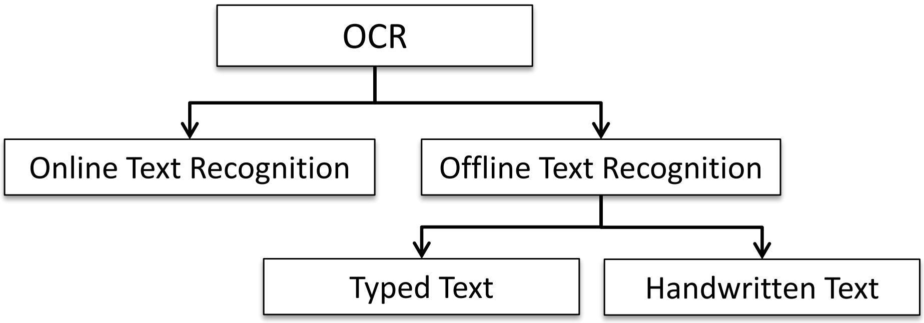


Figure 1: Optical Character Recognition Classifications

1.2.1. Online text recognition

In online text recognition, characters and words are recognized real time as soon as they are written, and therefore, contain temporal information. Online systems obtain the position of the pen as a function of time directly from the interface. This is usually done through pen-based interfaces where the writer writes with a special pen on an electronic tablet.

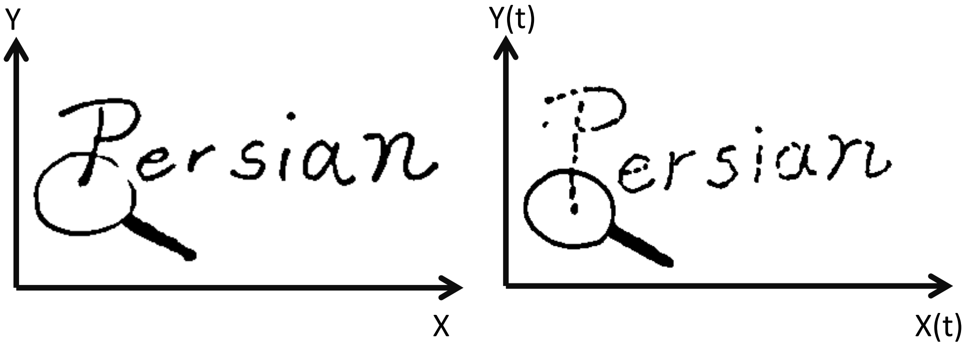


Figure 2: Offline text containing just special information (left), online text containing temporal sequence of points traced out by the pen (right)

Dynamic information, which are usually available for online text recognitions, are number of strokes, order of strokes, direction for each stroke, and speed of writing within each stroke. This valuable information assists in recognition of documents and frequently leads to better performing systems compared to offline recognition. Some of applications of online optical recognition are in PDAs, smart phones, and Tablet computers. The advantage of online recognition system over offline systems is interactivity, adaptation of writer to digitizer (or vice versa), less prone noise, and available temporal information. The disadvantage is that the whole document is not available for processing; and therefore, information needs to be processed dynamically.

1.2.1.1. Handwriting recognition development

Figure 3 shows major milestones in the development of handwriting recognition systems. The effort dates back to 1915 when Goldberg filed the first patent in handwriting recognition. Today, after a century of research and commercial developments, reliable systems exist in online text recognition. Today, handwritten texts on a screen of an iPAD can be quickly digitized and stored efficiently. One caveat is that handwriting recognition in non-Roman text suffers a lag in technology and challenges that are language specific. Therefore, there is a lot of room for research and development in improving OCR for non-Roman text.

1915	U.S. Patent on handwriting recognition user interface with a stylus by Hyman Goldberg
1938	U.S. Patent on machine recognition of handwriting by George Hansel
1957	Tom Dimond invents the first handwriting recognition tool (Stylator Tablet)
1961	RAND Tablet invention: the first two-dimensional Tablet (more advanced than Stylator)
1969	GRAIL system: handwriting recognition with electronic ink , gesture commands
1973	Newman and Sproull developed “The Ledeen Character Recognizer"
1980	First retail handwriting recognition systems: Pencept, CIC, Inforite point-of-scale terminal
1989	Portable handwriting recognition computer by GRiD systems
1997	Address interpretation system deployed by United States Postal Service
2001	Windows XP Tablet PC with handwriting recognizer
2007	CEDAR-FOX: a commercial Forensic comparison of handwriting

Figure 3: Timeline of handwriting recognition

1.2.2. Offline text recognition

An Offline text recognition system processes a static representation of a document. Offline text recognition is divided to two subcategories of typed and handwritten documents. In both subcategories, an image of the document obtained from a scanner or camera is processed. Obviously, due to the variety of handwriting styles and non-standard nature of handwritings, the problem of offline handwriting recognition is the most challenging problem in OCR and it usually requires language specific methods. On the other hand, OCR of typed documents are very much in demand for practical applications such as historical document analysis, official letter and document processing, and vehicle plate recognition.

Optical character recognition of typed document for English has become one of the most successful applications of technology in pattern recognition and artificial intelligence.

Most of the active research in this field is to deal with very complex documents, noisy and skewed documents, as well as improving recognition rates for non-Roman based texts.

1.2.2.1. Evolution of offline text recognition

Research in OCR of documents started at the beginning of the 1960s with the development of the digital computers. It is the first time OCR was considered as a viable data processing application with a wide commercial use. The first generation machines used special letter shapes which the OCRs could read. These shapes were specially designed for machine reading, and they appeared like symbols. The first commercialized OCR of this generation was from IBM, which could read a special IBM font. This OCR worked based on template matching, which compared the character image with a library of prototype images for each character of each font. In early 1970s, OCR systems were able to recognize regular typed and handwritten characters with limited numerals and a few letters and symbols. The first automatic letter-sorting machine for postal code numbers from Toshiba was developed during this period. The early OCR systems were mostly based on the structural analysis approach or standardization. In late 1970, an American standard OCR character set OCR-A font was defined, which was designed to facilitate optical recognition while still readable to humans. In 1980s, OCR systems faced the challenge of improving recognition rate for documents of poor quality and large printed and handwritten character sets. Today, a reliable OCR system should be able to perform accurately on complex documents intermixing with text, graphics, tables and mathematical symbols, color documents, low-quality noisy and skewed documents, etc.

1.3. OCR Systems Classical Flow

The input of an OCR system is an image. The image can be from a scanner, a camera, or simply a print-screen of a page. The image may contain not just the text but pictures, equations, tables, etc. Therefore, the first step of any OCR system is preprocessing.

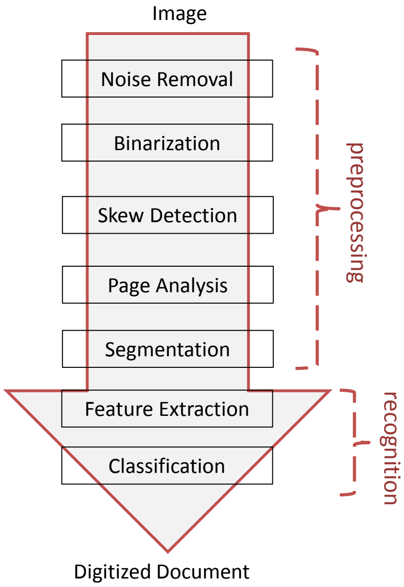


Figure 4: Offline Optical Character Recognition Classic process flow

The goal of preprocessing is to extract the text and convert the raw image of the text to segmented components. Then, in the recognition phase, the features of these segmented components are extracted and fed into a classification module. Finally, the outcome of the algorithm is a machine editable text (Figure 4).

1.3.1. Preprocessing Phase

Typical preprocessing includes the following steps, not necessarily in this order:

- Noise removal

Documents may contain noise for many reasons. One type of noise is the marginal noise which appears as a large dark region around the document image. Another type of noise is background noise which appears from uneven contrast, background spots, printer and scanner malfunctions. Finally, page rule lines are another source of noise interfering with text objects. There are several methods in dealing with each noise type.

For page rule lines, mathematical morphology based methods trace line like structures as candidate for rule lines [1]. Hough transform [2] can also be used to find imperfect instances of objects with certain class of shapes using a voting procedure. Finally, projection profile based methods work by creating a horizontal histogram in which the hills of the histogram are the center locations of the horizontal rule lines [3].

For marginal noise, there are two groups of methods. One group aims at identifying noise components. In this method, the noise patterns in an image are searched by extracting noise features [4]. For example Peerawit's method [5] uses Sobel edge detection and identifies noises to be removed by comparing the edge density of marginal noise and text. Another group for marginal noise removal is by Identifying Text Components [6].

For background noise, many techniques have been introduced. One common method is to use a low-pass filter to remove as much of the noise as possible while retaining the entire foreground pixels. More advanced methods are Binarization and Thresholding Based Methods [7], Fuzzy Logic Based Methods [8], Histogram Based Methods [9], and morphology based methods [10].

○ Binarization

Binarization is the process of converting gray scale images to binary images. A large number of methods have been proposed for binarization. These methods generally fall into two main approaches. One approach is based on global thresholding and the other is based on local thresholding.

In global thresholding, based on statistical attributions of a document, a single value is used as threshold between background and foreground pixels. Otsu method [11] is one of the most commonly used global binarization techniques. The main drawback of this method is that it cannot adapt well to noise and illuminations. A recent work by Lazzara [12] focuses on Sauvola binarization method. This method performs relatively well on classical documents, however, three main defects remain: the window parameter of Sauvola's formula does not fit automatically to the contents, it is not robust to low contrasts, is not invariant with respect to contrast inversion.

In local thresholding, the threshold value is varied based on the local content of the image. Commonly used Niblack binarization method [13] is based on local thresholding.

- Skew detection and correction

This step is an important preprocessing step discussed in chapter 2. After overview of common skew detection methods in chapter 2.2, a new technique in skew detection and correction of documents is introduced.

- Page Analysis

Page Analysis is the process of identifying and categorizing regions of interest in the image of a document. In this step, the text regions are labelled and then fed into the OCR flow. In chapter 3, we focus on this important preprocessing step and propose a new method in segmenting different regions of a document.

- Segmentation

This step is another important preprocessing step. Segmentation performance of an OCR system directly affects the recognition performance, as the output of the segmentation step is directly fed into the recognition engine. Figure 5 depicts hierarchy of offline segmentation step.

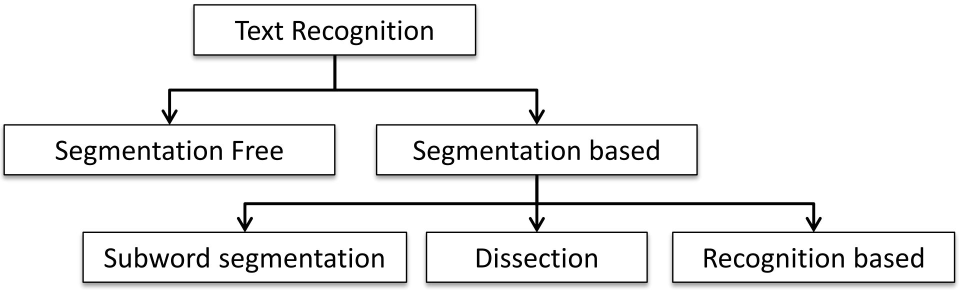


Figure 5: Hierarchy of Offline segmentation step of Optical Character Recognition

Text segmentation of Persian text is not a trivial step as characters could overlap, slant, and have different styles and fonts. Some researches skip the segmentation step entirely and instead take a holistic approach. The idea is to recognize whole words against a dictionary. The obvious problem of such an approach is the number of classes present in the recognition phase. In Persian there are 114 contextual forms of its 32 alphabets. Characters in the Persian language can have different shapes depending on its position within a word. A character that is used in the beginning of a word will have a different appearance than one that is in the middle or end of a word and these might be different in appearance than stand-alone characters. A technique that segments a Persian word into characters and then uses a classifier trained on all 114 shapes can potentially recognize any text.

On the other hand a holistic classifier needs to learn as many classes as the number of dictionary words, names of individuals and countries. Therefore in holistic approach, training of a classifier for many classes is one of the major issues. Based on this limitation, segmentation based approach is more practical than the holistic approach for real world problems. Many techniques have been developed for holistic approaches.

Benouareth [14] used holistic method for Arabic word recognition. To build a feature vector sequence, two segmentation schemes are incorporated to divide a word into frames. The first one is uniform segmentation, which vertically divides a word into equal sized frames. The second one is non-uniform segmentation, which has variable frame size. After segmenting word into frames, statistical and structural features are extracted by capturing ascenders, descenders, concavity, dots, and stroke direction. Vinciarelli [15] used similar technique in which prior to the information retrieval, the individual words on the handwritten document need to be recognized correctly. Using a fixed size sliding window, density feature is extracted for HMM to perform recognition by calculating the likelihood of a word against dictionary. To reduce the computational cost of holistic approach, Mozaffari [16] proposed a lexicon reduction scheme for offline Farsi handwriting recognition by analyzing dots within characters.

Segmentation based approaches, can be performed by either the dissection or recognition-based technique. Dissection is decomposition of the image into a sequence of sub-images using general features. Projection analysis, connected component processing, and white space are some of the common dissection techniques used by OCR systems [17]. These techniques are suitable for scripts which have spaces between characters.

The basic principle of recognition-based character segmentation is to use a mobile window of variable width to provide the tentative segmentations. Characters are by-products of the character recognition for systems using such a principle to perform character separation. The main advantage of this technique is that it bypasses serious character separation problems.

In some language scripts, like Persian, segmenting words to characters is a very difficult task as characters overlap. For this reason, in chapter 4 in segmenting Persian texts, we neither segment words to their characters nor skip the segmentation step entirely. Instead, we try to get the best of each method by segmenting text to its sub-words, a much easier process with a much better success rate. On the other hand, rather than dealing with a huge class size, we deal with a more manageable database of sub-words. The largest Persian sub-words dictionary reported in literature [18] contains 7317 subwords, which is just a small fraction of more than a million words in Persian language.

1.3.2. Recognition Phase

In this phase, segmented text is fed into a feature extraction algorithm and finally to classification module.

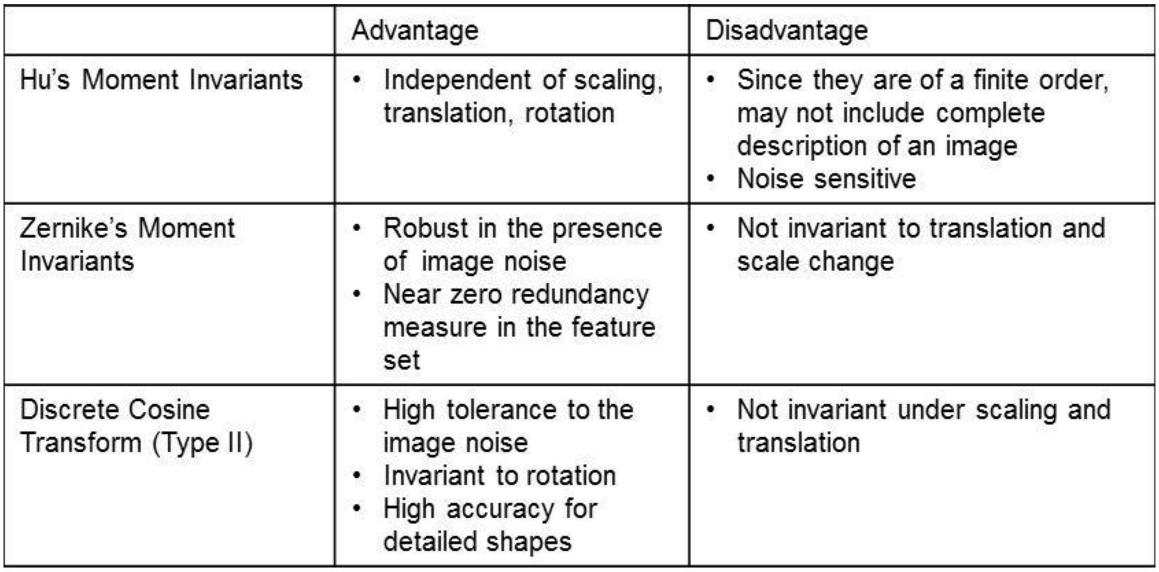
- Feature extraction

Feature extraction is to find a set of features that define the shape of the underlying character as precisely and uniquely as possible. Selection of feature extraction method is probably the most important factor in achieving high performance in recognition. Feature extraction methods are very much application specific and there is no universally accepted set of feature vectors in document image recognition. Some of the available methods in feature extraction include image invariant, projection histograms, zoning, and n-tuples. Image invariant features are popular choice in many OCR systems. Image invariant methods can be categorized to boundary-based and region-based methods.

A classical boundary-based method is Discrete Fourier Transform. In this method, Fourier transform is used to analyze a closed planar curve. Several variations of Fourier Transform for feature extraction have been introduced. For examples, Zahn [19] applies the Fourier transform to the sequence of angular differences between line segments in the curve.

In chapter 4, we apply the Discrete Fourier Transform (DCT) to segmented sub-words coordinate components. Transformation of images to Fourier domain provides important information about their structure. In the Fourier domain, there are low and high frequency components. High frequency components denote the fine details of a shape; whereas, low frequency components denote to basic shape structure. For feature extraction of characters, usually limited number of lowest frequency components in the image spectrum is considered and higher frequency components are discarded, as they mostly represent image noise.

In region-based methods, moments of different points present in a character are used as a feature. Hu [20] introduced the concept of classical moment invariant in 1962. Hu's other moments are statistical measure of the pixel distribution about the center of gravity of the character. In 1982, Teh [21] defined a set of moments called Zernike based on theory of orthogonal polynomials. Zernike moments have been used successfully in many OCR systems as well. The advantage and disadvantage of each of these methods is tabulated in Table 1.



	Advantage	Disadvantage
Hu's Moment Invariants	• Independent of scaling, translation, rotation	• Since they are of a finite order, may not include complete description of an image • Noise sensitive
Zernike's Moment Invariants	• Robust in the presence of image noise • Near zero redundancy measure in the feature set	• Not invariant to translation and scale change
Discrete Cosine Transform (Type II)	• High tolerance to the image noise • Invariant to rotation • High accuracy for detailed shapes	• Not invariant under scaling and translation

Table 1: Comparison of commonly used OCR's feature extraction methods

In chapter 4, we introduce a hybrid approach based on Hu, Zernike, and Discrete Fourier Transform for feature extraction of Persian sub-words.

- <u>Classification</u>

In classification stage, based on the features extracted and relationships among the features, an OCR process assigns labels to character images. This step is the final stage of the OCR system in which characters or sub-words are recognized and are output to machine editable form. Classification methods can be divided into two categories: learning-based and non-parametric classifiers. The learning-based classifiers require an intensive learning phase of the classifier parameters. Neural network [22, 23], Support Vector Machine (SVM) [24], Boosting [25], and decision tree [26] are examples of commonly used classifiers in this category.

Decision tree is used to create an expansive classifier ensemble. Different types of decision trees are discussed in [27]. Since in decision tree each node asks a question on only one feature, the learning phase is quicker compared to other methods in this category.

SVM is a binary classifier with discriminant function being the weighted combination of kernel functions over all training samples. After learning by quadratic programming, the samples of nonzero weights are called support vectors. For multi-class classification, binary SVMs are combined in either one-against others or one-against-one scheme [24]. Due to the high complexity of training and execution, this method is not suitable to classify a big set of data.

Learning-based classifiers, such as Artificial Neural Networks, usually offer more accuracy in performance; however, they impose a high computational cost for OCR systems. Nonparametric classifiers base their classification decision solely on the data, and require no learning of parameters. The most commonly used non-parametric method is Nearest-Neighbor (N-N) method [28], which classifies an image by the class of its nearest image in the database. These classifiers are suitable to handle a huge number of classes and do not suffer from over-fitting of parameters, which is a major issue in learning-based methods. More importantly, these methods do not require training. Although training is a one-time preprocessing step, retraining of parameters in large dynamic databases is a big overhead on the system. For smaller database however, learning based classifiers are recommended as they provide unsurpassed accuracy.

CHAPTER 2

SKEW DETECTION AND CORRECTION OF SCANNED DOCUMENTS

2.1. Introduction

A text document consists of several text lines. To estimate the skew angle of a text line, a straight line can be drawn through its characters. The angle of this straight line with the horizontal edges of the page is the skew angle of the text line (Figure 6).

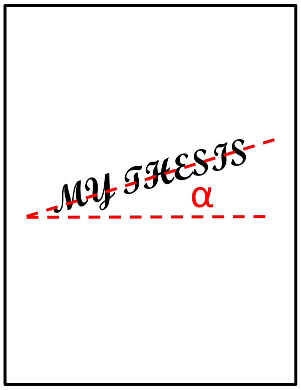


Figure 6: A skewed text line

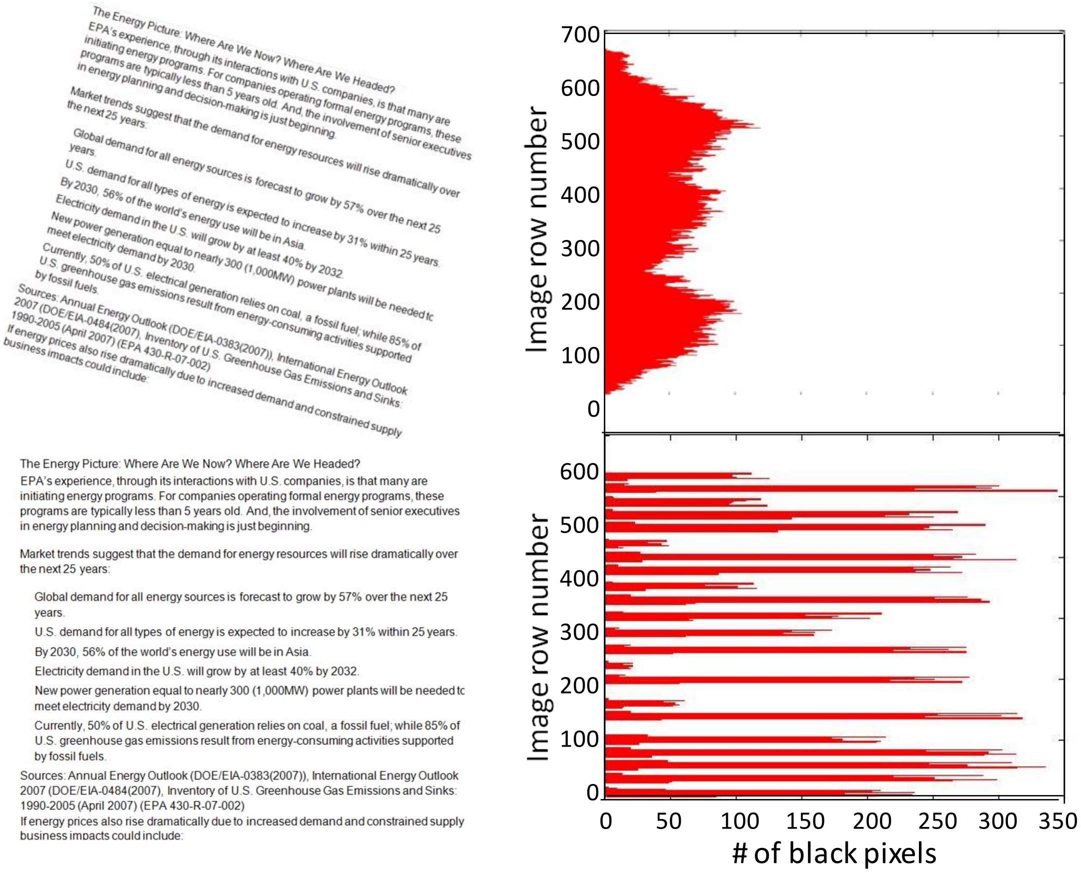
The dominant skew angle of the text lines in a page determines the skew angle of that page. A document originated electronically with a text editor has skew angle of zero. However, when a document is printed, photocopied or scanned, a non-zero skew angle will be introduced. Since document analysis algorithms such as text recognition or page layout analyzers usually assume a zero skewed page, skew detection and correction is considered a required preprocessing step. Moreover, to improve the quality of scanned documents, many scanners perform document skew correction immediately after a scan and before a document image is displayed on a computer monitor.

2.2. Skew Angle Detection Methods

Skew angle detection methods mainly fall into one of the following four categories:

2.2.1. Projection Profile

This method was initially proposed by Postl [29]. In this method, histograms of the number of black pixels along horizontal parallel sample lines through the document for a range of angles are calculated. For a non-skewed document, horizontal projection profile will have peaks whose width are equal to the characters' height with maximum peak heights at the text lines and valleys whose width are equal to the between-the-line spacing. Therefore, for each angle a measure of the variation in the bin heights (such as variance) along the projection profile is tracked and the angle with the most variation gives the skew angle.


The Energy Picture: Where Are We Now? Where Are We Headed?
EPA's experience, through its interactions with U.S. companies, is that many are initiating energy programs. For companies operating formal energy programs, these programs are typically less than 5 years old. And, the involvement of senior executives in energy planning and decision-making is just beginning.

Market trends suggest that the demand for energy resources will rise dramatically over the next 25 years:

Global demand for all energy sources is forecast to grow by 57% over the next 25 years.

U.S. demand for all types of energy is expected to increase by 31% within 25 years.

By 2030, 56% of the world's energy use will be in Asia.

Electricity demand in the U.S. will grow by at least 40% by 2032.

New power generation equal to nearly 300 (1,000MW) power plants will be needed tc meet electricity demand by 2030.

Currently, 50% of U.S. electrical generation relies on coal, a fossil fuel; while 85% of U.S. greenhouse gas emissions result from energy-consuming activities supported by fossil fuels.

Sources: Annual Energy Outlook (DOE/EIA-0383(2007)), International Energy Outlook 2007 (DOE/EIA-0484(2007), Inventory of U.S. Greenhouse Gas Emissions and Sinks: 1990-2005 (April 2007) (EPA 430-R-07-002)
If energy prices also rise dramatically due to increased demand and constrained supply business impacts could include:

Figure 7: Histograms of a skewed document (top) and a non-skewed document (bottom)

Figure 7 shows histograms of a document for two different skew angles. As shown, when a document has a non-zero skew angle, the horizontal histogram is spread at different row numbers. However, in a zero skewed page the histogram is clearly discreet with strong peaks and separated by valleys. Also, since the number of black pixels in a document is constant, the spectrum energy of the histogram is constant. Therefore, for a non-skewed page, the same amount of energy has to spread in a smaller area (sum of each line's height in a page). That is why for a non-skewed page, histogram peaks are much stronger than the peaks on a skewed page. Refer to Appendix A for a MATLAB source code.

Projection Profile Limitations:

The limitation of this method is that the document needs to be free from images, diagrams and graphs as these have greater contribution than text lines in the profile, therefore compromising the accuracy of angle detection. Moreover, this method is not very efficient, as projection profile of all possible angles need to be calculated. Additionally, projection profile methods are limited to estimate skew angle within ±10° to 15° and the accuracy of skew detection depends upon the angular resolution of the projection profile. Finally, projection profile methods are very sensitive to noise.

Improved Projection Profile Methods

In order to reduce high computational costs, several variations of Postl's original method have been proposed:

Baird [30] proposed a technique to select a subset of the points to be projected. He suggested finding the connected components and projecting only the bottom-centers of the connected components bounding box parallel to set of angles. For each angle, the sum of squares of the values in the projection profile is used as a criterion function. Finally, by using least squares procedure, the peak of the criterion function is approximated.

Ciardiella [31] selected a sub-region of a text based on high density of black pixels per row and used projection profile of this selected sub-region. The criterion function used in this method is the mean square deviation of the profile. Since, only a sub-region of the whole document is selected, the computational cost of the algorithm reduces.

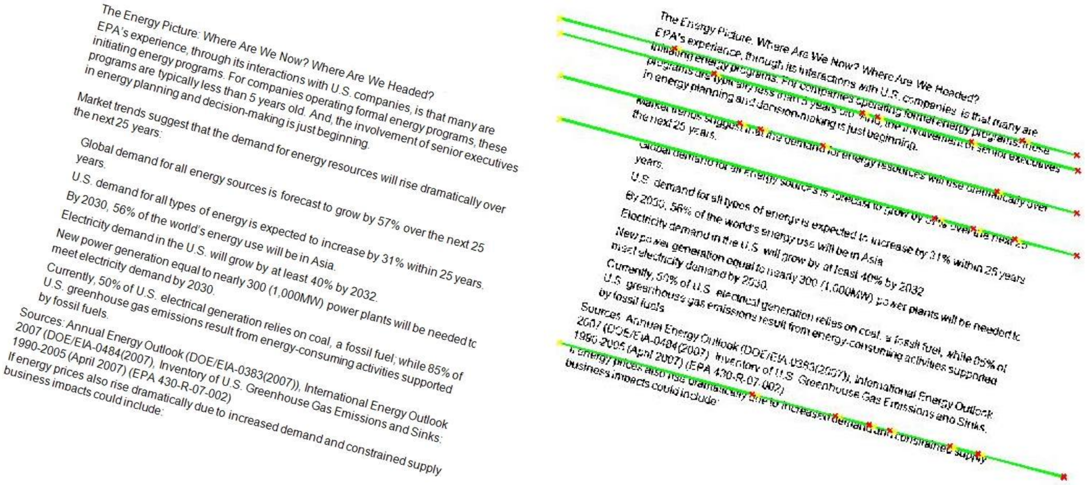
Ishitani [32] selected a cluster of parallel lines on the image and the bins of the profile stored the number of black to white transitions along the lines. Again, the aim of this modification was to improve the high computational cost of rotating the whole document for a wide range of angles.

Bloomberg [33] extracted a sample image in the skewed document. Skew angle was calculated by sample image rather than whole document which supposedly results in a faster skew estimation method. Same as Ciardiella's method, the goal is to apply the algorithm on a section of a page.

2.2.2. Hough Transform [HT]

The Hough transform was first introduced by Duda [34], who generalized the idea of Paul Hough [35]. In 1981, Ballard [36] applied the HT method to detect arbitrary shapes. In 1988, Illingworth [37] used HT to correct skew angle of a page. In this method, each point in Cartesian space (x,y) is mapped to a sinusoidal curve in ρ- θ Hough space using transform function $\rho = x\cos\theta + y\sin\theta$. When multiple points are on the same line, their transformation will intersect at the same point on the transform plane. Therefore, an accumulator is defined to track number of intersections that sinusoidal curves have at various ρ and θ values. As the number of intersections increases at a particular value of θ

so does the possibility of having a line in the original image corresponding to that θ value. Finally, peaks at each ρ value give the angles at which straight lines can be fit through the original pixels. The skew angle of the documents is found by averaging the θs with highest accumulator peaks.



The Energy Picture: Where Are We Now? Where Are We Headed?
EPA's experience, through its interactions with U.S. companies, is that many are
initiating energy programs. For companies operating formal energy programs, these
programs are typically less than 5 years old. And, the involvement of senior executives
in energy planning and decision-making is just beginning.
Market trends suggest that the demand for energy resources will rise dramatically over
the next 25 years:
Global demand for all energy sources is forecast to grow by 57% over the next 25
years.
U.S. demand for all types of energy is expected to increase by 31% within 25 years.
By 2030, 56% of the world's energy use will be in Asia.
Electricity demand in the U.S. will grow by at least 40% by 2032.
New power generation equal to nearly 300 (1,000MW) power plants will be needed tc
meet electricity demand by 2030.
Currently, 50% of U.S. electrical generation relies on coal, a fossil fuel; while 85% of
U.S. greenhouse gas emissions result from energy-consuming activities supported
by fossil fuels.
Sources: Annual Energy Outlook (DOE/EIA-0383(2007)), International Energy Outlook
2007 (DOE/EIA-0484(2007)), Inventory of U.S. Greenhouse Gas Emissions and Sinks:
1990-2005 (April 2007) (EPA 430-R-07-002)
If energy prices also rise dramatically due to increased demand and constrained supply
business impacts could include:

Figure 8: A skewed image of a page (left), Hough Transform lines with the highest accumulator values (right)

Figure 8 shows detected lines with highest HT accumulator peaks on a skewed page. The average skew angles of these lines are considered the skew angle of the page. Figure 9 is the Hough parameter space for the skewed page shown in Figure 8. In this space, each black pixel of the page transforms to a sinusoid. The point of intersect between two sinusoids indicates the parameters of the line passing through both points. The white squares in Figure 9 show the five points in HT space with the most values of Hough accumulator, corresponding to the lines found in Figure 8. Please refer to Appendix A for the MATLAB source code used in this dissertation for HT method skew angle correction.

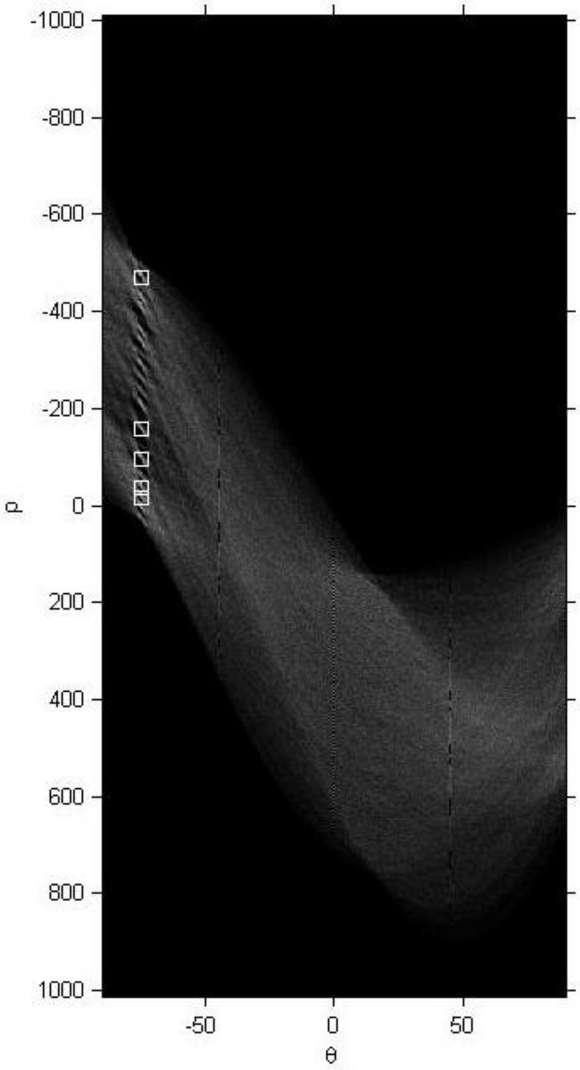


Figure 9: Hough Parameter space for the skewed page in Figure 8

Hough Transform Limitations

Even though this technique has shown high accuracy but it has its own shortcomings. Figure 10 shows an example of HT unsuccessful skew detection. Having different page columns and existence of a title and a foot note makes skew detection of this sample page particularly difficult. As shown, in this case, highest values of the accumulator do not represent the skew angle of the page.

HT is known as a skew detection method with high accuracy but relatively slow. As explained earlier, in this method every black pixel of a page needs to transfer to Hough space which makes this method computationally expensive. Also, this method is slow in the presence of noise, and in case of sparse text; it is difficult to choose a peak in Hough space [38], [39], [40]. Moreover, due to high computational cost, usually the skew is assumed within a range and relatively coarse angle intervals ($\theta > 0.5^\circ$) are calculated.

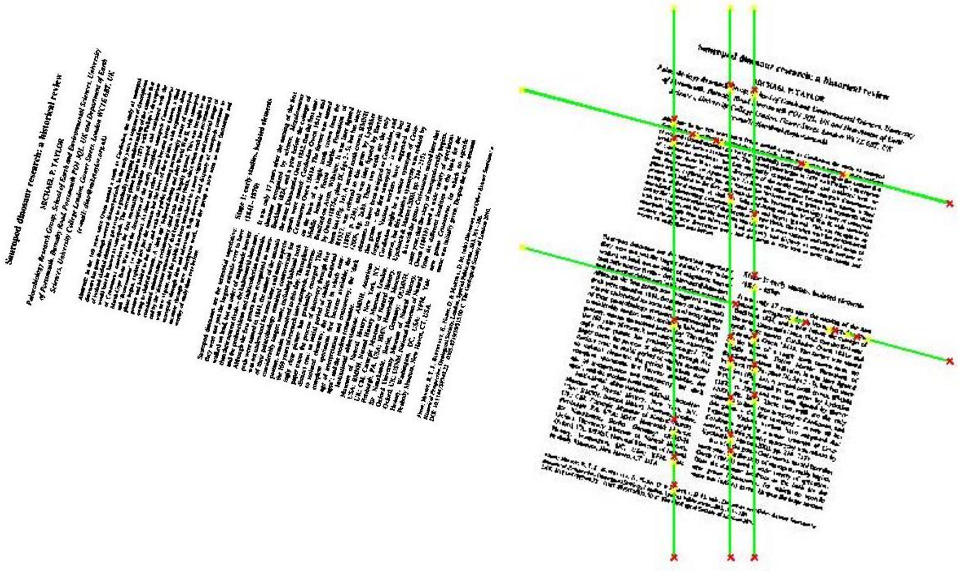


Figure 10: A skewed image of a page (left), HT lines with highest accumulator values (right)

Improved Hough Transform Methods

Hinds [38]**:** In order to decrease the amount of data within a binary image, Hinds [38] suggested using “burst image”. “Burst image” is a gray scale image in which each pixel’s intensity represents the vertical run-length of a column of black pixels in the original binary image. After finding “burst image”, Hough transform is applied to either vertical or horizontal burst image. The drawback is that in order to increase the speed of the algorithm, the resolution of the document image has to be decreased before creating "burst image". Also, this method does not perform well when the majority of the document is non-textual [39].

Manjunath [41] reduced number of Hough transforms by taking centroids of connected components instead of using all image pixels.

Le [42] and others used the bottom pixels of the candidate objects within a selected region for Hough transformation.

Most of the improvements to the HT method aim at reducing input data to the Hough space; however, they usually increase the complexity of the algorithm and reduce the accuracy of the skew angle detection.

2.2.3. Nearest Neighbor [N-N]

This method is based on connected components in which the first nearest-neighbors of all connected components are found and the histogram of the direction vectors for all nearest-neighbors is obtained [43].

By using Histogram peak, the skew angle can be found. Here are the main steps in the NN algorithm (refer to Appendix C for the MATLAB source code):

Step 1. Determine connected components
Step 2. Find nearest neighbor of each component using Euclidean distance
Step 3. Find the angle between centroids of nearest neighbor components
Step 4. Accumulate the angles in a histogram
Step 5. The dominant peak in the histogram indicates the skew angle

Figure 11 is an example of connected components detected on a skewed page and Figure 12 is the histogram of the accumulated angles.

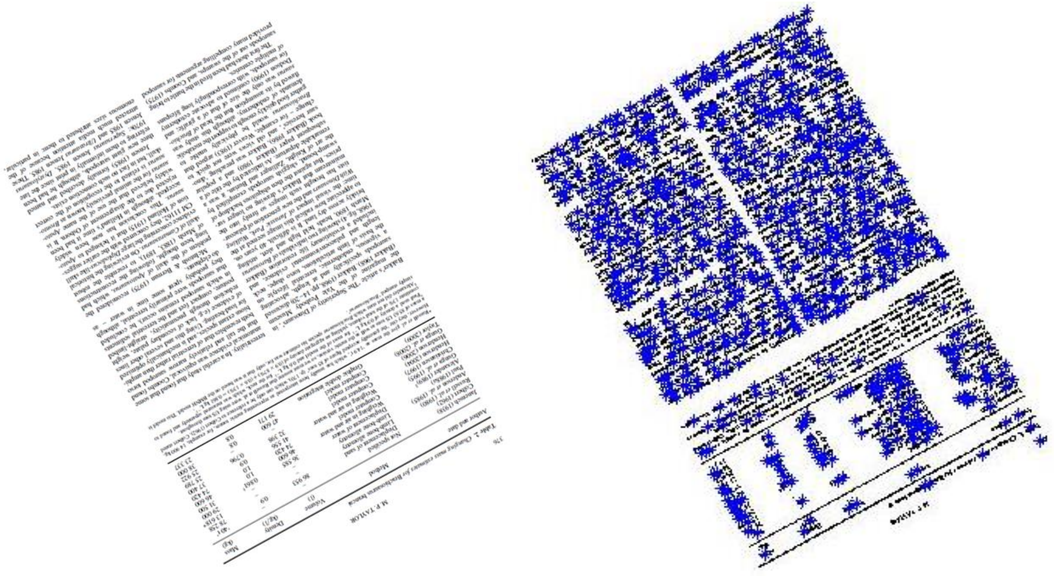


Figure 11: A skewed image of a page (left), connected components used for N-N skew detection method (right)

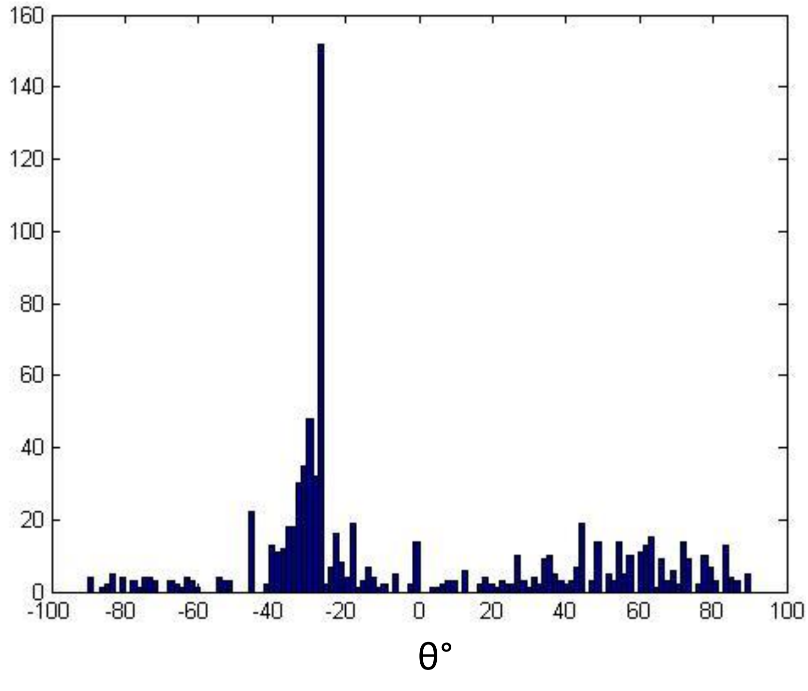
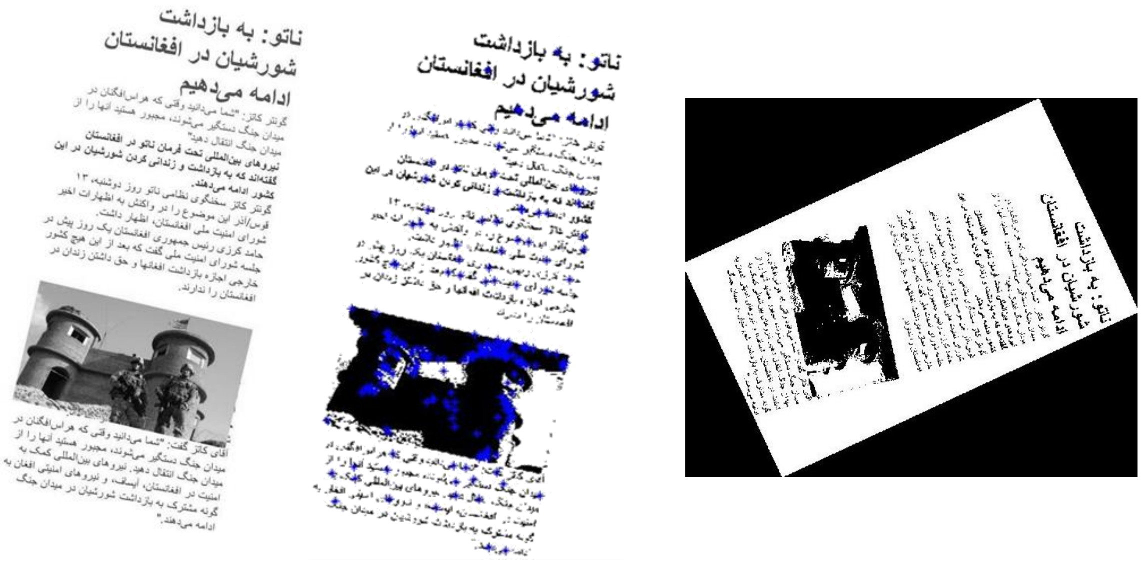


Figure 12: Histogram of the image in Figure 11

The histogram clearly show a strong peak at $-25°$, therefore, by rotating the image by $+25°$ the skew of the image is corrected. Please note that by correcting the skew of the image, the image is going to be up-side-down. Therefore, an orientation correction algorithm is needed to correct the image orientation. Figure 13 is another example of applying N-N method on a skewed page. In this case, however, the image skew angle is not correctly identified. This example demonstrates one of the weaknesses of the N-N method as its accuracy decreases when the image includes graphical pictures.



ناتو: به بازداشت
شورشیان در افغانستان
ادامه می‌دهیم

Figure 13: A skewed image of a page (left), connected components used for N-N skew detection method (middle), incorrect skew correction (right)

Figure 14 shows the histogram of this image. From the histogram, we observe that due to the presence of the graphical picture, the N-N accumulator has several strong peaks, which leads to an incorrect skew detection of the image.

N-N method limitations

The advantage of N-N method is that it is not limited to any range of skew angles. However, in the presence of noise and subparts of characters, accuracy of this method decreases significantly. In addition, this method requires special attention for dealing with different scripts, and connected or broken characters, and heavily depends on the quality of the binarization process output. This dependency can be very problematic when dealing with complex or degraded data, such as historical documents.

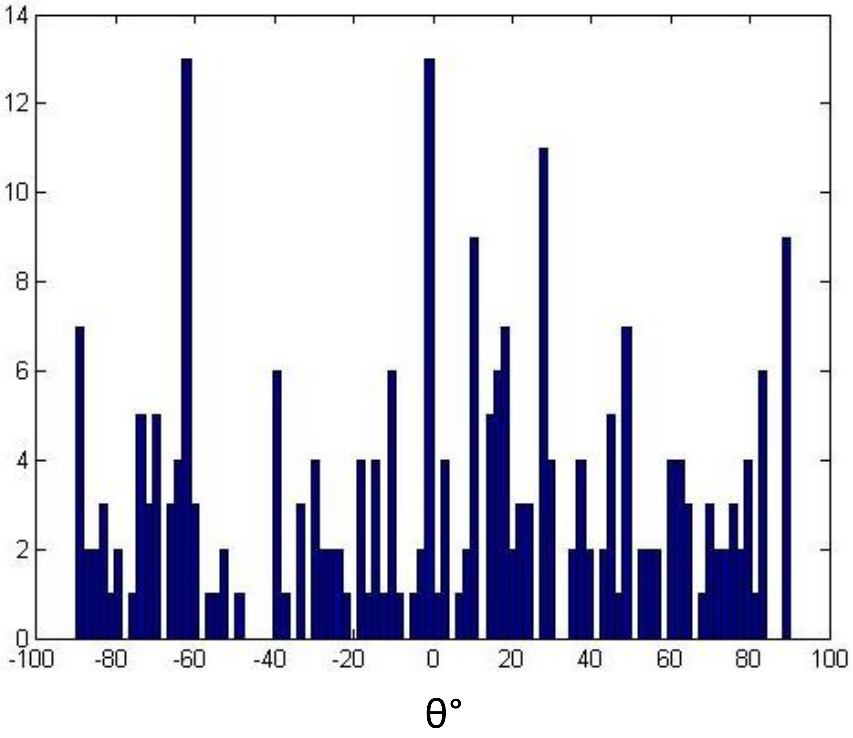


Figure 14: Histogram of the image in Figure 13

2.2.4. Fourier Transform

In this method first 2-D Fourier transform will be applied to the image plane. Then, coefficients of the power spectrum are calculated and stored in a spectrum [29]. A directional criterion for each angle is then calculated. The angle that maximizes the directional criterion is assumed to give the skew angle of the image. Peake [44] extended the earlier work of Postl [29] by calculating skew angle from Fourier spectrums calculated from a number of equally sized blocks of the original image. Peake noted that a directional alignment of energy occurs in Fourier space along an angle corresponding to the skew of the image. He speculates this directional alignment is due to spacing with the lines of a text.

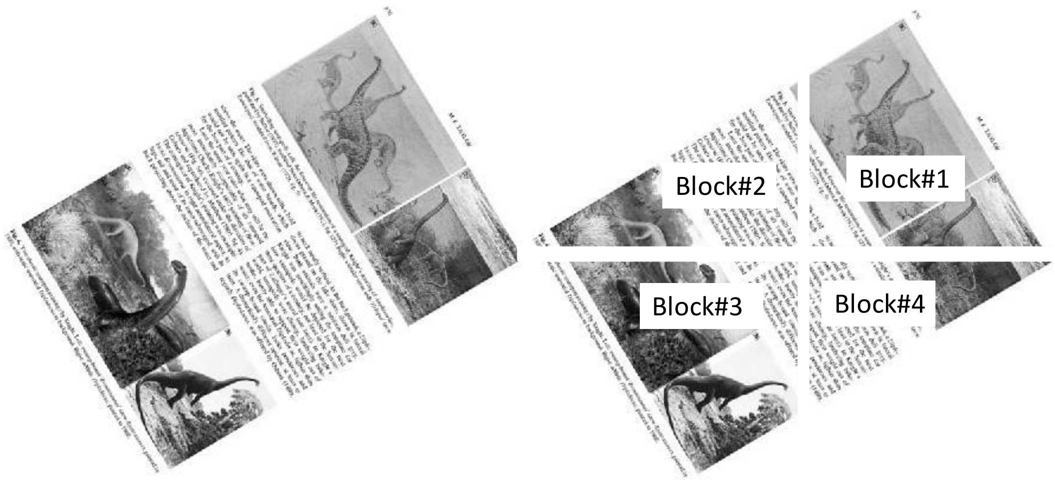


Figure 15: A skewed image of a page (left), original image sliced into 4 equal blocks (right)

To evaluate the Fourier transform in skew detection of documents, we implemented Peake's method (refer to Appendix D). Based on this method, we slice an image to 4 blocks of equal size (Figure 15). Then, using FFT, the Fourier spectrum is computed for each block. Based on Peak's recommendation, a small window of size $W \times W$, centered at the origin, is removed to improve the accuracy of the skew detection (Figure 16).

For consistency between the blocks, all values are normalized and 5 highest pairs of peaks are found and the angle of the line connecting each point and the center of the block with respect to the vertical axis is calculated.

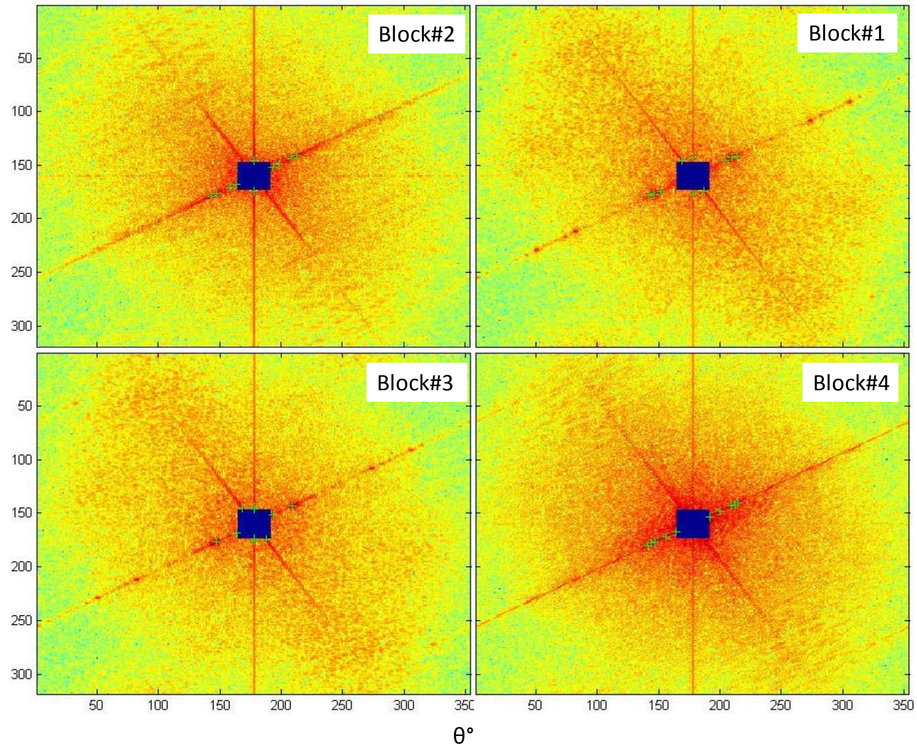


Figure 16: Fourier spectrum of the 4 blocks in Figure 15

Finally, in a histogram, integrals of normalized value of the peaks in the Fourier spectrum for each angle are collected. In the histogram, the bin with the highest peak is assumed to give the skew angle (Figure 17).

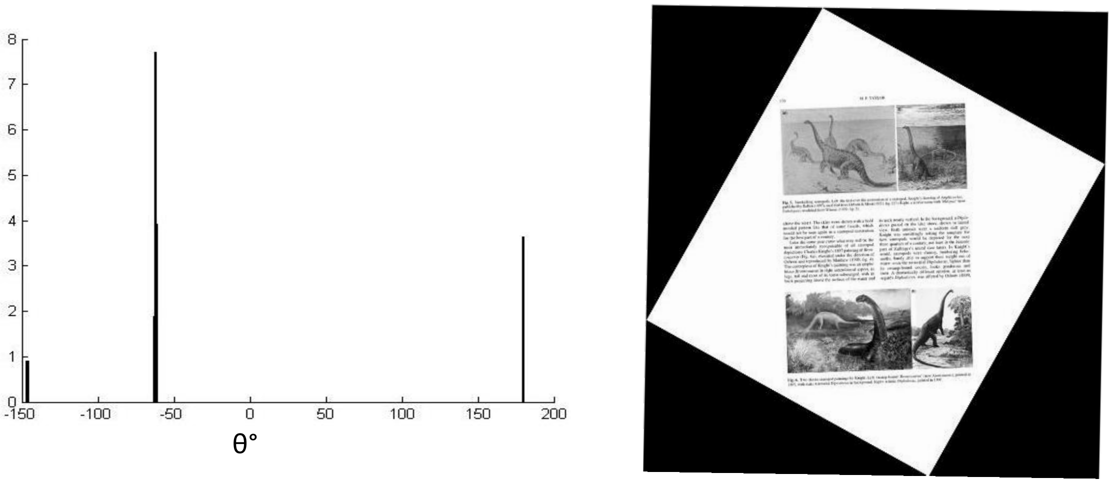
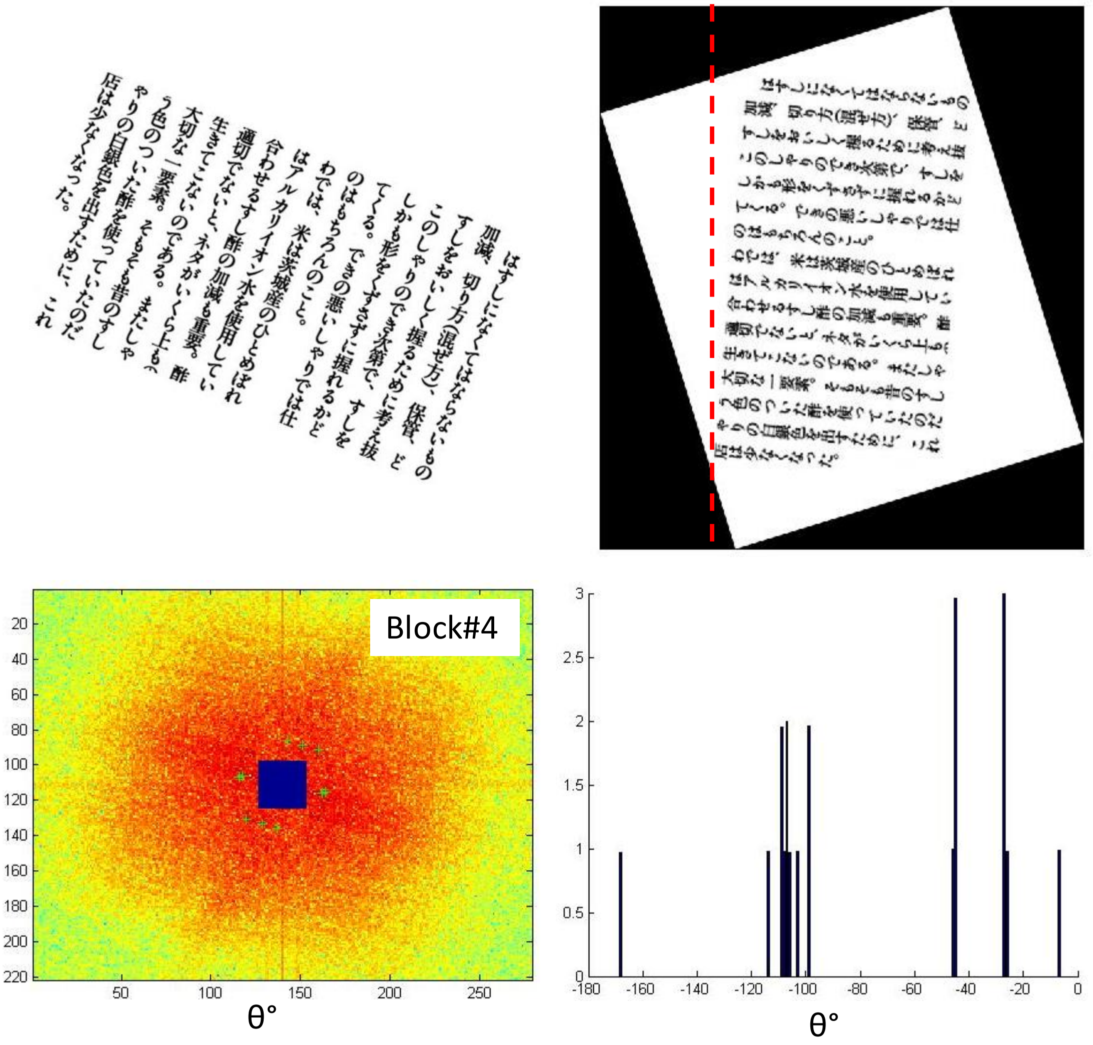


Figure 17: Histogram of the integrals of normalized value of the peaks in the Fourier spectrum (Left), corrected skew of the image in Figure 15 using Fourier transform

Limitations of Fourier Transform method

For large images, this method is computationally expensive since 2D Fourier transform of each pixel in the document has to be computed.

Also very often for a document image, the largest density direction of Fourier space may be different than the true skew direction. Figure 18 shows a skewed Japanese text which Fourier transform failed to correct.



はすしになくてはならないもの
加減、切り方(混ぜ方)、保管、ど
すしをおいしく握るために考え抜
このしゃりのでき次第で、すしを
しかも形をくずさずに握れるかど
てくる。できの悪いしゃりでは仕
のはもちろんのこと。
わでは、米は茨城産のひとめぼれ
はアルカリイオン水を使用してい
合わせるすし酢の加減も重要。酢
適切でないと、ネタがいくら上も
生きてこないのである。またしゃ
大切な一要素。そもそも昔のすし
う色のついた酢を使っていたのだ
やりの白銀色を出すために、これ
店は少なくなった。

Figure 18: A skewed image of a Japanese text (top-left), outcome of the skew correction algorithm (top-right), Fourier spectrum of one of the image blocks (bottom-left), and histogram of the normalized value of the peaks in the Fourier spectrum (bottom-right)

2.3. Skew correction mechanism

Once the skew angle α is identified, the document is simply rotated by a rotation matrix:

$$R_\alpha = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{1}$$

2.4. Skew Detection based on an axes-parallel bounding rectangle

In this dissertation, we propose a new method based on geometrical features of a skewed document. In this method, we calculate the area of an axes-parallel rectangle bounding box (Figure 19 and Figure 20). By rotating only the peripheral pixels of a text using a simple numerical method (explained in page 30), the area of the axes-parallel rectangles is minimized. The angle with the least area of the axes-parallel rectangle represents the skew angle. We would like to mention that our algorithm differs from typical minimum area bounding box methods such as the ones offered in [45] and [46]. Safabakhsh's method [45] is based on minimizing the area of a bounding box found from boundary pixels of connected components in the text. This method is computationally intensive, as all connected components of a text is held in memory for bounding box area calculation and for all possible angles. In the patent filed by Rudak [46], the boundary pixels are used to create a polygon and at each rotational angle a criterion is defined as the difference between the area of the polygon and the bounding box. Again, Rudak's method requires several computationally intensive calculations like creating the encompassing polygon, calculating the area of the polygon, vertex transformations for each rotation and more importantly not a methodological approach to search for its criterion minimization.

In our method however, only the outer peripheral pixels of a document are used to form an axes-parallel rectangle. Then, using the Bisection numerical method, skew angle is detected efficiently in a few iterations.

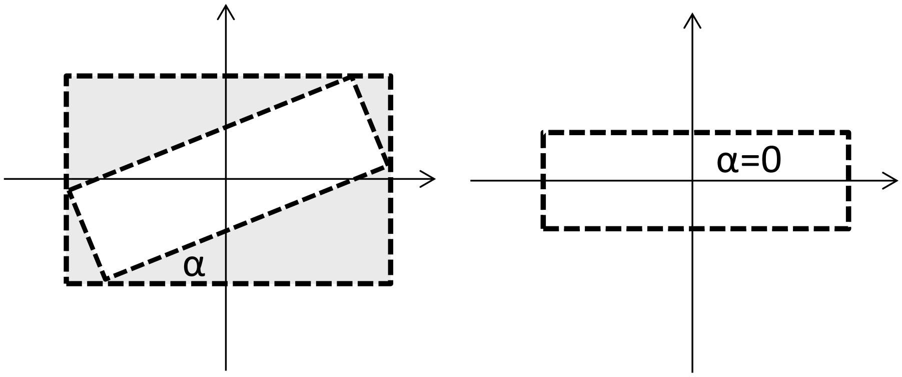


Figure 19: Left: An angled rectangle embedded in an axes-parallel rectangle. Right: A rectangle at zero angle

2.4.1. The proposed algorithm and its implementation

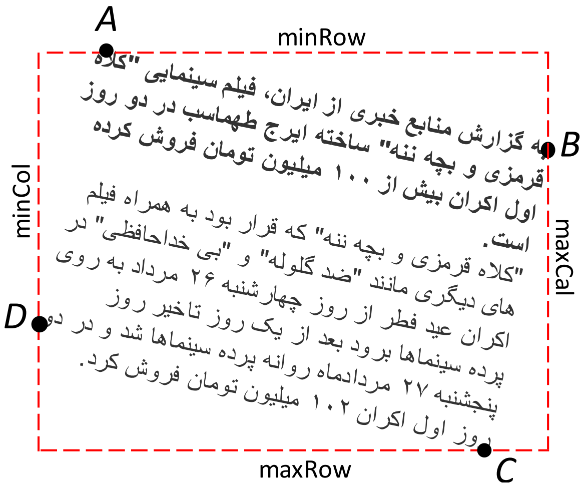
Our algorithm relies on correcting the skew angle of a text using the outer peripheral pixels; and therefore, can correct the skew of a text regardless of the script or the contents of the text. This method performs well compared to many of the available techniques because it can correct texts containing images, diagrams, etc. and works in wide range of angles.

Underlying Trigonometry

The following theorem is proposed as the basis of our skew correction methodology:

Theorem: In Euclidean plane geometry, assume a rectangle centered at origin and rotated by angle α. Using 4 corner vertices of this rectangle; create another rectangle oriented parallel to the axes. The area of this axes-parallel rectangle is minimum when α is a multiple of $\pi/2$.

The proof of the above theorem is trivial, as a skewed rectangle can always be “fitted” in a rectangle parallel to the axes made from its vertices (demonstrated in Figure 19). In our algorithm, we use this basic trigonometry fact, to minimize the area of the rectangle created from four most corner pixels of an image and parallel to the axes. In other words, we rotate the text in a way to eliminate the gray regions of the picture in Figure 19.



به گزارش منابع خبری از ایران، فیلم سینمایی "کلاه
قرمزی و بچه ننه" ساخته ایرج طهماسب در دو روز
اول اکران بیش از ۱۰۰ میلیون تومان فروش کرده
است.
"کلاه قرمزی و بچه ننه" که قرار بود به همراه فیلم
های دیگری مانند "ضد گلوله" و "بی خداحافظی" در
اکران عید فطر از روز چهارشنبه ۲۶ مرداد به روی
پرده سینماها برود بعد از یک روز تاخیر روز
پنجشنبه ۲۷ مردادماه روانه پرده سینماها شد و در دو
روز اول اکران ۱۰۲ میلیون تومان فروش کرد.

Figure 20: Corner pixels detection of a text

Step-by-step description of our algorithm:

Step 0: Preprocessing: After image binarization, apply noise removal and border elimination techniques to prepare the image for skew detection [47], [48], and [49].

Step 1: Text border detection: Simply scan the document row by row to detect the first and the last black pixel in each row. From this point forward just use the text border pixels as a representative of the document.

Step 2: Corner pixel detection (refer to Figure 20):

minRow: From step 1, find pixels with the minimum row number. Among them, the pixel with the minimum column number is our *minRow* pixel (Point A in Figure 20).

maxRow: From step 1, find pixels with the maximum row number. Among them, the pixel with maximum column number is our *maxRow* pixel (Point C in Figure 20).

maxCol: From step 1, find pixels with the maximum column number. Among them, the pixel with minimum row number is our *maxCol* pixel (Point B in Figure 20).

minCol: From Step 1, find the pixels with the minimum column. Among them, the pixel with maximum row number is our *minCol* pixel (Point D in Figure 20).

Step 3: Calculate the area of a rectangle parallel to axes (dashed rectangle in Figure 20):

$$A = (maxCol - minCol).(maxRow - minRow) \tag{2}$$

Step 4: Find direction of rotation to minimize the area:

Rotate border pixels $\pm 1°$.

If $A_{+1^\circ} < A$ *then direction is positive (CCW)*
If $A_{-1^\circ} < A$ *then direction is negative (CW)*
Otherwise, there is no direction. Skip Step 5 and set $\alpha = 0$.

Step 5: Rotate the picture $8°$ in a loop in the direction found in Step 4 and monitor area of the rectangle.

If $A_{current} \geq A_{previous}$ *then* $\alpha = \alpha_{previous}$, $A = A_{previous}$, *exit the loop.*

Step 6: Use Bisection method with steps of 0.5 to find the angle giving the minimum area:

```
step size=8
while step size>0.125      //we are targeting 0.125° accuracy
step size=step size/2
Rotate original border pixels $\alpha \pm step\ size°$.
$A = min(A,\ A_{+stepsize^\circ}$ , and $A_{-stepsize^\circ})$ .
Set $\alpha$ to the corresponding angle of A.
loop
```

Step 7: α is the skew angle. Rotate the original picture by $-\alpha$ to correct the skew angle of the text (refer to skew correction mechanism described in section 2.3).

Based on theorem mentioned in section 2.4.1, the result of the skew correction by minimizing the calculated area may be a multiple of $\pi/_2$. In step 8 we correct a possible $90°$ turned document.

Step 8: Scan the document row-wise between *rowMin* and *rowMax* and column-wise between *minCol* and *maxCol*. If column-wise scan had more white columns than row-wise white rows, then the picture need to be rotated $90°$. If the direction obtained from step 4 was positive, rotate $-90°$ and if the direction obtained from step 4 was negative or it was no direction, rotate $+90°$.

Performance and limitations

Angle Limitation: Our algorithm corrects skews of any angle. This is a big advantage over existing algorithms. However, if the original text is between 135° to 225° or exactly at 270°, our algorithm corrects the skew but not its orientation (the document will be 180° flipped). Fixing document orientation requires language specific algorithms. For example, orientation of a Roman script can be fixed based on the fact that ascenders[1] are more likely to occur than descenders[2].

Text Content Dependency: Our algorithm is content independent. Since we only use the outer periphery of a text, the content of the text does not matter. Therefore, the skew of a text with graphs, tables, diagrams, etc. can be corrected.

Speed: Our algorithm works very fast (in average less than 1 second on a typical PC) as it only works with the periphery pixels and number of rotations has been optimized using a binary search.

---

[1] The ascenders are the parts of lowercase characters that lie above the mean line.

[2] A descender is the portion of a letter that extends below the baseline of a font.

Accuracy: Our algorithm corrected skews of our tested documents with accuracy of 0.06 degrees. There is no need to correct for lesser angles as angles less than 0.1 degree have minimal effect on performance of character recognition algorithms.

2.5. Our method compared to existing methods

To compare our algorithm to the existing state-of-the-art skew detection algorithms, we programmed 4 conventional skew detection techniques: PP, HT, *1*st-NN, FT (source codes provided in Appendix A to Appendix D) and compared them against our axes-parallel bounding rectangle method. Test environment consisted of a PC with Intel i5-2540M CPU @ 2.60GHz with 16 Gb of memory and the tests were performed in MATLAB. To measure the computing time, MATLAB's "clock" function was used. Total of 130 images were used from the following sources:

- *Our database of 30 images*
- *University of Maryland Tobacco800 image database*
- *www.mediateam.oulu.f-i image data base*

We categorized our database to two groups of "Typical Samples" consisted of 100 images obtained from online sources and "Extreme Samples" consisted of 30 images.

2.5.1. Typical Samples

Table 2 and Figure 21 show the results of our method compared with the other conventional methods. As indicated in Figure 21, the goal is to approach a 100% success rate as fast as possible. It is clear that our method is superior compared to other methods both in the success rate and average time required for skew correction.

	PP	*1st*-NN	HT	FT	Our Method
Success Rate	86%	48%	92%	82%	95%
Avg. time (s)	16.8	2.5	4.9	2.8	1.56

Table 2: Success rate and average skew correction time

As mentioned earlier, our method is very efficient as it works only with the peripheral boundaries of the text. On the other hand, the success of our method is highly dependent to a successful noise removal preprocessing.

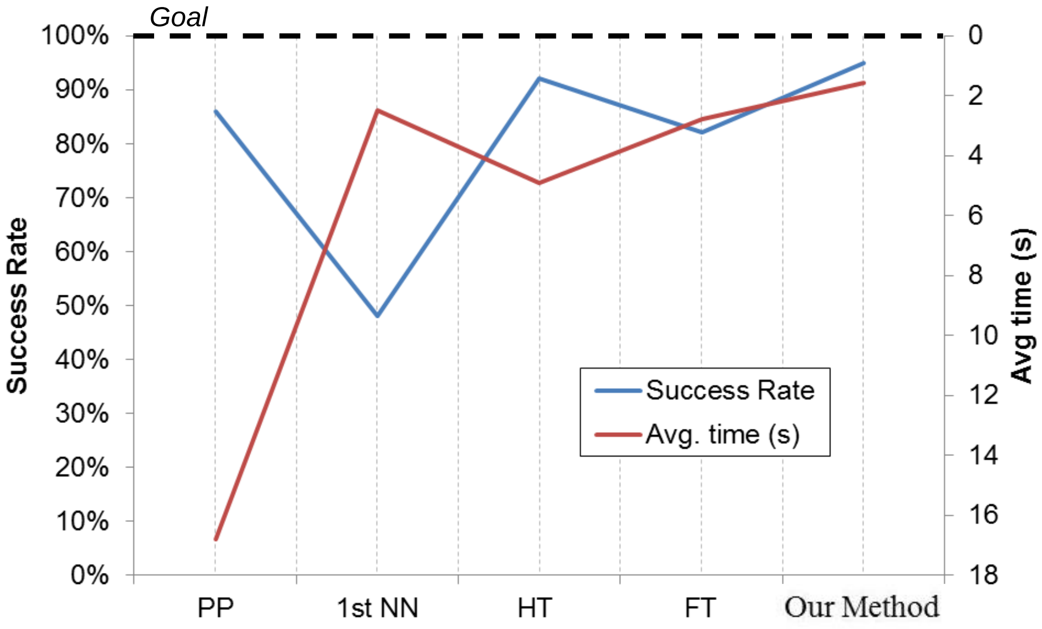


Figure 21: Comparison of success rate and average skew correction time

2.5.2. Extreme Samples

In the extreme samples category we created skewed pages in which typical skew correction techniques usually perform poorly. One example is when a page is skewed more than 45 degrees, includes images/tables, and consists of more than one column. Each image in this category was carefully scanned, then corrected in Photoshop for a 0˚.

Then, each image was arbitrarily rotated using the rotate function in Photoshop. Success is assumed when the difference between the estimated skew angle and the tagged skew angle is less than 1˚. Figure 22 shows samples of the images in this category.

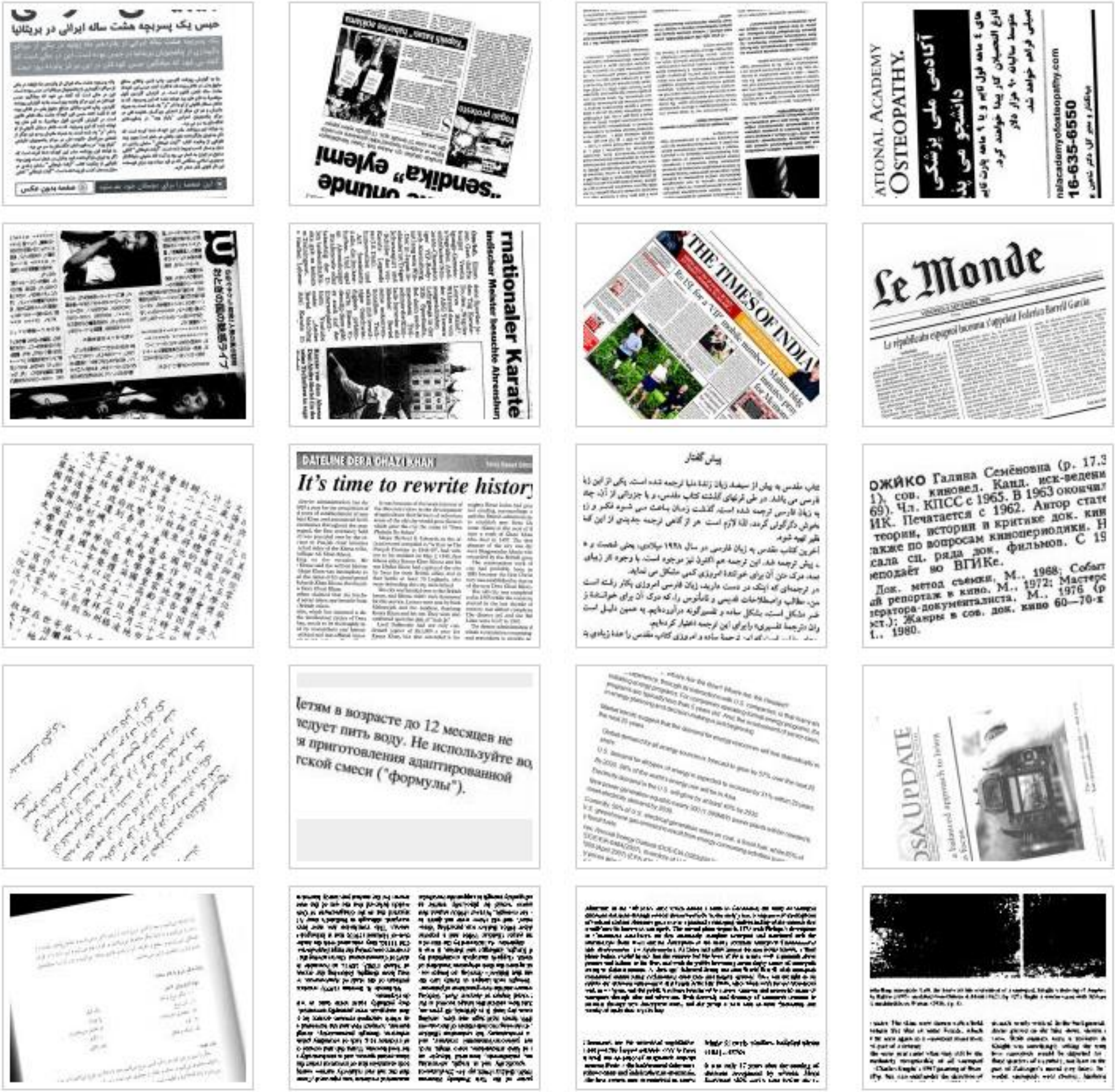


Figure 22: Sample images in the extreme samples category

Table 3 shows the results of our method compared with the other conventional methods. As shown, our method is successful in correcting all 30 images in this category.

However, in some cases the orientation of the image is not corrected. Among the conventional methods, PP performed best but not so much successful when images include pictures. *1st*-NN method performed poorly as this method works best in simple text images.

Img#	Skew Angle	PP	HT	1st-NN	FT	Our Method
1	25	Y-90	Y	Y	Y	Y
2	9.5	**N**	Y	**N**	**N**	Y
3	15	Y	Y	Y	Y	Y
4	15	Y	**N**	Y	**N**	Y
5	59	**N**	Y	Y	Y	Y
6	-29	Y	Y	**N**	Y	Y
7	-76	Y	**N**	**N**	**N**	Y
8	-68	Y	**N**	**N**	**N**	Y
9	7	Y+90	Y	**N**	Y	Y
10	36	Y	Y	Y	Y	Y
11	7	**N**	Y	**N**	Y	Y
12	36	Y	Y	**N**	Y	Y
13	82.5	**N**	Y-90	Y	Y-90	Y-90
14	-9.3	Y+180	Y-180	**N**	Y-180	Y-90
15	12	Y+90	Y+180	**N**	Y+180	Y+180
16	-3.5	**N**	**N**	**N**	Y	Y+180
17	88.3	Y	**N**	**N**	Y	Y
18	89	**N**	**N**	**N**	**N**	Y+90
19	-2	Y+180	Y+180	Y	Y+180	Y+90
20	-4	Y-90	**N**	**N**	Y	Y-90
21	-44	Y	Y	**N**	Y	Y-90
22	-12	Y	Y+180	**N**	**N**	Y
23	23	Y	**N**	**N**	Y	Y
24	1	Y	Y	Y	Y	Y
25	10	Y	Y	**N**	**N**	Y
26	-7	Y	**N**	Y	**N**	Y
27	2	Y	Y	**N**	Y	Y
28	-43.5	Y+180	Y	Y	Y	Y+180
29	20	**N**	Y	**N**	**N**	Y
30	-10	Y	Y	**N**	Y	Y

Table 3: Skew detection success rate comparison
"Y" indicates a successful detection, "N" indicates an unsuccessful detection, and ± indicates deviations from the correct skew angle.

2.5.1. Experimental Results

Below are some selected examples showing the skewed image (left) and the skew corrected image using our method (right).

First page of an English technical paper: As shown below the skew of the image is corrected accurately.

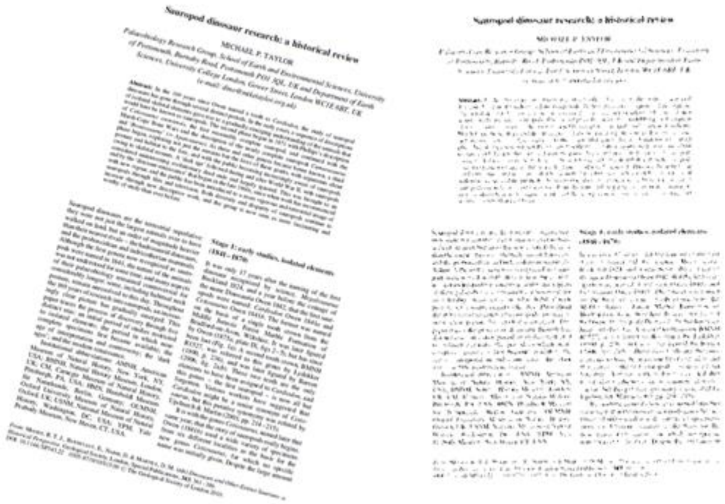


Figure 23: Skew correction of a page of an English technical paper

A page of English book including several pictures: The page is de-skewed correctly.

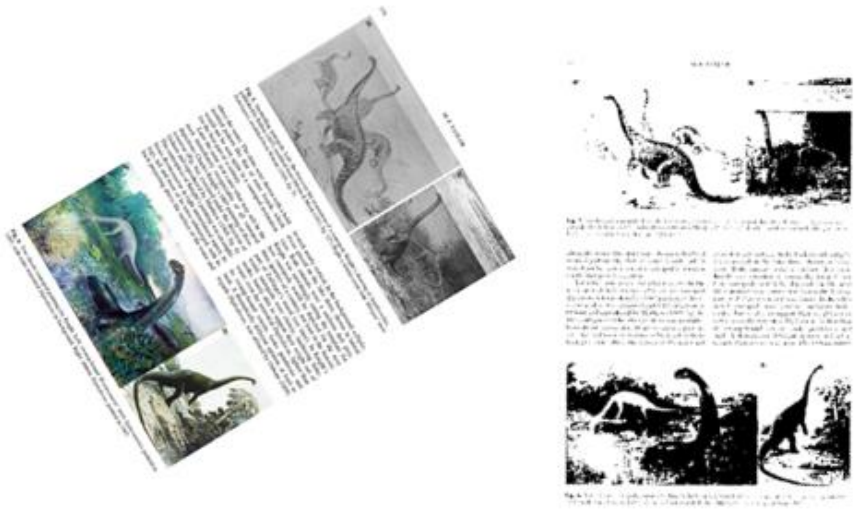


Figure 24: Skew correction of a page of an English book with graphical images

A page of English book including a table: As shown the page is de-skewed correctly. However, the page is upside down. As noted above, if the original text is between 135° to 225° or exactly at 270°, our algorithm corrects the skew but not its orientation. Here the image is at 209°, therefore it gets de-skewed to an upside down orientation.

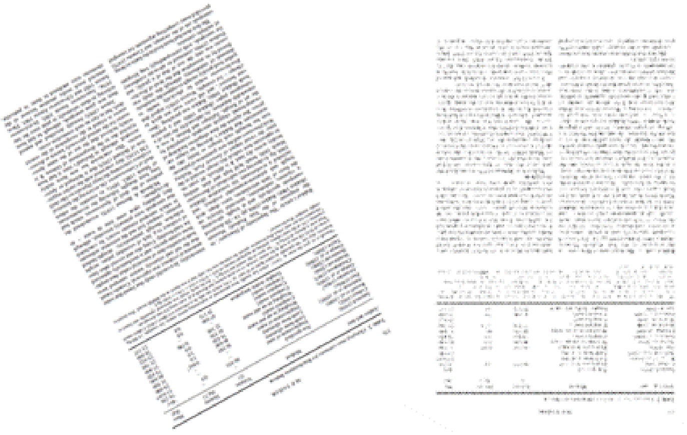
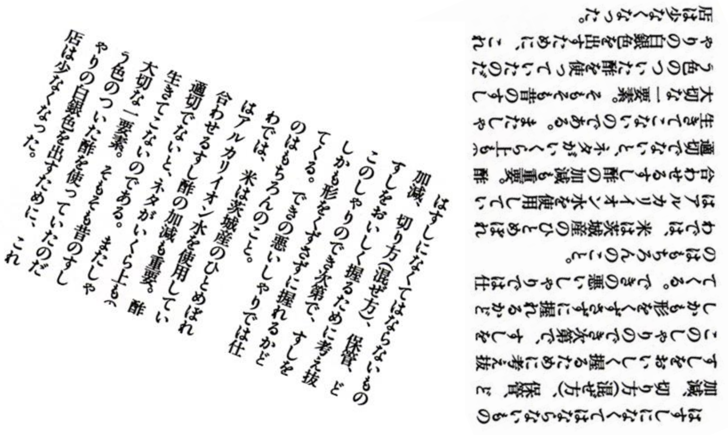


Figure 25: Skew correction of a page of an English book with a table at 209°

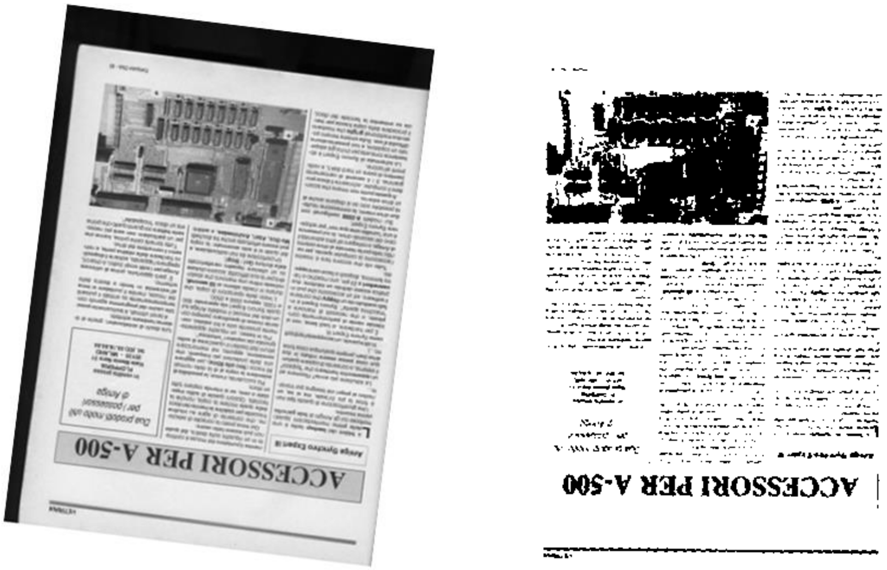
A Japanese text: As shown below, the text is de-skewed correctly. However since this Japanese text is written column-wise, its orientation is not correct. Here, an orientation correction algorithm specific to Japanese script is needed. This correction algorithm could be a simple smearing algorithm such as the one used in step 8, tuned for a top-to-bottom documents.



はすしになくてはならないもの
加減、切り方(混ぜ方)、保管、ど
すしをおいしく握るために考え抜
このしゃりのでき次第で、すしを
しかも形をくずさずに握れるかど
てくる。できの悪いしゃりでは仕
のはもちろんのこと。
わでは、米は茨城産のひとめぼれ
はアルカリイオン水を使用してい
合わせるすし酢の加減も重要。酢
適切でないと、ネタがいくら上も
生きてこないのである。またしゃ
大切な一要素。そもそも昔のすし
う色のついた酢を使っていたのだ
やりの白銀色を出すために、これ
店は少なくなった。

Figure 26: Skew correction of a Japanese text

A scanned image of an English document with graphical image: As shown, the image is de-skewed correctly. The preprocessing step of our algorithm removed the black borders and then the algorithm finds and corrects the skew of the image. However, because the original image is skewed 170°, our algorithm de-skews the image to an upside down orientation. Please note that the complexity of this document image as it includes an embedded table and picture and a wide bordered header. Also, the image has three columns with page number and footer. Other state-of-the-art algorithms such as projection profile were not successful in de-skewing this image.



ACCESSORI PER A-500

Figure 27: Skew correction of a Japanese text

A scanned image of an English document with a graphical image next to the text, a title, and a page number: The image is de-skewed correctly from 36 degrees skew angle. As shown in the right picture, our algorithm detects four corners of the image; and then, finds the area of the bounding box, parallel to the axes.

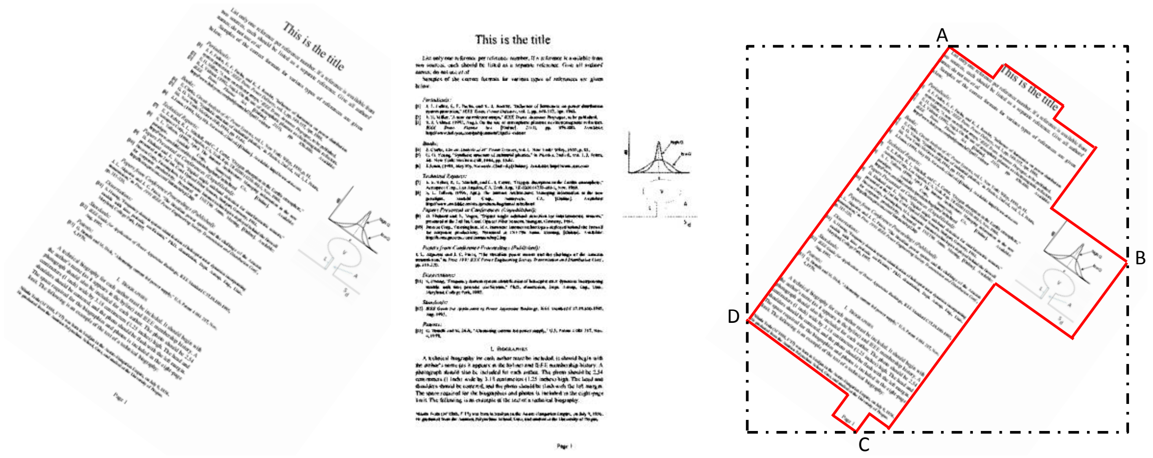


Figure 28: Skew correction of an English text

2.6. Conclusions

In this chapter we presented a novel technique for skew correction of a document. Our technique is based on an axes-parallel bounding box and works regardless of the content of the document. Therefore, our algorithm works in existence of graphical images, tables, charts, etc. with no angle limitations. A comparison of this algorithm with the existing state-of-the-art skew angle algorithms proved a reliable and fast algorithm, outperforming the compared methods.

CHAPTER 3

PAGE SEGMENTATION

3.1. Introduction

Page segmentation is a crucial step for a successful document image analysis. Published page segmentation algorithms often rely on some predetermined parameters such as font sizes, skews, and document scan resolutions. Variations of these parameters greatly affect the performance of the segmentation algorithms. In this dissertation, we introduce a new technique for page segmentation of complex documents in any format or skew angle. Our method simply performs a down-sampling followed by up-sampling. The sampling scale factor is calculated by estimating the white spaces between the rows. Our extensive evaluations confirm an efficient and robust technique for page segmentation capable of segmenting complex images.

3.2. Introduction

An image of a document is composed of not just pure text but a variety of segments such as pictures, tables, background, etc. For automatic text recognition of an arbitrary image of a document, segments of the image need to be separated and then analyzed. This process is called page layout analysis. The algorithms for layout analysis are classified primarily into two groups depending on the approach used. *Top-down algorithms* [50, 51, 52, 53, 54, 55, 56] start with the complete document image and divide it repeatedly to form smaller and smaller regions. In contrast, *Bottom-up algorithm* [57, 58, 59, 60, 61] start with the smallest components of a document (pixels or connected components) and repeatedly group them to form larger, homogenous, regions. Each approach has its own advantage and work well in specific situations. In addition, one could employ a hybrid approach that uses a combination of top-down and bottom-up strategies [62, 63, 64, 65]. There are a variety of algorithms that has been proposed to perform layout analysis of documents.

3.2.1. Top-Down Analysis Methods

Typical top-down approaches proceed by dividing a document image into smaller regions using the horizontal and vertical projection profiles.

Horizontal and vertical projection profile [50, 52] is a top-down method. In this method histograms of the number of black pixels along both horizontal and vertical sample lines through the document will be used for page layout analysis. Based on observations, it can be concluded that blank spaces between paragraphs are greater that the interline spacing. Therefore, by evaluating these spaces (valleys) in horizontal projection profile of a text, paragraphs and text lines within each paragraph can be located. In reality however, in order to improve efficiency and results, some sort of smoothing algorithm is used to decrease the original resolution of an image.

The Run Length Smearing Algorithm (RLSA) [53] is another Top-down algorithm which works only on binary images. This method consists mainly of two steps: block segmentation and classification. In block segmentation step, two bitmaps of an image one horizontally and one vertically are created. In order to create these bitmaps, the document image will be scanned vertically and horizontally. For each scan, sequences of white pixels that their run-length are less than or equal to a predefined threshold will be changed to black pixels in the current bitmap and all black pixels existed in the original image will be unchanged. Usually the threshold values for horizontal and vertical scans are different. The effect of this process is that all the neighboring characters will be merged into words, and words into text lines, and text lines into paragraphs depending on the value of the specified threshold and the distribution of white and black pixels within a document. In order to merge characters within a word, the threshold has to be chosen greater than character spacing within a word and less than between words spacing. After finding two bitmaps of the document image, bitmaps are then combined using logical AND operation. An extra horizontal smoothing may apply for finer segmentation. After finding all regions within the document, the algorithm moves on to classification step which calculates some features from each block and uses linear classifier to discriminate between text and non-text areas. The algorithm is fast but the values of thresholds have to be known in advance and the document should have a Manhattan page layout, that is, all columns can be isolated by a set of horizontal and vertical line-segments drawn through white spaces.

The Recursive X-Y Cut (RXYC) [54] is a tree-based top-down method which uses vertical and horizontal projection profiles alternatively to segment a page document to its smaller sub-blocks. In this method the whole document page is considered as a root of a tree and then based on the valleys on horizontal or vertical projection profiles the algorithm recursively split the documents into smaller blocks, representing the nodes of the tree. At each step of recursion, all valleys of the projection profile with values larger than predefined threshold will be found and based on the occurrence of valleys the chosen block will be segmented into smaller blocks. At each level of recursion the value of threshold may be different which requires some level of knowledge about the document structure. Therefore, in this method the threshold value and a criterion for stopping the recursion are required. Beside the disadvantage of requiring a threshold, the RXYC algorithm is only suited to pages where the layout is Manhattan. Furthermore, this method is sensitive to page skew.

Whitespace Analysis is another typical top-down algorithm. Since in many languages white space is used in similar ways as a delimiter of the layout, using background white space provides an advantage over foreground text. Another advantage of using background white space is that fewer parameters need to be specified. On the other hand, choice of maximal rectangles depends on the document format. This algorithm was proposed first by Baird [55]. In this method all the maximal white rectangular blocks whose union will cover the whole document background will be found. Then for each cover a sort key will be calculated which uses the area of the cover and a weighting function. After that, covers will be sorted based on their key values. The goal of using weighting function in calculating keys for covers is to assign higher weight to tall and long rectangular blocks which act as separators in the document. In the next step, the rectangular blocks one by one are combined and every time the uncovered area left by the union of the covers will be added to the segmentations sequence. The segmentations sequence is empty in the beginning and as the covers start merging the sequence will grow as well. The unification of rectangular covers continues until stopping rule has been met. Finally, after the unification process ends, the candidate text regions are the connected components within remaining uncovered sections. Baird algorithm is very complicated to implement. Also, it is sensitive to the stopping condition. If algorithm stops too early it results in a higher number of merge errors. And if it stops too late it will results in greater split errors. Shafait [56] also used background white rectangles to find text-lines in the document. In this method which is analogous to quicksort or branch-and-bound method, all tall whitespace rectangles in order of decreasing area are found. Then the whitespace rectangles which lines of text do not cross will be considered as obstacles. After finding obstacles a least square globally optimal text-line detection algorithm will be used to detect the text lines and eventually compute the bounding boxes of all characters making the text-line. In this method the page must have a Manhattan layout and by vertical and horizontal run must be separable into blocks. Also the choice of "maximal" rectangle might be non-intuitive for differently formatted documents. Advantage of this method is that it is language independent.

3.2.2. Bottom-Up Analysis Methods

Bottom-up approaches need to define primitive components to start the grouping process. The Docstrum algorithm by O'Gorman [59, 60] method is based on bottom-up k-nearest neighbor clustering of connected components of the page. Before implementing this algorithm, the page needs to be preprocessed. To do so, first *kFill* filter will be used to remove all salt and paper noises and then all the connected components will be found and the histogram of their bounding box sizes is made. By analyzing the histogram, the large font size components can be detected and handled separately by the Docstrum algorithm. The reason for this step is that within line and between line spacing will be different when the characters' font sizes become very large compared to the rest of the page. After preparing the page, the components are sorted based on their x value, and after that, for each component its k nearest neighbors are found. For each neighboring pair the angle and the distance between the components' centers will be kept. Next, the nearest-neighbor angle histogram will be compiled and the peak in the angle histogram will specify the rotation angle of the text. Then, two spacing histograms will be found: One histogram of nearest-neighbor distances for all angles within estimated orientation angle, and the other one for angles perpendicular to the orientation angle. The first histogram is called within-line histogram or between-line histogram. The peak found in above histograms gives within-line and between-line spacing. In order to find the text lines, a transitive closure is performed on within line neighbors to group them together. Then to find the text lines the centroids of the connected components are fed to the regression fit algorithm. Finally, pairs of text lines will be examined to determine if they belong to the same block or different blocks based on the approximately parallel criteria. If the criteria have been satisfied for each pair of lines they will be assigned to the same block. Some advantages of this method are robustness with respect to input parameters, independency from page orientation, relatively tolerant to random noises of an image. Disadvantages include; computationally more expensive compared to top-down approaches, the title and body text with larger font sizes have to be analyzed separately due to the larger font size spacing, the document image needs to be picture free before using Docstrum algorithm, the characters have to be well separated since the algorithm uses nearest-neighbor pairs to measure some features.

The Voronoi Diagrams algorithm [61, 62] is another bottom-up algorithm. In this method first all the connected components in a document image are found and sample points from connected components boundaries will be extracted using a predefined sample rate Rs. Then using a maximum noise size threshold T_n and other thresholds like width, height, aspect ratio, all noises will be extracted from the document. After that using sample points the Voronoi diagram will be generated and then the area Voronoi diagram is constructed by deleting the Voronoi edges that pass through a connected component. Finally the superfluous Voronoi edges that satisfy some specified criterion will be deleted to obtain boundaries of document components. An advantage of using this method is that it works with non-Manhattan layout documents. The disadvantages are that this algorithm is computationally expensive and causes split errors in case of varying font sizes and styles within a document.

Recent works [66, 67, 68, 69, 70, 71, 72, 73, 74, 75] in the area of page segmentation mostly focus on improvement of the classical methods or combining different methods to create new hybrid methods. In this document, we present a new method which does not use any aspect of the classical methods; instead, it is based on sampling technique as will be described next.

3.3. Page Segmentation Using Resampling Technique

In page segmentation the goal is to separate a document image into homogeneous zones, each consisting of only one physical layout structure. To achieve this goal, we eliminate or reduce white spaces in a segment to form a blob. Blobs then are separated using a simple connected component algorithm. In our technique we perform downsampling (zooming out) of the image followed by upsampling (zooming in) using a calculated scaling factor.

3.3.1. Image Resampling

Digital images consist of pixels which are measurements or samples of light from a subject. The original samples are usually obtained using a digital camera or a scanner by averaging the amount of red, green, and blue light that falls on the sensitive area of each of its sensing elements. Resampling is the mathematical technique used to create a new version of the image with a different number of pixels. Reducing the image size is called downsampling and increasing its size is called upsampling. When images are downsampled, information in the original image has to be discarded. When images are upsampled, the number of pixels increases however, new image details cannot be created (Figure 29). As a result, images normally become softer with upsampling. The reason is that the amount of information per pixel goes down. Thus downsampling followed by upsampling causes information loss or picture degrading (Figure 30).

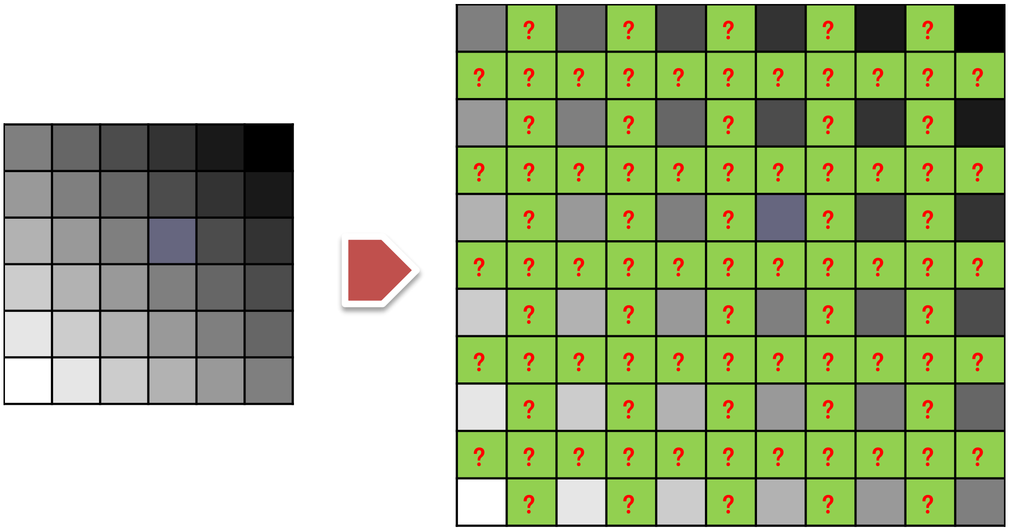


Figure 29: Illustration of image upsampling and required interpolation for missing information

In our technique for page segmentation, we take advantage of this fact to let the picture degrade in a downsampling-upsampling sequence (Figure 30). With an appropriate sampling rate the picture converts to a set of blobs which can then be separated using a simple connected component algorithm.

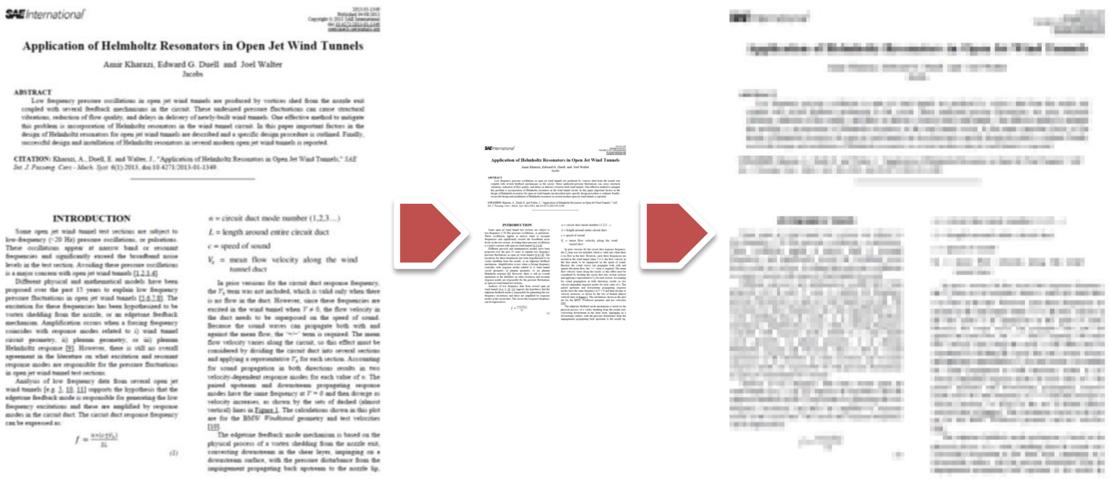
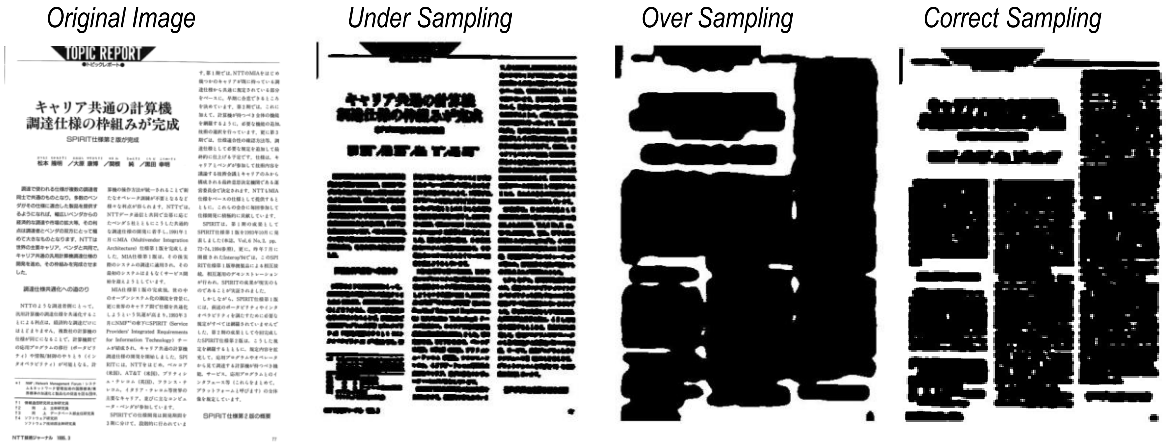


Figure 30: Illustration of image degrading in a sequence of downsampling-upsampling

3.3.2. Sampling Rate

Sampling rate or scaling factor is a number by which an image is resized. An optimized sampling rate is a requirement for our technique as too much sampling rate will smear blobs and in an extreme case the whole image becomes a single blob. On the other hand, not enough sampling will cause blob separation within a segment, leading to incorrect page segmentation. Figure 31 demonstrates effect of sampling rate in forming image blobs.



Original Image

Under Sampling

Over Sampling

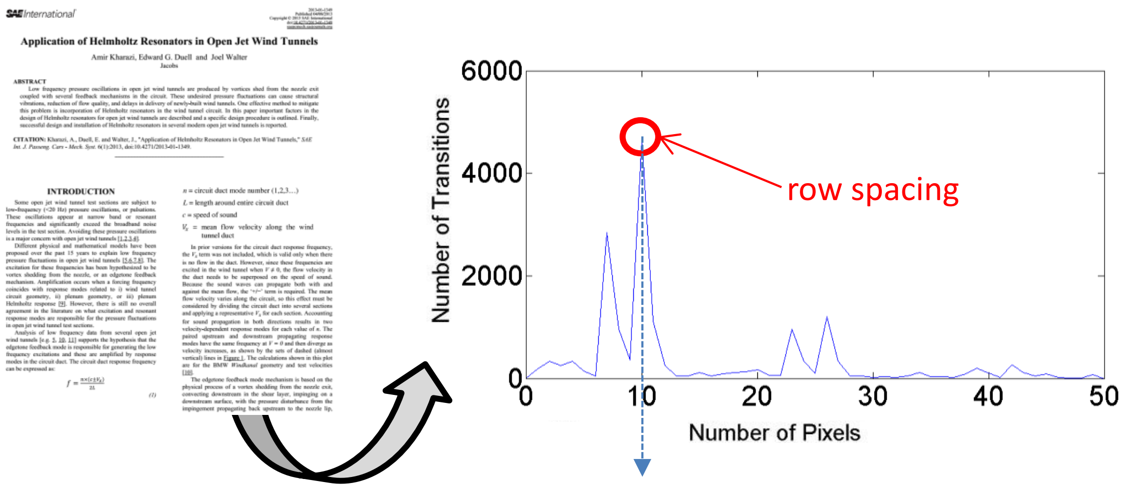
Correct Sampling

Figure 31: Segmentation of an image with different sampling rates

In our technique, sampling rate is calculated by a simple white space analysis of the image, as explained in the next section.

3.3.3. Determination of Sampling Rate for Page Segmentation

An optimal sampling rate is just enough sampling to let each segment form a continuous blob and prevent neighboring blobs to connect. The idea is to reduce or eliminate white spaces within a blob by estimating the distance between the rows of the text segments of an image. A sampling rate higher than the gap between the rows but lower than the space between the paragraphs results in correctly segmented blobs.



row spacing

Sampling rate=11

Figure 32: White space transition peaks of a vertical scan of an image

To calculate the sampling rate, we use a simple white gap counting algorithm. In this algorithm, after determining the outer boundaries of the image and filling in the connected components, we run a vertical white gap counting algorithm. The strongest peak in the spectrum corresponds to the gap between the rows. Our sampling rate is the number of pixels corresponding to the largest peak plus 1 pixel. By sampling at this rate, gaps between the words and lines are eliminated and blobs of texts will form. Figure 32 shows an example of white space transition peaks of a vertical scan of an image. For up-down languages, instead of vertical, a horizontal scan of the image is performed.

3.3.4. Work Flow of Our Resampling Method

Our algorithm starts with making the image gray scale. Then, the image is downsampled (zoomed out) with the scale factor determined from our white gap transition algorithm. Then, the resulting image is upsampled (zoomed in) with the same scaling factor. Finally, the image is binarized by converting all gray pixels to black. The resulting image is a set of blobs which are separated after a simple connected component analysis. Figure 33 illustrates the workflow of our resampling method.

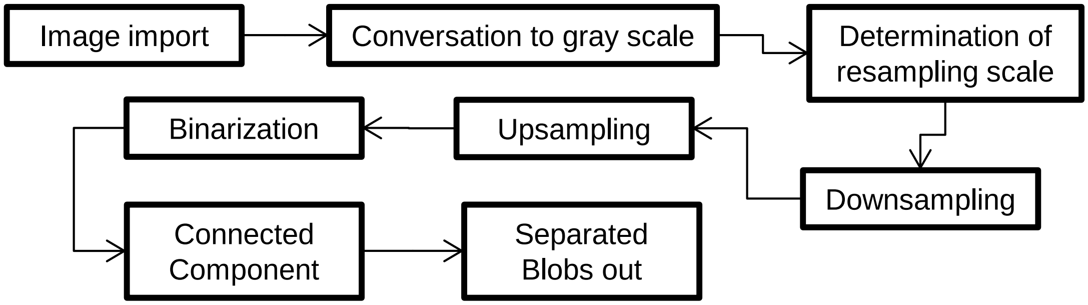


Figure 33: Workflow of our resampling algorithm.

Downsampling

In this stage of our algorithm, we downsample the image with the scaling factor determined from our white space gap transition algorithm. Downsampling involves computing a weighted average of the original pixels that overlap each new pixel. Many different resampling schemes are possible [73]. Most techniques work by computing new pixels as a weighted average of the surrounding pixels. The weights depend on the distance between the new pixel location and the neighboring pixels. The simplest methods consider only the immediate neighbors; more advanced methods examine more of the surround pixels to attempt to produce a more accurate result. For our algorithm, we tested several methods such as Nearest Neighbor (N-N), Bilinear, BiCubicle, and Lanczos. Except the N-N, other methods achieved similar segmentation results. For our evaluation tests we used the BiCubicle method. Bicubic resampling computes new pixels using cubic splines. When upsampling, this method operates on a 4 by 4 cell of pixels surrounding each new pixel location. This is the recommended resampling method for most images as it represents a good trade-off between accuracy and speed [74].

Upsampling

The resulting image from the downsampling stage is upsampled using the same factor used for downsampling. Upsampling involves interpolating between the existing pixels to obtain an estimate of their values at the new pixel locations. In this stage, we effectively replace the white gaps between the words and the rows with interpolated gray pixels. We used the same BiCubicle sampling method for upsampling.

Binarization

To eliminate or reduce the gap between words and text rows, all gray pixels are replaced with black. This operation forms a continuous blob for each text segment.

Connected Component

In this stage, the image consists of several continuous blobs which can be separated using a connected components algorithm. Separated blobs boundaries are then used to separate the original image.

3.4. Validations and Examples

We performed extensive validation tests of our segmentation technique. For our validation we used total of 230 images from the following sources:

- University of Maryland Tobacco800 image database
- www.mediateam.oulu.f-i image data base.
- Our database of 30 complex images

On the 200 random images from the online resources, our method achieved success rate of 98% (4 images not segmented correctly). The cause of the errors was the very low resolution (75 dpi) of some of the images, which led to incorrect calculation of the sampling factor. Since the images of the online resources were "typical", we created our own database of very complex images. Our algorithm perfectly detected the areas of different blobs for 29 out of 30 images. Moreover, due to a simple algorithm with no complex mathematical calculations, our algorithm works very efficiently, detecting images in less than 1 second on a typical PC.

Existing performance evaluations in the literature for different segmentation methods report much lower success rate for state-of-the-art page segmentation algorithms. For example, Mao [75] performed an empirical performance evaluation of five different page segmentation algorithms with three representative research algorithms and two well-known commercial products on a 978 image database. Table 1 shows the error rates reported by Mao.

Despite a more complex database, our algorithm outperforms algorithms reported by Mao both in error rate and processing time. Our algorithms' error rate, on a more challenging database, is about 2% with average processing time of less than a second on a personal computer.

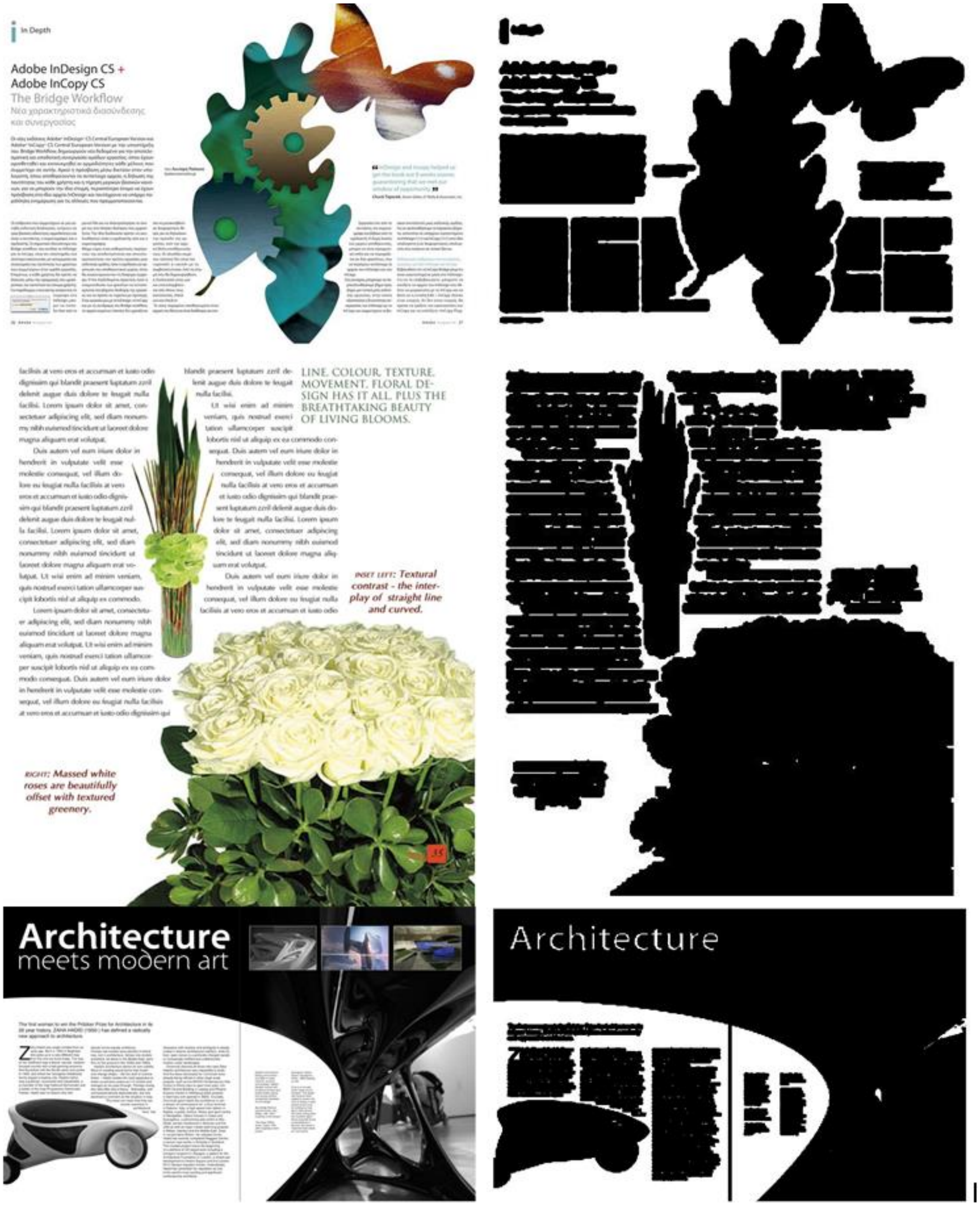

In Depth
Adobe InDesign CS +
Adobe InCopy CS
The Bridge Workflow
LINE, COLOUR, TEXTURE, MOVEMENT, FLORAL DESIGN HAS IT ALL, PLUS THE BREATHTAKING BEAUTY OF LIVING BLOOMS.
INSET LEFT: Textural contrast - the interplay of straight line and curved.
RIGHT: Massed white roses are beautifully offset with textured greenery.
Architecture
meets modern art
Architecture

Figure 34: Examples of complex page segmentation using our algorithm

Algorithm	Error Rate (%)	Average Processing Time (sec)
X-Y cut	14.7	2.1
Docstrum	5.0	4.1
Voronoi	4.7	2.8
Caere	6.0	2.0
ScanSoft	12.7	3.1

Table 4: Error rate and processing time for five page segmentation algorithms [75]

3.5. Text/Non-text Classification

Given the segmented document zones, correctly identifying the type of a zone is important for subsequent processes within any OCR system. One common method in separating text/non-texts blocks in document images including both text and non-text blocks is by representing the connected components of a block as feature vectors. Each feature vector consists of a set of measurements of pre-defined properties. Then a probabilistic model like decision tree [76] could be used in classifying each zone on the basis of its feature vector.

Most of the text components are smaller than non-text components. Therefore, size information is an important feature for classification. But size only information is not enough for classifying the big text and the small non-text components. One additional feature that could be used is that the shapes of non-text connected components are irregular and random. On the other hand, the shapes of text components are uniformly structured. Here are some commonly used feature vectors of bounding rectangles of the connected components for text/non-text classification: area of the connected components of the block, number of black pixels in the block in the original document image, mean horizontal black run lengths of the original image within the blocks, and the height and width of the bounding rectangles of the block [77].

For testing and evaluation purpose, the feature vector for each connected component of a test document image is extracted and then a class label is assigned to each connected component based on classification probabilities of text and non-text. For example, a decision tree classifier makes the assignment through a hierarchical decision procedure. The classification process can be described by means of a tree, in which at least one terminal node is associated with each class and non-terminal nodes represent various collections of mixed classes [76].

3.5.1. Proposed method for text/non-text classification

Since in our page segmentation method a page has already been segmented, the remaining task is to identify each segment as text or non-text. We use three criteria to decide if a segment is text or not.

A) Connected Component Area: One useful discriminator for texts in images is the variation in connected components area. Characters' size in most languages do not change much compared to variations in size of components in images. Therefore, we define a feature as

$$d = \frac{Stdev(connected\ components\ area)}{mean(connected\ components\ area)}$$

Feature d effectively calculates variations in the area of connected component of an image. We calculated this feature on segments extracted from images in our library of 500 images and found that this feature is less than 1.5 for texts segments and greater than 50 for image segments. Therefore, the threshold is not too sensitive and therefore we used $d < 2$ as our text identifier.

B) Stroke Width: Characters in most languages have similar stroke width or thickness throughout. Therefore, segments which exhibit too much variation in stroke width are identified as non-text [78].

$$\frac{stdev(strokewidths)}{mean(strokewidths)} > 5 \rightarrow NonText\ Segment$$

C) Finally, in our algorithm, we require a minimum of 10 connected components to identify a segment as text. Some images are just few connected components which may skew the decision making on connected component or stroke width criteria. Moreover, in most cases a text segment has at least 10 connected components.

Figure 35 shows segmentation and labeling of two complex document images using our segmentation method and text/non-text classification.

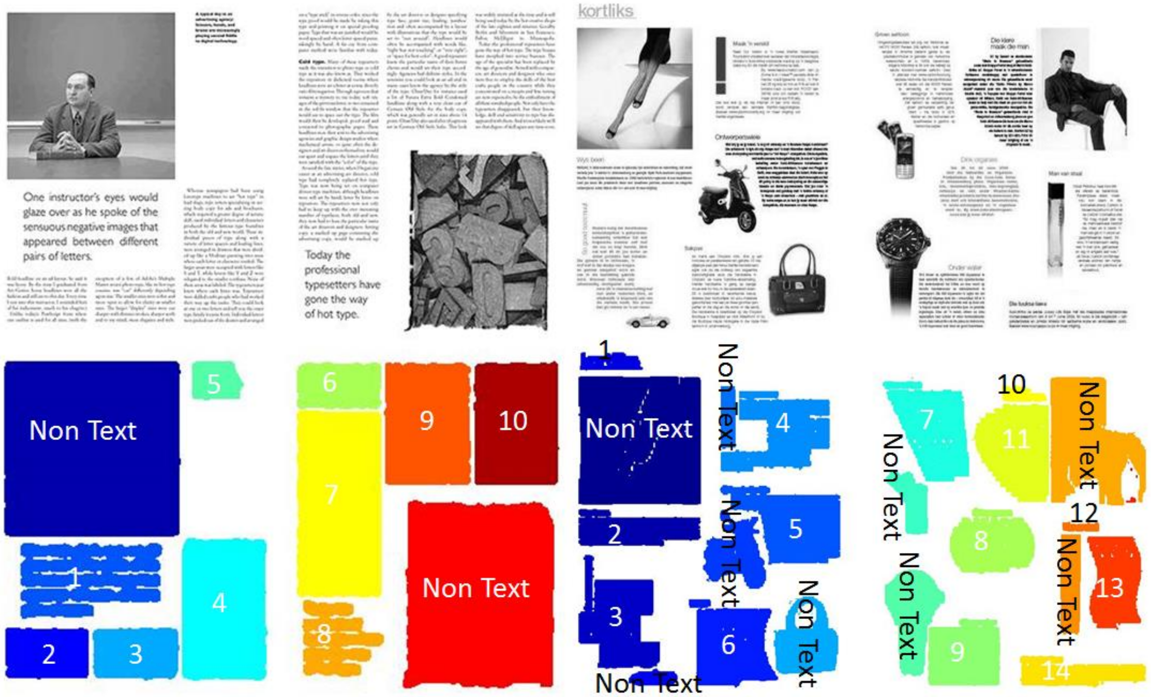

One instructor's eyes would glaze over as he spoke of the sensuous negative images that appeared between different pairs of letters.
Today the professional typesetters have gone the way of hot type.
kortliks
Non Text
5
1
4
2
3
6
9
10
7
Non Text
8
1
Non Text
Non Text
4
2
Non Text
5
3
Non Text
6
Non Text
10
Non Text
7
11
Non Text
12
8
Non Text
Non Text
13
9
14

Figure 35: Examples of complex page segmentation and labeling

3.6. Conclusions

In this chapter, we proposed a new segmentation technique based on image resampling method. Our method simply performs a downsampling (zoom-out) flowed by upsampling (zoom-in) with a calculated scaling factor. The sampling scale is calculated by utilizing a white gap transition algorithm and determining between the text rows gaps. This process converts the image to blobs of segments which are then separated using a simple connected component algorithm. Finally, each segment is identified as text or non-text using three criteria. Our extensive evaluations confirmed an efficient segmentation method capable of segmenting complex images.

CHAPTER 4

PRINTED PERSIAN/ARABIC TEXT RECOGNITION

4.1. Introduction

Arabic is spoken by over 200 million people in Middle East and Africa and belongs to the Afro-Asiatic language family. Persian is spoken by over 100 million people in Middle East and belongs to Indo-European language family. Although, Persian and Arabic are from different language families, they share similar scripts. Modern Persian is written using a modified variant of the Arabic alphabet. Although Persian alphabet was derived from Arabic, minor yet important differences exist in their alphabets and their styles of writing. For example, Persian script has four more alphabets than Arabic. Also, Persian has many exclusive fonts and cursive styles of writing.

Due to similarities of the two scripts, an OCR system could usually work on both scripts with little modifications. Since the author of this dissertation is a native Persian, and therefore more familiar with Persian script, she puts more emphasis on OCR of Persian script. Moreover, Arabic OCR has been published and introduced internationally; however, most of the research in Persian OCR has been presented only in Persian Journals and Iranian conferences.

4.2. Characteristics of Persian Texts

Some characteristics of Persian scripts make it especially challenging for segmentation and recognition:

1- Persian is Right-to-Left. However, most work in OCR is based on English and other languages that are written from Left-to-Right.

2- Persian script is very sensitive to dots. For example, two sets of four stand-alone Persian characters have the same basic shape but differ only in the absence or presence, location and number of dots (ج چ ح خ or ث ب ت پ).Five Persian characters when used as part of a connected sub-word or word differ only in location and number of dots (ب ت پ ن ث).

3- Existence of oblique strokes like ک گ creates problem of overlapping components.

4- Persian and Arabic are highly cursive and connected components make sub-words. In both of these scripts, the position of each letter in the word and its preceding or following letter in the same word are the factors that determine the shapes of the letter. In the Persian alphabet, similar to the Arabic alphabet, a letter can appear in four different forms: detached, initial, middle and final. All Persian letters with the exception of seven can be connected to other letters from both the right and the left sides. The seven exceptional characters can only be connected to other letters from the right side. Therefore, if any of those seven letters appear in the middle of a word, there will be a gap in connectivity [79, 80, 81].

5- Existence of some diacritics marks written above or below the letters (more common in Arabic than Persian). These diacritics indicate vowels, where consonant letters are connected together to make their pronunciation easier.

4.3. Segmentation of Persian Scripts

As explained in Chapter 1.3.1, one important preprocessing stage of OCR is segmentation, which directly affects the recognition success rate. For a Persian text, this step is specifically challenging due to special characteristics of Persian scripts, as explained in section 4.2. Much work has been done in segmentation, especially for

English to the point that segmentation in English is considered a solved problem [82]. However, many algorithms developed do not apply to Persian [83, 84]. Although many attempts have been made in segmentation of Persian texts [85, 86], based on our search, Persian text segmentation techniques that work for different font styles and sizes have relatively low success rates, consequently limiting the success rate of the recognition system. For example, Jelodar [79] used morphological hit/miss transformation for OCR of Persian text. Although, they claim a very high success rate, their experimental analysis is limited to specific font sizes and their method is very sensitive to noise. A more recent work from Broumandnia [87] successfully identifies texts from other regions of a Persian document such as pictures or tables. This method is based on pyramidal image structure which provides several resolutions of an image [88]. This method stays short of offering a solution in segmentation of the identified texts to characters or sub-words.

To avoid the challenges of the character segmentation, one unique approach is to segment a text into its component sub-words. Few recent works in offline Persian OCR have been focused in sub-word recognition. Nasrollahi [18] used Wavelet packet descriptors to recognize sub-words. They report a 97.6% recognition rate but using Wavelet packet descriptor is inherently slow for a large class of data and relatively complex to implement [89]. Fouladi [90] proposes a writer-dependent approach in which the system is trained to recognize the sub-words written by a particular writer. A contour alignment is the central part of their proposed algorithm.

In this chapter we propose a simple yet fast and accurate sub-word segmentation algorithm. Later in this chapter, we propose a hybrid feature extraction algorithm and 1st

Nearest Neighbor classification technique for successful recognition of Persian sub-words.

4.3.1. Proposed Algorithm

Figure 36 shows our proposed algorithm to segment a Persian word to its sub-words components.

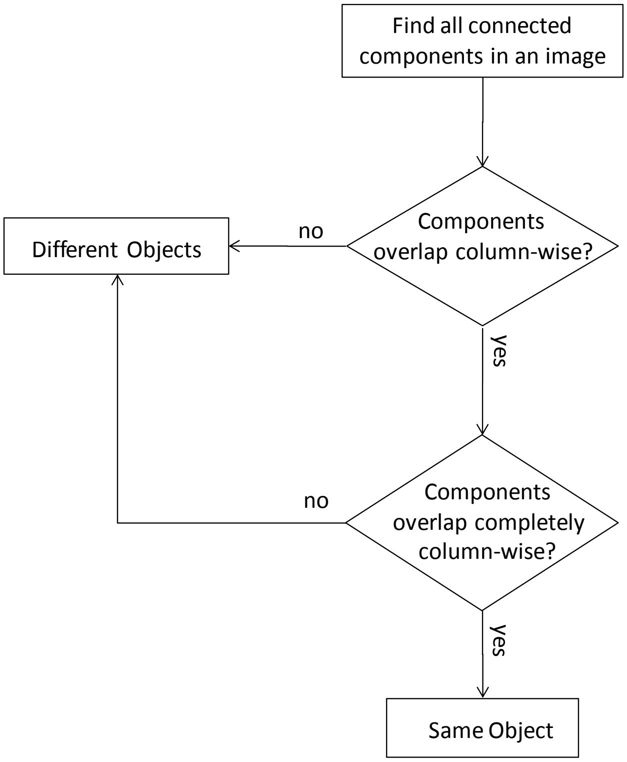
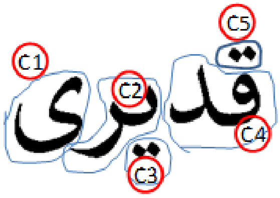

Find all connected components in an image
Components overlap column-wise?
no
Different Objects
yes
Components overlap completely column-wise?
no
yes
Same Object

Figure 36: Our proposed algorithm for segmentation of a word to its sub-words

In this algorithm, first, simply all connected components of an image of a word are found. For example, the below Persian word consists of five components.



Components of a word may or may not overlap column-wise, with the following three possibilities:

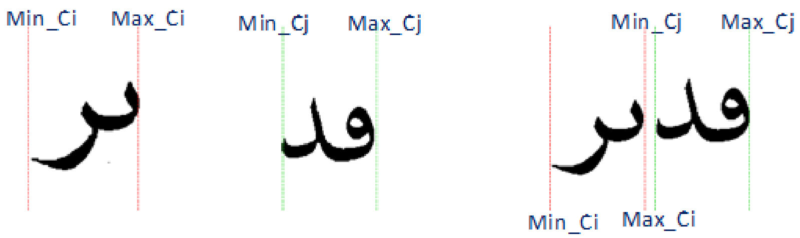
Possibility 1: Components do not overlap column-wise

In this case, the following condition must be satisfied:

$\mathrm{C_i maxColumn} < \mathrm{C_j minColumn}$

, *where* $C_i maxColumn$ *is the maximum column number of component i and* $C_j minColumn$ *is the minimum column number of the subsequent component j.*

The following example shows components #2 and #4 of the above Persian word are not overlapping:



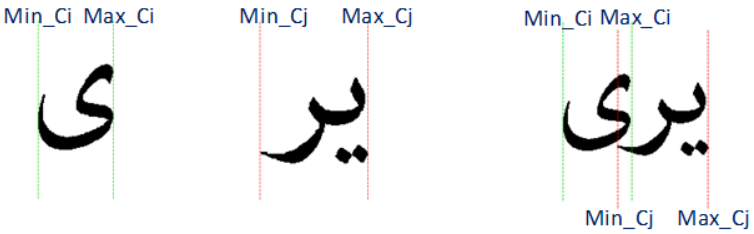
Decision: based on Figure 36, when two components do not overlap, they are considered as two different objects.

Possibility 2: Components overlap partially column-wise

For two components overlap partially, these conditions must be satisfied:

$$C_j minColumn < C_i maxColumn$$
$$C_j maxColumn > C_i minColumn$$
$$C_j maxColumn > C_i maxColumn$$

Here, components #4 and #5 are partially overlapping:



Decision: based on our proposed algorithm, partially overlapping components are considered as different objects.

Possibility 3: Components overlap completely column-wise

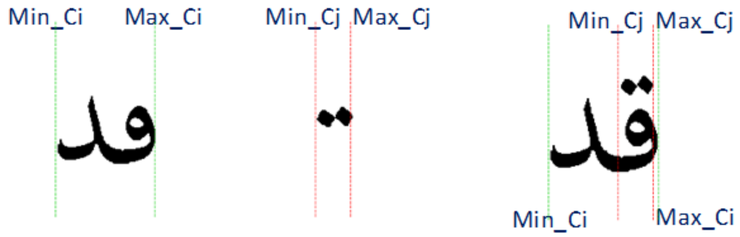
Two components are completely overlapping, if the following conditions are satisfied:

$$C_j minColumn < C_i maxColumn$$

$$C_j maxColumn > C_i minColumn$$

$$C_j maxColumn < C_i maxColumn$$

In this example, components #1 and #2 are completely overlapping:



Decision: this possibility is the only case that our proposed algorithm considers the two components as one object.

4.3.2. Label Settings

Based on the decisions made from our proposed algorithm (Figure 36), current labels of components may change. Since columns are scanned from left to right, the components are labeled from left to right, as shown in Figure 37. When our algorithm determines two components are the same, we change the label for one to match the other. Figure 37 shows how labeling for our example is changed.

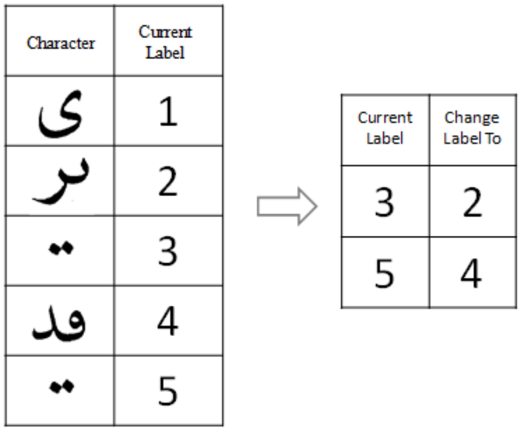


Figure 37: An example of labeling change when two components are the same

Finally, the algorithm puts the labels in sequential order.

4.3.3. Experimental Results

Our several tests of the proposed segmentation algorithm using four different fonts resulted in 100% segmentation success rate. Here are some examples of word segmentation using our proposed algorithm:

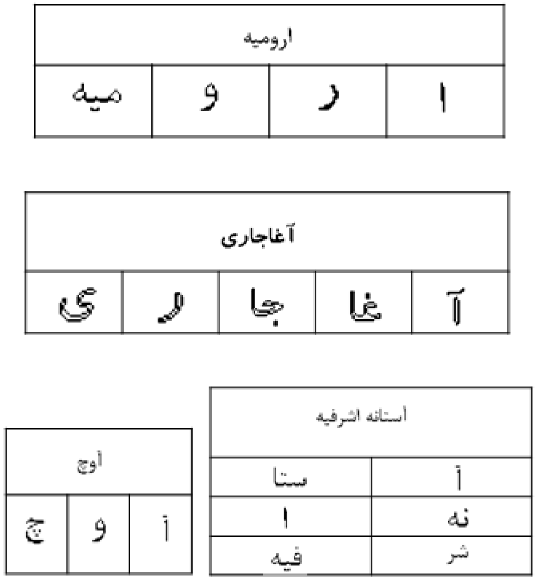


Figure 38: Examples of Persian word segmentation to its sub-words

Please note that this algorithm is especially designed to correctly segment words with overlapping components:

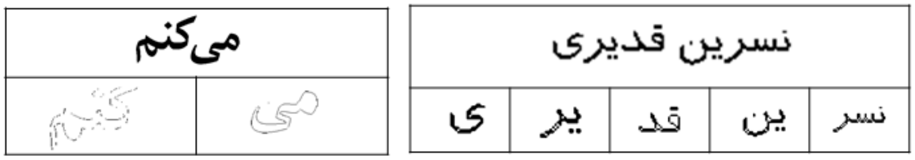


Figure 39: Examples of overlapping sub-words

Although, in the above left example, “می” overlaps with “ک “of “کنم” and in the right example, “ی” of “قدیری” overlaps with “ر”, the segmentation algorithm works accurately.

4.4. Algorithm for Segmentation of a Text Page

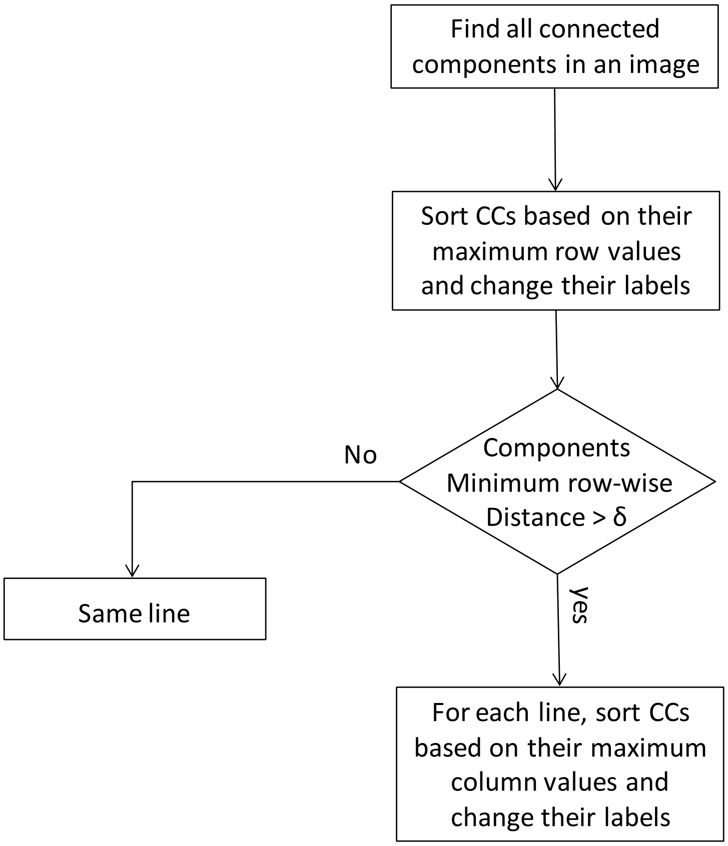

Find all connected components in an image
Sort CCs based on their maximum row values and change their labels
Components Minimum row-wise Distance > δ
No
Same line
yes
For each line, sort CCs based on their maximum column values and change their labels

Figure 40: Our proposed algorithm for segmentation of a text page

The goal of our recognition system is to digitalize a typed text. Therefore, we expand our proposed algorithm to segment a page of a Persian text to its sub-words, a necessary step before feature extraction.

Similar to the word segmentation, first, we find all components of a text. The initial labeling of these components is based on the columns order from top left corner to bottom right corner. Figure 41 is an example for initial labeling.

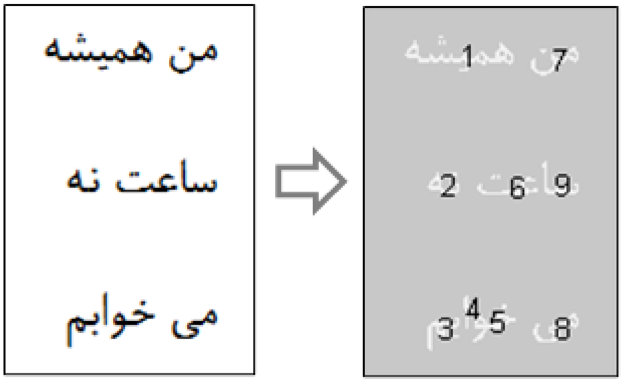


Figure 41: Initial labeling of a sample Persian text

The next step in our algorithm is sorting sub-words based on their maximum row values and changing their labels accordingly. After sorting, all the sub-words in the first line will be labeled before the second line and so on. Figure 42 shows an example for applying this step of our algorithm.

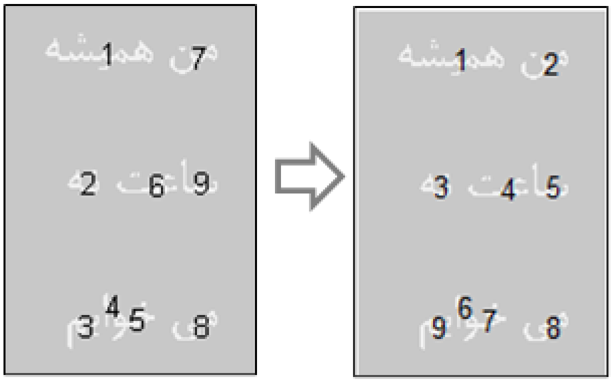


Figure 42: An example for sorting sub-words based on their maximum row values

Then, we need to classify components to their respective row number. To achieve this goal, we define a constant δ equal to the height of the longest component times a multiple κ:

$$\delta = \kappa \max\{\exists\ (C_i maxRow - C_i minRow)\}$$

δ is an indicator for the font size of the text and κ can be adjusted to define the distance between rows. Therefore, in our algorithm, we compare the difference between the maximum row and the minimum row of two consecutive components with δ to differentiate rows. Figure 43 shows how δ is calculated for different components.

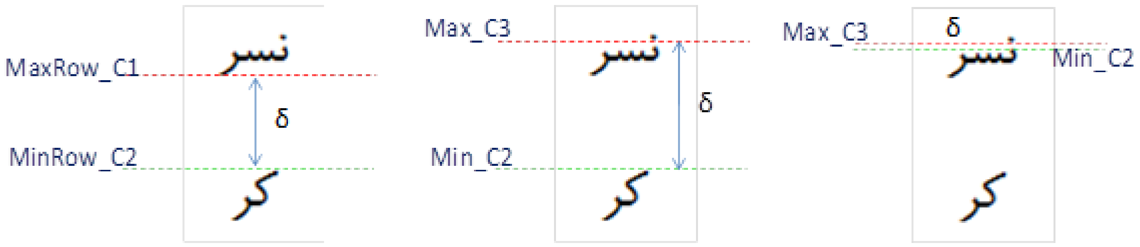


Figure 43: Examples for δ calculation for different components

Since Persian script is written from right to left, in the final step, we sort and re-label components based on their maximum column values. In this step, when a sub-word is located on the right side of another sub-word, its column number will be larger. Therefore, it will be labeled first. Figure 44 shows this step, in an example.

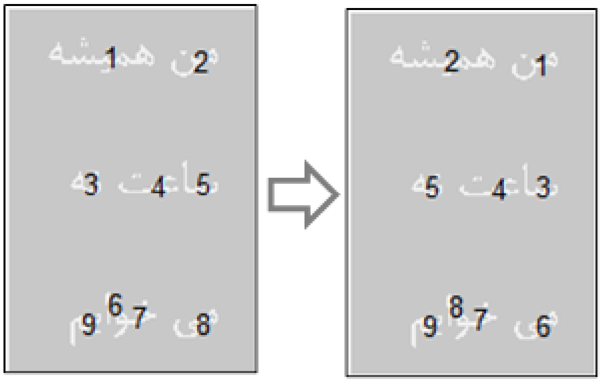


Figure 44: An example for sorting sub-words based on their maximum column values

After setting the labels, components are passed to our sub-word segmentation, as described in 4.3.1. We tested the above algorithm for several Persian texts with 4 different fonts and achieved error free segmentations for tested cases.

4.5. Feature extraction

Feature extraction step is applied on two sets of training and testing samples. For the training set, we created a library of 500 labeled Persian sub-words with 5 different samples for each sub-word.

In chapter 1.3.2, we discussed the feature extraction step of an OCR system and described three commonly used methods of Hu, Zernike, and DCT. In the training phase, for each labeled sub-word image, three feature extraction methods of Hu, Zernike, and DCT were applied and their respective feature vectors were calculated (Figure 45).

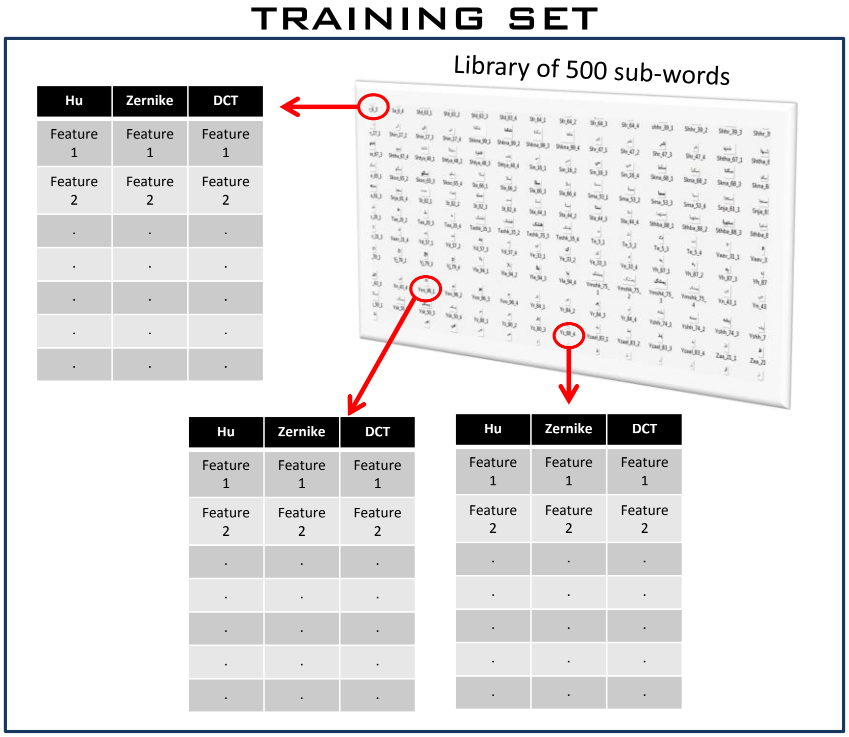
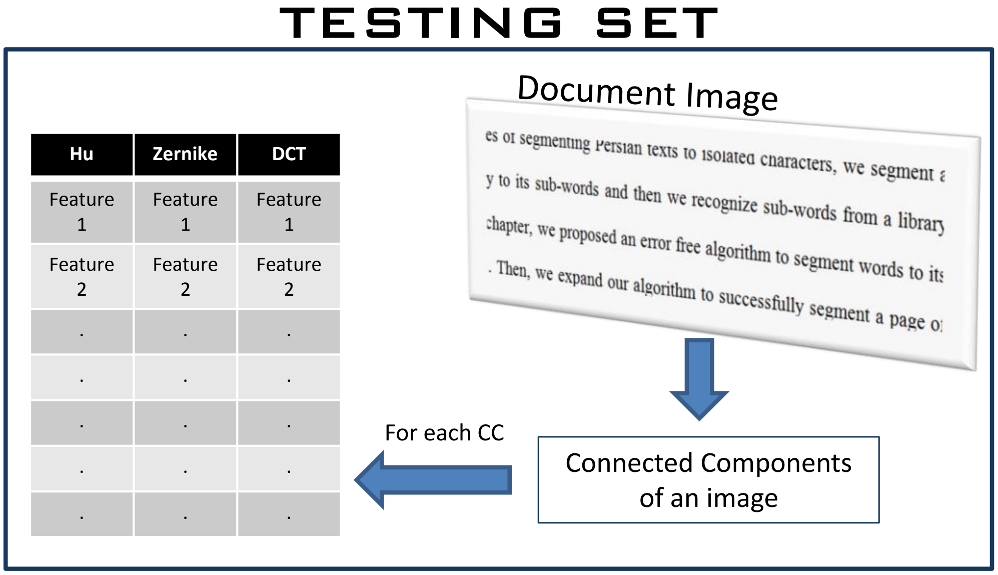


Figure 45: Extracted feature vectors of labeled sub-words training images

Similarly, in the testing phase, each feature extraction method was applied to every connected components of the document image and their respective feature vectors were stored to be used in classification phase (Figure 46).



es of segmenting Persian texts to isolated characters, we segment
y to its sub-words and then we recognize sub-words from a library
chapter, we proposed an error free algorithm to segment words to its
. Then, we expand our algorithm to successfully segment a page of

Figure 46: Extracted feature vectors of connected components of the document image

4.6. Classification

As explained in chapter 1.3.2, although learning-based classifiers usually offer more accuracy in performance compared to nonparametric classifiers, they are not suitable for large classes of data. Since in our sub-word recognition method a relatively large library of sub-words is maintained, the 1st Nearest-Neighbor (1-N-N) classifier is a practical choice. The 1-N-N classifier is one the oldest method known. The idea is very simple: to classify X find its closest neighbor among the training points (call it X’) and assign to X the label of X’. Here are the steps in applying 1-N-N method in the recognition phase of our OCR system for each feature extraction method:

- Calculate Euclidean distances of previously extracted feature vector of each testing connected component with respect to every feature vector of the training set.
- Find the closest distant neighbor of each testing connected component.
- Assign the nearest neighbor's label from the training set to the corresponding testing connected component.

4.7. Hybrid labeling decision

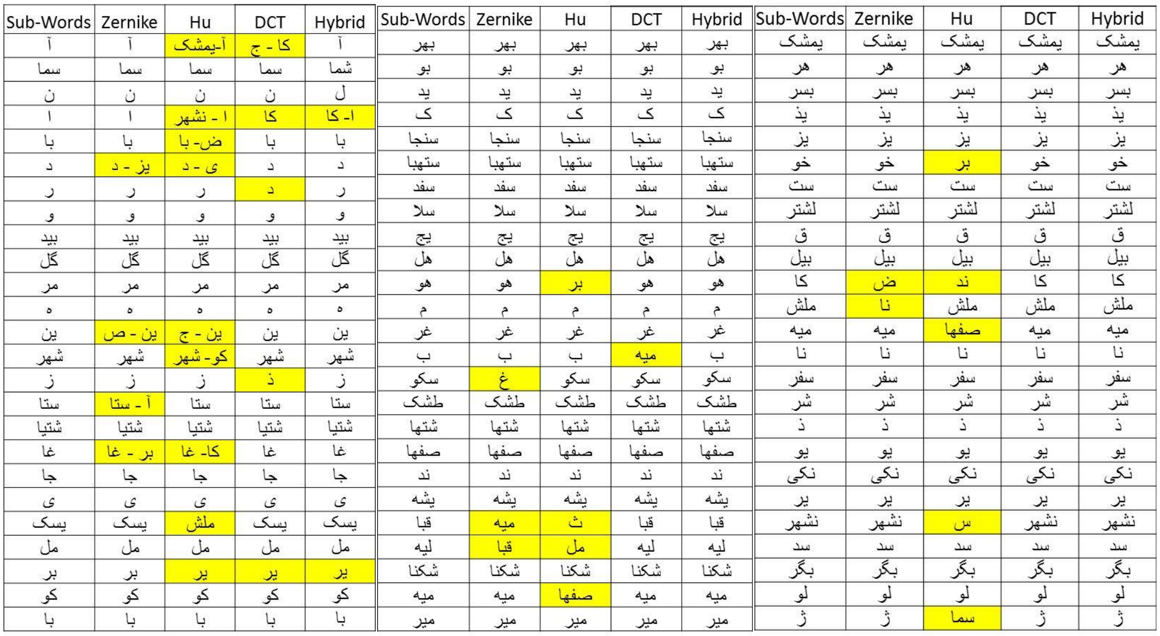
The classification results for each feature extraction method proved DCT as the most accurate method with 92.8% accuracy (accuracy is defined as percent of correct classifications). Hu had accuracy of 79.7% and Zernike had accuracy of 86%. Based on the results of this study, we implemented a hybrid labeling decision, based on the following observation of the results:

a) When DCT is incorrect the other two methods are correct.

b) When Hu and Zernike disagree, DCT gives the correct class-tag.

Therefore, in our labeling decision, if Hu and Zernike give identical results, that result is labeled; otherwise, DCT's label is applied.

By implementing this simple labeling method in our library of 500 sub-words, we achieved 97% accuracy (fifteen missed classifications in our library of 500 sub-words). Figure 47 shows a selection of sub-words in our library with the results of each feature extraction method and the assigned label by our hybrid method.



Sub-Words	Zernike	Hu	DCT	Hybrid	Sub-Words	Zernike	Hu	DCT	Hybrid	Sub-Words	Zernike	Hu	DCT	Hybrid
آ	آ	آ-یمشک	کا - ج	آ	بهر	بهر	بهر	بهر	بهر	یمشک	یمشک	یمشک	یمشک	یمشک
سما	سما	سما	سما	شما	بو	بو	بو	بو	بو	هر	هر	هر	هر	هر
ن	ن	ن	ن	ل	ید	ید	ید	ید	ید	بسر	بسر	بسر	بسر	بسر
ا	ا	ا - نشهر	کا	ا- کا	ک	ک	ک	ک	ک	یذ	یذ	یذ	یذ	یذ
با	با	ض- با	با	با	سنجا	سنجا	سنجا	سنجا	سنجا	یز	یز	یز	یز	یز
د	یز - د	ی - د	د	د	ستهبا	ستهبا	ستهبا	ستهبا	ستهبا	خو	خو	بر	خو	خو
ر	ر	ر	د	ر	سفد	سفد	سفد	سفد	سفد	ست	ست	ست	ست	ست
و	و	و	و	و	سلا	سلا	سلا	سلا	سلا	لشتر	لشتر	لشتر	لشتر	لشتر
بید	بید	بید	بید	بید	یج	یج	یج	یج	یج	ق	ق	ق	ق	ق
گل	گل	گل	گل	گل	هل	هل	هل	هل	هل	بیل	بیل	بیل	بیل	بیل
مر	مر	مر	مر	مر	هو	هو	بر	هو	هو	کا	ض	ند	کا	کا
ه	ه	ه	ه	ه	م	م	م	م	م	ملش	نا	ملش	ملش	ملش
ین	ین - ص	ین - ج	ین	ین	غر	غر	غر	غر	غر	میه	میه	صفها	میه	میه
شهر	شهر	کو- شهر	شهر	شهر	ب	ب	ب	میه	ب	نا	نا	نا	نا	نا
ز	ز	ز	ذ	ز	سکو	غ	سکو	سکو	سکو	سفر	سفر	سفر	سفر	سفر
ستا	آ - ستا	ستا	ستا	ستا	طشک	طشک	طشک	طشک	طشک	شر	شر	شر	شر	شر
شتیا	شتیا	شتیا	شتیا	شتیا	شتها	شتها	شتها	شتها	شتها	ذ	ذ	ذ	ذ	ذ
غا	بر - غا	کا- غا	غا	غا	صفها	صفها	صفها	صفها	صفها	یو	یو	یو	یو	یو
جا	جا	جا	جا	جا	ند	ند	ند	ند	ند	نکی	نکی	نکی	نکی	نکی
ی	ی	ی	ی	ی	یشه	یشه	یشه	یشه	یشه	یر	یر	یر	یر	یر
یسک	یسک	ملش	یسک	یسک	قبا	میه	ث	قبا	قبا	نشهر	نشهر	س	نشهر	نشهر
مل	مل	مل	مل	مل	لیه	قبا	مل	لیه	لیه	سد	سد	سد	سد	سد
بر	بر	یر	یر	یر	شکنا	شکنا	شکنا	شکنا	شکنا	بگر	بگر	بگر	بگر	بگر
کو	کو	کو	کو	کو	میه	میه	صفها	میه	میه	لو	لو	لو	لو	لو
با	با	با	با	با	میر	میر	میر	میر	میر	ژ	ژ	سما	ژ	ژ

Figure 47: A selection of our library of sub-words used in evaluation of feature extraction methods. Highlighted yellow cells indicate incorrect classification for each feature extraction method

4.8. Conclusions

To avoid the challenges of segmenting Persian texts to isolated characters, we segment a Persian document only to its sub-words and then we recognize sub-words from a library of sub-words. In this chapter, we proposed an error free algorithm to segment words to its component sub-words. Then, we expand our algorithm to successfully segment a page of a Persian text to sub-words, placing them in their respective rows.

In recognition phase of our Persian OCR system, three feature extraction methods of Hu, Zernike, and DCT were used in a simple yet accurate 1st nearest neighbor classification method. Finally, based on a hybrid method, connected components of the testing set were labeled. We tested the recognition phase of our OCR system on a library of 500 Persian sub-words and achieved 97% accuracy.

CHAPTER 5

CONCLUSION AND FUTURE WORK

5.1. Overview of the Dissertation and Conclusions

Optical Character Recognition is an active area of research in the field of pattern recognition. Despite several decades of research and development, even an elementary school student's reading skills outperforms the most advanced OCR systems. For example, while through a simple observation, human brain simply extracts the text from very complex documents, complicated algorithms are needed to replicate the very same task using computers.

In this dissertation, we made a brief introduction to the field of Optical Character Recognition. After discussing different branches of OCR, we narrowed down our focus to the recognition of typed offline documents. After a brief overview of history and evolution of different OCR systems, we explained in detail the classical flow of OCR systems. We explained that a typical OCR system is divided to two phases. The 1st phase is the preprocessing phase and the 2nd phase is the recognition phase. Then, we explained typical steps in preprocessing and recognition phase. For each step, we reviewed literature and gave our opinion on the best approach. According to the literature, quite diverse research directions are still being explored and standard procedures for building offline handwriting recognizers could not be established so far. However, some trends toward unified approaches can be identified, for example, the quite widely used Otsu method is mostly used for the binarization step or more recent efforts in Persian word segmentation focus on sub-word segmentation rather than isolated character segmentation.

In chapter 2, the important preprocessing step of skew detection is studied. After a literature review and brief description of four commonly used skew detection methods, we proposed a new technique in skew detection of documents using an axes-parallel bounding box. A comparison of this algorithm with the existing state-of-the-art skew angle algorithms proved a reliable and fast algorithm, outperforming the compared methods.

In chapter 3, we proposed a new segmentation technique based on image resampling method. Our method simply performs a downsampling (zoom-out) flowed by upsampling (zoom-in) with a calculated scaling factor. The sampling scale is calculated by utilizing a white gap transition algorithm and determining between the text rows gaps. This process converts the image to blobs of segments which are then separated using a simple connected component algorithm. Our extensive evaluations confirmed an efficient segmentation method capable of segmenting complex images.

In chapter 4, we focused on recognition of Perso-Arabic scripts with emphasis on Persian scripts. After reviewing the characteristics of Persian texts, we explained that due to the highly cursive nature of Persian texts, typical segmentation algorithms have poor success rates. Therefore, to avoid the challenges of segmenting Persian texts to isolated characters, we segment a Persian document only to its sub-words and then we recognize sub-words from a library of sub-words. Based on our 500 library of sub-words, our algorithm is capable for segmenting Persian words to sub-word without error and it is computationally very efficient. In the recognition phase, due to a relatively large library of sub-words, we use a hybrid scheme among Hu, Zernike, and DCT methods and use 1st nearest neighbor for classification. Our experimental tests on our library of 500 words result in 97% recognition rate.

In conclusion, the main contributions of this dissertation are in extensive research in two preprocessing steps of an OCR system: skew detection and page segmentation. In each step, we proposed state-of-the-art methods improving the performance of OCR systems. In addition, we focused on developing a Persian OCR system by segmenting words to sub-words and proposed new sub-word segmentation technique with 100% success rate. We introduced a hybrid scheme for feature extraction and chose a nonparametric classification technique to achieve 97% sub-words recognition accuracy.

5.2. Future Work

Although substantial progress has already been made toward the ultimate goal of automatic reading systems, still many challenges exists and OCR systems compared to human reading capabilities are primitive.

In the preprocessing phase of OCR systems in general, a single algorithm to deal with all sorts of imperfections in a document such as noise, skew, and page complexities needs to be developed. An OCR system needs to be so efficient that it works fast on small processors like mobile phones.

As for the Persian OCR, typical English or Chinese OCR systems do not work efficiently. Specific to Persian script methods, such as the ones introduced in this dissertation, are needed to improve Persian OCR systems and narrow the gap between the successes of Roman based script recognition systems and Persian OCR systems.

REFERENCES/BIBLIOGRAPHY

[1] J. Said, M. Cheriet and C. Suen, "Dynamic morphological preprocessing: a fast method for baseline extraction," *ICDAR*, pp. 8-12, 1996.

[2] L. Xu, E. Oja and P. Kultanen, "A new curve detection method: Randomized Hough Transform (RHT)," *Pattern Recognition Letters,* vol. 11, pp. 331-338, 1990.

[3] H. Cao, R. Prasad and P. Natarjan, "A stroke regeneration method for cleaning rule lines in handwritten document images," in *Proceeding of the international workshop on multilingual OCR, pp. 1-10,* New York, NY, 2009.

[4] T. Hoang, E. Smith and S. Tabbone, "Sparsity-based edge noise removal from bilevel graphical document images," *International Journal on Document Analysis and Recognition,* vol. 17, no. 2, pp. 161-179, 2014.

[5] W. Peerawit and A. Kawtrakul, "Marginal noise removal from document images using edge density," in *Proceeding of fourth information and computer engineering*, 2004.

[6] F. Shafait, J. van Beusekom and D. Keysers, "Document cleanup using page frame detection," *IJDAR,* vol. 11, no. 2, pp. 81-96, 2008.

[7] J. Sauvola and M. Pietikainen, "Adaptive document image binarization," *Pattern Recognition,* vol. 33, no. 2, pp. 225-236, 2000.

[8] R. Parvathi, N. Javanthi and et al., "Intuitionistic fuzzy approach to enhance text documents," in *Proceedings 3rd IEEE international conf. on intel. systems*, 2006.

[9] C. Leung, H. Chan and et al., "A new approach for image enhancement applied to low-contrast low-illumination documents," *Pattern Recognition Letters,* vol. 26, no. 6, pp. 769-778, 2005.

[10] S. Nomura, K. Yamanaka and et al., "Morphological preprocessing method to thresholding degraded word images," *Pattern Recognition Letters,* vol. 30, no. 8, pp. 729-744, 2009.

[11] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Itrans. Syst.,* vol. 9, no. 1, pp. 62-66, 1979.

[12] G. Lazzara and T. Géraud, "Efficient multiscale Sauvola's binarization," *International Journal on Document Analysis and Recognition,* vol. 17, no. 2, pp. 105-123, 2014.

[13] W. Niblack, An introduction to digital image processing, Englewood Cliffs: Prentice Hall, N. J., pp. 115-116, 1995.

[14] A. Benouareth, ,. A. Ennaji and M. Sellami, Semi, "Semi-continuous HMMs with explicit state duration for Arabic word modeling and recognition," *Pattern Recognition Letters,* vol. 29, pp. 1742-1752, 2008.

[15] A. Vinciarell, "Application of information retrieval techniques to single writer documents," *Pattern Recognition Letters,* vol. 22, pp. 1043-1050, 2001.

[16] S. Mozaffari, K. Faez and et al., "Lexican reduction using dots for off-line Farsi/Arabic handwritten word recognition," *Pattern Recognition Letters,* pp. 724-734, 2008.

[17] H. Al-Youse and S. S. Udpa, "Recognition of Arabic character," *IEEE Trans. Pattern. Anal. Mach. Intell.* , vol. 14, no. 8, pp. 853-857, 1992.

[18] S. Nasrollah and A. Ebrahimi , "Printed Persian Subword Recognition Using Wavelet Packet Descriptors," *Journal of Engineering (Open Access),* vol. Article ID 465469, 2013.

[19] C. T. Zahn and R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Transactions on Computers,* 1972.

[20] M. K. Hu, "Visual pattern recognition by moment invarients," *IRE Trans. info. theory,* vol. 8, pp. 179-187, 1962.

[21] C. H. Teh and R. T. Chin, "Invariant image recognition by Zernike moments," *IEEE transactions on pattern analysis and machine inteligence,* vol. 10, no. 4, 1988.

[22] Y. LeCun and B. Boser, "Handwritten digit recognition with a back-propagation

network," in *Advances in neural information processing (NIPS 89), pp. 396-404*, 1990.

[23] C. Burges, O. Matan and et al., "Shortest path segmentation: a method for training a neural network to recognize charac. strings," in *International Joint Conf. on Neural Networks (IJCNN)*, Baltimore, MD, 1992.

[24] A. Bosch, A. Zisserman and X. Munoz, "Representing shape with a spatial pyramid kernel," in *CIVR*, 2012.

[25] A. Opelt, M. Fussenegger and et al., "Weak hypotheses and boosting for generic object detection and recognition," in *ECCV*, 2004.

[26] A. Bosch, A. Zisserman and X. Munoz, "Image classification using random forests and ferns," in *ICCV*, 2007.

[27] L. Breiman and J. Friedman, Classification and Regression Trees, Belmont, California.: Wadsworth International, 1984.

[28] B. V. Dasarathy, "Nearest Neighbor (NN) norms: NN pattern classification techniques," in *IEEE Computer Society Press*, 1991.

[29] W. Postl, "Detection of linear oblique structures and skew scan in digitized documents," in *Proc. 8th international conference on pattern recognition*, 1986.

[30] H. Baird, "The skew angle of printed documents," in *Proc. SPSE 40th symposium hybrid imaging*, Rochester, NY, 1987.

[31] G. Ciardiello, G. Scafuro, M. T. Degrandi and e. al., "An experimental system for office document handling and text recognition," in *Proc. 9th international conference on pattern recognition*, 1988.

[32] Y. Ishitani, "Document skew detection based on local region complexity," in *Proc. 2nd international on document analysis and recognition*, Japan, 1993.

[33] D. S. Bloomberg, G. E. Kopec and L. Dasari, "Measuring document image skew and orientation," in *Document recognition II SPIE*, 1995.

[34] R. O. Duda and P. E. Hart, "Use of Hough transformation to detect lines and curves

in pictures," *Comm. ACM,* vol. 15, pp. 11-15, 1972.

[35] P. Hough, "Machine Analysis of bubble chamber pictures," in *2nd International conference on high-energy accelerators*, 1959.

[36] D. H. Billard, "Generalizing the Hough Transform to Detect Arbitrary Shapes," *Pattern recognition,* vol. 13, pp. 111-122, 1981.

[37] J. Illingworth and J. Kittler, "A survery of the Hough transforms," *Computer graphics and image processing,* vol. 44, pp. 87-116, 1988.

[38] S. C. Hinds, J. L. Fisher and D. P. D'Amoto, "A document skew detection method using run-length encoding and Hough transform," in *Proc. 10th Int'l conf. on pattern recognition*, New York, 1990.

[39] B. Yu and A. Jain, "A robust and fast skew detection algorithm for generic documents," *Pattern recognition,* vol. 29 (10), pp. 1599-1629, 1996.

[40] S. N. Srihari and V. Govindaraju, "Analysis of textual images using the Hough transform," *Machine vision and applications,* vol. 2, pp. 141-153, 1989.

[41] V. N. Manjunath, G. H. Kumar and P. Shivakumara, "Skew detection technique for binary document images based on Hough transform," *International journal technologies,* vol. 13 (3), pp. 194-200, 2006.

[42] D. X. Le, G. R. Thoma and H. Wechsler, "Automated page orientation and skew angle detection for binary document," vol. 27 (10), 1997.

[43] A. Hashizume, P.-S. Yeh and A. Rosenfeld, "A method of detecting the orientation of aligned components," *Pattern recognition letters,* vol. 4, pp. 125-132, 1986.

[44] G. S. Peake and T. N. Tan, "A general algorithm for document skew angle estimation," in *International conference on image processing*, 1997.

[45] R. Safabakhsh and S. Khadivi, "Document skew detection using minimum-area bounding rectangle," in *Information Technology: Coding and Computing,* 2000.

[46] P. Rudak, Y. Lee and T. Morgan, "Method for determining skew angle and location of a document in an over-scanned image". Patent US 7,027,666 B2, 2006.

[47] B. T. Avila and R. D. Lins, "Efficient removal of noisy borders from monochromatic documents," in *International conference on image analysis and recognition*, Portugal, 2004.

[48] D. X. Le, G. R. Thoma and H. Wechsler, "Automated borders detection and adaptive segmentation for binary document images," in *13th international conference on pattern recognition*, Austria, 1996.

[49] K. C. Fan, Y. K. Wang and T. R. Lay, "Marginal noise removal of document images," *Pattern Recognition*, vol. 35, 2002.

[50] H. S. Baird, S. E. Jones and S. Fortune, "Image Segmentation by Shape-Directed Covers," in *Proc. Int'l Conf. Pattern Recognition, pp. 820-825*, 1990.

[51] A. Antonacopoulos and D. Karatzas, "Semantics-Based Content Extraction in Typewritten Historical Documents," in *Eighth International Conference on Document Analysis and Recognition, pp. 48-53, 2005*.

[52] A. Antonacopoulos and R. T. Ritchings, "Representation and classification of complex-shaped printed regions using white tiles," in *Proceedings of the 3rd International Conference on Document Analysis and Recognition (ICDAR’95)*, Montreal, Canada, Vol. 2, pp. 1132-1135, 1995.

[53] F. Wahl, . K. Wong and R. Casey, "Block Segmentation and Text Extraction in Mixed Text/Image Documents," *Graphical Models and Image Processing,* vol. 20, pp. 375-390, 1982.

[54] G. Nagy and S. Seth, "Hierarchical representation of optically scanned documents," in *Proceedings of International Conference on Pattern Recognition, Vol 1, pp. 347–349*, 1984.

[55] H. S. Baird, "Background structure in document images," *Advances in Structural and Syntactic Pattern Recognition,* p. 253–269, 1992.

[56] F. Shafait, D. Keysers and T. M. Breuel, "Performance comparison of six algorithms for page segmentation," *7th IAPR Workshop on Document Analysis Systems,* p. 368–379, 2006.

[57] A. Jain and B. Yu, "Document Representation and Its Application to Page Decomposition," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 20, pp. 294-308, 1998.

[58] L. Fletcher and R. Kasturi, "A Robust Algorithm for Text String Separation from Mixed Text/Graphics Images," *IEEE Trans. Pattern Analysis and Machine Intelligence,* vol. 10, pp. 910-918, 1988.

[59] L. O'Gorman, "The document spectrum for page layout analysis," *IEEE Tras. on Page Analysis,* vol. 15, no. 11, pp. 1162-1173, 1993.

[60] M. Agrawal and D. Doermann, "Voronoi++: A Dynamic Page Segmentation Approach Based on Voronoi and Docstrum Features," in *10th International Conference on Document Analysis (ICDAR), pp. 1011-1015*, 2009.

[61] K. Kise, A. Sato and M. Iwata, "Segmentation of page images using the area voronoi diagram," *Computer Vision and Image Understanding,* vol. 70, no. 3, pp. 370-382, 1998.

[62] K. Chen, F. Yin and C. Liu, "Hybrid Page Segmentation with Efficient Whitespace Rectangles Extraction and Grouping," in *12th international Conf. on Document Analysis (ICDAR), pp. 958-962*, 2013.

[63] R. W. Smith, "Hybrid page layout analysis via tab-stop detection," in *Document Analysis and Recognition,10th International Conference on Document Analysis and Recognition(ICDAR), pp. 241–245*, 2009.

[64] T. Pavlidis and J. Zhou, "Page segmentation and classification," *CVGIP:Graphical Models and Image Processing,* vol. 54, pp. 484-496, 1992.

[65] A. Antonacopoulos, C. Clausner and . C. Papadopoulos, "Historical document layout analysis competition," in *Proc. 11th ICDAR, pp.1516-1520*, 2011.

[66] Y. Wang, X. Ding and C. Liu, "Topic Language Model Adaption for Recognition of Homologous Offline Handwritten Chinese Text Image," *IEEE Signal Processing Letters,* vol. 21, no. 5, p. 550–553, 2014.

[67] S. Bukhari, F. Shafait and T. Breuel , "Coupled snakelets for curled text-line

segmentation from warped document images," *International Journal on Document Analysis and Recognition (IJDAR),* vol. 16, no. 1, pp. 33-53, 2013.

[68] A. Winder, T. Andersen and E. H. B. Smith, "Extending Page Segmentation Algorithms for Mixed-Layout Document Processing," in *2011 Internation Conf. on Document Analysis (ICDAR), pp. 1245-1249*, 2011.

[69] F. Zirari and A. Ennaji, "A simple text/graphic separation method for document image segmentation," in *2013 ACS International Conf. on Computer Systems (AICCSA), pp. 1-4*, 2013.

[70] D. Deryagin, "Unified Performance Evaluation for OCR Zoning: Calculating Page Segmentation's Score, That Includes Text Zones, Tables and Non-text Objects," in *12th Internation Conf. on Document Analysis (ICDAR), pp. 953-957*, 2013.

[71] A. Gordo, M. R. usinol and e. al., "Document Classification and Page Stream Segmentation for Digital Mailroom Applications," in *12th Internation Conf. on Document Analysis (ICDAR), pp. 621-625*, 2013.

[72] E. Kostopoulou, E. Zacharia and D. Maroulis, "Accurate segmentation of 2D-PAGE images," in *2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO) Signal Processing Conference (EUSIPCO), pp. 2258-2262*, 2012.

[73] J. Parker, R. Kenyon and D. Troxel, "Comparison of Interpolating Methods for Image Resampling," *IEEE Trans Med Imaging,* vol. 2, no. 1, pp. 31-39, 1983.

[74] J. Wakerly, Digital Design: Principles and Practices, 3rd ed., Prentice Hall, New Jersey, 2001.

[75] S. Mao, A. Rosenfeld and T. Kanungo, "Document structure analysis algorithms: a literature survey," *International Society for Optics and Photonics Electronic Imaging.,* pp. 197-207, 2003.

[76] R. Haralick and L. Shapiro, Computer and Robot Vision, Reading, MA: Addison Wesley, 1997.

[77] R. M. Haralick, "Document Image Understanding: Geometric and Logical Layout," in *Proc. of the Conference on Computer Vision and Pattern Recognition*, 1994.

[78] C. Huizhong and et al., "Robust Text Detection in Natural Images with Edge-Enhanced Maximally Stable Extremal Regions," in *18th IEEE International Conference on Image Processing (ICIP)*, 2011.

[79] M. Jelodar and e. al., "A Persian OCR System using Morphological Operators," *World Academy of Science, Engineering and Technology*, no. 4, 2005.

[80] J. Sadri, S. Izadi and et al., "State-of-the-art in Farsi scripts recognition," in *9th International Symposium on Signal Processing and Its Applications, 2007*, Sharjah, 2007.

[81] N. Tagougui, M. Kherallah and A. Alimi, "Online Arabic handwriting recognition: a survey," *International Journal on Document Analysis and Recognition*, vol. 16, no. 3, pp. 209-226, 2013.

[82] A. Mao and M. Kanungo, "A Methodology for Empirical Performance Evaluation of Page Segmentation Algorithms," *IEEE Transaction on Pattern Analysis and Machine Intelligence*, vol. 23, no. 3, pp. 242-256, 2001.

[83] M. Shridhar, J. CAO and M. Ahmadi, "Recognition of Handwritten Numerals with Multiple Feature and Multistage Classifier," *Pattern Recognition Society*, 1995.

[84] Q. Trier, A. Jain and T. Taxt, "Feature Extraction Methods for Character Recognition–A SURVEY," *Pattern Recognition*, vol. 29, no. 4, pp. 641-662, 1996.

[85] A. Johnston, *Classifying Persian Characters with Artificial Neural Networks and Inverted Complex Zernike Moments*, PhD Thesis, Imperial College of London, 2005.

[86] G. Amayeh, S. Kasaei and A.R. Tavakkoli, "A Modified Algorithm to Obtain Translation, Rotation and Scale Invariant. Zernike Moment Shape Descriptors," in *International Workshop on Computer Vision, IPM*, Tehran, 2004.

[87] M. A. Broumandnia, "Persian/Arabic Document Segmentation Using Hybrid Methods," *International Journal of Advanced Computer Science*, vol. 1, no. 2, pp. 65-71, 2011.

[88] R. Gonzalez and R. Woods, Digital Image Processing, Second Edition, Prentice-Hall, 2002.

[89] X. Qian and et al., "Object Categorization Using Hierarchical Wavelet Packet Texture Descriptors," in *11th IEEE International Symposium on Multimedia*, 2009.

[90] K. Fouladi, B. Araabi and E. Kabir, "A fast and accurate contour-based method for writer-dependent offline handwritten Farsi/Arabic subwords recognition," *International Journal on Document Analysis and Recognition,* vol. 17, no. 2, pp. 181-203, 2014.

[91] M. Badreldin and S. A., "High Accuracy Character Recognition Algorithm Using Fourier and Topological Descriptors," *Pattern Recognition,* vol. 17, 1984.

[92] L. O. Gorman, "The document spectrum for page layout analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* vol. 15, no. 11, p. 1162–1173, 1993.

[93] N. Singh, N. Bhatia and A. Kaur, "Hough transform based fast skew detection and accurate skew correction methods," *Pattern Recognition,* vol. 41, no. 2, 2008.

.

APPENDICES

Appendix A. Projection Profile skew detection MATLAB source code

```
function my_prjprofile(BImg,i)
tic
Theta = -90:89;
horizproj_var = cell(1,1);
for T = -90:89
    RImg = imrotate(BImg,T,'bilinear');
    horizproj= sum(RImg == 0,2);
    horizproj_var{1} = cat(1, horizproj_var{1},var(horizproj));
end

 [~,Locs] = findpeaks(horizproj_var{1},'NPEAKS',1);
Rotation_Angle = Theta(Locs)
if( Rotation_Angle ~= 0)
    Img = imrotate(BImg,Rotation_Angle,'bilinear');
else
    Img = BImg;
end
toc
```

Appendix B. Hough transform skew detection MATLAB source code

```
function [H, P , theta, Rotation_Angle] = my_hough(Img,i)
 %find edges
 tic
 sigma=1;
 [g3, t3]=edge(Img, 'canny', [0.04 0.10], sigma);
 %Do the Hough transform
 [H t r] = hough(g3,'RhoResolution',0.5,'Theta',-90:0.5:89.5);

 %Display the transform in such a way that we can draw points on it
later...
  imshow(H, [], 'XData', t, 'YData', r );

 %Add axis labels to the picture
 xlabel('\theta'), ylabel('\rho');
 axis on, axis normal;
 P  = houghpeaks(H,5);
 %draw peaks over Hough transform
 x = t(P(:,2)); y = r(P(:,1));
 hold on;
 plot(x,y,'s','color','white');
 % Find lines and plot them
 lines = houghlines(Img,t,r,P,'FillGap',5,'MinLength',7);
 figure, imshow(Img), hold on
 theta = cell(1,1);
 for k = 1:length(lines)
    xy = [lines(k).point1; lines(k).point2];
    plot(xy(:,1),xy(:,2),'LineWidth',2,'Color','green');

    % Plot beginnings and ends of lines
    plot(xy(1,1),xy(1,2),'x','LineWidth',2,'Color','yellow');
    plot(xy(2,1),xy(2,2),'x','LineWidth',2,'Color','red');
    theta{1} = cat(1,theta{1},lines(k).theta);
 end
Angle_Range = unique(theta{1}(:));
Accum_Array = hist( theta{1}(:), numel(Angle_Range) );
if size(Accum_Array,2)> 1
    Loc = find(Accum_Array == max(Accum_Array))
     Rotation_Angle = Angle_Range(Loc);
else
    Rotation_Angle = Angle_Range(1);
end
toc
```

Appendix C. Nearest-Neighbor skew detection MATLAB source code

```
function   [P1,   P2   ,   theta,   Locs,Angle_Range,Rotation_Angle]   =
my_nearest_neighbor(BImg,i)
tic
CC = bwconncomp(BImg);
S = regionprops(CC,'Centroid');
centroids = cat(1, S.Centroid); %% centroids(col,row)
figure(1),imshow(BImg)
hold on
plot(centroids(:,1), centroids(:,2), 'b*')
hold off
[idx,                                  dist]                          =
knnsearch(centroids,centroids,'K',2,'dist','euclidean');

% idx(:,1) includes each centroid and idx(:,2) includes its nearest
neighbor
P1 = centroids(idx(:,1),:);
P2 = centroids(idx(:,2),:);

% Find angle between each centroids and its nearest neighbor
for j = 1: size(P1,1)
    if P1(j,1) > P2(j,1)
        temp = P1(j,:);
        P1(j,:) = P2(j,:);
        P2(j,:) = temp;
    end
end
dp = bsxfun(@minus, P2, P1);
theta = atan2(dp(:,2), dp(:,1));
theta = round((theta*180)./pi);
Angle_Range = unique(theta);
figure(2)
hist( theta, numel(Angle_Range))
Accum_Array = hist( theta, numel(Angle_Range) );
[~, Locs] = max(Accum_Array);
Rotation_Angle = Angle_Range(Locs);
% [~,Locs] = findpeaks(Accum_Array,'NPEAKS',2);
% Rotation_Angle = mean(Angle_Range(Locs));
toc
if( Rotation_Angle ~= 0)
    Img = imrotate(BImg,Rotation_Angle,'bilinear');
else
    Img = BImg;
end
```

Appendix D. Fourier method skew detection MATLAB source code

```
function [T,Real_Angel,Rotation_Angle,val] = my_FFT(Img,i)
figure,imshow(Img)

tStart = tic;
T = cell(1,2);
Real_Angel = cell(1,1);
j = 1;
[m, n] = size(Img);
M = floor(m/2)+1;
N = floor(n/2)+1;


I1 = Img(1:M,N:n);
j = j+1;
figure(j), imshow(I1)
[S1, j,rq,cq] = my_FFT_Spec(I1,j);
j = j+1;
[T,Real_Angel,j] = Accum_Array(S1,j,T,Real_Angel,rq,cq);

I2 = Img(1:M,1:N);
j = j+1;
figure(j), imshow(I2)
[S2, j,rq,cq] = my_FFT_Spec(I2,j);
j = j+1;
[T,Real_Angel,j] = Accum_Array(S2,j,T,Real_Angel,rq,cq);

I3 = Img(M:m,1:N);
j = j+1;
figure(j), imshow(I3)
[S3, j,rq,cq] = my_FFT_Spec(I3,j);
j = j+1;
[T,Real_Angel,j] = Accum_Array(S3,j,T,Real_Angel,rq,cq);

I4 = Img(M:m,N:n);
j = j+1;
figure(j), imshow(I4)
[S4, j,rq,cq] = my_FFT_Spec(I4,j);
j = j+1;
[T,Real_Angel,j] = Accum_Array(S4,j,T,Real_Angel,rq,cq);
% [tmp,ind] = sort(T{1},'ascend');
% T{1} = tmp;
% T{2} = T{2}(ind);
loc = [];
if size(T{2},1) > 2
    [~, loc] = findpeaks(T{2},'npeaks',1);
end
if isempty(loc)
    [~,loc] = max(T{2});
end
% [~,loc] = max(T{2});
[rx, ~] = find(T{1}(loc)- 5 < Real_Angel{1} & Real_Angel{1} <
T{1}(loc)+ 5);
[val ~] = sort(Real_Angel{1}(rx));
val
```

```
angle = median(val);
figure(j)
hold on;
bar(T{1},T{2})
tElapsed = toc(tStart)
if  angle >= 45
Rotation_Angle = -90 + angle;
else
Rotation_Angle = angle;
end
```

```
function [S,j,rq,cq] = my_FFT_Spec(I,j)

F = fft2(I);
Fc = fftshift(F);
S = log(1 + abs(Fc));
% j = j+1;
% figure(j), imshow(S, [])
title('Fig.2: Spectrum of the Input Image')
[M,N] = size(S);
rq = floor(M/2)+1;
cq = floor(N/2)+1;
S(rq-13:rq+13,cq-13:cq+13)= 0;
% j = j+1;
% figure(j), imshow(S, [])
mmin = min(S(:));
mmax = max(S(:));
S = (S-mmin) ./ (mmax-mmin);
```

```
function [T, RA,j] = Accum_Array(S,j,T,RA,rq,cq)
[r,c,v] = FastPeakFind(S, 'NPEAKS',10);

figure(j),imagesc(S); hold on
plot(c(1:end),r(1:end),'g+')
j = j+1;
t = atan2(c(1:2:end)-cq, r(1:2:end)-rq);
% % t = atan2(r(1:2:end),c(1:2:end));
t = (t*180)./pi;
theta = round(t);
v = v(1:2:end);
for k = 1: size(theta,1)
    RA{1} = cat(1,RA{1},t(k));
    [rx, cy] = find(T{1}(:) == theta(k));
    if( isempty(rx))
        T{1} = cat(1,T{1},theta(k));
        T{2} = cat(1,T{2},v(k));
    else
        T{2}(rx)= T{2}(rx)+v(k);
    end
end
```

Appendix E. Our skew detection method MATLAB source code

```
clc;
clear all;
close all;
imtool close all; % Close all imtool figures.

cd('C:\Personal\Windsor\Image processing code')
%cd('../')
samplefolder = './Whole_page/';
file_l = dir(samplefolder);
Step = pi/180;
Theta = [];
Area = zeros(size(Theta));
%// the first two in filelist are . and ..

 for i= 15:15% size(file_l,1)
    %// filelist is not a folder
    if file_l(i).isdir ~= true
        fname = file_l(i).name
        Org_Img = imread([samplefolder fname]);
        if length(size(Org_Img))>2 % checking if rgb image;
             Img =rgb2gray(Org_Img);
             Img=im2bw(Img,graythresh(Img));
        else
            Img = Org_Img;
        end

        if  Img(1,1) ~= 0
            Img = ~Img;
        end

        figure,imshow(Img)
        cd('C:\Personal\Windsor\Image processing code\Rotated_results')
        fn = sprintf('Original_Image_%s.png',num2str(i-2,'%02i'));
        imwrite(~Img,fn,'png');
        cd('C:\Personal\Windsor\Image processing code')

        Final_Img = zeros(size(Img));
        [n,m] = size(Img);
        Pivot(1)= (n+1)/2;
        Pivot(2)= (m+1)/2;
        PP = Find_perimeter_points(Img);
        [Corners Area_Org] = Find_rec_area(PP);
        crop                            =                            Img(
min(Corners{1}(:)):max(Corners{1}(:)),min(Corners{2}(:)):max(Corners{2}
(:)));
        figure,imshow(~crop)
        PP{1}(:)  = PP{1}(:)-Pivot(1);
        PP{2}(:)  = PP{2}(:)-Pivot(2);
        Rotated_PP = Rotate_Perimeter_Points(PP,Step);
        Rotated_PP{1}(:)  = Rotated_PP{1}(:)+Pivot(1);
        Rotated_PP{2}(:)  = Rotated_PP{2}(:)+Pivot(2);
        [x Area_Up] = Find_rec_area(Rotated_PP);
        Rotated_PP = Rotate_Perimeter_Points(PP,-1*Step);
        Rotated_PP{1}(:)  = Rotated_PP{1}(:)+Pivot(1);
        Rotated_PP{2}(:) = Rotated_PP{2}(:)+Pivot(2);
        [x Area_Down] = Find_rec_area(Rotated_PP);

        if Area_Up < Area_Org
            Turn_dir = 1;
        else if Area_Down < Area_Org
                Turn_dir = -1;
            else
                Turn_dir = 0;
                cd('C:\Personal\Windsor\Image                 processing
code\Rotated_results\My_SkewFun')
                fn      =      sprintf('Original_Image_%s.png',num2str(i-
2,'%02i'));
                imwrite(~Img,fn,'png');
            end
        end
        Rotation_Step = 4*pi/180;
        R = 0;
        tic
        if Turn_dir ~= 0
            Min_Area = Area_Org;
            R = R + Turn_dir*Rotation_Step;
            RPP = Rotate_Perimeter_Points(PP,R);
            RPP{1}(:) = RPP{1}(:)+Pivot(1);
            RPP{2}(:) = RPP{2}(:)+Pivot(2);
            [Corners Area_maxr] = Find_rec_area(RPP);
            while Min_Area > Area_maxr
                Min_Area = Area_maxr;
                R = R + Turn_dir*Rotation_Step;
                RPP = Rotate_Perimeter_Points(PP,R);
                RPP{1}(:) = RPP{1}(:)+Pivot(1);
                RPP{2}(:) = RPP{2}(:)+Pivot(2);
                [Corners Area_maxr] = Find_rec_area(RPP);
            end
            if R*180/pi~=0
                R = R-Turn_dir*Rotation_Step;
            end
            Rotation_Step = Rotation_Step/2;
            while Rotation_Step >= (0.25*pi/180) &&  Turn_dir ~= 0
                [R,Min_Area,Rotation_Step]                          =
Find_Min_Area2(R,Rotation_Step,PP, Min_Area,Pivot);
            end
        end
        t = R*180/pi
        Img1 = imrotate(Img,R*180/pi,'bilinear');
        PP = Find_perimeter_points(Img1);
        Flag = Smearing_Func(Img1,PP);
        if Flag == 0
            Img1 = imrotate(Img1,-1*Turn_dir*90,'bilinear'); %Img1
            t = -1*Turn_dir*90 + R*180/pi;
        end
        toc
        t
        figure,imshow(Img1)
       cd('C:\Personal\Windsor\Image                          processing
code\Rotated_results\My_SkewFun')
```

```
       fn             =           sprintf('Image_%s_%s_%s.png',num2str(i-
2,'%02i'),num2str(t,'%02i'),num2str(toc,'%04f'));
       imwrite(~Img1,fn,'png');
        cd('C:\Personal\Windsor\Image processing code')
    end

 end
cd('C:\Personal\Windsor\Image processing code')
```

```
function PPoints = Find_perimeter_points(Img)
% Receives an image and finds only the surrounding pixel's coordinates
PPoints = cell(1,2);

for r = 1:size(Img,1)
    c = find(Img(r,:),1,'first');
    if size(c,2)~= 0
        PPoints{1} = cat(1,PPoints{1},r);
        PPoints{2} = cat(1,PPoints{2},c);
    end
    c = find(Img(r,:),1,'last');
    if size(c,2)~= 0
        PPoints{1} = cat(1,PPoints{1},r);
        PPoints{2} = cat(1,PPoints{2},c);
    end
end
```

```
function [CP Area] = Find_rec_area(P)
% Having perimiter points of a text area find the corner points of
% the surrounding rectangle and the coverage area
PP = cell(1,2);
xmin = min(P{1}(:));
ymin = min(P{2}(:));
if xmin < 0
    PP{1}= P{1}(:)-xmin;
else
    PP{1} = P{1}(:);
end
if ymin < 0
    PP{2} = P{2}(:)-ymin;
else
    PP{2} = P{2}(:);
end
% PP{1}(:)
% PP{2}(:)

CP = cell(1,2);
[val ind1]= min(PP{1});
ind2 = find(PP{2}(ind1)== min(PP{2}(ind1)),1,'first');
CP{1} = cat(1,CP{1},PP{1}(ind1(1,ind2)));
CP{2} = cat(1,CP{2},PP{2}(ind1(1,ind2))) ;      % Point A

[val ind1]= max(PP{1});
ind2 = find(PP{2}(ind1)== max(PP{2}(ind1)),1,'first');
```

```
CP{1} = cat(1,CP{1},PP{1}(ind1(1,ind2)));
CP{2} = cat(1,CP{2},PP{2}(ind1(1,ind2))); % Point C


[val ind1]= min(PP{2});
ind2 = find(PP{1}(ind1)== max(PP{1}(ind1)),1,'first');
CP{1} = cat(1,CP{1},PP{1}(ind1(1,ind2)));
CP{2} = cat(1,CP{2},PP{2}(ind1(1,ind2)));        % Point D

[val ind1]= max(PP{2});
ind2 = find(PP{1}(ind1)== min(PP{1}(ind1)),1,'first');
CP{1} = cat(1,CP{1},PP{1}(ind1(1,ind2)));
CP{2} = cat(1,CP{2},PP{2}(ind1(1,ind2))); % Point B
CP{1}(:) = ceil(CP{1}(:));
CP{2}(:) = ceil(CP{2}(:));
Area = (CP{1}(2)-CP{1}(1))*(CP{2}(4)-CP{2}(3));
```

```
function [MinT,Area,RS] = Find_Min_Area2(MinT,RS,Points,Area,P)
% Rotates inputed points in +RS and -RS degree and calculates the area
of
% the minimum rectangle surrounding the points and returns the theta
value
% that gives the minimum area between two opposite rotations.
% figure,imshow(Img)

 PP1 = Rotate_Perimeter_Points(Points,MinT + RS);
 PP1{1}(:) = PP1{1}(:)+P(1);
 PP1{2}(:) = PP1{2}(:)+P(2);
 [CP1 Area_Up] = Find_rec_area(PP1);
 PP2 = Rotate_Perimeter_Points(Points,MinT - RS);
 PP2{1}(:) = PP2{1}(:)+P(1);
 PP2{2}(:) = PP2{2}(:)+P(2);
 [CP2 Area_Down] = Find_rec_area(PP2);
if Area > Area_Up
     MinT = MinT + RS;
     Area = Area_Up;
 else if Area > Area_Down
         MinT = MinT - RS;
         Area = Area_Down;
     end
 end
 RS = RS/2;
```

```
function PP = Rotate_Perimeter_Points(Input_coordinates,theta)
% This function rorates the Input_coordinates, theta degrees.
% % Rotate input coordinates theta degree.
In = cell2mat(Input_coordinates);
m = [cos(theta) -sin(theta);sin(theta) cos(theta)]';
Rotated = In*m;
PP = mat2cell(Rotated,size(Rotated,1),[1 1]);
```

```
function Flag = Smearing_Func(Img,PP)
% Smearing function checks if the text documents need to be rotated 90
% degree clockwise or counter clockwise. If number of white column
counts
% are more than white row counts the flag will be 0, otherwise it will
be
% 1.
WCC = 0;
WRC = 0;
Flag = 1;

rmin = min(PP{1}(:));
rmax = max(PP{1}(:));
cmin = min(PP{2}(:));
cmax = max(PP{2}(:));
WC = sum(sum(Img(rmin:rmax,cmin:cmax) == 1,2))
WR = sum(sum(Img(rmin:rmax,cmin:cmax) == 1,1))
for c = cmin:cmax
    x = find(Img(rmin:rmax,c),1,'first');
    if(isempty(x))
        WCC = WCC + 1;
    end
end
for r = rmin:rmax
    x = find(Img(r,cmin:cmax),1,'first');
    if(isempty(x))
        WRC = WRC + 1;
    end
end
WCC
WRC
if WCC > WRC
    Flag = 0;
end
```

Appendix F. Our Page Segmentation MATLAB source code

```
clc;
clear all;
close all;
%// list all the files in some folder
cd('C:\Personal\Windsor\Image processing code')
somefolder = './Images/';
filelist = dir(somefolder);
FV_T = cell(1,2);
imcell = cell(1,((numel(filelist)-2) / 4));
counter = 0;
for i=1:size(filelist,1)
    %// filelist is not a folder
    if filelist(i).isdir ~= true
        fname = filelist(i).name;
            temp = regexp (fname,'_','split');
            classtag = str2double(temp{1,2});
            Img = imread([somefolder fname]);

            %// convert it to grayscale image if tmp is a color
            %// image/picture
            if size(Img,3) == 3
            Img = rgb2gray(Img);
            end
            counter = counter + 1;
            if counter < 2
                imcell{classtag} = Img;
%                 j = j + 1;
            elseif counter == 4
                counter = 0;
            end
            level = graythresh(Img); % find the threshold based on otsu's
method
            BinaryImg = im2bw(Img,level);    % convert intensity image to
binary
            if BinaryImg(1,1) == 1
                BinaryImg = ~BinaryImg;
            end
            BinaryImg = padarray(BinaryImg,[5 5]);
            [r c] = find(BinaryImg == 1);
            Img_b = bound2im([r c]);

            FV_T{1} = cat(1,FV_T{1},invmoments(Img_b));
            FV_T{2} = cat(1,FV_T{2},classtag);
            classtag = [];
            %FV = cat(2,FV,classtag);
        %end
    end
end

% sort the value of the moments based on the classtag
Y = cell(1,2);
B = FV_T(:,2);
[Y{2},ix] = sort(B{1});
Y{1} = FV_T{1}(ix,:);
```

```
FV_T = Y;
clear Y;

%// Extract feature vectors of the sample data.
cd('C:\Personal\Windsor\Image processing code')
%cd('../')
samplefolder = './S_Images/';
file_l = dir(samplefolder);
FV_S = [];
class = [];
%// the first two in filelist are . and ..

for i=1: size(file_l,1)
%     size(file_l,1)
    %// filelist is not a folder
    if file_l(i).isdir ~= true
        fname = file_l(i).name;
            Img = imread([samplefolder fname]);
            %// convert it to grayscale image if tmp is a color
            %// image/picture
            if size(Img,3) == 3
            Img = rgb2gray(Img);
            end
            level = graythresh(Img); % find the threshold based on otsu's
method
            BinaryImg = im2bw(Img,level);    % convert intensity image to
binary
            if BinaryImg(1,1) == 1
                BinaryImg = ~BinaryImg;
            end
            BinaryImg = padarray(BinaryImg,[5 5]);

            cd('C:\Personal\Windsor\Image processing code\Results\Hu')
            fn = sprintf('Image_%s.jpeg',num2str(i-2,'%04i'));
            imwrite(BinaryImg,fn,'jpeg');
            cd('C:\Personal\Windsor\Image processing code')

            CC = bwconncomp(BinaryImg,8);
            L = labelmatrix(CC);
            [Regions,R_order] = sort_labels(BinaryImg,L,CC.NumObjects);
            [Match_matrix,Nmobj]                                   =
labeling_dist(Regions,CC.NumObjects,R_order);
            [changed_labled_image,N_obj]                           =
change_label(Regions,Match_matrix,Nmobj,CC.NumObjects);


            for k = 1:N_obj
                [r,c] = find(changed_labled_image == k);
                q = bound2im([r,c]);

                FV_S = cat(1,FV_S,invmoments(q));
            end

            class = knnclassify(FV_S,FV_T{1},FV_T{2},1);
            for j = 1:numel(class)
                  TestImg = imcell{class(j)};
                  cd('C:\Personal\Windsor\Image               processing
code\Results\Hu')
                  fn       =       sprintf('Image_%s_%s.jpeg',num2str(i-
2,'%04i'),num2str(j,'%04i'));
                  imwrite(TestImg,fn,'jpeg');

            end
            cd('C:\Personal\Windsor\Image processing code')
            class = [];
            clear TestImg;
        %end
    end
    FV_S = [];
end
 cd('C:\Personal\Windsor\Image processing code')
clear imcell;
close all

function          [Z_T_FV,H_T_FV,DCT_T_FV,Tr_classtag,imcell]          =
Get_Training_FVs()
cd('C:\Personal\Windsor\Image processing code')
somefolder = './Images/';
filelist = dir(somefolder);
NClass_sample = 4;
order = 1:10;
Tr_classtag = cell(1,1);
Z_T_FV = cell(1,1);
H_T_FV = cell(1,1);
DCT_T_FV = cell(1,1);

imcell = cell(1,((numel(filelist)-2) / 4));
counter = 0;
% classtag = 0;

%// the first two in filelist are . and ..
%size(filelist,1)

for i=1:size(filelist,1)
    %// filelist is not a folder
    if filelist(i).isdir ~= true

        fname = filelist(i).name;
            temp = regexp (fname,'_','split');
            classtag = str2double(temp{1,2});
            Img = imread([somefolder fname]);

            %// convert it to grayscale image if tmp is a color
            %// image/picture
            if size(Img,3) == 3
                Img = rgb2gray(Img);
            end
            counter = counter + 1;
            if counter < 2
                imcell{classtag} = Img;
            elseif counter == NClass_sample
                counter = 0;
            end
            level = graythresh(Img); % find the threshold based on
otsu's method
            BinaryImg = im2bw(Img,level);    % convert intensity image
to binary
            if BinaryImg(1,1) == 1
                BinaryImg = ~BinaryImg;
            end
            BinaryImg = padarray(BinaryImg,[5 5]);
            [r c] = find(BinaryImg == 1);
            Img_b = bound2im([r c]);
            Img_b = padarray(Img_b,[5 5]);
            Img_b = imresize(Img_b,[200 NaN],'bilinear');
            [Img_trans,~,~] = lans_invariant(Img_b,'scale 1 translation
1');
            [A_nm,~,~]  = lans_zmoment(Img_trans,order);
            clear Img_trans;
            Temp = abs(A_nm);
            Z_T_FV{1} = cat(1,Z_T_FV{1},Temp);
            clear A_nm;
            clear Temp;
            H_T_FV{1} = cat(1,H_T_FV{1},invmoments(Img_b));
            [Img_trans,~,~] = lans_invariant(BinaryImg,'scale 1 translation
1');
            B = boundaries(Img_trans); %Trace object boundaries
            b = cat(1,B{:}); % Concatenate all the found boundaries
            DCT_Coef_Train = DCTFVec([b(:,1)', b(:,2)'], 20, 0);
            clear Img_trans;
            DCT_T_FV{1} = cat(1,DCT_T_FV{1},DCT_Coef_Train');
            Tr_classtag{1} = cat(1,Tr_classtag{1},classtag);
            classtag = [];
            clear BinaryImg;
            clear Img_b;
            clear B;
            clear b;
    end
end
```

```
function [ DCTCoef ] = DCTFVec(ExtFVec, fnumb, mode)

% input mode:
%   mode = 1 => Simulation & Testing mode
%   mode = 0 => Training mode

% Performing Cosine Transform
if (mode ~= 3)

    [c,d] = size(ExtFVec);

    Apat = ExtFVec(:,1:d/2);
    Dpat = ExtFVec(:,(d/2+1):end);
    if (d/2) < fnumb

        Apat = [Apat zeros(c,fnumb - d/2)];
        Dpat = [Dpat zeros(c,fnumb - d/2)];
    end

    [m,n]=size(Apat);


    CosTr = cos((pi/n)*(repmat(([1:n]'+1/2),1,n))*diag([1:n]));

    a = Apat*CosTr.*repmat(sqrt(2/n),m,n);
    b = Dpat*CosTr.*repmat(sqrt(2/n),m,n);


    for i = 1:m
        for j = 1:fnumb
            if isnan(a(i,j))
                a(i,j)=0;
            end
            if isnan(a(i,j))
                a(i,j)=0;
            end
        end
    end

    DCTCoef = [a(:,1:(fnumb)) b(:,1:(fnumb))]';

else

    d = size(ExtFVec,2);

    Apat = ExtFVec(:,1:d/3);
    Bpat = ExtFVec(:,(d/3+1):(2*d/3));
    Dpat = ExtFVec(:,((2*d/3)+1):end);
    [m,n]=size(Apat);
    CosTr = cos((pi/n)*(repmat(([1:n]'+1/2),1,n))*diag([1:n]));

    a = Apat*CosTr.*repmat(sqrt(2/n),m,n);
    b = Bpat*CosTr.*repmat(sqrt(2/n),m,n);
    DCTCoef = [a(:,3:(2+fnumb)) b(:,3:(2+fnumb))]';
end
end
```

```
function [z_T_FVs,H_T_FVs,DCT_T_FVs] = Extract_Test_FVs(BImg,order)
z_T_FVs = [];
H_T_FVs = [];
DCT_T_FVs = [];
BinaryImg = padarray(BImg,[5 5]);
[changed_labled_image,N_obj] = Extracted_subwords(BinaryImg);

for k = 1:N_obj
    [r,c] = find(changed_labled_image == k);
    q = bound2im([r,c]);
    g = imresize(q,1.5);
    figure,imshow(g);
    H_T_FVs = cat(1,H_T_FVs,invmoments(q));
    Img_b = padarray(q,[5 5]);
    [TImg_trans,~,~]       =  lans_invariant(Img_b,'scale  1  translation
1');
    B = boundaries(TImg_trans); %Trace object boundaries
    b = cat(1,B{:}); % Concatenate all the found boundaries
    Boundary_Img = bound2im([ b(:,1), b(:,2)]);
    figure,imshow(Boundary_Img)
    DCT_Coef_Test = DCTFVec([b(:,1)', b(:,2)'], 20, 0); %dct2
    clear TImg_trans;
    DCT_T_FVs = cat(1,DCT_T_FVs,DCT_Coef_Test');
    clear TA_nm;
    Img_b = imresize(Img_b,[200 NaN],'bilinear');
    [TImg_trans,~,~]       =  lans_invariant(Img_b,'scale  1  translation
1');
    [TA_nm,~,~] = lans_zmoment(TImg_trans,order);
    clear TImg_trans;
    z_T_FVs = cat(1,z_T_FVs,abs(TA_nm));
    clear TA_nm;
end
```

```
function [changed_labled_image,N_obj] = Extracted_subwords(BinaryImg)
 CC = bwconncomp(BinaryImg,8);
 L = labelmatrix(CC);
 [Regions,R_order] = sort_labels(BinaryImg,L,CC.NumObjects);
 [Match_matrix,Nmobj] = labeling_dist(Regions,CC.NumObjects,R_order);
 [changed_labled_image,N_obj]                                          =
change_label(Regions,Match_matrix,Nmobj,CC.NumObjects);
```

```
function  [z,Region_order] = sort_labels(bw,L,num)

%  Components  are  sorted  based  on  their  maximum  row  value.  Therefore,
all
% the components with smaller maximum row values will be labeled before
the
% next row's components.
Region_info = cell(1,2);
Region_order = cell(1,1);
for i = 1:num
    [r ~] = find(L == i);
    Region_info{1} = cat(1,Region_info{1},max(r));
end
% Region_info{1}(:,1)
[sorted,sorted_order] = sort(Region_info{1},1);
L2 = zeros(size(L));
for i = 1:num
    L2(L == sorted_order(i)) = i ;
end
clear Region_info
Region_info = cell(1,2);
Region_info{1} = sorted;
for i = 1:num
    [r ~] = find(L2 == i);
    Region_info{2} = cat(1,Region_info{2},min(r));
end
```

```
Region_order{1} = cat(1,Region_order{1},1);
for i = 1:num-1
   if abs(Region_info{1}(i,1)-Region_info{2}(i+1,1))> 25
       Region_order{1} = cat(1,Region_order{1},i+1);
   end
end
Region_order{1} = cat(1,Region_order{1},num+1);
% Region_order{1}(:,1)
clear Region_info;
clear sorted;
clear sorted_order;
clear r;
clear c;
% Look at the components of each row and sort them from maximum column
% value to minimum.
Region_info = cell(1,2);
z = zeros(size(L2));

for M = 1:num
    [r c] = find(L2 == M);
    Region_info{2} = cat(1,Region_info{2},max(c));
    Region_info{1} = cat(1,Region_info{1},M);
end
 for i = 1: size(Region_order{1})-1
     for j = Region_order{1}(i): (Region_order{1}(i+1))-1
         for k = j+1: (Region_order{1}(i+1))-1
             if(Region_info{2}(j) < Region_info{2}(k))
                 temp1 = Region_info{2}(j);
                 temp2 = Region_info{1}(j);
                 Region_info{2}(j) = Region_info{2}(k);
                 Region_info{1}(j) = Region_info{1}(k);
                 Region_info{2}(k)= temp1;
                 Region_info{1}(k)= temp2;
             end
         end
     end
 end

 for i = 1: num
     z(L2 == Region_info{1}(i)) = i ;
 end
```

```
function [L_matrix,Nmatched_obj] = labeling_dist(L,num,Line_seg)
% LABELING labels overlapping objects to the same group.
% [L_matrix,Nmatched_obj] = LABELING(L,NUM)receives a specified
% connected component matrix and
% the number of connected components in an image and figures out
wheather
% either of those objects belong to eachother. If so, it changes the
lables
% to reflect this effect. The output of this function is L_matrix,
% the matrix of the lables that need to be changed, and number of
% overlapped objects that has been found,Nmatched_obj.
%L_matrix = zeros(factorial(num),2);
L_matrix = zeros(num,2);
rp = 1;
```

```
Nmatched_obj = 0;
% Line_seg{1}(:,1);
for i = 1: size(Line_seg{1})-1
     for k = Line_seg{1}(i): (Line_seg{1}(i+1))-1
         [r, c] = find(L==k);
         for j = k+1: (Line_seg{1}(i+1))-1
             [rj, cj] = find(L == j);
             if   (max(c,[],1)   >=   min(cj,[],1))   &&   (min(c,[],1)<=
max(cj,[],1))
                 if  (  max(cj,[],1)  <=max(c,[],1))  &  (min(cj,[],1)  >=
min(c,[],1))...
                         |(max(c,[],1)<= max(cj,[],1)+ 5)

                     if(size(c,1) >= size(cj,1))
                         L_matrix(rp,1:2)=[j,k]; % j Label has to change to k
label
                         rp = rp+1;
                     else
                         L_matrix(rp,1:2)=[k,j];
                         rp = rp+1;
                     end
                  end
                  Nmatched_obj  =  Nmatched_obj  +  1;  %  number  of  matched
objects
             end
         end
     end
end
return

function [LImage,remain_obj] = change_label(LImage,M,N_match,N_obj)
%CHANGE_LABEL  changes  the  label  of  the  objects  have  to  be  in  the  same
group.
%[LImage,remain_obj] = CHANGE_LABEL(LImage,M,N_match,N_OBJ) find the
%objects that need to be in the same group and changes their label to
%match them togheter. Then arrange all the label in sequence.
%  It  receives  the  labeled  image,  LImage,  and  the  matrix  contains  the
matching
%  number  of  the  labels,M,  number  of  matched  has  been  found,N_match,  and
the
% total number of objects,N_obj.
% It returns the corrected label matrix ,LImage, and number of objects
% remained in the Image,remain_obj.

if(N_obj > 0)
    for i = 1:N_match
        [r,c] = find(LImage == M(i,1));
        for j = 1:size(r,1)
            LImage(r(j),c(j))= M(i,2);
        end
    end
end
x = zeros(1,N_obj);
remain_obj = 0;
for i = 1:N_obj
    y = find(M(:,1) == i);
    if(size(y,1) == 0)
```

```
        remain_obj = remain_obj + 1;
        x(remain_obj) = i;
    end
end
% x
for i = 1:remain_obj
    [r,c] = find(LImage == x(1,i));
    for j = 1:size(r,1)
        LImage(r(j),c(j))= i;
    end
end
```

```
function class = Voting_Scheme(Z,H,D)
class = cell(1,1);
for i = 1:numel(Z)
    if Z(i) == H(i)
        class{1} = cat(1,class{1},Z(i));
    else
        class{1} = cat(1,class{1},D(i));
    end
end
```

```
function [z,H,D,tag] = Sort_Tr_FVs(z,H,D,tag)
% sort the value of the moments based on the classtag
Temp_tag = cell(1,1);
[Temp_tag{1},ix] = sort(tag{1});
Temp_FV{1} = z{1}(ix,:);
z{1} = Temp_FV{1};
Temp_FV{1} = [];
Temp_FV{1} = H{1}(ix,:);
H{1} = Temp_FV{1};
Temp_FV{1} = [];
Temp_FV{1} = D{1}(ix,:);
D{1} = Temp_FV{1};
tag = Temp_tag;
clear Temp_tag;
```

VITA AUCTORIS

NAME: Mahnaz Shafii

PLACE OF BIRTH: Tehran, Iran

YEAR OF BIRTH: 1980

EDUCATION: Azad University, B.Sc., Tehran, Iran, 2003

Michigan State University, M.Sc., MI, USA, 2008

University of Windsor, Ph.D., Ontario, CANADA, 2014