OCR Engine

Optical character recognition

(Built in JAVA using Tesseract)

Yash Patel [1124832]
MS. Computer Science,
New York Institute of Technology
New York, USA.

Poojan Patel [1098083]
MS. Computer Science,
New York Institute of Technology
New York, USA.

Ishita Lad [1116445]
MS. Computer Science,
New York Institute of Technology
New York, USA.

Hamali Patel [1123186]
MS. Computer Science,
New York Institute of Technology
New York, USA.

Abstract

This paper represent a development and deployment and/or Implementation of Optical Character Recognition (OCR) to translate images of typewritten or handwritten characters into electronically editable format by preserving font properties. OCR can do this by applying pattern matching algorithm. The Recognized characters are stored in editable format. Thus OCR make the computer read the printed documents discarding noise.

Keywords- Optical Character Recognition, Image convert to character, Image cropping.

Introduction

Optical character recognition (optical character reader) (OCR) is the mechanism to convert images of handwritten or printed text into encoded editable text, whether from a scanned document, a photo of a document. The application domains where it uses for maintaining data entry from printed paper data records, e.g. Passport, documents, invoices, bank statements, computerized receipts, business cards, mail.

It is a very convenient method of digitizing printed texts so that it can be edited, searched, stored more compactly.

It is used in machine processes such as cognitive computing, machine translation, and text-to-speech. OCR is a field of research in pattern recognition, artificial intelligence and computer vision.

OCR are recognition engines used for imaging. OCR have largely develop independent standards and metrics. In the Recent years, the advancement of pattern recognition has accelerated due to many applications. Determining what text is in an image is a difficult task. Optical Recognition is more demanding and it is an evident for the pattern advancement. OCR software identifies character by comparing shapes to those stored in library. OCR is speedy and most cost effective method. OCR recognizes wide varieties of fonts. OCR provides an alpha-numeric recognition of printed or handwritten recognition. OCR focusses on accuracy.

1. History and Motivations

Initially might be optical character recognition invented to create reading devices for the blind. In 1914, Emanuel Goldberg developed a machine that read characters and converted them into standard telegraph code [citation needed]. In the late 1920s and into the 1930s he developed "Statistical Machine" for searching microfilm archives using an optical code recognition system. In 1931 he was acquired the invention patent by IBM.[3]

Tesseract was an open-source OCR engine developed at HP between 1984 and 1994.After a joint project between HP Labs Bristol, and
HP's scanner division in Colorado, Tesseract had a significant lead in accuracy over the commercial engines, but did not become a product. The next stage of its development was back in HP Labs Bristol as an investigation of OCR for compression. Work concentrated more on improving rejection efficiency than on base-level accuracy. At the end of this project, at the end of 1994, development ceased entirely. In late 2005, HP released Tesseract for open source. It is now available at http://code.google.com/p/tesseract-ocr.[6]

Ease of use : Related work

2. Existing Methodology

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 only Windows and Ubuntu are rigorously tested by developers. Tesseract does not come with a GUI and is instead run from the command-line interface. There are many separate projects which provide a GUI for Tesseract. One notable example is **OCRFeeder**.

OCRFeeder is an optical character recognition suite for GNOME, which also supports virtually any command-line OCR engine, such as Cuneiform, GOCR, Ocrad and **Tesseract**. It converts paper documents to digital document files and can serve to make them accessible to visually impaired users.[4]

Figure.1

OCRFeeder is free and open-source software subject to the terms of the GNU General Public License (GPL) version 3 or later. It is available for Linux and other Unix-like operating systems.

So the motivation behind this project is to learn in depth about OCR GUI system using existing methodology and explore the functionality for the same.

https://www.youtube.com/watch?v=GETE8KfkbKA Watch video to understand the existing tool built for GNOME Linux. So after deploying this project to make this tool platform independent.

3. Problem Definition

In this section, the research problem that is addressed in this paper will be described in more details, including the module as well and next generation implementation.

3.1. How does Tesseract OCR work

OCR program developed in java because we need platform independency. We are here using existing methodology to deploy the OCR system to make more convenient and user friendly as well. So here we are using *Tesseract API* to make this implementation simple. To summarize this whole process will be like below figure.

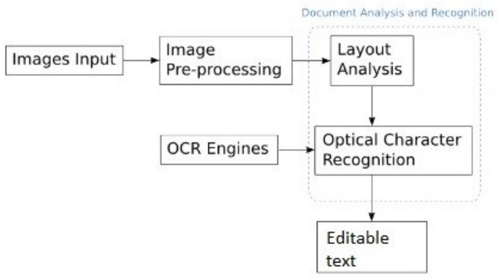


Figure.2

Tesseract OCR engine `libtesseract` library contains

1. ghost4j-0.3.1.jar
2. jai_imageio.jar
3. jna.jar
4. junit-4.10.jar
5. tools.jar

Packages built in for use. Developer can directly deployed their functionality using these packages as API.[5]

3.2. Study the Methodology of Tesseract

Tess4J is a Java Native Access (JNA) wrapper for Tesseract OCR API; it provides character recognition support for common image formats, and multi-page images. The library has been developed and tested on Windows and Linux.

3.3. Tesseract Algorithm

3.3.1. Line Finding Algorithm

The line finding algorithm is one of the few part of Tesseract that has previously been published. The line finding algorithm is designed so that a skewed page can be recognized without having to de-skew, thus saving loss of quality because of process are blob filtering and line construction. Assuming that page layout analysis has already provided text regions of roughly uniform text size, a simple percentile height filter removes drop-caps and vertically touching characters. The median height approximates the text size in the region, so it is safe to filter out blobs that are smaller than some fraction of the median height, being most likely punctuation, marks and noise.[2]

3.3.2. Baseline fitting

Once the text lines have been found, the baselines are fitted more precisely using quadratic spline. This was enabled Tesseract to handle pages with curved baselines. Which are a common artifact in scanning and not just at book bindings. The baseline are fitted by partitioning the blobs into group with reasonable continuous displacement for the original straight baseline. A quadratic spline is fitted to the most populous partition be a least squares fit.

3.3.3. Fixed pitch detection and chopping

Tesseract tests the text lines to determine whether they are fixed pitch. Where it finds pitch text, Tesseract chops the words into character using the pitch, and disables the chopper and association on these words for the word recognition step.[1]

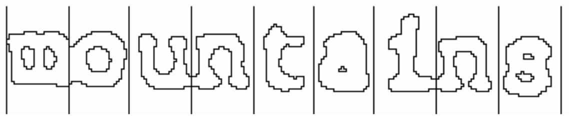


Figure.3

3.3.4. **Proportional word finding**

Non-fixed-pitch or proportional text spacing is a highly non-trivial task. Tesseract solves most of the problems by measuring gaps in a limited vertical range between the baseline and mean line. Space that are close to the threshold at this stage are made fuzzy. So the final step can be made after word recognition.

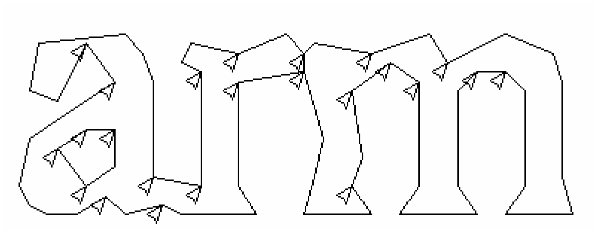


Figure.4

While the result from a word is unsatisfactory, Tesseract attempts to improve the results by chopping the blob with worst confidence form the character classifier. [1]

We are implementing java program to make Tesseract simpler and user friendly using this Tesseract OCR API. Deployed code can be easily make OCR process simpler and it also uses GUI rather than command line for platform independency.

4. **Code implementation (use of Tesseract OCR API)**

Software requirement: **-** We were using Java development tool-kit and Java Run time environment of amd64 (windows – 64 bit) for deploying OCR engine in windows 64 bit operating system. And also used **libtesseract302.dll** and **liblept168.dll** of 64-bit version.

Tesseract API contains two built in packages Package *net.sourceforge.tess4j* and *net.sourceforge.vietocr* in it.[5]

The whole package is now available at this link :
http://tenet.dl.sourceforge.net/project/tess4j/tess4j/3.1/Tess4J-3.1-src.zip
Extract it and you can use this source to deploy OCR.
net.sourceforge.tess4j package contains Tesseract API a Java wrapper for Tesseract OCR DLL using JNA Interface Mapping.
Public interface **TessDllAPI** extends ***com.sun.jna.Library***.

public class EANYCODE_CHAR extends com.sun.jna.Structure, It should be noted that the format for char_code for version 2.0 and beyond is UTF-8, which means that ASCII characters will come out as one structure but other characters will be returned in two or more instances of this structure with a single byte of the UTF-8 code in each, but each will have the same bounding box.

- Programs which want to handle languages with different characters sets will need to handle extended characters appropriately, but all code needs to be prepared to receive UTF-8 coded characters for characters such as bullet and fancy quotes.

Public class **ETEXT_DESC** extends com.sun.jna, Structure Description of the output of the OCR engine. This structure is used as both a progress monitor and the final output header, since it needs to be a valid progress monitor while the OCR engine is storing its output to shared memory.[5]

During progress, all the buffer info is -1. Progress starts at 0 and increases to 100 during OCR. No other constraint. Every progress callback, the OCR engine must set ocr_alive to 1. The HP side will set ocr_alive to 0. Repeated failure to reset to 1 indicates that the OCR engine is dead. If the cancel function is not null, then it is called with the number of user words found. If it returns true, then operation is canceled.

Public class **Tesseract** extends java.lang.Object, An object layer on top of TessDllAPI, provides character recognition support for common image formats, and multi-page TIFF images beyond the uncompressed, binary TIFF format supported by Tesseract OCR engine. The extended capabilities are provided by the Java Advanced Imaging Image I/O Tools.

- Support for PDF documents is available through Ghost4J, a JNA wrapper for GPL Ghost script, which should be installed and included in system path.

- Any program that uses the library will need to ensure that the required libraries (the .jar files for jna, jai-imageio, and ghost4j) are in its compile and run-time class path. It provides **doOCR** method to perform OCR operation for any image file.

net.sourceforge.vietocr package contains java wrapper classes to handle Image operations. It contains ImageHelper, ImageOHelper and PdfUtilities. A Java/.NET GUI frontend for Tesseract OCR engine. Supports optical character recognition for Vietnamese and other languages supported by Tesseract.

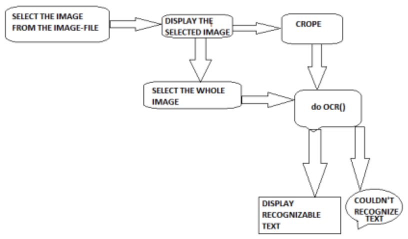

SELECT THE IMAGE FROM THE IMAGE-FILE
DISPLAY THE SELECTED IMAGE
CROPE
SELECT THE WHOLE IMAGE
do OCR()
DISPLAY RECOGNIZABLE TEXT
COULDN'T RECOGNIZE TEXT

Figure.5

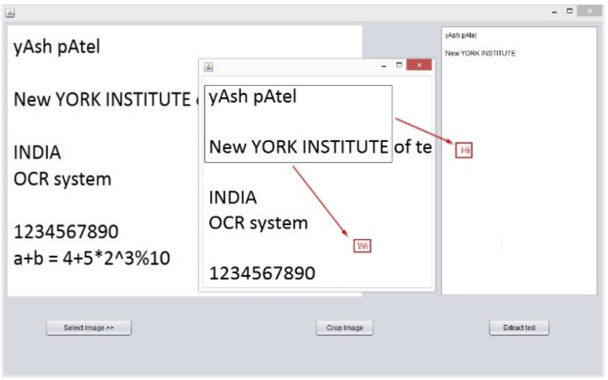
OCR stands for optical character recognition. This software implement based In java language.in this project we can convert image form into text form. For converting image into text form we first of all browse image from the image file than display selected image into textarea.

Then, if we want to recognize whole image or there is also a cropping functionality .with the use of cropping functionality we recognize only those character from the image which we want to display. Than using Tesseract library, we can recognize text using doOCR () function from the image. And display recognizable text into text-box. If the image are not supported for this software Like .pdf than it display message like “character couldn’t recognize”.

Cropping algorithm

Image data: (wi, hi) and define **ri = wi / hi**

Screen resolution: (ws, hs) and define **rs = ws / hs**


yAsh pAtel

New YORK INSTITUTE

INDIA
OCR system

1234567890
a+b = 4+5*2^3%10

Figure.6

Scaled image dimensions:
rs > ri ? (wi * hs/hi, hs) : (ws, hi * ws/wi)

5. Results and Performance Evaluations

As we were trying to run same test cases on existing **OCRFeeder** and on our **OCR engine**. We got some ordinary and expected result sets and evaluations were good.

On performance evaluation criteria our OCR engine as better functionality and also enough accurate as well.
We took some different image format files with variations on font style, size, color and background too.

We got some good result sets which describes below:

Sr.	Test Cases			Manual check	OCRFeeder	OCR Engine
1	Small font size	Black font	White paper	100 %	78 %	80 %
2	Medium font size	Black font	White paper	100 %	97 %	97 %
3	Large font size	Black font	White paper	100 %	90 %	90 %
4	Small font size	White font	Blue paper	100 %	90 %	95 %
5	Medium font size	White font	Blue paper	100 %	97 %	97 %
6	Large font size	White font	Blue paper	100 %	90 %	90 %
7	Small font size	Red font	White paper	100 %	-	-
8	Medium font size	Red font	White paper	100 %	-	-
9	Large font size	Red font	White paper	100 %	-	-
10	Small font size	Black font	yellow paper	100 %	95 %	95 %
11	Medium font size	Black font	yellow paper	100 %	97 %	97 %
12	Large font size	Black font	yellow paper	100 %	95 %	95 %
13	Small font size	White font	Black Paper	100 %	70 %	-
14	Medium font size	White font	Black Paper	100 %	70 %	-
15	Large font size	White font	Black Paper	100 %	70 %	-

Table.1

Analysis: We took some real test cases and analysis some results that would be like this above described in table the existing OCRFeeder can analyze Black color font on white paper normally. But whenever we use other colored font then the results will be changed. We double checked our results with manually checked results. Here we taken manually checked result as 100 % accurate and we are comparing our results with respect to the manual checked results. And we got the proper idea about the combinations of the results. Like we found that our OCR engine cannot detect font when we use white color font on black colored paper layout.

Our OCR Engine can detect handwritten paper document as well as scanned photo copy of document. Our machine cannot detect very fancy font style like Disney font, cookie font, and comic font style. And we can also implement this for any other language, here we were only used English UTF-8 language for simplicity. This API can be implemented for any other language.

The next step will be to export the detected document into MS Word (.docx) to make it portable and more usable. After exporting into word document we check whether our result varied or not.

We Compared the below graphs and at the end we knew that the graph shows that the platform dependency removed but because of that somewhat accuracy also been changed like existing OCRFeeder was not available to detect handwritten documents but our OCR does. summarize this all here demonstrate the overall analysis at the end of this page so it proved that our OCR was good in all cases where we were using black font and it also extracted proper .docx file editable format without corruption of data.[9]

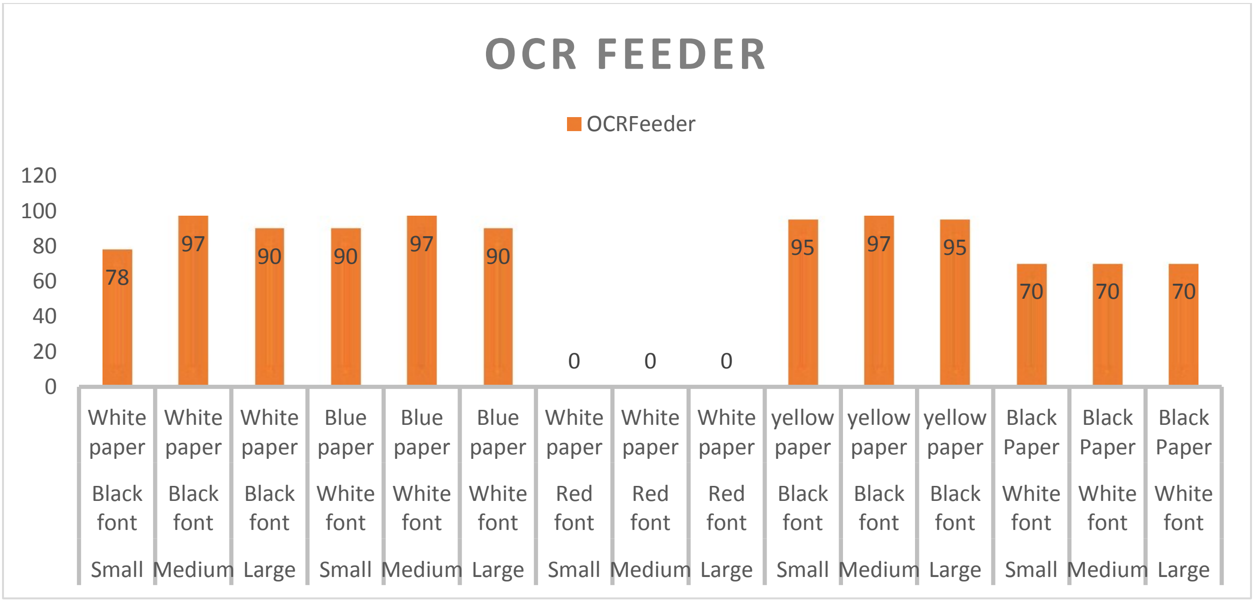


Figure.7

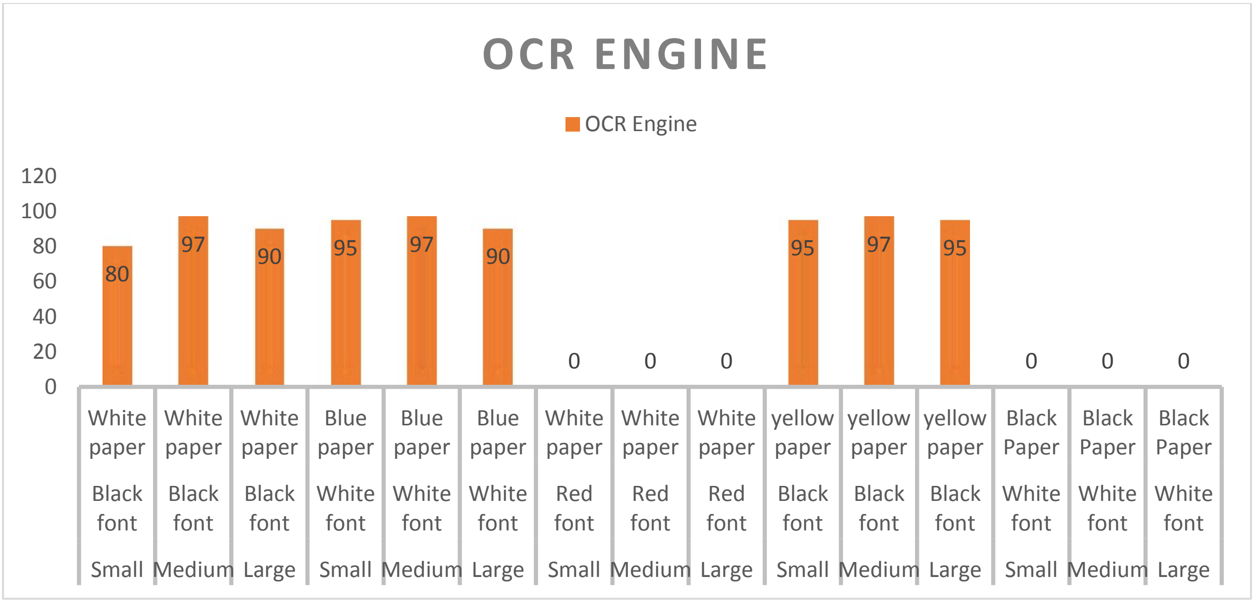
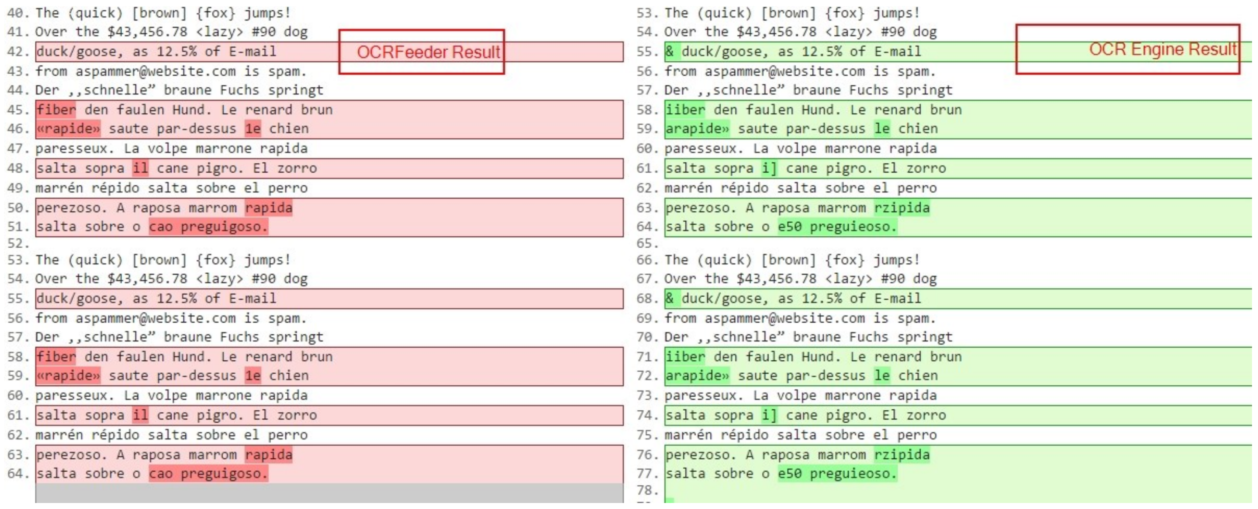


Figure.8



```
The (quick) [brown] {fox} jumps!
Over the $43,456.78 <lazy> #90 dog
duck/goose, as 12.5% of E-mail
from aspammer@website.com is spam.
Der ,,schnelle” braune Fuchs springt
fiber den faulen Hund. Le renard brun
«rapide» saute par-dessus 1e chien
paresseux. La volpe marrone rapida
salta sopra il cane pigro. El zorro
marrén répido salta sobre el perro
perezoso. A raposa marrom rapida
salta sobre o cao preguigoso.

The (quick) [brown] {fox} jumps!
Over the $43,456.78 <lazy> #90 dog
duck/goose, as 12.5% of E-mail
from aspammer@website.com is spam.
Der ,,schnelle” braune Fuchs springt
fiber den faulen Hund. Le renard brun
«rapide» saute par-dessus 1e chien
paresseux. La volpe marrone rapida
salta sopra il cane pigro. El zorro
marrén répido salta sobre el perro
perezoso. A raposa marrom rapida
salta sobre o cao preguigoso.
```

```
The (quick) [brown] {fox} jumps!
Over the $43,456.78 <lazy> #90 dog
& duck/goose, as 12.5% of E-mail
from aspammer@website.com is spam.
Der ,,schnelle” braune Fuchs springt
iiber den faulen Hund. Le renard brun
arapide» saute par-dessus le chien
paresseux. La volpe marrone rapida
salta sopra i] cane pigro. El zorro
marrén répido salta sobre el perro
perezoso. A raposa marrom rzipida
salta sobre o e50 preguieoso.

The (quick) [brown] {fox} jumps!
Over the $43,456.78 <lazy> #90 dog
& duck/goose, as 12.5% of E-mail
from aspammer@website.com is spam.
Der ,,schnelle” braune Fuchs springt
iiber den faulen Hund. Le renard brun
arapide» saute par-dessus le chien
paresseux. La volpe marrone rapida
salta sopra i] cane pigro. El zorro
marrén répido salta sobre el perro
perezoso. A raposa marrom rzipida
salta sobre o e50 preguieoso.
```

Figure.9

In some test case the overall detection was same for both but some words or pattern cannot be detected by both of the OCR systems.

Here above check the difference between two results. We take the same test case of yellow background paper and black font case for demonstrating the differences and drawback of system as well.[10] [11]

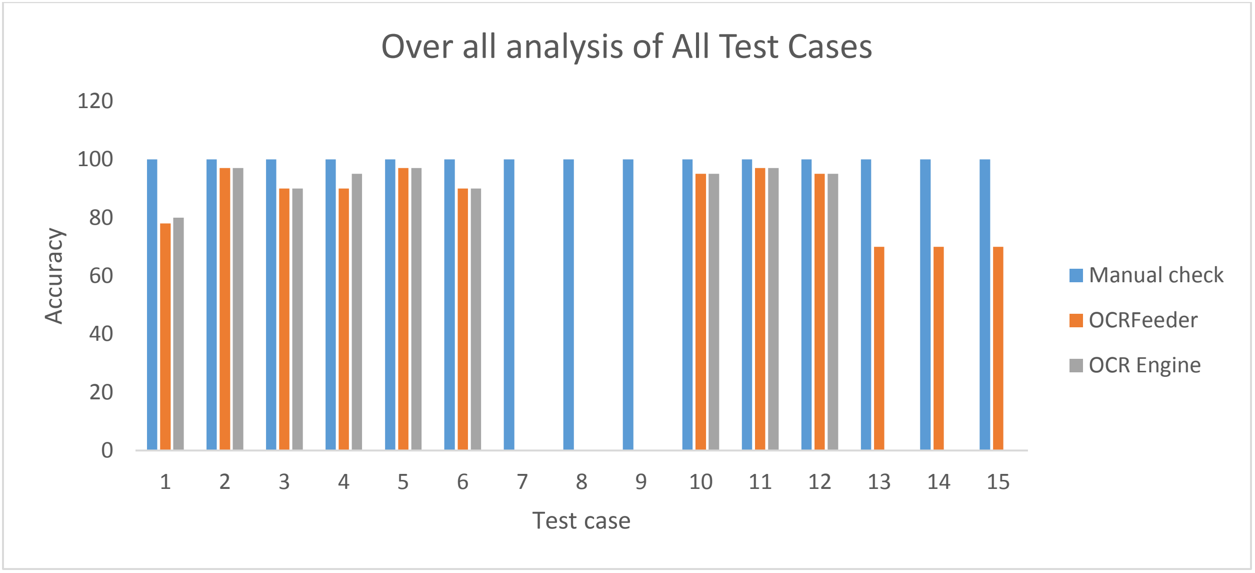


Figure.10

In summary, we can clearly identify from Figs. 6 –9 that when we uses black font's test case we get accurate result. If fonts are simple and style not much fancy then it will be definitely detected by our OCR engine.

Here, as we described algorithm for cropping and image detection, line finding, baseline fitting using **Tesseract OCR API** is really convenient approach.

6. *CONCLUSION*

In this report, **Optical Character Recognition** is completely presented. This OCR system have the ability to yield excellent results. The recognition system presented in this project is computationally efficient. The feature extraction step of optical character recognition is the most important. It can be used with existing OCR methods, especially for English text. We were able to give the character recognition accuracy by 85% for a text to be recognized from the image file. Users can also crop or save the image text which is extracted from the image file. The main advantage of this project is that, that from now there would be no more retyping, quick digital searches, edit text, save space, accessibility and many other. The future scope for the project which can be done in our project to extract the text and save it from any other file format .e.g. .psd .pdf file.

We did this whole project by following some references and existing ideas, as the idea was given by Poojan Patel and Hemali Patel specifically the terminology of Tesseract OCR and the project definition and problem description of project.

Then we were trying to collect the existing research data and better implementation ideas from research paper and other cyber tutorials. Yash Patel and Ishita lad did the java applet GUI development and then Poojan Patel helped to solve this problem using Tesseract API implementation over our OCR System, End of the project Yash Patel shared another idea to extract as docx editable MS word from OCR system and implemented this as another functionality of the OCR system.

In future this area of research can be used in authentication and security field as image based authentication data transfer, Text hiding into image such as steganography, online banking system will be more flexible when we use OCR system to convert handwritten documents into plain text.

REFRENCES

[1] Optical Character Recognition Implementation Using Pattern Matching by Faisal Mohammad

Available at: http://www.ijcsit.com/docs/Volume%205/vol5issue02/ijcsit20140502254.pdf

[2] An Overview of the Tesseract OCR Engine by Ray Smith

Available at:http://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf

[3] Wikipedia article

Available at: https://en.wikipedia.org/wiki/Tesseract_(software)

[4] OCRFeeder Wikipedia article

Available at: https://en.wikipedia.org/wiki/OCRFeeder

[5] Tesseract library description

Link: http://tesseract-ocr.googlecode.com/git/vs2008/doc/programming.html

[6] Tesseract implementation

Link: https://www.raywenderlich.com/93276/implementing-tesseract-ocr-ios

[7] OCR exists in Ubuntu

Link: https://www.howtoforge.com/ocr_with_tesseract_on_ubuntu704

[8] Open source extract text from image

Link: https://opensource.com/life/15/9/open-source-extract-text-images

[9] Difference calculator

Link: http://www.calculatorsoup.com/calculators/algebra/percent-difference-calculator.php

[10] Content difference checker

Link: https://www.diffchecker.com/sigtabdr

[11] Text compare

Link: http://text-compare.com/

[12] A survey of modern optical character rec

Link: http://arxiv.org/pdf/1412.4183.pdf

ACKNOWLEDGEMENT

It is a great opportunity for us to write research paper about subject like “**OCR Engine using Tesseract OCR API**" resolution. At the time of preparing this paper we are gone through different books, reference papers, tutorials and websites which help us to get acquainted with this new topics. We are actually focusing on those topics which are important for us to understand about this subject easily.

Every project big or small is successful largely due to efforts of number of people who has always given their advice and have also helped. I sincerely thank to all those people for their support, inspiration and guidance in making this project. We acknowledge with gratitude to assistant professor Wenjia Li, my respective teacher, who has always been sincere and helpful in making me understanding the different system of legal research and conceptual problems in my term research paper.

Apart from us this term paper will certainly be immense importance for those who are interesting to know about this subject. I hope they will find it comprehensible.

We have tried hard and soul to gather all relevant documents regarding this subject. We don’t know how far we want able to do that ‘Furthermore we don’t claim all the information in this term paper is included perfectly. There may be shortcoming, factual error, mistaken opinion which are all ours and we alone are responsible for those but we will try to give a better volume in future.

Thank you,

Yash Patel

Poojan Patel

Ishita Lad

Hemali Patel