PROJECT REPORT

On

OPTICAL CODE RECOGNITION

Submitted for Partial Fulfilment of Award of

BACHELOR OF TECHNOLOGY
In
Computer Science and Engineering
(2017)

By
Raheela Zaidi [1312210079]
Shaivya Chandra [1312210091]

Under the Guidance
Of
Dr R.G. Tiwari

SHRI RAMSWAROOP MEMORIAL GROUP OF PROFESSIONAL COLLEGES, LUCKNOW
Affiliated to
Dr. APJ ABDUL KALAM TECHNICAL UNIVERSITY, LUCKNOW

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRMGPC

CERTIFICATE

Certified that the project entitled **"Optical Code Recognition"** submitted by **Raheela Zaidi [1312210079]** and **Shaivya Chandra [1312210091]** in the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (CS) of Dr. A.P.J. Abdul Kalam Technical University, is a record of students' own work carried under our supervision and guidance. The project report embodies results of original work and studies carried out by the student and the contents do not forms the basis for the award of any other degree to the candidate or to anybody else.

Dr. R.G.Tiwari	**Mr. Alok Misra**
Assistant Professor	HOD, CSE
(Project Guide)	(Head of Department)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

SRMGPC

DECLARATION

We hereby declare that the project entitled **"Optical Code Recognition"** submitted by using the partial fulfilment of the requirements for the award of the degree of Bachelor of Technology (CSE) of Dr. A.P.J. Abdul Kalam Technical University, is a record of our own work carried under the supervision and guidance of **Dr.R.G.Tiwari** (Assistant Professor, Department of Computer Science). To the best of our knowledge this project has not been submitted to Dr. A P J Abdul Kalam Technical University or any other University or institute for the award of any degree.

Raheela Zaidi **Shaivya Chandra**

(1312210079) **(1312210091)**

ACKNOWLEDGEMENTS

We take this opportunity to express our profound gratitude and deep regards to our guide and our coordinator **Dr. R.G.Tiwari** for their exemplary guidance, monitoring and constant encouragement. The blessing, help and guidance given by them time to time shall carry us a long way in the journey of life on which we are about to embark.

We also take this opportunity to express a deep sense of gratitude to **Computer Science and Engineering Department, SRMGPC, Lucknow** for their cordial support, valuable information and guidance, which helped us in this task through various stages.

We are obliged to staff members of **Computer Science and Engineering Department, SRMGPC**, for the valuable information provided by them in their respective fields. We are grateful for their cooperation. We would like to express my special gratitude and thanks to them for giving us such attention and time.

Raheela Zaidi **(1312210079)**

Shaivya Chandra **(1312210091)**

PREFACE

Today with the advancement of web, people are placing their comments & opinions on social media so that they can be seen by other people too. Survey has shown that such opinion also affect the people reading those opinions. So the remarks related to a product or issue are to be analysed by the associated organization so that they can be improved based upon the remarks of people. Opinion mining and sentiment analysis are used to extract such remarks and analyse them on the basis of its polarity respectively. In this project we try to get users opinion with the help of sentiment analysis. The proposed concept is explained with the help of movie reviews.

The project report has been divided into 6 chapters. The topics covered under each are as follows:

- **INTRODUCTION:** This chapter gives our problem definition along with the aims and objectives. This part also includes a section on classification,data source and extraction of the project.
- **LITERATURE REVIEW:** This chapter explains the general theory of the project Sentiments analysis and based benefits.
- **PROPOSED METHODOLOGY:** This chapter describes the way in which the process can take place, software/hardware requirements and specifications.
- **RESULTS AND DISCUSSIONS:** This section gives the tasks that were accomplished module-wise.
- **FUTURE SCOPE:** This section gives the future enhancements that can be made in the project idea and its implementation.

ABSTRACT

Typing code ties a programmer to their laptop. This is problematic for lecturers because it reduces mobility, reduces eye contact and engagement with the class, and makes it difficult and unnatural for the lecturer to point out things in the code. Handwriting code on the board solves all of these problems. However, it also introduces another one: the code can not actually run. Often times, it is useful to show the end result or step through the program. Our aim is to create an application which can identify the handwritten code to an compliable source code.

Current prevalent OCR-engines can recognize printed text with high accuracy, but can hardly handle hand-written text very well. The difficulty of hand-written text recognition is due to variation of characters and poor alignment of text line. Therefore, we design to achieve a workable solution for hand-written code recognition system.

Keywords- Optical Character Recognition, Pre Processing, Tesseract, Post Processing, Compilation

Table of Contents

CHAPTER-1

INTRODUCTION

Optical Character recognition has been a subject of research. Pattern recognition has three main steps: observation, pattern segmentation, and pattern classification. Optical Character Recognition (OCR) systems is transforming large amount of documents, either printed alphabet or handwritten into machine encoded text without any transformation, noise, resolution variations and other factors.

In general, handwriting recognition is classified into two types as off-line and On-line character recognition. Off-line handwriting recognition involves automatic conversion of text into an image into letter codes which are usable within computer and text-processing applications. Off-line handwriting recognition is more difficult, as different people have different handwriting styles. But, in the on-line system, On-line character recognition deals with a data stream which comes from a transducer while the user is writing. The typical hardware to collect data is a digitizing tablet which is electromagnetic or pressure sensitive. When the user writes on the tablet, the successive movements of the pen are transformed to a series of electronic signal which is memorized and analysed by the computer.

Optical character recognition (also optical character reader, OCR) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text, whether from a scanned document, a photo of a document, a scene-photo (for example the text on signs and billboards in a landscape photo) or from subtitle text superimposed on an image (for example from a television broadcast). It is widely used as a form of information entry from printed paper data records, whether passport documents, invoices, bank statements, computerised receipts, business cards, mail, printouts of static-data, or any suitable documentation. It is a common method of

digitising printed texts so that they can be electronically edited, searched, stored more compactly, displayed on-line, and used in machine processes such as cognitive computing, machine translation, (extracted) text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.
OCR engines have been developed into many kinds of domain-specific OCR applications, such as receipt OCR, invoice OCR, and check OCR, legal billing document OCR.

They can be used for:

- Data entry for business documents, e.g. check, passport, invoice, bank statement and receipt
- Automatic number plate recognition.
- Automatic insurance documents key information extraction
- Extracting business card information into a contact list.
- More quickly make textual versions of printed documents, e.g. book scanning for Project Gutenberg
- Make electronic images of printed documents searchable, e.g. Google Books
- Converting handwriting in real time to control a computer (pen computing)
- Defeating CAPTCHA anti-bot systems, though these are specifically designed to prevent OCR.
- Assistive technology for blind and visually impaired users

Optical Character recognition is an application of Digital image processing (DIP).Digital image processing deals with manipulation of digital images through a digital computer. It is a subfield of signals and systems but focus particularly on images. DIP focuses on developing a computer system that is able to perform processing on an image. The input of that system is a digital image and the system process that image using efficient algorithms, and gives an image as an output. The most common example is Adobe Photoshop. It is one of the widely used applications for processing digital images.
Signal processing is a discipline in electrical engineering and in mathematics that deals with analysis and processing of analog and digital signals, and deals with storing, filtering, and other operations on signals. These signals include transmission signals, sound or voice signals, image signals, and other signals etc.

Out of all these signals, the field that deals with the type of signals for which the input is an image and the output is also an image is done in image processing. As its name suggests, it deals with the processing on images.

It can be further divided into analog image processing and digital image processing.

- **Analog Image Processing**

Analog image processing is done on analog signals. It includes processing on two dimensional analog signals. In this type of processing, the images are manipulated by electrical means by varying the electrical signal. The common example include is the television image.

Digital image processing has dominated over analog image processing with the passage of time due its wider range of applications.

- **Digital Image Processing**

The digital image processing deals with developing a digital system that performs operations on a digital image.

1.1 WHAT IS AN IMAGE?

An image is nothing more than a two dimensional signal. It is defined by the mathematical function f(x, y) where x and y are the two co-ordinates horizontally and vertically.

The value of f(x, y) at any point is gives the pixel value at that point of an image.

Figure 1.1 Image

The above figure is an example of digital image that you are now viewing on your computer screen. But actually, this image is nothing but a two dimensional array of numbers ranging between 0 and 255.

Relationship between a digital image and a signal

If the image is a two dimensional array then what does it have to do with a signal? In order to understand that, we need to first understand what is a signal?

SIGNAL

In physical world, any quantity measurable through time over space or any higher dimension can be taken as a signal. A signal is a mathematical function, and it conveys some information. A signal can be one dimensional or two dimensional or higher dimensional signal. One dimensional signal is a signal that is measured over time. The common example is a voice signal.

The two dimensional signals are those that are measured over some other physical quantities. The example of two dimensional signals is a digital image. We will look in more detail in the next tutorial of how a one dimensional or two dimensional single and higher signals are formed and interpreted.

Relationship

Since anything that conveys information or broadcast a message in physical world between two observers is a signal. That includes speech or (human voice) or an image as a signal. Since when we speak, our voice is converted to a sound wave/signal and transformed with respect to the time to person we are speaking to. Not only this , but the way a digital camera works, as while acquiring an image from a digital camera involves transfer of a signal from one part of the system to the other.

1.2 HOW A DIGITAL IMAGE IS FORMED?

Since capturing an image from a camera is a physical process. The sunlight is used as a source of energy. A sensor array is used for the acquisition of the image. So when the sunlight falls upon the object, then the amount of light reflected by that object is sensed by the sensors, and a continuous voltage signal is generated by the amount of sensed data. In order to create a digital image, we need to convert this data into a digital form. This involves sampling and quantization. (They are discussed later on). The result of sampling and quantization results in a two dimensional array or matrix of numbers which are nothing but a digital image.

There is much type of images, and we will look in detail about different types of images, and the colour distribution in them.

The binary image

The binary image as it name states, contain only two pixel values 0 and 1.

In our previous tutorial of bits per pixel, we have explained this in detail about the representation of pixel values to their respective colours.

Here 0 refers to black colour and 1 refers to white colour. It is also known as Monochrome.

- **Black and white image:**

The resulting image that is formed hence consists of only black and white colour and thus can also be called as Black and White image.

- **No gray level**

One of the interesting this about this binary image that there is no gray level in it. Only two colours that are black and white are found in it.

Format

Binary images have a format of PBM (Portable bit map) 2, 3, 4, 5, 6 bit colour format

The images with a colour format of 2, 3, 4, 5 and 6 bit are not widely used today. They were used in old times for old TV displays, or monitor displays.

But each of these colours have more than two gray levels, and hence has gray colour unlike the binary image.

In a 2 bit 4, in a 3 bit 8, in a 4 bit 16, in a 5 bit 32, in a 6 bit 64 different colours are present.

- **8 bit colour format**

8 bit colour format is one of the most famous image formats. It has 256 different shades of colours in it. It is commonly known as Grayscale image. The range of the colours in 8 bit varies from 0-255. Where 0 stands for black, and 255 stands for white, and 127 stands for gray colour.

This format was used initially by early models of the operating systems UNIX and the early colour Macintoshes.

Format:

The format of these images is PGM (Portable Gray Map).

This format is not supported by default from windows. In order to see gray scale image, you need to have an image viewer or image processing toolbox such as Mat lab.

Behind gray scale image:

As we have explained it several times in the previous tutorials, that an image is nothing but a two dimensional function, and can be represented by a two dimensional array or matrix. So in the case of the image of Einstein shown above, there would be two dimensional matrix in behind with values ranging between 0 and 255.

But that’s not the case with the colour images.

16 bit colour format

It is a colour image format. It has 65,536 different colours in it. It is also known as High colour format. It has been used by Microsoft in their systems that support more than 8 bit colour format. Now in this 16 bit format and the next format we are going to discuss which is a 24 bit format are both colour formats. The distribution of colour in a colour image is not as simple as it was in grayscale image.

A 16 bit format is actually divided into three further formats which are Red, Green and Blue. The famous RGB format. Now the question arises, that how would you distribute 16 into three. If you do it like this, 5 bits for R, 5 bits for G, 5 bits for B

Then there is one bit remains in the end. So the distribution of 16 bit has been done like this.5 bits for R, 6 bits for G, 5 bits for B.

The additional bit that was left behind is added into the green bit. Because green is the colour which is most soothing to eyes in all of these three colours.

Note this is distribution is not followed by all the systems. Some have introduced an alpha channel in the 16 bit. Another distribution of 16 bit format is like this: 4 bits for R, 4 bits for G, 4 bits for B, 4 bits for alpha channel.

Or some distribute it like this5 bits for R, 5 bits for G, 5 bits for B, 1 bits for alpha channel.

- **24 bit colour format**

24 bit colour format also known as true colour format. Like 16 bit colour format, in a 24 bit colour format, the 24 bits are again distributed in three different formats of Red, Green and Blue.

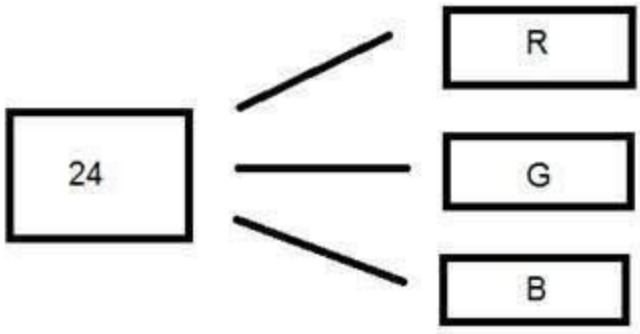

Figure 1.2 24 bit Image

Since 24 is equally divided on 8, so it has been distributed equally between three different colour channels.

Their distribution is like this.8 bits for R, 8 bits for G, 8 bits for B.

1.3 OVERLAPPING FIELDS

Machine/Computer vision

Machine vision or computer vision deals with developing a system in which the input is an image and the output is some information. For example: Developing a system that scans human face and opens any kind of lock. This system would look something like this.

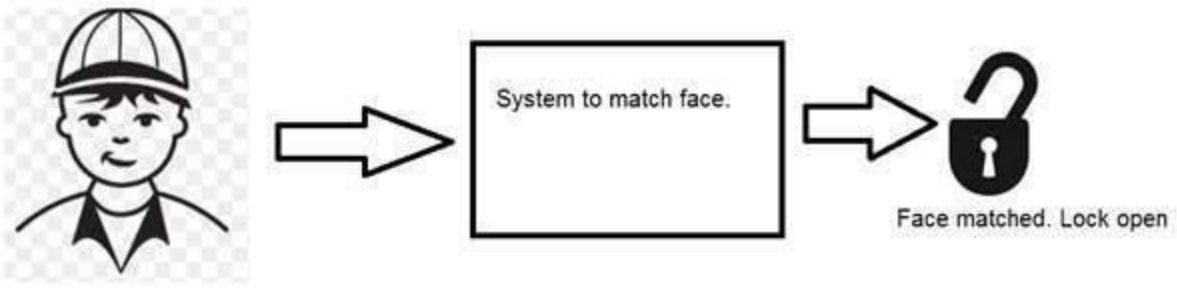

Figure 1.3 Computer Vision

Computer graphics

Computer graphics deals with the formation of images from object models, rather than the image is captured by some device. For example: Object rendering. Generating an image from an object model. Such a system would look something like this.

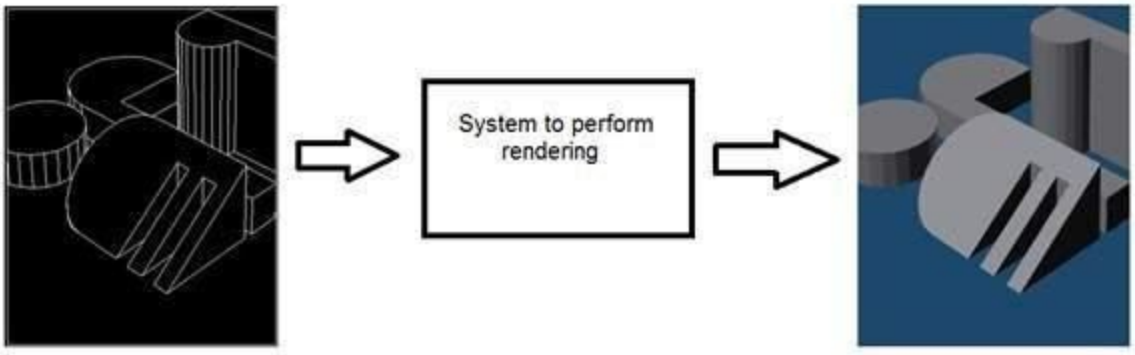

Figure 1.4 Computer Graphics

Artificial intelligence

Artificial intelligence is more or less the study of putting human intelligence into machines. Artificial intelligence has many applications in image processing. For example: developing computer aided diagnosis systems that help doctors in interpreting

images of X-ray , MRI etc. and then highlighting conspicuous section to be examined by the doctor.

Signal processing

Signal processing is an umbrella and image processing lies under it. The amount of light reflected by an object in the physical world (3d world) is passing through the lens of the camera and it becomes a 2d signal and hence result in image formation. This image is then digitized using methods of signal processing and then this digital image is manipulated in digital image processing.

Optical Character Recognition (OCR) is a field of research in pattern recognition, artificial intelligence and machine vision, signal processing. Optical character recognition (OCR) is usually referred to as an off-line character recognition process to mean that the system scans and recognizes static images of the characters. It refers to the mechanical or electronic translation of images of handwritten character or printed text into machine code without any variation. We aims to bridge the gap between handwritten source code and compliable source code.

Optical Character recognition has been a subject of research. Pattern recognition has three main steps: observation, pattern segmentation, and pattern classification. Optical Character Recognition (OCR) systems is transforming large amount of documents, either printed alphabet or handwritten into machine encoded text without any transformation, noise, resolution variations and other factors.

In general, handwriting recognition is classified into two types as off-line and

On-line character recognition. Off-line handwriting recognition involves automatic conversion of text into an image into letter codes which are usable within computer and text-processing applications. Off-line handwriting recognition is more difficult, as different people have different handwriting styles. But, in the on-line system, On-line character recognition deals with a data stream which comes from a transducer while the user is writing. The typical hardware to collect data is a digitizing tablet which is electromagnetic or pressure sensitive. When the user writes on the tablet, the successive movements of the pen are transformed to a series of electronic signal which is memorized and analysed by the computer.

Optical Character Recognition (OCR) is a field of research in pattern recognition, artificial intelligence and machine vision, signal processing. Optical character recognition (OCR) is usually referred to as an off-line character recognition process to mean that the system scans and recognizes static images of the characters. It refers to the mechanical or electronic translation of images of handwritten character or printed text into machine

code without any variation. We aims to bridge the gap between handwritten source code and compliable source code.

1.4 PROJECT OBJECTIVE

Current prevalent OCR-engines can recognize printed text with high accuracy, but can hardly handle hand-written text very well. The difficulty of hand-written text recognition is due to variation of characters and poor alignment of text line. Therefore, to achieve a workable solution for hand-written code recognition system, we first customize and train an OCR-engine, to make it able to deal with my own handwriting. Besides, several image processing methods and text-level post processing algorithms have been adopted to enhance the system accuracy. Test results show that the system can have a reasonable accuracy on both characters set and realistic code text.

Typing code ties a programmer to their laptop. This is problematic for lecturers because it reduces mobility, reduces eye contact and engagement with the class, and makes it difficult and unnatural for the lecturer to point out things in the code. Handwriting code on the board solves all of these problems. However, it also introduces another one: the code cannot actually run. Often times, it is useful to show the end result or step through the program. Our aim is to create an application which can identify the handwritten code to a compliable source code.

1.5 PROJECT METHODOLOGY:

Since OCR would require high computation complexity, running the entire project on mobile phone is infeasible. So we a number of steps in order to get a nearly accurate result:

1. Capture and upload Image: Using Android device to capture the image and upload it to the server on the web page.
2. Pre-processing: In this step, several image processing methods are applied to the uploaded image to make it ready for recognition.
3. Text recognition: In this step, we run the OCR engine to extract text from the pre-processed image. It cannot recognize handwritten text originally, and some training process is required to make it capable of doing so.
4. Post-processing: In this step, some heuristic algorithms are applied to the retrieved text.
5. Manual adjustment: Even with all the previous steps, the system still cannot guarantee 100% correctness of recognition, and even 1 error would lead to compilation failure so we copy the code on the compiler plugin.
6. Compile & execute & display: After manually modified by user, ideally everything should be correct now.

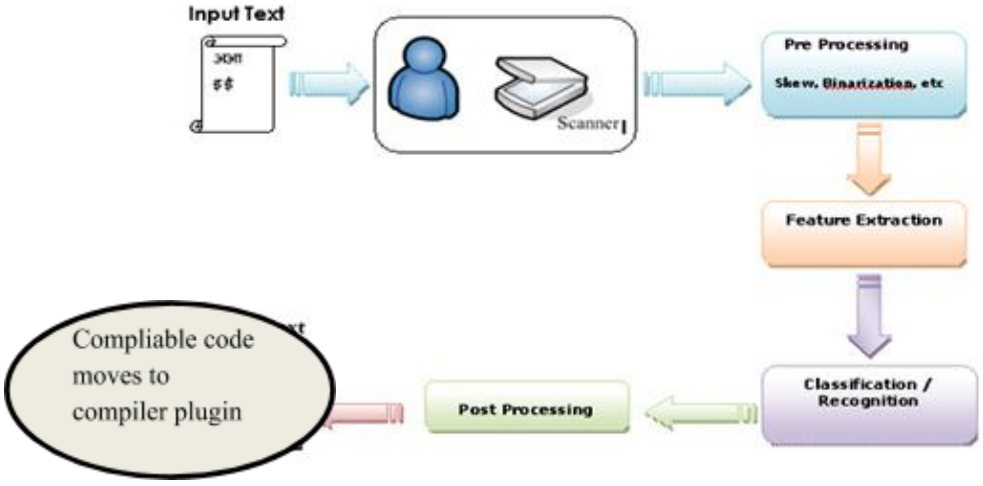

Figure 1.5 Flow Chart

Optical character recognition (optical character reader) (OCR) is the mechanical or electronic conversion of images of typed, handwritten or printed text into machine-encoded text. It is widely used as a form of data entry from printed paper data records, whether passport documents, invoices, bank statements, computerised receipts, business cards, mail, printouts of static-data, or any suitable documentation. It is a common method of digitising printed texts so that it can be electronically edited,

searched, stored more compactly, displayed on-line, and used in machine processes such as machine translation, text-to-speech, key data and text mining. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

Early versions needed to be trained with images of each character, and worked on one font at a time. Advanced systems capable of producing a high degree of recognition accuracy for most fonts are now common. Some systems are capable of reproducing formatted output that closely approximates the original page including images, columns, and other non-textual components.

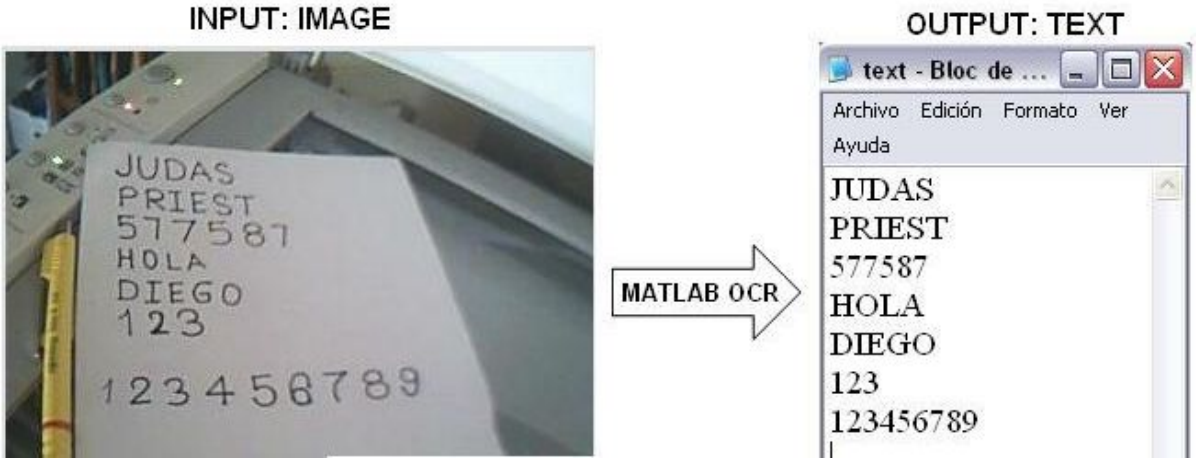

Figure 1.6 OCR Example

1.6 RELATED WORK

Early optical character recognition may be traced to technologies involving telegraphy and creating reading devices for the blind. In 1914, Emanuel Goldberg developed a machine that read characters and converted them into standard telegraph code. Concurrently, Edmund Fournier d'Albe developed the Optophone, a handheld scanner that when moved across a printed page, produced tones that corresponded to specific letters or characters.

Iris is a system for evaluating Ruby source code on-the-go. Iris consists of a web client where the user can upload an image of source code and a server where image is processed by Tesseract. Tesseract's output is then post-processed with rules to substitute common error strings with correct versions. Ruby was chosen because its syntax rules are relatively loose compared to other languages like C. For example, semicolons are optional, parenthesis are optional in Ruby's "print" statement, and variables do not have a static type. Since Ruby is an interpreted language like Python and MATLAB, it is possible to get partial output even in presence of syntax errors, e.g. a syntax error on line 100 will not necessarily cause the whole program to fail.

In the 2000s, OCR was made available online as a service (WebOCR), in a cloud computing environment, and in mobile applications like real-time translation of foreign-language signs on a smartphone.

Various commercial and open source OCR systems are available for most common writing systems, including Latin, Cyrillic, Arabic, Hebrew, Indic, Bengali (Bangla), and Devanagari, Tamil, Chinese, Japanese, and Korean characters.

CHAPTER-2

LITREARY REVIEW

2.1 THE VERY FIRST ATTEMPTS.

To replicate the human functions by machines, making the machine able to perform tasks like reading is an ancient dream. The origins of character recognition can actually be found back in 1870. This was the year that C.R.Carey of Boston Massachusetts invented the retina scanner which was an image transmission system using a mosaic of photocells.

Two decades later the Polish P. Nipkow invented the sequential scanner which was a major breakthrough both for modern television and reading machines. During the first decades of the 19'Th century several attempts were made to develop devices to aid the blind through experiments with OCR. However, the modern version of OCR did not appear until the middle of the 1940's with the development of the digital computer. The motivation for development from then on, was the possible applications within the business world.

The start of OCR

By 1950 the technological revolution was moving forward at a high speed, and electronic data processing was becoming an important field. Data entry was performed through punched cards and a cost-effective way of handling the increasing amount of data was needed. At the same time the technology for machine reading was becoming sufficiently mature for application, and by the middle of the 1950's OCR machines became commercially available.

The first true OCR reading machine was installed at Reader's Digest in 1954. This equipment was used to convert typewritten sales reports into punched cards for input to the computer.

2.2 HISTORY OF MACHINE RECOGNITION OF SCRIPTS

The overwhelming volume of paper-based data in corporations and offices challenges their ability to manage documents and records. Computers, working faster and more efficiently than human operators, can be used to perform many of the tasks required for efficient document and content management. Computers understand alphanumeric characters as ASCII code typed on a keyboard where each character or letter represents a recognizable code. However, computers cannot distinguish characters and words from scanned images of paper documents. Therefore, where alphanumeric information must be retrieved from scanned images such as commercial or government documents, tax

returns, passport applications and credit card applications, characters must first be converted to their ASCII equivalents before they can be recognized as readable text.

Optical character recognition system (OCR) allows us to convert a document into electronic text, which we can edit and search etc. It is performed off-line after the writing or printing has been completed, as opposed to on-line recognition where the computer recognizes the characters as they are written. For these systems to effectively recognize hand-printed or machine printed forms, individual characters must be well separated. This is the reason why most typical administrative forms require people to enter data into neatly spaced boxes and force spaces between letters entered on a form. Without the use of these boxes, conventional technologies reject fields if people do not follow the structure when filling out forms, resulting in a significant overhead in the administration cost.

Optical character recognition for English has become one of the most successful applications of technology in pattern recognition and artificial intelligence. OCR is the machine replication of human reading and has been the subject of intensive research for more than five decades. To understand the evolution of OCR systems from their challenges, and to appreciate the present state of the OCRs, a brief historical survey of OCRs is in order now. Depending on the versatility, robustness and efficiency, commercial OCR systems may be divided into the following four generations [Line, 1993; Pal & Chaudhuri, 2004]. It is to be noted that this categorization refers specifically to OCRs of English language.

2.2.1 FIRST GENERATION OCR SYSTEMS

Character recognition originated as early as 1870 when Carey invented the retina scanner, which is an image transmission system using photocells. It is used as an aid to the visually handicapped by the Russian scientist Tyurin in 1900. However, the first generation machines appeared in the beginning of the 1960s with the development of the digital computers. It is the first time OCR was realized as a data processing application to the business world [Mantas, 1986]. The first generation machines are characterized by the "constrained" letter shapes which the OCRs can read. These symbols were specially designed for machine reading, and they did not even look natural. The first commercialized OCR of this generation was IBM 1418, which was designed to read a special IBM font, 407. The recognition method was template matching, which compares the character image with a library of prototype images for each character of each font.

2.2.2 SECOND GENERATION OCR SYSTEMS

Next generation machines were able to recognize regular machine-printed and hand printed characters. The character set was limited to numerals and a few letters and symbols. Such machines appeared in the middle of 1960s to early 1970s. The first automatic letter sorting machine for postal code numbers from Toshiba was developed during this period. The methods were based on the structural analysis approach. Significant efforts for standardization were also made in this period. An American standard OCR character set: OCR-A font was defined, which was designed to facilitate optical recognition, although still readable to humans. A European font OCR-B was also designed.

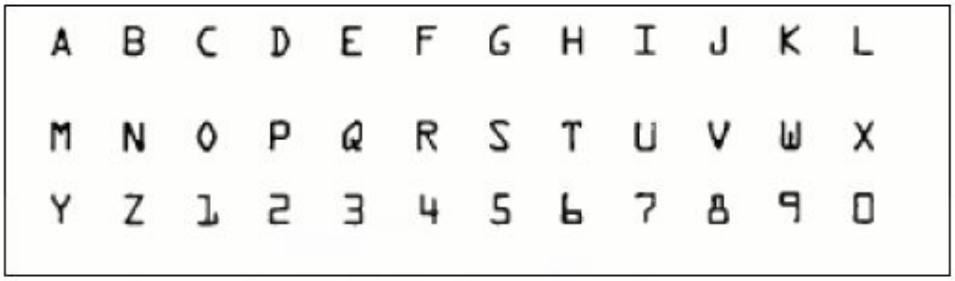

Figure 2.1 OCR Text Format

2.2.3 THIRD GENERATION OCR SYSTEMS

For the third generation of OCR systems, the challenges were documents of poor quality and large printed and hand-written character sets. Low cost and high performance were also important objectives. Commercial OCR systems with such capabilities appeared during the decade 1975 to 1985.

2.2.4 OCRS TODAY (FOURTH GENERATION OCR SYSTEMS)

The fourth generation can be characterized by the OCR of complex documents intermixing with text, graphics, tables and mathematical symbols, unconstrained handwritten characters, colour documents, low-quality noisy documents, etc. Among the commercial products, postal address readers, and reading aids for the blind are available in the market. Nowadays, there is much motivation to provide computerized document analysis systems. OCR contributes to this progress by providing techniques to convert large volumes of data automatically. A large number of papers and patents advertise recognition rates as high as 99.99%; this gives the impression that automation problems seem to have been solved. Although OCR is widely used presently, its accuracy today is still far from that of a 14 seven-year old child, let alone a moderately skilled typist [Nagy, Nartker & Rice, 2000].

Failure of some real applications show that performance problems still exist on composite and degraded documents (i.e., noisy characters, tilt, mixing of fonts, etc.) and that there is still room for progress. Various methods have been proposed to increase the accuracy of optical character recognizers. In fact, at various research laboratories, the challenge is to develop robust methods that remove as much as possible the typographical and noise restrictions while maintaining rates similar to those provided by limited-font commercial machines [Belaid,1997].

Thus, current active research areas in OCR include handwriting recognition, and also the printed typewritten version of non-Roman scripts (especially those with a very large number of characters).

2.3 COMPONENTS OF AN OCR SYSTEM

Before we present a survey on various approaches used in the literature for recognizing fonts and characters, a brief introduction to the general OCR techniques is given now. The objective of OCR software is to recognize the text and then convert it to editable form. Thus, developing computer algorithms to identify the characters in the text is the principal task of OCR. A document is first scanned by an optical scanner, which produces an image form of it that is not editable. Optical character recognition involves translation of this text image into editable character codes such as ASCII. Any OCR implementation consists of a number of pre-processing steps.

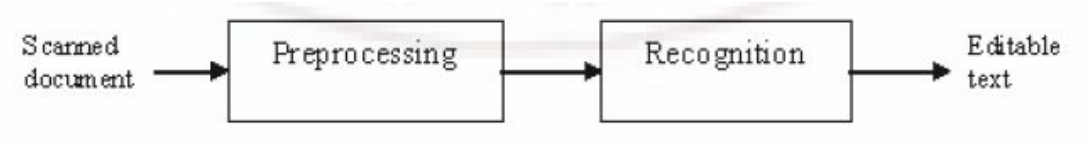

Figure 2.2 Components

The number and types of pre-processing algorithms employed on the scanned image depend on many factors such as age of the document, paper quality, resolution of the scanned image, amount of skew in the image, format and layout of the images and text,

kind of the script used and also on the type of characters: printed or handwritten .

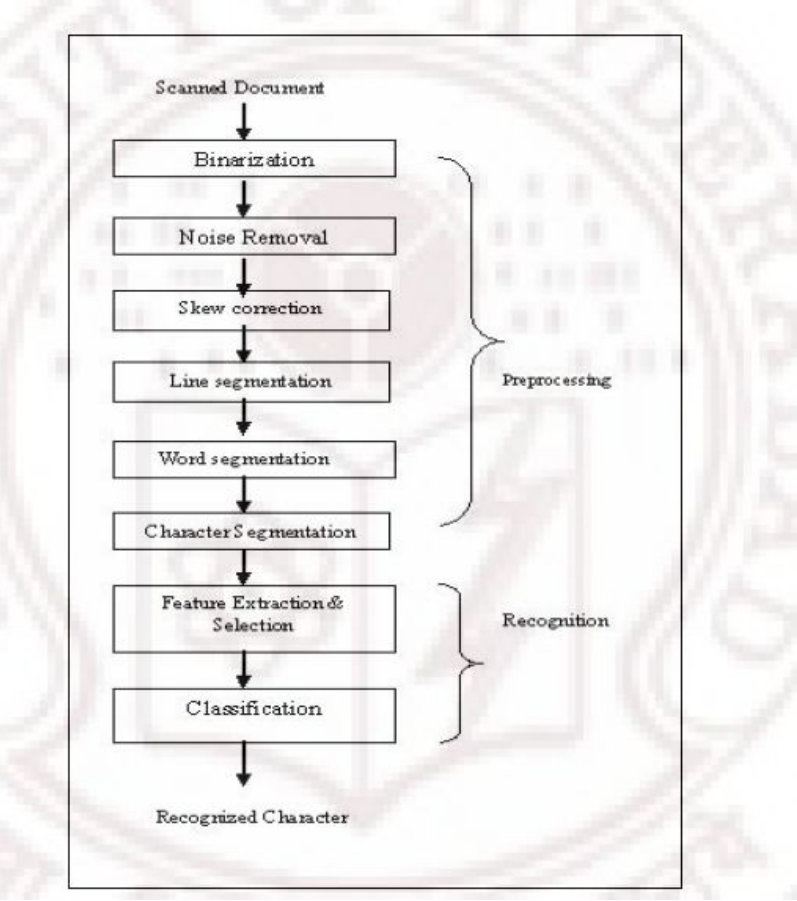

Figure 2.3 Basic Process

2.4 TYPES OF OCR

OCR systems may be subdivided into two classes. The first class includes the special purpose machines dedicated to specific recognition problems. The second class covers the systems that are based on a PC and a low-cost scanner.

Dedicated hardware systems

The first recognition machines were all hardwired devices. Because this hardware was expensive, throughput rates had to be high to justify the cost, and parallelism was exploited. Today such systems are used in specific applications were speed is of high importance, for instance within the areas of mail-sorting and check-reading. The cost of these machines are still high, up to a million dollars, and they may recognize a wide range of fonts.

Software based PC versions

Advancements in the computer technology has made it possible to fully implement the recognition part of OCR in software packages which work on personal computers. Present PC systems are comparable to the large scaled computers of the early days, and as little additional equipment is required, the cost of such systems is low.

However, there are some limitations in such OCR software, especially when it comes to speed and the kinds of character sets read. Hand held scanners for reading do also exist. These are usually limited to the reading of numbers and just a few additional letters or symbols of fixed fonts. They often read a line at a time and transmits it to application programs.

Three commercial software products are dominant within the area of recognition of European languages. These are systems produced by Caera Corporation, Kurzweil and Calera Corporation, with prices in the range of $500 - $1000. The speed of these systems is about 40 characters per second

2.6 OCR CAPABILITIES

The sophistication of the OCR system depends on the type and number of fonts recognized.

Below a classification, by the order of difficulty, based on the OCR systems' capability To recognize different character sets, is presented.

Fixed font.

OCR machines of this category deals with the recognition of one specific typewritten font.

Such fonts are OCR-A, OCR-B, Pica, Elite, etc. These fonts are characterized by fixed spacing between each character. The OCR-A and OCR-B are the American and European standard fonts specially designed for optical character recognition, where each character has a unique shape to avoid ambiguity with other characters similar in shape. Using these character sets, it is quite common for commercial OCR machines to achieve a recognition rate as high as 99.99% with a high reading speed.

The systems of the first OCR generation were fixed font machines, and the methods applied were usually based on template matching and correlation.

Multifont.

Multifont OCR machines recognize more than one font, as opposed to a fixed font system, which could only recognize symbols of one specific font. However, the fonts recognized by these machines are usually of the same type as those recognized by a fixed font system.

These machines appeared after the fixed-font machines. They were able to read up to about ten fonts. The limit in the number of fonts were due to the pattern recognition algorithm, template matching, which required that a library of bit map images of each character from each font was stored. The accuracy is quite good, even on degraded images, as long as the fonts in the library are selected with care.

Omnifont.

An omnifont OCR machine can recognize most nonstylized fonts without having to maintain huge databases of specific font information. Usually omnifont-technology is characterized by the use of feature extraction. The database of an omnifont system will contain a description of each symbol class instead of the symbols themselves. This gives flexibility in automatic recognition of a variety of fonts.

Although omnifont is the common term for these OCR systems, this should not be understood literally as the system being able to recognize all existing fonts. No OCR machine performs equally well, or even useably well, on all the fonts used by modern typesetters.

A lot of current OCR-systems claim to be omnifont.

Constrained handwriting.

Recognition of constrained handwriting deals with the problem of unconnected normal handwritten characters. Optical readers with such capabilities are not yet very common, but do exist. However, these systems require well-written characters, and most of them can only recognize digits unless certain standards for the hand-printed characters are followed. The characters should be printed as large as possible to retain good resolution, and entered in specified boxes. The writer is also instructed to keep to certain models provided, avoiding gaps and extra loops. Commercially the term ICR (Intelligent Character Recognition) is often used for systems able to recognize hand printed characters.

CHAPTER-3 PROPOSED METHODOLOGY

3.1 PROBLEM STATEMENT

Whenever we computer science engineers try to do some programming we begin with writing the code on paper instead on directly writing on the computer, in general. Thus we end a writing the code twice. This leads to time wastage.so through our application we will have to write the code just once on paper, click the image upload it and we will get the machine encoded version of our program. This will help in saving our time and also make the task easy and interesting.

3.2 MODULES AND THEIR FUNCTIONALITY

3.2.1 Image Pre-Processing

In image pre-processing, image data recorded by sensors on a satellite restrain errors related to geometry and brightness values of the pixels. These errors are corrected using appropriate mathematical models which are either definite or statistical models. Image enhancement is the modification of image by changing the pixel brightness values to improve its visual impact. Image enhancement involves a collection of techniques that are used to improve the visual appearance of an image, or to convert the image to a form which is better suited for human or machine interpretation.

Sometimes images obtained from satellites and conventional and digital cameras lack in contrast and brightness because of the limitations of imaging sub systems and illumination conditions while capturing image... Image enhancement is useful in feature extraction, image analysis and an image display. The enhancement process itself does not increase the inherent information content in the data. It simply emphasizes certain specified image characteristics. Enhancement algorithms are generally interactive and application dependent.

Image processing usually refers to digital image processing, but optical and analog image processing also are possible. The acquisition of images (producing the input image in the first place) is referred to as imaging. Closely related to image processing are computer graphics and computer vision. In computer graphics, images are manually made from physical models of objects, environments, and lighting, instead of being acquired (via imaging devices such as cameras) from natural scenes, as in most animated movies. Computer vision, on the other hand, is often considered high-level image processing out of which a machine/computer/software intends to decipher the physical contents of an image or a sequence of images (e.g., videos or 3D full-body magnetic resonance scans).

In modern sciences and technologies, images also gain much broader scopes due to the ever growing importance of scientific visualization (of often large-scale complex

scientific/experimental data). Examples include microarray data in genetic research, or real-time multi-asset portfolio trading in finance.

Typical pre-processing includes the following stages:

• Binarization,

• Noise removing, (De-Speckle)

• Skew detection and correction,

• Line segmentation,

• Word segmentation,

3.2.1.1 Binarization

A binary image is a digital image that has only two possible values for each pixel. Typically, the two colours used for a binary image are black and white, though any two colours can be used. The colour used for the object(s) in the image is the foreground colour while the rest of the image is the background colour. In the document-scanning industry, this is often referred to as "bi-tonal".

Binary images are also called bi-level or two-level. This means that each pixel is stored as a single bit—i.e., a 0 or 1. The names black-and-white, B&W, monochrome or monochromatic are often used for this concept, but may also designate any images that have only one sample per pixel, such as grayscale images. In Photoshop parlance, a binary image is the same as an image in "Bitmap" mode.

Binary images often arise in digital image processing as masks or as the result of certain operations such as segmentation, thresholding, and dithering. Some input/output devices, such as laser printers, fax machines, and bi-level computer displays, can only handle bi-level images.

A binary image can be stored in memory as a bitmap, a packed array of bits. A 640×480 image requires 37.5 KiB of storage. Because of the small size of the image files, fax machine and document management solutions usually use this format. Most binary images also compress well with simple run-length compression schemes.

Binary images can be interpreted as subsets of the two-dimensional integer lattice Z2; the field of morphological image processing was largely inspired by this view.

The **Binarization Method** converts the grey scale image (0 up to 256 gray levels) in to black and white image (0 or 1). The result of OCR highly depends upon the binarization. The high quality binarized image can give more accuracy in character recognition as

compared original image because noise is present in the original image. In fact problem is that which binarization algorithm is appropriate for all images. The selection of most optimal binarization algorithm is difficult, because different binarization algorithm gives different performance on different data sets. This is especially true in the case of historical documents images with variation in contrast and illumination.

The algorithms divide into two categories

a) Global Binarization

b) Local Binarization.

The global binarization methods used single threshold value for whole image and the local binarization method where the threshold value calculated locally pixel by pixel or region by region.

The figure below shows the basic block diagram of binarization:

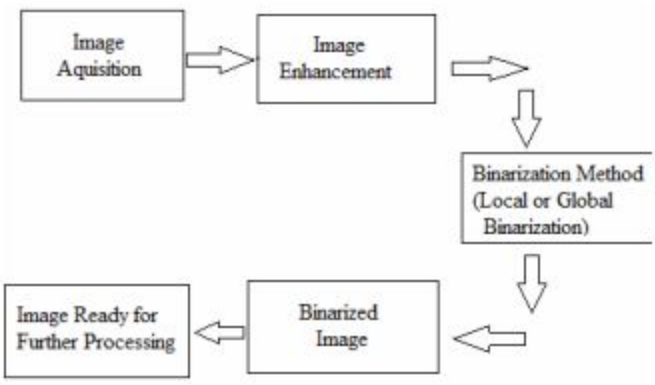

Figure 3.1 Basic Block Diagram of Binarization.

3.2.1.2 Noise Removal

When an image is acquired or is transmitted for Image processing applications, there are chances of image degradation. The degradation could be noise. Noise can be Gaussian, salt & pepper, Speckle, Poisson etc. Noise removal is an important task of image processing. Different types of noise can make image unreadable perfectly and cause barrier in many applications of image processing. Different type of linear and non-linear filters can be used to remove the speckles to make the region of the image under study clearer. There are many classical based filters for noise reduction, many fuzzy inspired filters have been developed [a]

Digital noise is generally caused by insufficient light levels at the site location. It may also occur when the imaging sensors come under the effect of environmental conditions at the time of image acquisition. Another cause is heat, the image sensor heats up causing photons to get separated from the photo sites and taint other photo sites. A very slow shutter speed allows the noise to enter. Presence of dust particles on the scanner screen and the presence of inference in the transmission channel can also corrupt the image.

Median, Adaptive median filter and proposed median filter.

The degraded image due to noise need to be treated for advanced processing. Different types of filters are available for the removal of noise. In this paper, median, adaptive median and new proposed median filters are discussed. These filters are used to eliminate salt and pepper noise.

Median filter: The median filter is a non-linear filtering tool to remove noise. Its hardware implementation is straightforward and does not require many resources. This filter is used traditionally to remove impulse noise as it is the most popular used non-linear filter. The standard median filter does not perform well when impulse noise is greater than 0.2. A simple median utilizing 3×3 or 5×5-pixel window is sufficient only when the noise intensity is less than approx. 10-20%. When the intensity of noise is increasing, a simple median filter remains many shots unfiltered. This filter does not preserve detail and it also smoothens non-impulsive noise.

Adaptive median filter: Adaptive median filter is a linear filtering tool that performs spatial processing to determine which pixels in an image have been affected by impulse noise. The adaptive median filter perform handle the noises than Median filter as it changes size the neighbourhood during operation. This filter has the following purposes:

1). Removal of impulse noise

2). Smoothing of other noise

3). Reduces distortion, like excessive thinning or thickening of object boundaries

Figure 3.2 Smoothening

3.2.1.3 Skew Detection and Correction

The field of image processing brings out the idea of automatic gathering and processing of the observed information. Like, most of the documents are present in the printed form. But if they are needed to be converted into electronic form, it has to be done through scanning. During document scanning, skew is inevitably introduced into the incoming document image. Skew refers to the text which neither parallel nor at right angles to a specified or implied line. Character recognition is very sensitive to the page skew, skew detection and correction in document images are the critical steps before layout analysis. The different types of skews within a document page can fall into these categories:- Global Skew, assuming that all page text have the same angle skew, Multiple-Skew, when certain area of the page have different slant than other, and NonUniform text line skew, when the slant fluctuates (such as lines having a wavy shape). Skew detection is used for text line position determination in Digitized documents, automated page orientation, and skew angle detection for binary document images, skew detection in handwritten scripts, in compensation for Internet audio applications and in the correction of scanned documents. The largest classes of methods for skew detection are based on projection profile analysis, Hough transform, nearest-neighbour clustering, Fourier-transformation, histogram analysis, moments and other methods. Describes a technique for skew detection. There are two steps for detecting the skew. The first step is dimension reduction; it is the process of decreasing the size of image pixels of features and finding another feature with lower dimensions. It includes transformation of initial set to other set for retaining maximum information. This will extract the initial set first and will generate missing information in it. After transformation, select an optimal subset of features based on an objective function. There are different dimension reduction methods, which we can use in skew detection of document images. The best subset includes all of the strongly relevant and weakly relevant but non-redundant features. In the second step, skew is estimated. Here, the deviation corresponding to the highest and the lowest value of the function is usually considered as the skew.

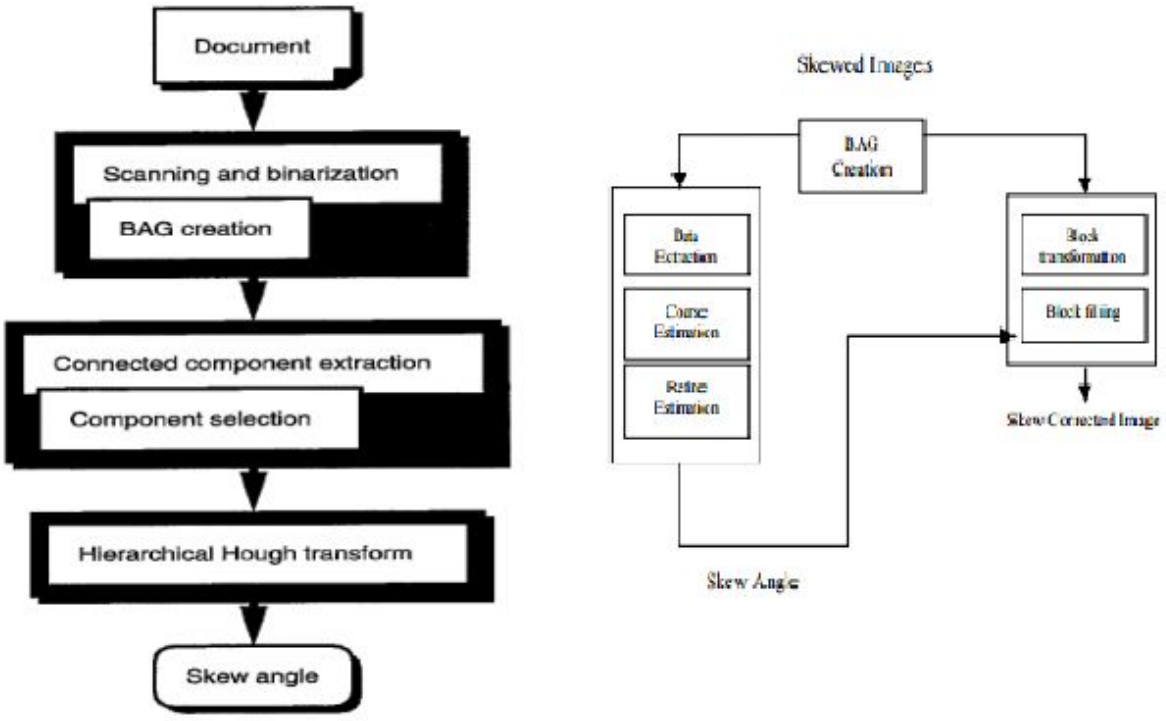

Figure 3.3 Block diagram of Skew detection and correction algorithm

3.2.2 Feature Extraction

In machine learning, pattern recognition and in image processing, feature extraction starts from an initial set of measured data and builds derived values (features) intended to be informative and non-redundant, facilitating the subsequent learning and generalization steps, and in some cases leading to better human interpretations. Feature extraction is related to dimensionality reduction.

When the input data to an algorithm is too large to be processed and it is suspected to be redundant (e.g. the same measurement in both feet and meters, or the repetitiveness of images presented as pixels), then it can be transformed into a reduced set of features (also named a feature vector). Determining a subset of the initial features is called feature selection. The selected features are expected to contain the relevant information from the input data, so that the desired task can be performed by using this reduced representation instead of the complete initial data.

Feature extraction involves reducing the amount of resources required to describe a large set of data. When performing analysis of complex data one of the major problems stems from the number of variables involved. Analysis with a large number of variables generally requires a large amount of memory and computation power, also it may cause a classification algorithm to overfit to training samples and generalize poorly to new samples. Feature extraction is a general term for methods of constructing combinations of the variables to get around these problems while still describing the data with sufficient accuracy.

The best results are achieved when an expert constructs a set of application-dependent features, a process called feature engineering. Nevertheless, if no such expert knowledge is available, general dimensionality reduction techniques may help. These include:

- Independent component analysis
- Isomer
- Kernel PCA
- Latent semantic analysis
- Partial least squares
- Principal component analysis
- Multifactor dimensionality reduction
- Nonlinear dimensionality reduction

One very important area of application is image processing, in which algorithms are used to detect and isolate various desired portions or shapes (features) of a digitized image or video stream. It is particularly important in the area of optical character recognition.

We have used TESSARACT for Feature Extraction.

3.2.2.1 Tesseract

Tesseract is an optical character recognition engine for various operating systems.[2] It is free software, released under the Apache License, Version 2.0 and development has been sponsored by Google since 2006] Tesseract is considered one of the most accurate open-source OCR engines currently available. Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License, Version 2.0 and development has been sponsored by Google since 2006.Tesseract is considered one of the most accurate open-source OCR engines currently available.

Tesseract was in the top three OCR engines in terms of character accuracy in 1995. It is available for Linux, Windows and Mac OS X, however, due to limited resources it is only rigorously tested by developers under Windows and Ubuntu.

Tesseract up to and including version 2 could only accept TIFF images of simple one-column text as inputs. These early versions did not include layout analysis, and so inputting multi-columned text, images, or equations produced garbled output. Since version 3.00 Tesseract has supported output text formatting, OCR positional information and page-layout analysis. Support for a number of new image formats was added using the Leptonic library. Tesseract can detect whether text is monospaced or proportionally spaced.

If Tesseract is used to process right-to-left text such as Arabic or Hebrew, the results are ordered as though it is left-to-right text. [

Tesseract is suitable for use as a backend and can be used for more complicated OCR tasks including layout analysis by using a frontend such as OCRopus.

Tesseract's output will have very poor quality if the input images are not preprocessed to suit it: Images (especially screenshots) must be scaled up such that the text x-height is at least 20 pixels, any rotation or skew must be corrected or no text will be recognized, low-frequency changes in brightness must be high-pass filtered, or Tesseract's binarization stage will destroy much of the page, and dark borders must be manually removed, or they will be misinterpreted as characters.

WORKING OF TESSERACT:

- It uses feature extraction.
- Each 15 X 15 pixel character is then compared with the characters present in Dictionary Data of tesseract.

Tesseract can produce plain text, PDF, and HTML output. Tesseract's standard output is a plain txt file.

3.2.3 Post Processing

OCR accuracy can be increased if the output is constrained by a lexicon – a list of words that are allowed to occur in a document. This might be, for example, all the words in the English language, or a more technical lexicon for a specific field. This technique can be problematic if the document contains words not in the lexicon, like proper nouns. Tesseract uses its dictionary to influence the character segmentation step, for improved accuracy.

The output stream may be a plain text stream or file of characters, but more sophisticated OCR systems can preserve the original layout of the page and produce, for example, an annotated PDF that includes both the original image of the page and a searchable textual representation.

The text returned by algorithm may contains mistakes. Before pass it to human adjustment, we apply some heuristic text-level processing on it to avoid as much manual modification as possible. A frequent word list is first generated just like what we did in IV-C. Then we tokenize the input text using all punctuation as delimiter. For each token, we calculate its edit distance between all the candidates in the frequent word list. If the distance is below some threshold, the token will be replaced by the candidate. The threshold is piecewise so that for the words with shorter length, the threshold is lower. Such method does have drawbacks. For example, since 'int' is in frequent word list, if user has a variable named 'ant', it will be replaced. Overall the post-processing helps, and more sophisticated method can be tried in the future

Image post processing enhances the quality of a finished image to prepare it for publication and distribution. It includes techniques to clean up images to make them visually clearer as well as the application of filters and other treatments to change the look and feel of a picture. Image post processing can also involve removing things from the frame when they don't belong or distract.

By this the technique we can recover the correct character even if it is not present in candidate list produced by the OCR system. Here some heuristic algorithms are applied to the retrieved text. The general idea is to make the text looks more like a piece of a code**.**

3.2.4 Compilation

- Once the code is recognized. It is sent to the Compilation Plug inning this area the user can check the code and manually adjust the minor errors due recognition
- After manual checking it can be compiled.

3.3 HARDWARE AND SOFTWARE SPECIFICATIONS

3.3.1 Software Requirements

Language used: JAVA

Operating system: Windows 7

IDE used: Netbeans 8.0

3.3.2 Hardware Requirements

Processor: Dual Core Intel I3

RAM: 2 GB RAM

Graphic Adapter: 8 bit graphic adapter and Display

3.5. APPLICATIONS

OCR engines have been developed into many kinds of object-oriented OCR applications, such as receipt OCR, invoice OCR, and check OCR, legal billing document OCR.

They can be used for:

1. Data entry for business documents, e.g. check, passport, invoice, bank statement and receipt
2. Automatic number plate recognition
3. Automatic insurance documents key information extraction
4. Extracting business card information into a contact list
5. More quickly make textual versions of printed documents, e.g. book scanning for Project Gutenberg
6. Make electronic images of printed documents searchable, e.g. Google Books
7. Converting handwriting in real time to control a computer (pen computing)
8. Defeating CAPTCHA anti-bot systems, though these are specifically designed to prevent OCR Assistive technology for blind and visually impaired users

Practical OCR Applications

In recent years, OCR (Optical Character Recognition) technology has been applied throughout the entire spectrum of industries, revolutionizing the document management process. OCR has enabled scanned documents to become more than just image files, turning into fully searchable documents with text content that is recognized by computers. With the help of OCR, people no longer need to manually retype important documents when entering them into electronic databases. Instead, OCR extracts relevant information and enters it automatically. The result is accurate, efficient information processing in less time.

Banking

The uses of OCR vary across different fields. One widely known OCR application is in banking, where OCR is used to process checks without human involvement. A check can be inserted into a machine, the writing on it is scanned instantly, and the correct amount of money is transferred. This technology has nearly been perfected for printed checks, and is fairly accurate for handwritten checks as well, though it occasionally requires manual confirmation. Overall, this reduces wait times in many banks.

Legal

In the legal industry, there has also been a significant movement to digitize paper documents. In order to save space and eliminate the need to sift through boxes of

paper files, documents are being scanned and entered into computer databases. OCR further simplifies the process by making documents text-searchable, so that they are easier to locate and work with once in the database.

Healthcare

Healthcare has also seen an increase in the use of OCR technology to process paperwork. Healthcare professionals always have to deal with large volumes of forms for each patient, including insurance forms as well as general health forms. To keep up with all of this information, it is useful to input relevant data into an electronic database that can be accessed as necessary. Form processing tools, powered by OCR, are able to extract information from forms and put it into databases, so that every patient's data is promptly recorded. As a result, healthcare providers can focus on delivering the best possible service to every patient.

OCR in Other Industries

OCR is widely used in many other fields, including education, finance, and government agencies. OCR has made countless texts available online, saving money for students and allowing knowledge to be shared. Invoice imaging applications are used in many businesses to keep track of financial records and prevent a backlog of payments from piling up. In government agencies and independent organizations, OCR simplifies data collection and analysis, among other processes. As the technology continues to develop, more and more applications are found for OCR technology, including increased use of handwriting recognition. Furthermore, other technologies related to OCR, such as barcode recognition, are used daily in retail and other industries

3.6. CODE

INDEX.JSP

```
<%@page contentType="text/html" pageEncoding="UTF-8"%>

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=ISO-8859-1">

<title>File Upload</title>

</head>

<center>

<body style="background-image: url(img/thumb-1920-587777.png)">

<form method="post" action="UploadServlet" enctype="multipart/form-data">

   <h1>Select file to upload:</h1>

</br>

<input type="file" class='myButton' name="dataFile" id="fileChooser"/>

<style>

.myButton {

        -moz-box-shadow: 0px 10px 14px -7px #f5978e;

        -webkit-box-shadow: 0px 10px 14px -7px #f5978e;

        box-shadow: 0px 10px 14px -7px #f5978e;

        background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #f24537), color-stop(1, #c62d1f));

        background:-moz-linear-gradient(top, #f24537 5%, #c62d1f 100%);

        background:-webkit-linear-gradient(top, #f24537 5%, #c62d1f 100%);

        background:-o-linear-gradient(top, #f24537 5%, #c62d1f 100%);
```

```
	background:-ms-linear-gradient(top, #f24537 5%, #c62d1f 100%);
	background:linear-gradient(to bottom, #f24537 5%, #c62d1f 100%);
	filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#f24537', endColorstr='#c62d1f',GradientType=0);
	background-color:#f24537;
	-moz-border-radius:8px;
	-webkit-border-radius:8px;
	border-radius:8px;
	display:inline-block;
	cursor:pointer;
	color:#ffffff;
	font-family:Arial;
	font-size:20px;
	font-weight:bold;
	padding:13px 32px;
	text-decoration:none;
	text-shadow:0px 1px 0px #810e05;
}
.myButton:hover {
	background:-webkit-gradient(linear, left top, left bottom, color-stop(0.05, #c62d1f), color-stop(1, #f24537));
	background:-moz-linear-gradient(top, #c62d1f 5%, #f24537 100%);
	background:-webkit-linear-gradient(top, #c62d1f 5%, #f24537 100%);
	background:-o-linear-gradient(top, #c62d1f 5%, #f24537 100%);
	background:-ms-linear-gradient(top, #c62d1f 5%, #f24537 100%);
```

```
        background:linear-gradient(to bottom, #c62d1f 5%, #f24537 100%);
        filter:progid:DXImageTransform.Microsoft.gradient(startColorstr='#c62d1f',
endColorstr='#f24537',GradientType=0);
        background-color:#c62d1f;
}
.myButton:active {
        position:relative;
        top:1px;
}
</style>
<br/><br/>
<input type="submit" class='myButton' value="Upload" />
</form>
</center></body>
</html>
```

FILEUPLOAD.JAVA

```
import java.io.File;

import java.io.IOException;

import java.util.List;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import org.apache.commons.fileupload.FileItem;

import org.apache.commons.fileupload.disk.DiskFileItemFactory;

import org.apache.commons.fileupload.servlet.ServletFileUpload;


/**

 * Servlet to handle File upload request from Client

 * @author Javin Paul

 */

public class UploadServlet extends HttpServlet {

   private final String UPLOAD_DIRECTORY = "C:/Tesseract-OCR";

 public  static String name = "";

   @Override

   protected void doPost(HttpServletRequest request, HttpServletResponse response)

         throws ServletException, IOException {


      //process only if its multipart content

      if(ServletFileUpload.isMultipartContent(request)){
```

```
        try {
            List<FileItem> multiparts;
                                        multiparts = new ServletFileUpload( new
DiskFileItemFactory()).parseRequest(request);

            for(FileItem item : multiparts){
               if(!item.isFormField()){
                  File input;
                   input = new File(item.getName());

                   name = input.getName();
                   item.write( new File(UPLOAD_DIRECTORY + File.separator + name));
               }
            }

           //File uploaded successfully
           request.setAttribute("message", "File Uploaded Successfully");

        } catch (Exception ex) {
           request.setAttribute("message", "File Upload Failed due to " + ex);
        }

     }else{
        request.setAttribute("message",
                        "Sorry this Servlet only handles file upload request");
```

```
        }

        request.getRequestDispatcher("/result.jsp").forward(request, response);

    }

}
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */
```

FEATUREEXTRACTION.JAVA

```
import java.io.BufferedReader;

import java.io.IOException;

import java.io.InputStreamReader;

import java.io.PrintWriter;

import javax.servlet.RequestDispatcher;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


/**

 *

 * @author shaivya chandra

 */

public class NewServlet extends HttpServlet {


    /**

     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>

     * methods.

     *

     * @param request servlet request

     * @param response servlet response

     * @throws ServletException if a servlet-specific error occurs

     * @throws IOException if an I/O error occurs
```

```
    */
     protected void processRequest(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
      response.setContentType("text/html;charset=UTF-8");
      PrintWriter out = response.getWriter();
      try {

        String upload_file = UploadServlet.name;
        String cm = "tesseract "+upload_file+" toberead";
        Grayscale.main();
        BlackWhite.main();



      ProcessBuilder builder = new ProcessBuilder(
        "cmd.exe", "/c", "cd \"C:\\Tesseract-OCR\" && "+cm);
      builder.redirectErrorStream(true);
      Process p = builder.start();
                                BufferedReader r = new BufferedReader(new InputStreamReader(p.getInputStream()));
      String line;
      while (true) {
        line = r.readLine();
        if (line == null) { break; }
```

```
            System.out.println(line);
        }
        RequestDispatcher rd=request.getRequestDispatcher("read");
                    rd.forward(request,response);




    } catch(Exception e) {
        System.out.println(e.toString());
        e.printStackTrace();
    }
         finally {
            out.close();
        }
    }


      // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the + sign on the left to edit the code.">
    /**
     * Handles the HTTP <code>GET</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
```

```
     */
    @Override
    protected void doGet(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Handles the HTTP <code>POST</code> method.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */
    @Override
    protected void doPost(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
```

```
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

GRAYSCALE.JAVA

```
import java.awt.Color;
import java.io.File;
import java.io.IOException;
import java.awt.image.BufferedImage;
import java.util.Arrays;
import javax.imageio.ImageIO;

public class Grayscale{

    /**
     *
     * @throws IOException
     */static File f = new File("C:\\Tesseract-OCR\\"+UploadServlet.name);
     static BufferedImage img;
    public static int main()throws IOException{
```

```
//read image
try{
 // noise();
  img = ImageIO.read(f);
}catch(IOException e){
  System.out.println(e);
}

//get image width and height
int width = img.getWidth();
int height = img.getHeight();

//convert to grayscale
for(int y = 0; y < height; y++){
  for(int x = 0; x < width; x++){
    int p = img.getRGB(x,y);

    int a = (p>>24)&0xff;
    int r = (p>>16)&0xff;
    int g = (p>>8)&0xff;
    int b = p&0xff;

    //calculate average
    int avg = (r+g+b)/3;
```

```
      //replace RGB value with avg
      p = (a<<24) | (avg<<16) | (avg<<8) | avg;

      img.setRGB(x, y, p);
    }
   }

   //write image
   try{
    f = new File("C:\\Tesseract-OCR\\"+UploadServlet.name);
    ImageIO.write(img, "jpg", f);

   }catch(IOException e){
    System.out.println(e);
   }
  return 1;}//main() ends here

/*
 * Logic: Captures the colour of 8 pixels around the target pixel.Including the target pixel there will be 9 pixels.
 *          Isolate the R,G,B values of each pixels and put them in an array.Sort the arrays.Get the Middle value of the array
 *          Which will be the Median of the color values in those 9 pixels.Set the color to the Target pixel and move on!
 */
```

```
public static void noise() throws IOException{
                                //Input Photo File
    Color[] pixel=new Color[9];
    int[] R=new int[9];
    int[] B=new int[9];
    int[] G=new int[9];



     img = ImageIO.read(f);
    for(int i=1;i<img.getWidth()-1;i++)
       for(int j=1;j<img.getHeight()-1;j++)
       {
          pixel[0]=new Color(img.getRGB(i-1,j-1));
          pixel[1]=new Color(img.getRGB(i-1,j));
          pixel[2]=new Color(img.getRGB(i-1,j+1));
          pixel[3]=new Color(img.getRGB(i,j+1));
          pixel[4]=new Color(img.getRGB(i+1,j+1));
          pixel[5]=new Color(img.getRGB(i+1,j));
          pixel[6]=new Color(img.getRGB(i+1,j-1));
          pixel[7]=new Color(img.getRGB(i,j-1));
          pixel[8]=new Color(img.getRGB(i,j));
          for(int k=0;k<9;k++){
             R[k]=pixel[k].getRed();
             B[k]=pixel[k].getBlue();
```

```
                G[k]=pixel[k].getGreen();
            }
            Arrays.sort(R);
            Arrays.sort(G);
            Arrays.sort(B);
            img.setRGB(i,j,new Color(R[4],B[4],G[4]).getRGB());
          }
       ImageIO.write(img,"jpg",f);
    }

}//class ends here
```

BINARIZATION.JAVA

```
import java.awt.Color;

import java.awt.image.BufferedImage;

import java.io.File;

import java.io.IOException;

import javax.imageio.ImageIO;



public class BlackWhite {


public static void main()

{

try

{

BufferedImage original = ImageIO.read(new File(UploadServlet.name));

BufferedImage binarized = new BufferedImage(original.getWidth(), original.getHeight(),BufferedImage.TYPE_BYTE_BINARY);


int red;

int newPixel;

int threshold =230;


for(int i=0; i<original.getWidth(); i++)

{
```

```
for(int j=0; j<original.getHeight(); j++)
{

// Get pixels
red = new Color(original.getRGB(i, j)).getRed();

int alpha = new Color(original.getRGB(i, j)).getAlpha();

if(red > threshold)
{
newPixel = 0;
}
else
{
newPixel = 255;
}
newPixel = colorToRGB(alpha, newPixel, newPixel, newPixel);
binarized.setRGB(i, j, newPixel);

}
}
ImageIO.write(binarized, "jpg",new File(UploadServlet.name) );
}
catch (IOException e)
{
```

```
e.printStackTrace();
}
}

private static int colorToRGB(int alpha, int red, int green, int blue) {
int newPixel = 0;
newPixel += alpha;
newPixel = newPixel << 8;
newPixel += red; newPixel = newPixel << 8;
newPixel += green; newPixel = newPixel << 8;
newPixel += blue;

return newPixel;
}
}
```

COMPILER.JSP

```
<html>
   <head>
      <meta http-equiv="Content-Type" content="text/html; charset=UTF-8">
      <title>JSP Page</title>
   </head>
   <body style="background-image: url(img/Q8Wq94.png)">
      <table>
       <tr>
            <td width="700px">
         <% String userlabel=(String)request.getAttribute("result"); %>
         <h1 style="color: whitesmoke"> Result : </h1>
         <h3 style="color: whitesmoke"><%=userlabel%> </h3>
            </td><td></td></tr>
         <tr><td></td><td>
                        <iframe frameborder="0" height="500px" width="700px"
src="https://www.compilejava.net/" >
         </iframe>
            </td></tr>
      </table>
```

READ.JAVA

```
/*
 * To change this license header, choose License Headers in Project Properties.
 * To change this template file, choose Tools | Templates
 * and open the template in the editor.
 */

import java.io.BufferedReader;
import java.io.FileReader;
import java.io.IOException;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.PrintWriter;
import javax.servlet.RequestDispatcher;
import javax.servlet.ServletContext;
import javax.servlet.ServletException;
import javax.servlet.http.HttpServlet;
import javax.servlet.http.HttpServletRequest;
import javax.servlet.http.HttpServletResponse;

/**
 *
 * @author shaivya chandra
 */
public class read extends HttpServlet {
```

```
    /**
     * Processes requests for both HTTP <code>GET</code> and <code>POST</code>
     * methods.
     *
     * @param request servlet request
     * @param response servlet response
     * @throws ServletException if a servlet-specific error occurs
     * @throws IOException if an I/O error occurs
     */public static String curline = " ";
     public static String curlin = " ";
    private static final String filepath = "C:\\Tesseract-OCR\\toberead.txt";
        protected void processRequest(HttpServletRequest request, HttpServletResponse response)
            throws ServletException, IOException {
        response.setContentType("text/html;charset=UTF-8");
        PrintWriter out = response.getWriter();
        BufferedReader br;

        try {
            curlin = "";
            curline= "";
            out.print("<html>");

           br = new BufferedReader(new FileReader(filepath));
```

```
                while ((curlin = br.readLine()) != null) {


            curline = curline + " \n " +curlin;


            out.print("<body>");
            out.print(" \n"+curlin);
                }
        out.print("</body>");
        out.print("</html>");
request.setAttribute("result",curline);
                br.close();


    }
    catch (IOException e) {
                e.printStackTrace();
    }
     RequestDispatcher rd=request.getRequestDispatcher("compile.jsp");
            rd.forward(request,response);
       out.close();
    }



    // <editor-fold defaultstate="collapsed" desc="HttpServlet methods. Click on the +
sign on the left to edit the code.">
```

```
/**
 * Handles the HTTP <code>GET</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doGet(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
    processRequest(request, response);
}

/**
 * Handles the HTTP <code>POST</code> method.
 *
 * @param request servlet request
 * @param response servlet response
 * @throws ServletException if a servlet-specific error occurs
 * @throws IOException if an I/O error occurs
 */
@Override
protected void doPost(HttpServletRequest request, HttpServletResponse response)
        throws ServletException, IOException {
```

```
        processRequest(request, response);
    }

    /**
     * Returns a short description of the servlet.
     *
     * @return a String containing servlet description
     */
    @Override
    public String getServletInfo() {
        return "Short description";
    }// </editor-fold>

}
```

3.7 SNAPSHOT

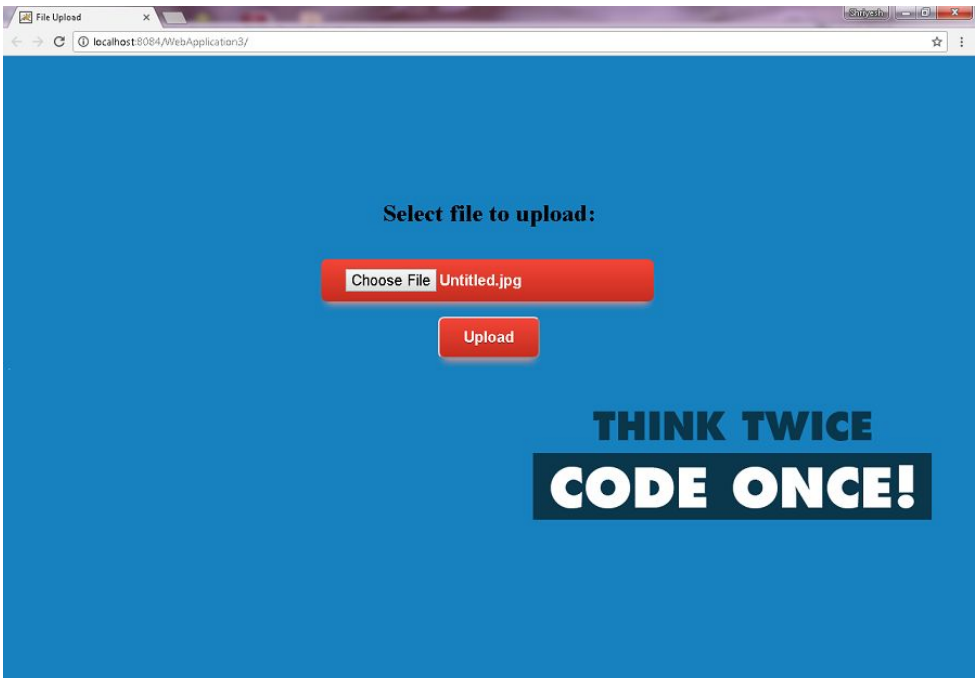

Figure 3.4 Index.jsp

```
import java.io.*;
class abc
{
System.out.print
("Shaivya Raheela");
}
```

Figure 3.5 Input Image

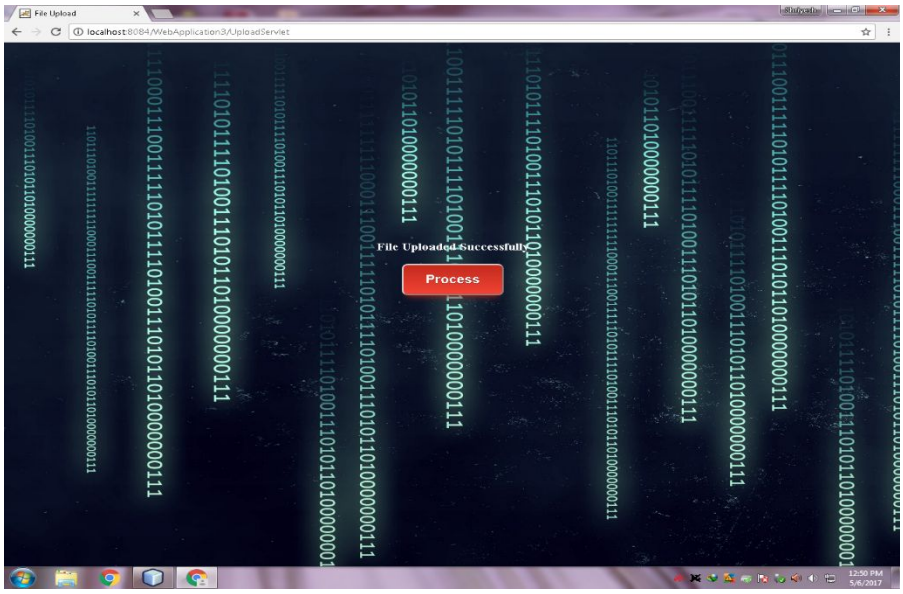

Figure 3.6 File Uploaded

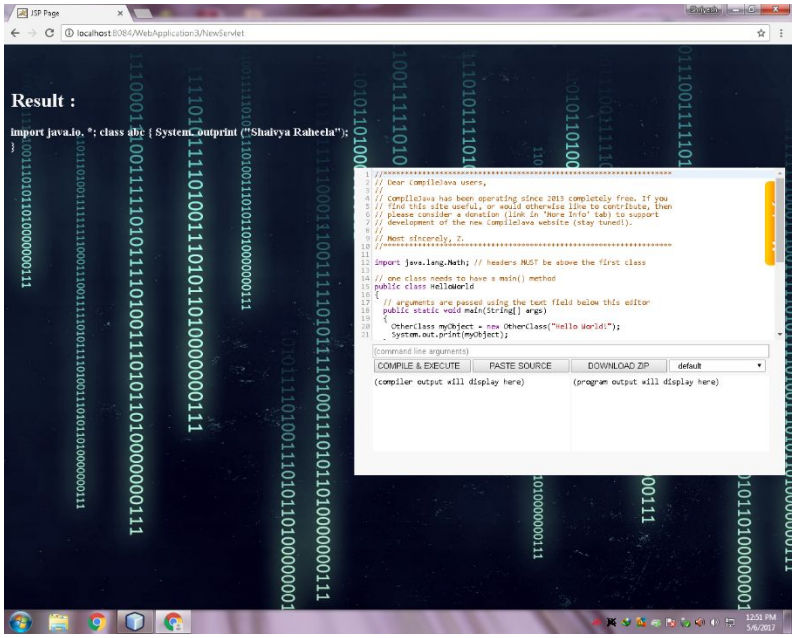

Figure 3.7 Result and Compile

CHAPTER-4

RESULT ANALYSIS AND DISCUSSIONS

4.1 TESTING

The purpose of testing is to discover errors. Testing is the process of trying to discover every conceivable fault or weakness in a work product. It provides a way to check the functionality of components, sub-assemblies, assemblies and/or a finished product It is the process of exercising software with the intent of ensuring that the Software system meets its requirements and user expectations and does not fail in an unacceptable manner. There are various types of test. Each test type addresses a specific testing requirement.

4.2 TYPES OF TESTS

- *Unit testing*

Unit testing involves the design of test cases that validate that the internal program logic is functioning properly, and that program inputs produce valid outputs. All decision branches and internal code flow should be validated. It is the testing of individual software units of the application .it is done after the completion of an individual unit before integration. This is a structural testing, that relies on knowledge of its construction and is invasive. Unit tests perform basic tests at component level and test a specific business process, application, and/or system configuration. Unit tests ensure that each unique path of a business process performs accurately to the documented specifications and contains clearly defined inputs and expected results.

- *Functional test*

Functional tests provide systematic demonstrations that functions tested are available as specified by the business and technical requirements, system documentation, and user manuals.

Functional testing is centred on the following items:

Valid Input : identified classes of valid input must be,

Accepted.

Invalid Input : identified classes of invalid input must be Rejected.

Functions : identified functions must be exercised.

Output : identified classes of application outputs.

Systems/Procedures: interfacing systems or procedures must be invoked.

Organization and preparation of functional tests is focused on requirements, key functions, or special test cases. In addition, systematic coverage pertaining to identify Business process flows; data fields, predefined processes, and successive processes must be considered for testing. Before functional testing is complete, additional tests are identified and the effective value of current tests is determined.

- *System Test*

System testing ensures that the entire integrated software system meets requirement. It tests a configuration to ensure known and predictable results. An example of system testing is the configuration oriented system integration test. System testing is based on process descriptions and flows, emphasizing pre-driven process links and integration points.

- *White Box Testing*

White Box Testing is a testing in which in which the software tester has knowledge of the inner workings, structure and language of the software, or at least its purpose. It is purpose. It is used to test areas that cannot be reached from a black box level.

- *Black Box Testing*

Black Box Testing is testing the software without any knowledge of the inner workings, structure or language of the module being tested. Black box tests, as most other kinds of

tests, must be written from a definitive source document, such as specification or requirements document, such as specification or requirements document.

4.2.1 TEST OBJECTIVES

- All field entries must work properly.
- Pages must be activated from the identified link.
- The entry screen, messages and responses must not be delayed.

- *Integration Testing*

Software integration testing is the incremental integration testing of two or more integrated software components on a single platform to produce failures caused by interface defects. The task of the integration test is to check that components or software applications, e.g. components in a software system or – one step up – software applications at the company level – interact without error.

- *Acceptance Testing*

User Acceptance Testing is a critical phase of any project and requires significant participation by the end user. It also ensures that the system meets the functional requirements.

Test Results: All the test cases mentioned above passed successfully. No defects encountered.

4.3 ADVANTAGES AND LIMITATIONS

4.3.1 Advantages

- OCR processing makes your documents 100% text-searchable which is a huge advantage to be able to search through your documents for names, reference numbers, addresses etc.

- OCR saves you a considerable amount of time compared to searching through your paper based documents in a complicated filing service. With a quick computer-based search, you can find the file you are looking for in seconds.

- Going by that old phrase, time is money, OCR will also save your organisation a lot of money.

- OCR can massively improve your customer services. If you take incoming calls which require you to access documents then having those document available instantly in digital form can make the overall customer experience better due to the speed of searching for the files they need and the ability to edit their contents easily.

- OCR can allow documents to be made editable. We can convert to Microsoft Word and other editable formats so, if contents need changing, they can be changed easily.

- OCR allows you to copy and paste from the document itself whether that's in PDF format or MS Word format.

- OCR is considerably more accurate than it used to be so you can expect precision. If you were put off by the lack of speed of the old OCR software, you should give it another try and see just how much has improved.

- OCR is low cost and is one of the few services which can improve the way your business operates as well as save you space and money, all for that very low cost.

- OCR is also known to boost staff morale when their working environment is easier to work within and less paper-centric.

4.3.2 Limitations

Although Optical Character Recognition (OCR) scanning technology has increased rapidly over the years, there are, however, limitations in regards to the source materials and character formatting.

- Text from a source with a font size of less than 12 points will results in more errors.

- Scanning of plain text files or spreadsheet print outs usually work, however the data needs to be reformatted to match the original.

- Document formatting may be lost during text scanning (i.e., bold, italic & underline are not always recognized).

- Not worth doing for small amount of text.
- Cursive handwriting is difficult to be recognized
- We do not get 100% accurate results there may be some mistakes.
- Image used should be of very high quality.

CHAPTER- 5
CONCLUSION

Optical Code Recognition system is an application which can identify the handwritten code to a compliable source code. Typing code ties a programmer to their laptop. This is problematic for lecturers because it reduces mobility, reduces eye contact and engagement with the class, and makes it difficult and unnatural for the lecturer to point out things in the code.

Current prevalent OCR-engines can recognize printed text with high accuracy, but can hardly handle hand-written text very well. The difficulty of hand-written text recognition is due to variation of characters and poor alignment of text line. Therefore, we design to achieve a workable solution for hand-written code recognition system.

The project has been implemented and it concludes that the characters can be recognized if the image is correctly aligned and properly executed. The recognized characters are displayed in separate window where it is in the text format. The region of interest and the background of the image must have optimum colour difference. The system recognizes the exact character using feature matching between the extracted character and the template of all characters as a measure of similarity.

Accuracy rates can be measured in several ways, and how they are measured can greatly affect the reported accuracy rate. For example, if word context (basically a lexicon of words) is not used to correct software finding non-existent words, a character error rate of 1% (99% accuracy) may result in an error rate of 5% (95% accuracy) or worse if the measurement is based on whether each whole word was recognised with no incorrect

letters. Web-based OCR systems for recognising hand-printed text on the fly have become well known as commercial products in recent years.

CHAPTER-6

FUTURE WORK

In general, the OCR System is functional and can recognize most code. However, there are still several drawbacks that can be improved in the future. First thing is its poor robustness against rotation. I've tried Hough transform to adjust skewness, but it doesn't produce ideal result. One way to fix it is to detect and draw text line on image before apply Hough Transform.

One possible way is first detect centroid of each character and use a straight line to fit them. Another issue is that currently OCR System cannot take in any input during running time, so its application is very limited. More interactive activity should be added to the app to deal with this case. Meanwhile, the post-processing part can also be improved.

Recognition of cursive text is an active area of research, with recognition rates even lower than that of hand-printed text. Higher rates of recognition of general cursive script will likely not be possible without the use of contextual or grammatical information. For example, recognising entire words from a dictionary is easier than trying to parse individual characters from script. Reading the *Amount* line of a cheque (which is always a written-out number) is an example where using a smaller dictionary can increase recognition rates greatly. The shapes of individual cursive characters themselves simply do not contain enough information too accurately (greater than 98%) recognise all handwritten cursive script.

For example, since a variable cannot be used before it's declared in C, if we can incorporate knowledge like this into recognition, the accuracy would be further boosted. The current recognition is limited to my own handwriting due to lack of training examples. If we want to generalize it to recognize handwriting from arbitrary people, more training example is needed. The last issue is the generalization to other programming language. One possible challenge here is the recognition of indentation, because some language like Java relies on indentation for interpretation. The Tesseract alone cannot recognize indentation accurately, so some other method should be introduced to handle it.

APPENDIX-A

LIST OF FIGURES

REFERENCES

[1] Gonzalez, Brian M. "Iris : A Solution for Executing Handwritten Code." *Iris : A Solution for Executing Handwritten Code*. University of Agder, 1 June 2012. Web. 06 June 2014.

[2] Rakshit, Sandip. "ArXiv.org Cs ArXiv:1003.5898." *[1003.5898] Recognition of Handwritten Roman Numerals Using Tesseract Open Source OCR Engine*. Jadavpur University, 30 May 2010. Web. 06 June2014.

[3] Eric Thong "Codeable: Generating Compilable Source Code from Handwritten Source Code" EE 368 Course project

[4] https://code.google.com/p/tesseract-ocr/

[5] https://code.google.com/p/tesseract-ocr/wiki/TrainingTesseract3

[6] http://pp19dd.com/tesseract-ocr-chopper/

[7] Vijay Laxmi Sahu, Babita Kubde Offline Handwritten Character Recognition Techniques using Neural Network: A Review International Journal of Science and Research (IJSR), India Online ISSN: 2319-7064 Volume 2 Issue 1, January 2013

[8] Gurpreet Singh Chandan Jyoti Kumar Rajneesh Rani Dr. Renu Dhir," Feature Extraction of Gurmukhi Script and Numerals: A Review of Offline Techniques" IJARCSSE Volume 3, Issue 1, January 2013 pp 257-263

[9] Majida Ali Abed, Hamid Ali Abed Alasadi," Simplifying Handwritten Characters Recognition Using a Particle Swarm Optimization Approac h" European Academic Research, Vol. I, Issue 5/ August 2013 pp-532-552

[10] Argha Roy, Diptam Dutta, Kaustav Choudhury," Training Artificial Neural Network using Particle Swarm

[11] Optimization Algorithm" IJARCS SE Volume 3, Issue 3, March 2013 pp 43—434

[12] Amir Bahador Bayat Recognition of Handwritten Digits Using Optimized Adaptive Neuro-Fuzzy Inference Systems and Effective Features Journal of Pattern Recognition and Intelligent Systems Aug. 2013, Vol. 1

[13] Swagatam Das, Arijit Biswas, Sambarta Dasgupta, and Ajith Abraham Bacterial Foraging Optimization Algorithm: Theoretical Foundations, Analysis, and Applications Hindawi Publishing Corporation Applied Computational Intelligence and Soft Computing Volume 2012, Article ID 897127.