International Journal of Trend in Scientific Research and Development (IJTSRD)
International Open Access Journal
ISSN No: 2456 - 6470 | www.ijtsrd.com | Volume - 1 | Issue – 6

# Text Extraction from Image using Python

**T. Gnana Prakash**
Assistant Professor, CSE Department,
VNR VJIET, Hyderabad, India

**K. Anusha**
Assistant Professor, CSE Department,
Vardhaman College of Engineering, Hyderabad, India

## ABSTRACT

With so much of our lives computerized, it is vitally important that machines and humans can understand one another and pass information back and forth. Mostly computers have things their way we have to & talk to them through relatively crude devices such as keyboards and mice so they can figure out what we want them to do. However, when it comes to processing more human kinds of information, like an old-fashioned printed book or a letter scribbled with a fountain pen, computers have to work much harder. That is where optical character recognition (OCR) comes in. Here we process the image, where we apply various pre-processing techniques like desk wing, binarization etc. and algorithms like Tesseract to recognize the characters and give us the final document.

***Keywords****: Open CV- Python; Image Processing; Text Extraction; Image threshold; Virtual Image*

## I. INTRODUCTION

Text data present in images contain useful information for automatic annotation, indexing, and structuring of images. Extraction of this information involves detection, localization, tracking, extraction, enhancement, and recognition of the text from a given image. However, variations of text due to differences in size, style, orientation, and alignment, as well as low image contrast and complex background make the problem of automatic text extraction extremely challenging. While comprehensive surveys of related problems such as face detection, document analysis, and image indexing can be found, the problem of text information extraction is not well surveyed. A large number of techniques have been proposed to address this problem, and the purpose of this paper is to classify and review these algorithms, discuss benchmark data and performance evaluation, and to point out promising directions for future research.

Content-based image indexing refers to the process of attaching labels to images based on their content. Image content can be divided into two main categories: perceptual content and semantic content. Perceptual content includes attributes such as color, intensity, shape, texture, and their temporal changes, whereas semantic content means objects, events, and their relations. A number of studies on the use of relatively low-level perceptual content for image and video indexing have already been reported. Studies on semantic image content in the form of text, face, vehicle, and human action have also attracted some recent interest. Among them, text within an image is of particular interest as

- It is very useful for describing the contents of an image;
- It can be easily extracted compared to other semantic contents, and
- It enables applications such as keyword-based image search, automatic video logging, and text-based image indexing.

## II. TEXT IN IMAGES

A variety of approaches to text information extraction (TIE) from images have been proposed for specific applications including page segmentation, address block location, license plate location, and content-based image indexing.

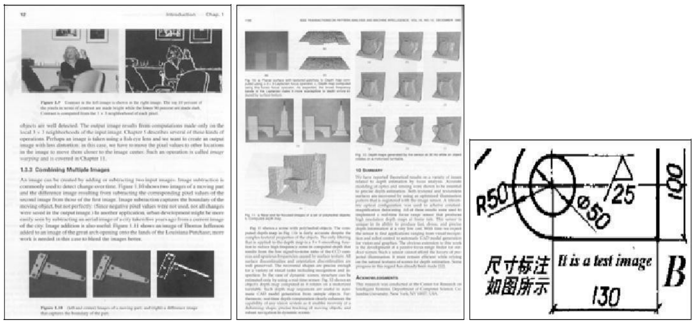



Fig. 1: Grayscale document images

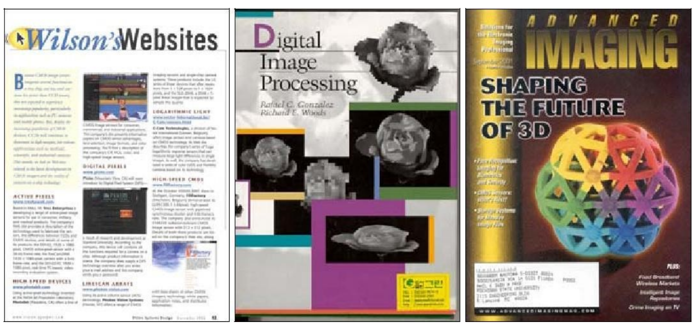



Fig. 2: Multi-color document images


Fig. 3: Images with caption text


Fig. 4: Scene text images

Text in images can exhibit many variations with respect to the properties like geometry, color, motion, edge and compression.

Table 1: Properties of text in images

| Properties | | Variants or sub-classes |
|---|---|---|
| **Geometry** | Size | Regularity in size of text |
| | Alignment | Horizontal/vertical |
| | | Straight line with skew (implies vertical direction) |
| | | Curves |
| | | 3D perspective distortion |
| | Inter-character distance | Aggregation of characters with uniform distance |
| **Colour** | | Gray |
| | | Colour (monochrome, polychrome) |
| **Motion** | | Static |
| | | Linear Movement |
| | | 2D rigid constrained movement |
| | | 3D rigid constrained movement |
| | | Free Movement |
| **Edge** | | Strong edges (contrast) at text boundaries |
| **Compression** | | Un-compressed image |
| | | JPEG, MPEG-compressed image |

The problem of Text Information Extraction TIE system receives an input in the form of a still image or a sequence of images. The images can be in gray scale or color, compressed or un-compressed, and the text in the images may or may not move. The TIE problem can be divided into the following sub-problems: (i) detection, (ii) localization, (iii) tracking, (iv) extraction and enhancement (v) Optical Character recognition (OCR).

## III. IMAGE THRESHOLDING

### *A. Threshold Binary*

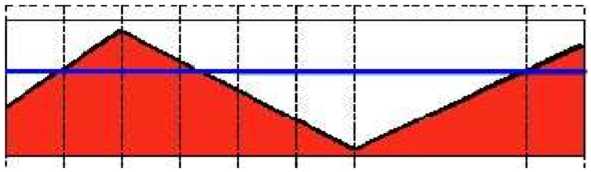


Fig.5: Threshold Binary

This thresholding operation can be expressed as:

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{maxVal} & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

So, if the intensity of the pixel $\mathtt{src}(x,y)$ is higher than $\mathtt{thresh}$, then the new pixel intensity is set to a $\mathtt{MaxVal}$. Otherwise, the pixels are set to $0$.

### *B. Threshold Binary, Inverted*

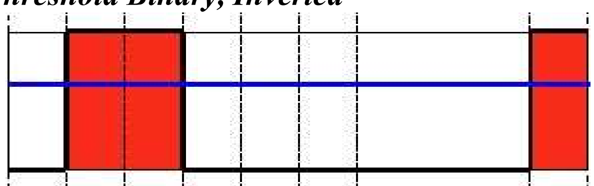


Fig. 5: Threshold Binary, Inverted

This thresholding operation can be expressed as:

$$\mathtt{dst}(x,y) = \begin{cases} 0 & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{maxVal} & \text{otherwise} \end{cases}$$

If the intensity of the pixel $\mathtt{src}(x,y)$ is higher than $\mathtt{thresh}$, then the new pixel intensity is set to a $0$. Otherwise, it is set to $\mathtt{MaxVal}$.

### *C. Truncate*

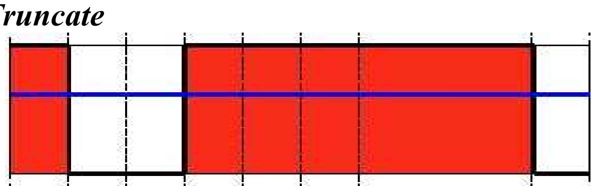


Fig. 6: Truncate

This thresholding operation can be expressed as:

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{threshold} & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{src}(x,y) & \text{otherwise} \end{cases}$$

The maximum intensity value for the pixels is $\mathtt{thresh}$, if $\mathtt{src}(x,y)$ is greater, then its value is *truncated*. See figure below:

### *D. Threshold to Zero*

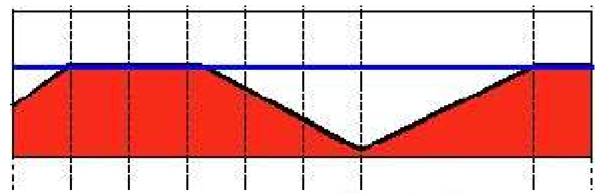


Fig. 7: Threshold to Zero

This operation can be expressed as:

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{src}(x,y) & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases}$$

If $\mathtt{thresh}$ is lower than $\mathtt{thresh}$, the new pixel value will be set to $0$.

### *E. Threshold to Zero, Inverted*

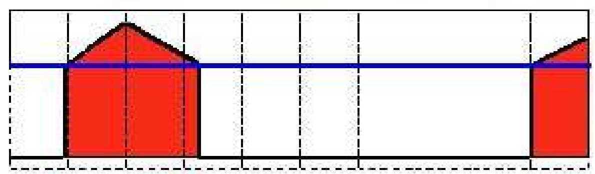


Fig. 8: Threshold to Zero, Inverted

This operation can be expressed as:

$$\mathtt{dst}(x,y) = \begin{cases} 0 & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{src}(x,y) & \text{otherwise} \end{cases}$$

If $\mathtt{thresh}$ is greater than $\mathtt{thresh}$, the new pixel value will be set to $0$.

### *F. Simple Thresholding*

If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). The function used is **cv2.threshold**. First argument is the source image, which **should be a grayscale image**. Second argument is the threshold value which is used to classify the pixel values. Third argument is the maxVal which represents the value to be given if pixel value is more than (sometimes less than) the threshold value. OpenCV provides different styles of thresholding and it is decided by the fourth parameter of the function. Different types are:

- cv2.THRESH_BINARY
- cv2.THRESH_BINARY_INV

- cv2.THRESH_TRUNC
- cv2.THRESH_TOZERO
- cv2.THRESH_TOZERO_INV

Two outputs are obtained. First one is a **retval** . Second output is our **thresholded image**.

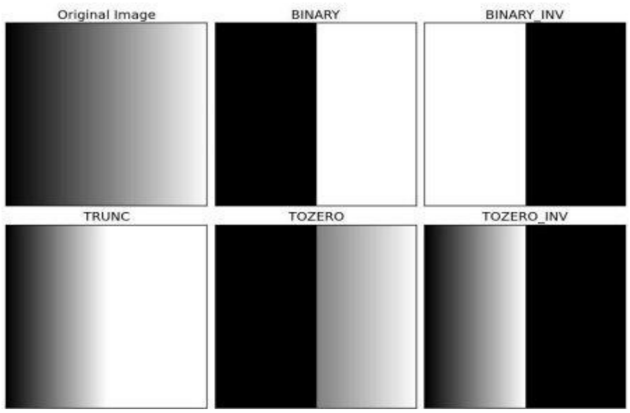



Fig. 9: Image Describing outputs of different Thresholding techniques

## IV. PYTHON ANYWHERE

**Python Anywhere** is an online Integrated Development Environment (IDE) and Web hosting service based on the Python programming language. It provides in browser access to server-based Python and Bash Command-line interfaces, along with a code editor with Syntax highlighting. One striking different between Python Anywhere and the usual Python Cloud Computing solution that we know of, is that you can totally work on it online using internet browser in developing your Python application. With this, you can bypass the usual delicacies on preparing a local workstation that meet cloud hosting service environment requirement and directly work inside your browser that connected to many consoles provided by Python anywhere, such as : Bash, Python/iPython 2.6/2.7/3.3 and MySQL.

This provides a step-by-step guide on how to deploy your Django applications. The service provides in-browser access to the server-based Python and Bash command line interfaces, meaning you can interact with Python Anywhere's servers just like you would with a regular terminal instance on your own computer. Currently, Python Anywhere are offering a free account which sets you up with an adequate amount of storage space and CPU time to get a Django application up and running.

### *A. Creating a Python Anywhere Account*

First sign up for a Beginner Python Anywhere account. If your application takes off and becomes popular, you can always upgrade your account at a later stage to gain more storage space and CPU time along with a number of other benefits (like hosting specific domains and ssh abilities).

Once your account has been created, you will have your own little slice of the World Wide Web at http://<username>.pythonanywhere.com, where <username> is your Python Anywhere username. It is from this URL that your hosted application will be available from.

### *B. The Python Anywhere Web Interface*

The Python Anywhere web interface contains a *dashboard*, which in turn provides a series of tabs allowing you to manage your application. The tabs as illustrated in Fig. 10 include:

- a *consoles* tab, allowing you to create and interact with Python and Bash console instances;
- a *files* tab, which allows you to upload to and organize files within your disk quota;
- a *web* tab, allowing you to configure settings for your hosted web application;
- a *schedule* tab, allowing you to setup tasks to be executed at particular times; and
- a *databases* tab, which allows you to configure a MySQL instance for your applications should you require it.

Of the five tabs provided, we'll be working primarily with the *consoles* and *web* tabs. The Python Anywhere help pages provide a series of detailed explanations on how to use the other tabs.

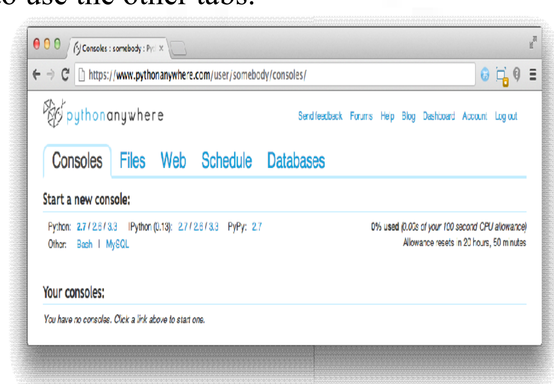



Fig. 10: The Python Anywhere dashboard, showing the *Consoles* tab.

### *C. Python Anywhere to upload the image*

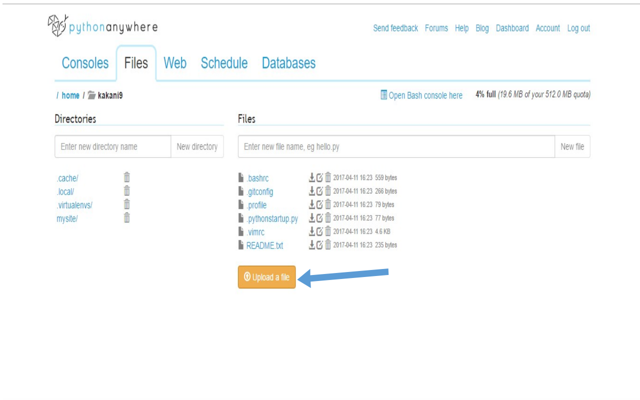



Fig. 11: Python Anywhere IDE to upload image

In the Python Anywhere IDE, the user can upload the image from which he or she wishes to extract the text. After logging into Python Anywhere account, a user has to go to the working directory where one can find "Upload a File" option. Clicking on it lets user chose the desired image and then uploads it to Python Anywhere cloud.

### *D. The Bash Console*


Fig. 12: Finding Bash Console in Python Anywhere

Python Anywhere allows a user to have two consoles for a free trial. On upgrading the account, a user can increase this number. To run the python files one must open the bash console.

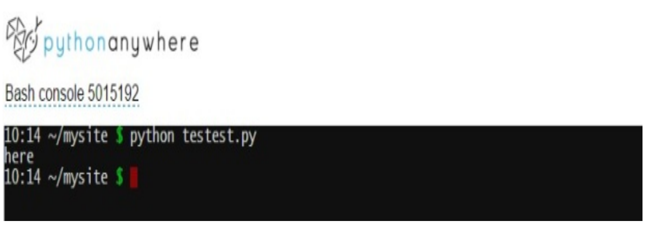



Fig. 13: Running Files in Bash Console

Here we specify the file we wish to run. Python is the keyword to specify that we are running a python file and testest.py is the file name.

### *E. Result File*

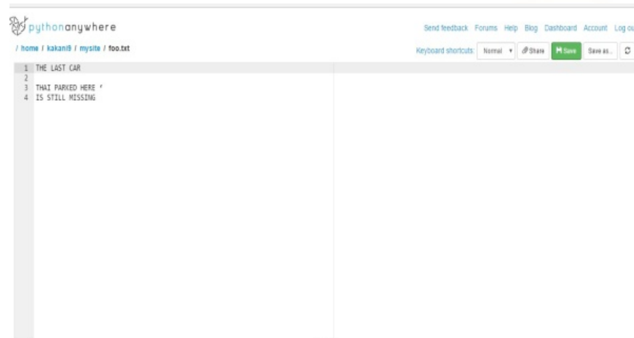



Fig. 14: Text files containing extracted text

The text extracted from the images is pipelined to a text file where the user can view, edit and modify its contents. User can thus save the obtained text file and download it from Python Anywhere.

## V. SYSTEM ANALYSIS

### *A. System Architecture*

The entire process can be depicted using these basic steps:

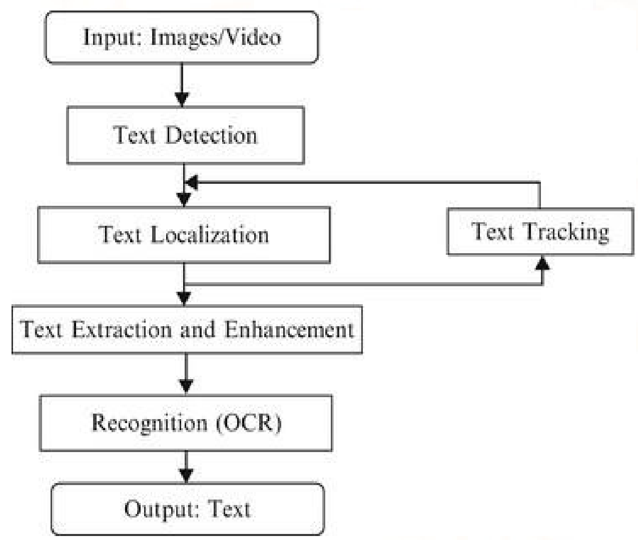



Fig. 15: Workflow in the system

The three basic steps involved in this process are detection, enhancement and extraction. This diagram defines the structure of the system.

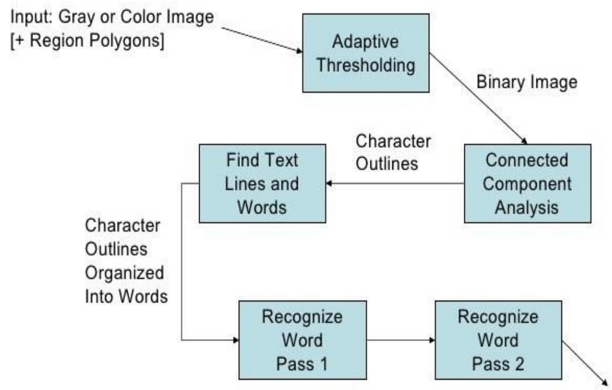



Fig. 16: Detailed Architecture of system

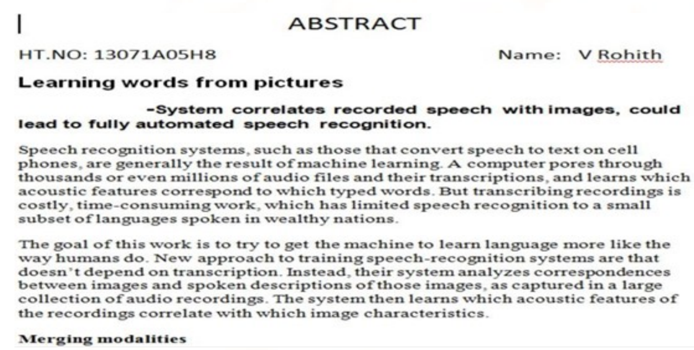
## VI. Test cases

Table 2: Test Cases

| S. No | Test Case | Expected Result | Actual | Result |
|---|---|---|---|---|
| 1 | Image with plain Text and plain background | Text extracted | Text extracted | Passed |
| 2 | Image with luminance | Text extracted | Text extracted | Passed |
| 3 | Tabular data which contains the rows and columns | Text extracted | Text extracted | Passed |
| 4 | Letter head | Text extracted | Text extracted | Passed |
| 5 | Bond paper with the text content which is in colour | Text extracted | Text extracted | Passed |
| 6 | Signboard containing text | Text extracted | Text extracted | Passed |
| 7 | Text with varying font size | Text extracted | Text extracted | Passed |
| 8 | Handwritten text | Text extracted | | Partially passed |
| 9 | Image with high text data of low details | Text extracted | Text extracted | Failed |
| 10 | Complex background image with tilted text containing mixed colours | Text extracted | Text extracted | Failed |
| 11 | Label on water bottle | Text extracted | Text extracted | Failed |

Below are the results of few test cases performed. The original image and the extracted text are shown below.

### *A. Example 1:*


ABSTRACT

HT.NO: 13071A05H8 Name: V Rohith

**Learning words from pictures**

**-System correlates recorded speech with images, could lead to fully automated speech recognition.**

Speech recognition systems, such as those that convert speech to text on cell phones, are generally the result of machine learning. A computer pores through thousands or even millions of audio files and their transcriptions, and learns which acoustic features correspond to which typed words. But transcribing recordings is costly, time-consuming work, which has limited speech recognition to a small subset of languages spoken in wealthy nations.

The goal of this work is to try to get the machine to learn language more like the way humans do. New approach to training speech-recognition systems are that doesn't depend on transcription. Instead, their system analyzes correspondences between images and spoken descriptions of those images, as captured in a large collection of audio recordings. The system then learns which acoustic features of the recordings correlate with which image characteristics.

**Merging modalities**

Fig.17: Image

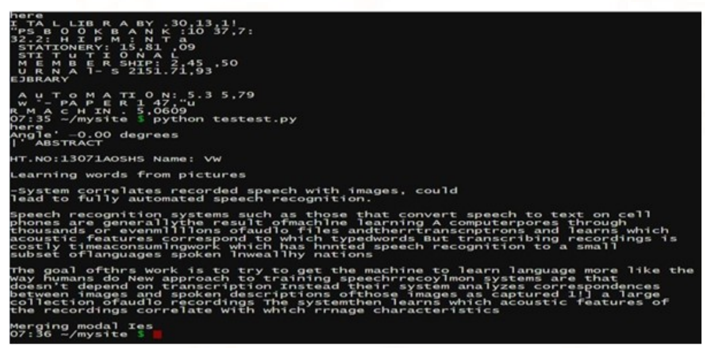



Fig. 18: Image with plain background

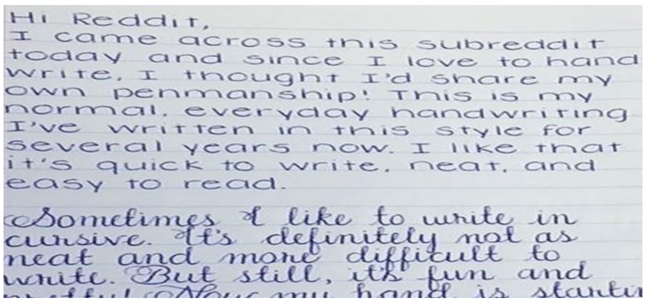
### *B. Example 2:*


Fig. 19: Bond Paper

```
' were 2")


' ngvtesseract imageilaislrinfl\mage.open('C:/Users/bkkakanw/Documenls/pmject/pytesseractr
10:55 ~/mysite $ python testest.py
here
1.

5 NAKA HANUMANT NAGAR, FAIZABAD (U.P.) PIN-224 001 in the
of CHAIRMAN, DULARI ENVIRONMENTAL EDUCATIONAL SOCIAL
ARE TRUST & MANAGER of DULARI MAI-[ILA MAHAVIDYALAYA,
DULARI NAGAR, PILAI, AMBEDKAR NAGAR (U.P.) PIN-224 168 take oath and
state as under:-

(
That I am Manager of DULARI MAHJLA MAHAVIDYALAYA, DULARI
NAGAR, PILAI, AMBEDKAR NAGAR (U.P.) PIN-224 168 and signing this
Affidavit on behalf of it for which I am fully authorized

That the managing society/trust has made an application to the NRC, NCTE for running
an institution namely DULARI MAHILA MAHAVIDYALAYA, DULARI
NAGAR, PlLAI, AMBEDKAR NAGAR (U.I'.) PIN-224 168 for B.Ed. with an
intake of 100 Seat. The institution alter fulfilling all the provisions of the NCTE Act,
Rules & Regulations has got letter of intent under section 7(9) of the NCTE Regulations,

2009 vide NRC, NCTE letter No. F.Na. NRC/NCTEINRCAPP-729M225"l
Meeting/75088 dated 07 March, 2014. /
i That the managing society/trust of the institution has constituted the Selection

Committee for appointment of faculty as per the policy of the State Govemment/
University/UGC and the following were the Members ofthe Committee:-
```

Fig. 20: Bond Paper with plain background

### *C. Example 3:*


Hi Reddit,
I came across this subreddit today and since I love to hand write, I thought I'd share my own penmanship! This is my normal, everyday handwriting I've written in this style for several years now. I like that it's quick to write, neat, and easy to read.

Sometimes I like to write in cursive. It's definitely not as neat and more difficult to write. But still, it's fun and

Fig. 21: Hand writing Image

```
H's qwCK to wme, mean and
cosy to read.
(obonwfi 1&4 Mao   Lam/{L m
'm £0
WM . mtdg mi. (1A
'mmt (find w to
mm.@uf AXLLL, 11' {in} and
' m Ahmtu
M. .1!"er Amman n/n
11:11 ~/mysite $ python testest.py
here
H\ Reddn.
I Come QCrcss Mme Subrcdmr
"mm and smut: 1 \cve m hcmd
Wruc. 1 thugh† I'd Show: my
own penrhonshup' Tms 16 nw
normal. evsryclcw handwnnnq
I've wnnen m nms awn: for
§€Veral ycors now, I Ime ThCIT
H's quch To wrlie, mean and
cosy to read.
LLth/{L 'UL
1129 3f Mm £0
méayfib .Wmidu mi. m
mat and mom dip/lurk kn
wKLiL.@LL£ Mitt, 11' {in}. and
m mm.
M. .JE..I HWM A'vl11 m1
11:11 ~/mysite $
```

Fig. 22: Image with plain background

### *D. Example 4:*

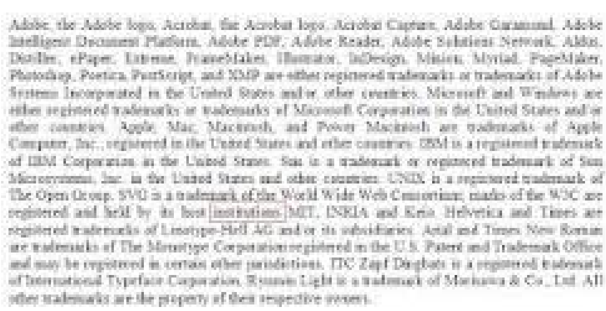


Fig. 23: Image with high text data of low details

```
06:03 ~/mysite $ python testest.py
here
é"'1a.Vc:: been ckosm 5:201 'Une xole 07f Q$Sodcd£
L',-"-:2',~SUEtQfl£ CUL miCfliOwH COYFOTGL'UOTL ("as
;-~<:;:<:r3~<\;si£ment Pmccss wank on 'FM' {we dcua
 {6) pf 06 Auaus'sp .2016. The job deécwif
tggm Sta-Leg "the, Package, 4,0 be CM'OLLHA,
 [QKA Pm' monum-
06:03 ~/mysite $
```

Fig. 24: Image with high text data of low details

### *E. Example 5:*

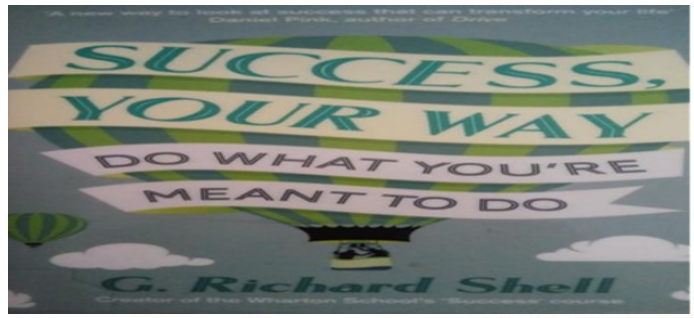



Fig. 25: Complex background image with tilted text containing mixed colors

```
C:\Python34>python testest.py
hi
QED B 553$. (Ed $5.9m
at... 53>, 55353. 53 "c5 nr$.x.,:m : 1:. I ,

C:\Python34>python testest.py
hi
1.; r'ook at success that can transform yam tife'
Dame! Pink, author of Drive

a


3: how»; Success" course

C:\Python34>
```

Fig. 25: Complex background image with tilted text containing mixed colors with plain background

## CONCLUSION

Even though a large number of algorithms have been proposed in the literature, no single method can provide satisfactory performance in all the applications due to the large variations in character font, size, texture, color, etc. Through this paper we are in the stream of deriving the satisfactory results by enhancing the input by fine tuning the image and deriving the optimum levels of accuracy from TESSERACT.

## FUTURE SCOPE

With machine learning algorithms constantly being developed and improved, massive amounts of computational power becoming readily available both locally and on the cloud, and unfathomable amounts of data can be extracted not only in the domain of image but also in terms of scene, video frames and scrolling types of data.

## REFERENCES


1) M. Flickner, H. Sawney et al., Query by Image and Video Content: The QBIC System, IEEE Computer 28 (9) (1995) 23-32.
2) J. Zhang, Y. Gong, S. W. Smoliar, and S. Y. Tan, Automatic Parsing of News Video, Proc. of IEEE Conference on Multimedia Computing and Systems, 1994, pp. 45-54.
3) M. H. Yang, D. J. Kriegman, and N. Ahuja, Detecting faces in Images: A Survey, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (1) (2002) 34-58.
4) Y. Cui and Q. Huang, Character Extraction of License Plates from Video, Proc. of IEEE Conference on Computer Vision and Pattern Recognition, 1997, pp. 502 –507.
5) C. Colombo, A. D. Bimbo, and P. Pala, Semantics in Visual Information Retrieval, IEEE Multimedia, 6 (3) (1999) 38-53.
6) T. Sato, T. Kanade, E. K. Hughes, and M. A. Smith, Video OCR for Digital News Archive, Proc. of IEEE Workshop on Content based Access of Image and Video Databases, 1998, pp. 52-60.
7) Atsuo Yoshitaka and Tadao Ichikawa, A Survey on Content-based Retrieval for Multimedia Databases, IEEE Transactions on Knowledge and Data Engineering, 11(1999) 81-93.
8) W. Qi, L. Gu, H. Jiang, X. Chen, and H. Zhang, Integrating Visual, Audio, and Text Analysis for News Video, Proc. of IEEE International Conference on Image Processing, 2000, pp. 10-13.
9) D.Wactlar, T. Kanade, M. A. Smith, and S. M. Stevens, Intelligent Access to Digital Video: The Informedia Project, IEEE Computer, 29 (5) (1996) 46-52.
10) H. Rein-Lien, M. Abdel-Mottaleb, A. K. Jain, Face Detection in Color Images, IEEE Transactions on Pattern Analysis and Machine Intelligence, 24 (5) (2002) 696-706.