1. INTRODUCTION

1.1 AIM OF THE PROJECT

We live in a society where in people are busy with their own lives. Leaving aside helping others. Our society has many handicapped individuals who depend on others for help. Our tool focuses on visually impaired by Capturing signs boards in public places, capturing images from packaged goods in supermarkets and capturing images of things in their surroundings etc.,

In this method, we define the region of interest(ROI) by using text localization and recognition to acquire the text information. In this, we will enhance the efficiency of the existing optical character recognition(OCR). Thus, there is a need of character recognition mechanisms which transforms text from images with complex background to a readable format.

1.2 GENERAL INTRODUCTION

1.2.1 TEXT EXTRACTION

Image processing is any form of signal processing for which the input is an image, the output of image processing may be either an image or a set of characteristics or parameters related to the image. Most image-processing techniques involve treating the image as a two-dimensional signal and applying standard signal-processing techniques to it.

The basic concept used here is of Contour. Contours can be explained simply as a curve joining all the continuous points (along the boundary), having same color or intensity. The contours are a useful tool for shape analysis and object detection and recognition [1]. Contour is only detected when there is a difference in the intensity of the Object boundary between the background and foreground image. So, the efficiency of contour also depends on the initial processing of the image. How can we make binarize the image using the different thresholding techniques? We know that a text or a letter is a closed area or a boundary, so we have taken benefit of this property of the text. Due to this reason, a contour is formed and the text can be easily extracted.

The project is in its initial stage which will be helpful in different field like:

Banking (To read Credit Card)

• Libraries (To convert Scanned Page to Image)

• Govt. Sector (Form Processing)

• Used in Car Number Plate Recognition System

• Undesirable Text removal from images once it is completed or taken to the next label.

1.2.2 IMAGE

Let us start with the word "image". The surrounding world is composed of images. Humans are using their eyes, containing 1.5x10^8 sensors to obtaining images from the surrounding world in the visible portion of the electromagnetic spectrum (wavelengths between 400 and 700 nanometers). The light changes on the retina are sent to image processor center in the cortex [2].

In the image database systems geographical maps, pictures, medical images, pictures in medical atlases, pictures obtaining by cameras, microscopes, telescopes, video cameras, paintings, drawings and architectures plan, drawings of industrial parts, space images are considered as images. There are different models for color image representation. In the seventeenth century, Sir Isaac Newton showed that a beam of sunlight passing through a glass prism comes into view as a rainbow of colors. Therefore, he first understood that white light is composed of many colors. Typically, the computer screen can display 2^8 or 256 different shades of gray. For color images this makes 2^ (3*8) = 16,777,216 different colors.

Clerk Maxwell showed in the late nineteenth century that every color image cough be created using three images – Red, green and Blue image. A mix of these three images can produce every color. This model, named RGB model, is primarily used in image representation. The RGB image could be presented as a triple (R, G, B) where usually R, G, and B take values in the range [0, 255]. Another

color model is YIQ model (lamination (Y), phase (I), quadrature phase (Q)). It is the base for the color television standard. Images are presented in computers as a matrix of pixels. They have finite area. If we decrease the pixel dimension the pixel brightness will become close to the real brightness.

1.2.3 IMAGE DATABASE SYSTEMS

Set of images are collected, analyzed and stored in multimedia information systems, office systems, Geographical information systems(GIS), CAD/CAM systems, earth resources systems, medical databases, information retrieval systems, art gallery and museum catalogues, animal and plant atlases, sky star maps, meteorological maps, catalogues in shops and many other places. There are sets of international organizations dealing with different aspects of image storage, analysis and retrieval. Some of them are: AIA (Automated Imaging/Machine vision), AIIM (Document imaging), ASPRES (Remote Sensing/ Photogram) etc.

There are also many international centers storing images such as: Advanced imaging, Scientific/Industrial Imaging, Microscopy imaging, Industrial imaging etc. There are also different international work groups working in the field of image compression, TV images, office documents, medical images, industrial images, multimedia images, graphical images, etc.

1.3 CANNY EDGE DETECTION

Canny Edge Detection is a popular edge detection algorithm. It was developed by John F. Canny in 1986. It is a multi-stage algorithm and we will go through each stage [5].

Noise Reduction

Since edge detection is susceptible to noise in the image, first step is to remove the noise in the image with a 5x5 Gaussian filter. We have already seen this in previous chapters.

Finding Intensity Gradient of the Image

Smoothened image is then filtered with a Sobel kernel in both horizontal and vertical direction to get first derivative in horizontal direction (G_x) and vertical direction (G_y). From these two images, we can find edge gradient and direction for each pixel as follows:

$$Edge\_Gradient\ (G) = \sqrt{G_x^2 + G_y^2}$$

$$Angle\ (\theta) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$$

Gradient direction is always perpendicular to edges. It is rounded to one of four angles representing vertical, horizontal and two diagonal directions.

Non-maximum Suppression

After getting gradient magnitude and direction, a full scan of image is done to remove any unwanted pixels which may not constitute the edge. For this, at every pixel, pixel is checked if it is a local maximum in its neighborhood in the direction of gradient. Check the image below:

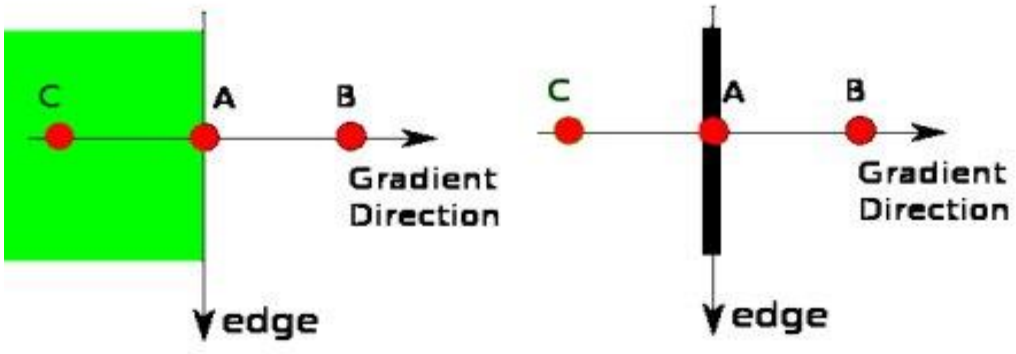


Fig. 1.1 – Non-maximum Suppression

Point A is on the edge (in vertical direction). Gradient direction is normal to the edge. Point B and C are in gradient directions. So, point A is checked with point B and C to see if it forms a local maximum. If so, it is considered for next stage, otherwise, it is suppressed (put to zero). In short, the result you get is a binary image with "thin edges".

Hysteresis Thresholding

This stage decides which are all edges are really edges and which are not. For this, we need two threshold values, *minVal* and *maxVal*. Any edges with intensity gradient more than *maxVal* are sure to be edges and those below *minVal* are sure to be non-edges, so discarded. Those who lie between these two thresholds are classified edges or non-edges based on their connectivity. If they are connected to "sure-edge" pixels, they are considered to be part of edges. Otherwise, they are also discarded. See the image below:

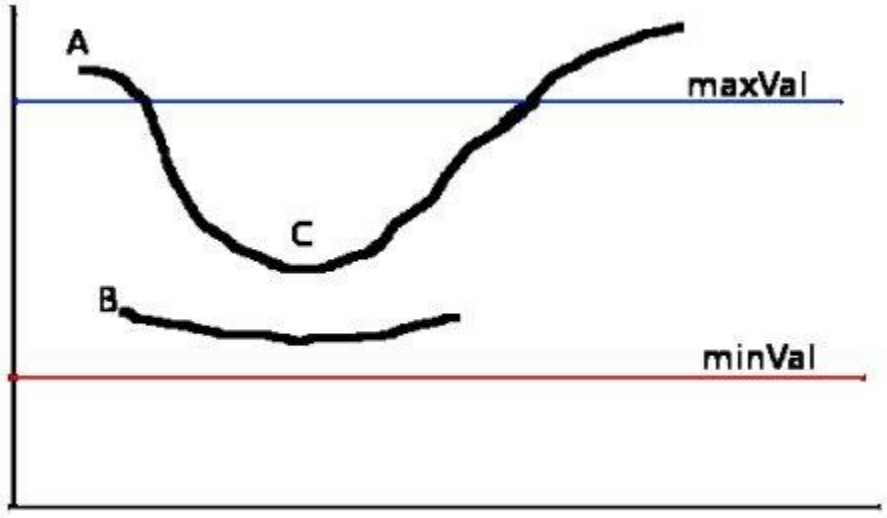


Fig. 1.2 – Hysteresis Thresholding

The edge A is above the *maxVal*, so considered as "sure-edge". Although edge C is below *maxVal*, it is connected to edge A, so that also considered as valid edge and we get that full curve. But edge B, although it is above *minVal* and is in same region as that of edge C, it is not connected to any "sure edge", so that is discarded. So, it is very important that we have to select *minVal* and *maxVal* accordingly to get the correct result. This stage also removes small pixels noises on the assumption that edges are long lines. So, what we finally get is strong edges in the image.

1.4 SUPPORT VECTOR MACHINE

"Support Vector Machine" (SVM) is a supervised machine learning algorithm which can be used for both classification or regression challenges. However, it is mostly used in classification problems. In this algorithm, we plot each data item as a point in n-dimensional space (where n is number of features you have) with the value of each feature being the value of a particular coordinate. Then, we perform classification by finding the hyper-plane that differentiate the two classes very well (look at the below snapshot)

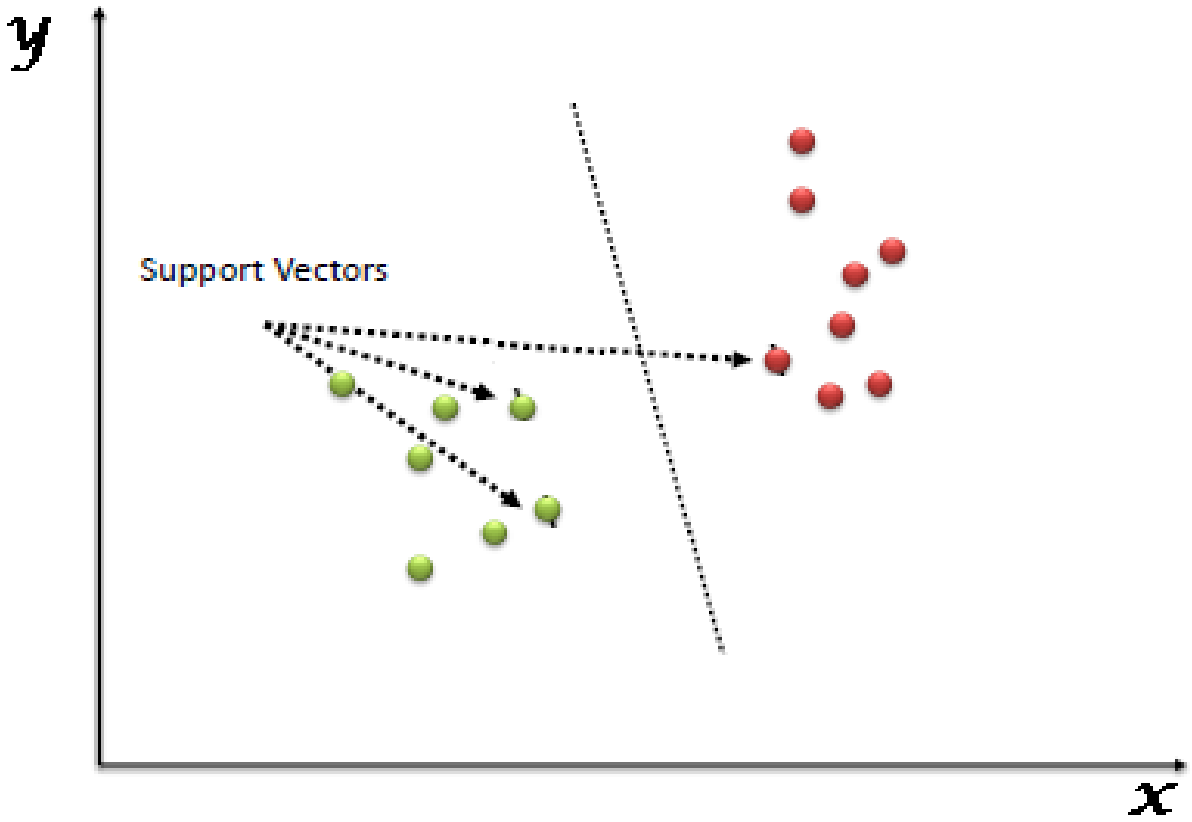


Fig. 1.3 – SVM Hyper Plane

Support Vectors are simply the co-ordinates of individual observation. Support Vector Machine is a frontier which best segregates the two classes (hyper-plane/ line).

2. EXISTING SYSTEM AND PROPOSED SYSTEM

2.1 EXISTING SYSTEM

The existing OCR technique works well with simple plain images but when the same technique is applied to images which are quite complex obtained from natural sceneries, this technique fails to recognize the text present in the images. OCR works best only with good quality typed documents. No OCR software is 100 percent accurate.

The number of errors depend upon the quality and type of document, including the font used. Errors that occur during OCR include misreading letters, skipping over letters that are unreadable, or mixing together text from adjacent columns or image captions. If high accuracy is required as with converting digital books to electronic format, then a clean-up of the electronic text will be needed.

2.2 PROPOSED SYSTEM

It detects text from natural scene images and extracts them regardless of the orientation is proposed. All existing methods are designed to operate under a certain constraint, like detecting text only in one direction. We have three stages in processing an image.

- Pre-processing
- Segmentation
- Recognition

3. LITERATURE STUDY

3.1 LITERATURE REVIEW

A large number of approaches have been proposed during the course of period for extracting the text from the images. The existing work on text extraction from images can be classified according to different criteria. This paper classifies methods according to the different types of image, analyzes those algorithms and discusses them. P. Nagabhushan [3] proposed an efficient approach to detect and extract the text in complex background color document images. The proposed method uses an algorithm called the canny edge detector algorithm to detect the edges. Once the edges are obtained, the dilation operation is International Journal of Modern Trends in Engineering and Research (IJMTER) Volume 04, Issue 4, [April– 2017] ISSN (Online):2349–9745; ISSN (Print):2393-8161 @IJMTER-2017, All rights Reserved 96 performed on them. It was found that it created holes in most of the connected components. These correspond to character strings. Connected components without holes were eliminated. Other non-text components were eliminated by computing and analyzing the standard deviation of each connected component. There was an unsupervised local thresholding which was devised to perform foreground segmentation in detected text regions. At the end the noisy text regions were identified and were processed to further enhance the quality of retrieved foreground. Thai [4] describes an approach for novel text extraction from graphical document images. The proposed method uses Morphological Component Analysis (MCA) algorithm. This algorithm describes an advancement of sparse representation framework. It shows two appropriately chosen discriminative over complete dictionaries. Two discriminative dictionaries were based on undecimated wavelet transform and curvelet transform. Angadi [5] proposed an automated method to detect and extract text prior to further image analysis. The proposed methodology uses Discrete Cosine Transform (DCT) based high pass filter to remove and prevent the dissemination of the constant background. The processed image was divided into 50x50 block and then the texture feature matrix was computed. A discriminant function which was anew was used to classify text blocks. The detected text blocks were merged back together to obtain new text like regions. Finally, the refinement phase was a post processing step used to improve the detection accuracy. This phase used to cover small portions of missed text present in adjacent undetected blocks and unprocessed regions. The

proposed method comprises of 5 phases; Background removal/suppression in the DCT domain, texture features computation on every 50x50 block and obtaining a feature matrix D, Classification of blocks, merging of text blocks to detect text regions, and refinement of text regions. This methodology had been conducted on 100 indoor and outdoor low resolution natural scene images containing text of different size, font, and alignment with complex backgrounds containing Kannada text and English text. The approach also detected nonlinear text regions and can be extended for text extraction from the images of other languages with little modifications. Thanh-Ha do [6] proposed an approach to extract text regions from graphical documents. In this method, first empirically two sequences are constructed of learned dictionaries for the text and graphical parts respectively. Then sparse representations are computed for all different sizes and nonoverlapped document patches in these learned dictionaries. Each patch can be classified based on these representations. It is classified into the text or graphic category by comparing its reconstruction errors. Same-sized patches in one category are then merged together to define the corresponding text or graphic layers which are combined to create a final text/graphic layer. Finally, in a postprocessing step, text regions are further filtered out by using some learned thresholds. Ranjini [7] talks about English text extraction from blob in comic image. Detecting text and extracting text from comic images helps to preserve the text and formatting during the conversion process and it provides very fine quality of text from the printed document. Initially, a pre-processing is done on the image. In the pre-processing step the RGB images are converted into a binary image by applying the threshold values between 0 to 1. Then the image is subjected to balloon detection. CCL algorithm is applied to the noise removed RGB images for detecting the connected components in the image which helps to detect the connected components in the image often it is used for Balloon detection. Text blob extraction is performed on them which are used to identify text blobs from non-text blobs, To avoid the false detection and to reduce the complexity the text blobs need to be identified exactly. The identification is done, based on the features of blob size. Finally, the text is recognized and extracted by using the optical character recognition method. Karin Sobottka [8] proposed a significant automatic approach for text location and identification on colored book and journals. In the pre-processing step a clustering algorithm is applied to reduce the amount of small variations in colors. For extracting text hypothesis two methods have been developed. One method is based on a top-down analysis which uses successive splitting of image regions. The other method is a bottom-up region growing algorithm. At the end,

both methods are combined to distinguish between text and non-text elements. Text elements are International Journal of Modern Trends in Engineering and Research (IJMTER) Volume 04, Issue 4, [April– 2017] ISSN (Online):2349–9745; ISSN (Print):2393-8161 @IJMTER-2017, All rights Reserved 97 binarized using automatically extracted information about text color. The binarized text regions can be used as input for a conventional OCR module. The proposed method is not restricted to cover pages, but can be applied to the extraction of text from other types of color images as well. S.Audithan [9] formulated a computationally fast method to extract text like regions from printed documents. It uses Haar discrete wavelet transform to detect edges of candidate text regions. Thresholding technique is used to remove non-text edges. Morphological dilation operator is used to connect the isolated candidate text edge and then a line feature vector graph is generated based on the edge map. This method explains an improved canny edge detector to detect text pixels. The stroke information extracts the spatial distribution of edge pixels. Finally, text regions are generated and filtered according to line features. Syed Saqibet [10] described a significant approach for curled text line information which is extracted from grayscale camera-captured document images. This approach is based on differential geometry, which uses local direction of gradients and second derivatives as the measure of curvature. The grayscale text line is enhanced by using multi-oriented multi-scale anisotropic Gaussian smoothing. Detection of central lines of curled text line is found using ridges. Hessian matrix is used for finding direction of gradients and derivatives. By using this information, ridges are detected by finding the zero-crossing of the appropriate directional derivatives of smoothed image. This method is robust against high degrees of curl and requires no post-processing.

The performance of each technique has been evaluated based on its precision and recall rates obtained. As explained in the earlier sections, precision and recall rates are calculated as follows:

$$\text{Precision Rate} = \frac{\text{Correctly detected words}}{\text{Correctly detected words} + \text{False positives}} \times 100\% \quad (1)$$

$$\text{Recall Rate} = \frac{\text{Correctly detected words}}{\text{Correctly detected words} + \text{False Negatives}} \times 100\% \quad (2)$$

3.2 FEASIBILITY STUDY

Preliminary investigation examines project feasibility; the likelihood the system will be useful to the organization. The main objective of the feasibility study is to test the Technical, Social and Economical feasibility for adding new modules and debugging old running system. All system is feasible if they are unlimited resources and infinite time. There are aspects in the feasibility study portion of the preliminary investigation:

1. Technical Feasibility
2. Social Feasibility
3. Economical Feasibility

3.2.1 TECHNICAL FEASIBILITY

This study is carried out to check the technical feasibility, that is, the technical requirements of the system. Any system developed must not have a high demand on the available technical resources. This will lead to high demands on the available technical resources. This will lead to high demands being placed on the client. The developed system must have a modest requirement, as only minimal or null changes are required for implementing this system.

3.2.2 SOCIAL FEASIBILITY

The aspect of study is to check the level of acceptance of the system by the user. This includes the process of training the user to use the system efficiently. The user must not feel threatened by the system, instead must accept it as a necessity. The level of acceptance by the users solely depends on the methods that are employed to educate the user about the system and to make him familiar with it. His level of confidence must be raised so that he is also able to make some constructive criticism, which is welcomed, as he is the final user of the system.

3.2.3 ECONOMICAL FEASIBILITY

This study is carried out to check the economic impact that the system will have on the organization. The amount of fund that the company can pour into the research and development of the system is limited. The expenditures must be justified. Thus, the developed system as well within the budget and this was achieved because most of the technologies used are freely available. Only the customized products had to be purchased.

4. SYSTEM ANALYSIS

4.1 SOFTWARE REQUIREMENTS

4.1.1 PYTHON

Python is an easy to learn, powerful programming language. It has efficient high-level data structures and a simple but effective approach to object-oriented programming. Python's elegant syntax and dynamic typing, together with its interpreted nature, make it an ideal language for scripting and rapid application development in many areas on most platforms.

The Python interpreter and the extensive standard library are freely available in source or binary form for all major platforms from the Python Web site, https://www.python.org/, and may be freely distributed. The same site also contains distributions of and pointers to many free third-party Python modules, programs and tools, and additional documentation.

The Python interpreter is easily extended with new functions and data types implemented in C or C++ (or other languages callable from C). Python is also suitable as an extension language for customizable applications.

If you do much work on computers, eventually you find that there's some task you'd like to automate. For example, you may wish to perform a search-and-replace over a large number of text files or rename and rearrange a bunch of photo files in a complicated way. Perhaps you'd like to write a small custom database, or a specialized GUI application, or a simple game.

If you're a professional software developer, you may have to work with several C/C++/Java libraries but find the usual write/compile/test/re-compile cycle is too slow. Perhaps you're writing a test suite for such a library and find writing the testing code a tedious task. Or maybe you've written a program that could use an extension language, and you don't want to design and implement a whole new language for your application.

Python is just the language for you.

You could write a Unix shell script or Windows batch files for some of these tasks, but shell scripts are best at moving around files and changing text data, not well-suited for GUI applications or

games. You could write a C/C++/Java program, but it can take a lot of development time to get even a first-draft program. Python is simpler to use, available on Windows, Mac OS X, and Unix operating systems, and will help you get the job done more quickly.

Python is simple to use, but it is a real programming language, offering much more structure and support for large programs than shell scripts or batch files can offer. On the other hand, Python also offers much more error checking than C, and, being a *very-high-level language*, it has high-level data types built in, such as flexible arrays and dictionaries. Because of its more general data types Python is applicable to a much larger problem domain than Awk or even Perl, yet many things are at least as easy in Python as in those languages.

Python allows you to split your program into modules that can be reused in other Python programs. It comes with a large collection of standard modules that you can use as the basis of your programs — or as examples to start learning to program in Python. Some of these modules provide things like file I/O, system calls, sockets, and even interfaces to graphical user interface toolkits like Tk.

Python is an interpreted language, which can save you considerable time during program development because no compilation and linking is necessary. The interpreter can be used interactively, which makes it easy to experiment with features of the language, to write throw-away programs, or to test functions during bottom-up program development. It is also a handy desk calculator. Python enables programs to be written compactly and readably. Programs written in Python are typically much shorter than equivalent C, C++, or Java programs, for several reasons:

- the high-level data types allow you to express complex operations in a single statement;
- statement grouping is done by indentation instead of beginning and ending brackets;
- no variable or argument declarations are necessary.

Python is extensible: if you know how to program in C it is easy to add a new built-in function or module to the interpreter, either to perform critical operations at maximum speed, or to link Python programs to libraries that may only be available in binary form (such as a vendor-specific graphics library). Once you are really hooked, you can link the Python interpreter into an application written in C and use it as an extension or command language for that application.

4.1.2 INSTALLING PYTHON

Unlike most Unix systems and services, Windows does not include a system supported installation of Python. To make Python available, the CPython team has compiled Windows installers (MSI packages) with every release for many years. These installers are primarily intended to add a per-user installation of Python, with the core interpreter and library being used by a single user. The installer is also able to install for all users of a single machine, and a separate ZIP file is available for application-local distributions.

Supported versions

A Python release only supports a Windows platform while Microsoft considers the platform under extended support. This means that Python 3.6 supports Windows Vista and newer. If you require Windows XP support, then please install Python 3.4.

Installation steps

Four Python 3.6 installers are available for download - two each for the 32-bit and 64-bit versions of the interpreter. The *web installer* is a small initial download, and it will automatically download the required components as necessary. The *offline installer* includes the components necessary for a default installation and only requires an internet connection for optional features. See Installing Without Downloading for other ways to avoid downloading during installation.

After starting the installer, one of two options may be selected:

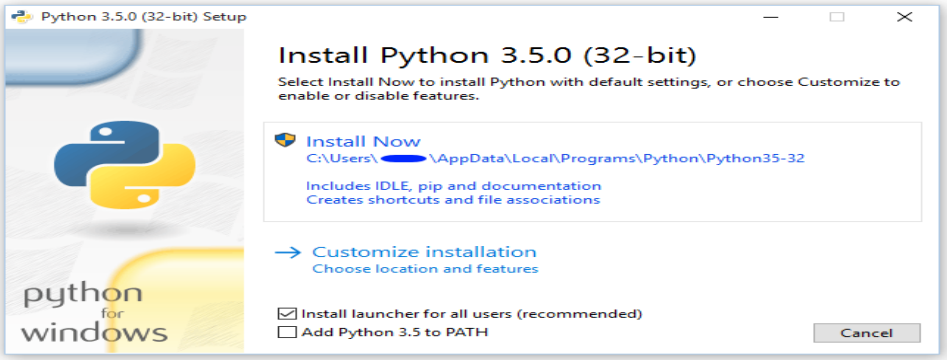


Fig. 4.1 – Python Installer

If you select “Install Now”:

- You will *not* need to be an administrator (unless a system update for the C Runtime Library is required or you install the Python Launcher for Windows for all users)
- Python will be installed into your user directory

- The Python Launcher for Windows will be installed according to the option at the bottom of the first page
- The standard library, test suite, launcher and pip will be installed
- If selected, the install directory will be added to your PATH
- Shortcuts will only be visible for the current user

Selecting “Customize installation” will allow you to select the features to install, the installation location and other options or post-install actions. To install debugging symbols or binaries, you will need to use this option.

To perform an all-users installation, you should select “Customize installation”. In this case:

- You may be required to provide administrative credentials or approval
- Python will be installed into the Program Files directory
- The Python Launcher for Windows will be installed into the Windows directory
- Optional features may be selected during installation
- The standard library can be pre-compiled to bytecode
- If selected, the install directory will be added to the system PATH
- Shortcuts are available for all users

Removing the MAX_PATH limitation

Removing MAX_PATH limitation is not Required. Changed in version 3.6: Support for long paths was enabled in python.

Installing without UI

All of the options available in the installer UI can also be specified from the command line, allowing scripted installers to replicate an installation on many machines without user interaction. These options may also be set without suppressing the UI in order to change some of the defaults.

To completely hide the installer UI and install Python silently, pass the /quiet option. To skip past the user interaction but still display progress and errors, pass the /passive option. The /uninstall option may be passed to immediately begin removing Python - no prompt will be displayed.

All other options are passed as name=value, where the value is usually 0 to disable a feature, 1 to enable a feature, or a path.

Installing without downloading

As some features of Python are not included in the initial installer download, selecting those features may require an internet connection. To avoid this need, all possible components may be downloaded on-demand to create a complete *layout* that will no longer require an internet connection regardless of the selected features. Note that this download may be bigger than required, but where a large number of installations are going to be performed it is very useful to have a locally cached copy. Execute the following command from Command Prompt to download all possible required files. Remember to substitute python-3.6.0.exe for the actual name of your installer, and to create layouts in their own directories to avoid collisions between files with the same name.

python-3.6.0.exe /layout [optional target directory]

You may also specify the /quiet option to hide the progress display.

Modifying an install

Once Python has been installed, you can add or remove features through the Programs and Features tool that is part of Windows. Select the Python entry and choose “Uninstall/Change” to open the installer in maintenance mode.

“Modify” allows you to add or remove features by modifying the checkboxes - unchanged checkboxes will not install or remove anything. Some options cannot be changed in this mode, such as the install directory; to modify these, you will need to remove and then reinstall Python completely.

“Repair” will verify all the files that should be installed using the current settings and replace any that have been removed or modified.

“Uninstall” will remove Python entirely, with the exception of the Python Launcher for Windows, which has its own entry in Programs and Features.

4.1.3 USING THE PYTHON INTERPRETER

Invoking the Interpreter

The Python interpreter is usually installed as /usr/local/bin/python3.6 on those machines where it is available; putting /usr/local/bin in your Unix shell’s search path makes it possible to start it by typing the command: python3.6 to the shell. Since the choice of the directory where the interpreter lives is an installation option, other places are possible; check with your local Python guru or system administrator. (E.g., /usr/local/python is a popular alternative location.)

On Windows machines, the Python installation is usually placed in C:\Python36, though you can change this when you’re running the installer. To add this directory to your path, you can type the following command into the command prompt in a DOS box:

```
set path=%path%;C:\python36
```

Typing an end-of-file character (Control-D on Unix, Control-Z on Windows) at the primary prompt causes the interpreter to exit with a zero-exit status. If that doesn’t work, you can exit the interpreter by typing the following command: quit ().

The interpreter’s line-editing features include interactive editing, history substitution and code completion on systems that support read line. Perhaps the quickest check to see whether command line editing is supported is typing Control-P to the first Python prompt you get.

The interpreter operates somewhat like the Unix shell: when called with standard input connected to a tty device, it reads and executes commands interactively; when called with a file name argument or with a file as standard input, it reads and executes a *script* from that file.

A second way of starting the interpreter is python -c command [arg] ..., which executes the statement(s) in *command*, analogous to the shell’s -coption. Since Python statements often contain spaces or other characters that are special to the shell, it is usually advised to quote *command* in its entirety with single quotes.

Some Python modules are also useful as scripts. These can be invoked using python -m module [arg] ..., which executes the source file for *module* as if you had spelled out its full name on the command line.

When a script file is used, it is sometimes useful to be able to run the script and enter interactive mode afterwards. This can be done by passing -ibefore the script.

ARGUMENT PASSING

When known to the interpreter, the script name and additional arguments thereafter are turned into a list of strings and assigned to the argv variable in the sys module. You can access this list by executing import sys. The length of the list is at least one; when no script and no arguments are given, sys.argv[0] is an empty string. When the script name is given as '-' (meaning standard input), sys.argv[0] is set to '-'. When -c *command* is used, sys.argv[0] is set to '-c'. When -m *module* is used, sys.argv[0] is set to the full name of the located module. Options found after -ccommand or -m *module* are not consumed by the Python interpreter's option processing but left in sys.argv for the command or module to handle.

INTERACTIVE MODE

When commands are read from a tty, the interpreter is said to be in *interactive mode*. In this mode it prompts for the next command with the *primary prompt*, usually three greater-than signs (>>>); for continuation lines it prompts with the *secondary prompt*, by default three dots (...). The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

```
$ python3.6
Python 3.6 (default, Sep 16 2015, 09:25:04)
[GCC 4.8.2] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

```
>>>
>>> the_world_is_flat = True
>>> if the_world_is_flat:
...     print("Be careful not to fall off!")
...
Be careful not to fall off!
```

Interpreter and it's Environment

SOURCE CODE ENCODING

By default, Python source files are treated as encoded in UTF-8. In that encoding, characters of most languages in the world can be used simultaneously in string literals, identifiers and comments — although the standard library only uses ASCII characters for identifiers, a convention that any portable code should follow. To display all these characters properly, your editor must recognize that the file is UTF-8, and it must use a font that supports all the characters in the file.

To declare an encoding other than the default one, a special comment line should be added as the *first* line of the file. The syntax is as follows:

```
# -*- coding: encoding -*-
```

where *encoding* is one of the valid codecs supported by Python.

For example, to declare that Windows-1252 encoding is to be used, the first line of your source code file should be:

```
# -*- coding: cp-1252 -*-
```

One exception to the *first line* rule is when the source code starts with a UNIX "shebang" line. In this case, the encoding declaration should be added as the second line of the file. For example:

#!/usr/bin/env python3

 # -- coding: cp-1252 -*-*

4.1.4 PYTHON PACKAGES

OpenCV-Python

This package contains only the OpenCV core modules without the optional contrib modules. If you are looking for a version which includes OpenCV contrib modules, please install OpenCV-contrib-python instead. The packages contain pre-compiled OpenCV binary with Python bindings. This enables super-fast (usually < 10 seconds) OpenCV installation for Python.

Installation and Usage

1. If you have previous/another version of OpenCV installed (e.g. cv2 module in the root of Python's site-packages), remove it before installation to avoid conflicts.

- To further avoid conflicts and to make development easier, Python's virtual environments are highly recommended for development purposes.

1. If you have an existing OpenCV-contrib- python installation, run pip uninstall OpenCV-contrib-python

2. Install this package:

pip install OpenCV-python

1. Import the package:

import cv2

The package contains haarcascade files. cv2.data.haarcascades can be used as a shortcut to the data folder. For example:

cv2.CascadeClassifier(cv2.data.haarcascades+"haarcascade_frontalface_default.xml")

NumPy

NumPy is the fundamental package for scientific computing with Python. It contains among other things:

- a powerful N-dimensional array object
- sophisticated (broadcasting) functions
- tools for integrating C/C++ and Fortran code
- useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases. NumPy is licensed under the BSD license, enabling reuse with few restrictions.

Installation via pip

Most major projects upload official packages to the Python Package index. They can be installed on most operating systems using Python's standard pip package manager. Note that you need to have Python and pip already installed on your system. You can install packages via commands such as:

python -m pip install --user numpy scipy matplotlib ipython jupyter pandas sympy nose

We recommend using an user install, using the -- user flag to pip (note: do not use sudo pip,which can cause problems). This installs packages for your local user, and does not write to the system directories.

gTTS (Google Text To Speech)

Creates an mp3 file from spoken text via the Google TTS (Text-to- Speech) API. A Python interface for Google's _Text to Speech_ API. It creates an _mp3_ file with the `gTTS` module or `gtts-cli` command line utility. It allows unlimited lengths to be spoken bytokenizing long sentences where the speech would naturally pause.

Install

pip install gTTS

Usage

You may either use `gTTS` as a python module or as a command-line utility.

A. Python Module

1. Import `gTTS`

>>from gtts import gTTS

2. Create an instance

>>tts = gTTS(text='Hello', lang='en', slow=True)

Parameters:

`text` - String - Text to be spoken.

`lang` - String - [ISO 639-1 language code] (#lang_list) (supported by the Google_Text to Speech_ API) to speak in.

`slow` - Boolean - Speak slowly. Default `False` (Note: only two speeds are provided by the API).

3. Write to a file

To disk using `save(file_name)`

>>tts.save("hello.mp3")

To a file pointer_ using `write_to_fp(file_object)`

>>f = TemporaryFile()

>>tts.write_to_fp(f)

>>#<do something=" "with=" "f=" ">

>>f.close()

B. Command line utility

Command

gtts-cli.py [-h] (["text to speak"] | -f FILE) [-l LANG] [-- slow] [-- debug] [-o destination_file]

Example:

$ # Read the string 'Hello' in English to hello.mp3

$ gtts-cli "Hello" -l 'en' -o hello.mp3

$ # Read the string 'Hello' in English (slow speed) to hello.mp3

$ gtts-cli "Hello"; -l 'en' -o hello.mp3 -- slow

$ # Read the contents of file 'hello.txt' in Czech to hello.mp3:

$ gtts-cli -f hello.txt -l 'cs'; -o hello.mp3

$ # Read the string 'Hello' from stdin in English to hello.mp3

$ echo "Hello" | gtts-cli -l 'en' -o hello.mp3 -

SUPPORTED LANGUAGES

* 'af' : 'Afrikaans'
* 'sq' : 'Albanian'
* 'ar' : 'Arabic'
* 'hy' : 'Armenian'
* 'bn' : 'Bengali'
* 'ca' : 'Catalan'
* 'zh' : 'Chinese'
* 'zh-cn' : 'Chinese (Mandarin/China)'
* 'zh-tw' : 'Chinese (Mandarin/Taiwan)'
* 'zh-yue' : 'Chinese (Cantonese)'
* 'hr' : 'Croatian'
* 'cs' : 'Czech'
* 'da' : 'Danish'
* 'nl' : 'Dutch'
* 'en' : 'English'
* 'en-au' : 'English (Australia)'
* 'en-uk' : 'English (United Kingdom)'
* 'en-us' : 'English (United States)'
* 'eo' : 'Esperanto'
* 'fi' : 'Finnish'
* 'fr' : 'French'
* 'el' : 'Greek'
* 'hi' : 'Hindi'
* 'hu' : 'Hungarian'
* 'is' : 'Icelandic'
* 'id' : 'Indonesian'
* 'it' : 'Italian'
* 'ja' : 'Japanese'
* 'km' : 'Khmer (Cambodian)'
* 'ko' : 'Korean'
* 'la' : 'Latin'
* 'lv' : 'Latvian'
* 'mk' : 'Macedonian'
* 'no' : 'Norwegian'
* 'pl' : 'Polish'
* 'pt' : 'Portuguese'
* 'ro' : 'Romanian'
* 'ru' : 'Russian'
* 'sr' : 'Serbian'
* 'si' : 'Sinhala'
* 'sk' : 'Slovak'
* 'es' : 'Spanish'
* 'es-es' : 'Spanish (Spain)'_____

PIL (Pillow)

The Python Imaging Library adds image processing capabilities to your Python interpreter. This library provides extensive file format support, an efficient internal representation, and fairly powerful image processing capabilities. The core image library is designed for fast access to data

stored in a few basic pixel formats. It should provide a solid foundation for a general image processing tool.

Image Archives:

The Python Imaging Library is ideal for image archival and batch processing applications. You can use the library to create thumbnails, convert between file formats, print images, etc. The current version identifies and reads a large number of formats. Write support is intentionally restricted to the most commonly used interchange and presentation formats.

Image Display:

The current release includes Tk Photo Image and Bitmap Image interfaces, as well as a Windows DIB interface that can be used with Python Win and other Windows-based toolkits. Many other GUI toolkits come with some kind of PIL support.

For debugging, there's also a show () method which saves an image to disk and calls an external display utility.

Image Processing:

The library contains basic image processing functionality, including point operations, filtering with a set of built-in convolution kernels, and colour space conversions. The library also supports image resizing, rotation and arbitrary affine transforms. There's a histogram method allowing you to pull some statistics out of an image. This can be used for automatic contrast enhancement, and for global statistical analysis.

Installation:

We provide Pillow binaries for Windows compiled for the matrix of supported Pythons in both 32 and 64-bit versions in wheel, egg, and executable installers. These binaries have all of the optional libraries included except for raqm and libimagequant.

>pip install Pillow

If the prerequisites are installed in the standard library locations for your machine (e.g. /usr or /usr/local), no additional configuration should be required. If they are installed in a non-standard

location, you may need to configure setup tools to use those locations by editing setup.py orsetup.cfg, or by adding environment variables on the command line:

$ CFLAGS="-I/usr/pkg/include" pip install pillow

If Pillow has been previously built without the required prerequisites, it may be necessary to manually clear the pip cache or build without cache using the -- no-cache-dir option to force a build with newly installed external libraries.

Using the Image class:

The most important class in the Python Imaging Library is the Image class, defined in the module with the same name. You can create instances of this class in several ways; either by loading images from files, processing other images, or creating images from scratch.

To load an image from a file, use the open() function in the Image module:

>>>from PIL import Image

>>>im = Image.open("hopper.ppm")

If successful, this function returns an Image object. You can now use instance attributes to examine the file contents:

>>>from __future__ import print_function

>>>print(im.format, im.size, im.mode

PPM (512, 512) RGB

The format attribute identifies the source of an image. If the image was not read from a file, it is set to None. The size attribute is a 2-tuple containing width and height (in pixels). The mode attribute defines the number and names of the bands in the image, and also the pixel type and depth. Common modes are “L” (luminance) for greyscale images, “RGB” for true color images, and “CMYK” for pre-press images. If the file cannot be opened, an IO Error exception is raised. Once you have an instance of the Image class, you can use the methods defined by this class to process and manipulate the image. For example, let’s display the image we just loaded:

>>>im.show()

Reading and writing images:

The Python Imaging Library supports a wide variety of image file formats. To read files from disk, use the open() function in the Image module. You don't have to know the file format to open a file. The library automatically determines the format based on the contents of the file. To save a file, use the save() method of the Image class. When saving files, the name becomes important. Unless you specify the format, the library uses the filename extension to discover which file storage format to use. Geometrical transforms The PIL.Image.Image class contains methods to resize() and rotate() an image. The former takes a tuple giving the new size, the latter the angle in degrees counter-clockwise.

out = im.resize((128, 128))

out = im.rotate(45) # degrees counter-clockwise

To rotate the image in 90 degree steps, you can either use the rotate() method or the transpose()method. The latter can also be used to flip an image around its horizontal or vertical axis.

Image enhancement:

The Python Imaging Library provides a number of methods and modules that can be used to enhance images.
The ImageFilter module contains a number of pre-defined enhancement filters that can be used with the filter() method.

from PIL import ImageFilter

out = im.filter(ImageFilter.DETAIL)

Pytesseract

Python-tesseract is a python wrapper for Google's Tesseract-OCR. Python-tesseract is an optical character recognition (OCR) tool for python. That is, it will recognize and "read" the text embedded in images. Python-tesseract is a wrapper for `Google's Tesseract-OCR Engine<https: github.com=" " tesseract-ocr=" " tesseract=" ">`_. It is also useful as a stand-alone invocation

script to tesseract, as it can read all image types supported by the Python Imaging Library, including jpeg, png, gif, bmp, tiff, and others, whereas tesseract-ocr by default only supports tiff and bmp. Additionally, if used as a script, Python-tesseract will print the recognized text instead of writing it to a file.

Installation

Prerequisites:

- Python-tesseract requires python 2.5+ or python 3.x

- You will need the Python Imaging Library (PIL) (or the Pillow fork).

Under Debian/Ubuntu, this is the package **python-imaging** or **python3-imaging**.

- Install `Google Tesseract OCR <https: github.com=" " tesseract-ocr=" " tesseract=" ">`_

(additional info how to install the engine on Linux, Mac OSX and Windows).

You must be able to invoke the tesseract command as *tesseract*. If this isn't the case, for example because tesseract isn't in your PATH, you will have to change the"tesseract_cmd" variable ``pytesseract.pytesseract.tesseract_cmd``.

Under Debian/Ubuntu you can use the package **tesseract-ocr**.

For Mac OS users. please install homebrew package **tesseract**.

Installing via pip:

Check the `pytesseract package page <https: pypi.python.org=" " pypi=" " pytesseract=" ">_

for more information.

code-block:: bash

$ (env)> pip install pytesseract

Or if you have git installed:

code-block:: bash

$ (env)> pip install -U git+https://github.com/madmaze/pytesseract.git

Installing from source:

.. code-block:: bash

$>git clone https://github.com/madmaze/pytesseract.git

$ (env)>cd pytesseract && pip install -U .

Usage

code-block:: python

try:

import Image

except ImportError:

from PIL import Image

import pytesseract

pytesseract.pytesseract.tesseract_cmd ='<full_path_to_your_tesseract_executable>'

Include the above line, if you don't have tesseract executable in your PATH

Example tesseract_cmd: 'C:\\Program Files (x86)\\Tesseract-OCR\\tesseract'

Simple image to string

print(pytesseract.image_to_string(Image.open('test.png')))

French text image to string

print(pytesseract.image_to_string(Image.open('test-european.jpg'), lang='fra'))

Get bounding box estimates

print(pytesseract.image_to_boxes(Image.open('test.png')))

Get verbose data including boxes, confidences, line and page numbers

print(pytesseract.image_to_data(Image.open('test.png')))

Support for OpenCV image/NumPy array objects

code-block:: python

import cv2

img = cv2.imread('/**path_to_image**/digits.png')

print(pytesseract.image_to_string(img))

OR explicit beforehand converting

print(pytesseract.image_to_string(Image.fromarray(img))

Add the following config, if you have tessdata error like:"Error opening data file...

code-block:: python

tessdata_dir_config='--tessdata-dir"<replace_with_your_tessdata_dir_path>'"

#Example config: '--tessdata-dir "C:\\Program Files (x86)\\Tesseract-OCR\\tessdata"'

It's important to add double quotes around the dir path.

pytesseract.image_to_string(image, lang='chi_sim', config=tessdata_dir_config)

Functions

* image_to_string: Returns the result of a Tesseract OCR run on the image to string

* image_to_boxes: Returns result containing recognized characters and their box boundaries

* image_to_data: Returns result containing box boundaries, confidences, and other information. Requires Tesseract 3.05+. For more information, please check the `Tesseract TSV documentation <https: github.com="" tesseract-ocr="" tesseract="" wiki="" command-line-usage#tsv- output-currently- available-in- 305-dev- in-master- branch-on- github="">_

Webbrowser

The webbrowser module provides a high-level interface to allow displaying Web-based documents to users. Under most circumstances, simply calling theopen() function from this module will do the right thing. Under Unix, graphical browsers are preferred under X11, but text-mode browsers will be used if graphical browsers are not available or an X11 display isn't available. If text-mode browsers are used, the calling process will block until the user exits the browser. If the environment variable BROWSER exists, it is interpreted to override the platform default list of browsers, as an os.pathsep-separated list of browsers to try in order. When the value of a list part contains the string %s, then it is interpreted as a literal browser command line to be used with the argument URL substituted for %s; if the part does not contain %s, it is simply interpreted as the name of the browser to launch. [1] For non-Unix platforms, or when a remote browser is available on Unix, the controlling process will not wait for the user to finish with the browser but allow the remote browser to maintain its own windows on the display. If remote browsers are not available on Unix, the controlling process will launch a new browser and wait.

The script webbrowser can be used as a command-line interface for the module. It accepts a URL as the argument. It accepts the following optional parameters: -n opens the URL in a new browser window, if possible; -t opens the URL in a new browser page ("tab").

Os

This module provides a portable way of using operating system dependent functionality. If you just want to read or write a file see open(), if you want to manipulate paths, see the os.path module, and if you want to read all the lines in all the files on the command line see the file input module. For creating temporary files and directories see the temp file module, and for high-level file and directory handling see the shutil module. Notes on the availability of these functions:

- The design of all built-in operating system dependent modules of Python is such that as long as the same functionality is available, it uses the same interface; for example the function os.stat(path) returns stat information about path in the same format (which happens to have originated with the POSIX interface).
- Extensions peculiar to a particular operating system are also available through the os module, but using them is of course a threat to portability.

- All functions accepting path or file names accept both bytes and string objects, and result in an object of the same type, if a path or file name is returned.
- An “Availability: Unix” note means that this function is commonly found on Unix systems. It does not make any claims about its existence on a specific operating system.
- If not separately noted, all functions that claim “Availability: Unix” are supported on Mac OS X, which builds on a Unix core.

Sys

This module provides access to some variables used or maintained by the interpreter and to functions that interact strongly with the interpreter. It is always available.

sys.argv

The list of command line arguments passed to a Python script. argv[0] is the script name (it is operating

system dependent whether this is a full pathname or not). If the command was executed using the -c
command line option to the interpreter, argv[0] is set to the string '-c'. If no script name was passed to the Python interpreter, argv[0] is the empty string. To loop over the standard input, or the list of files given on the command line, see the fileinput module.

4.2 HARDWARE REQUIREMENTS

Processor: Minimum Intel Pentium 200mhz or any higher version like i3, i5 or i7 are suggestable.
Ram: 1gb minimum and 2gb recommended and any higher version is very feasible to work
Disk: A space of minimum of 512 mb is needed to carry out the experiment with the given number of test cases to check the accuracy and high amount of space can allow user to work with more number of test cases test cases.
os: win 7 or higher version of windows is optimal for this experiment.

5. SYSTEM DESIGN

5.1 INTRODUCTION

System design is transition from a user-oriented document to programmers or data base personnel. The design is a solution, how to approach to the creation of a new system. This is composed of several steps. It provides the understanding and procedural details necessary for implementing the system recommended in the feasibility study. Designing goes through logical and physical stages of development, logical design reviews the present physical system, prepare input and output specification, details of implementation plan and prepare a logical design walkthrough. The database tables are designed by analyzing functions involved in the system and format of the fields is also designed. The fields in the Database tables should define their role in the system. The unnecessary fields should be avoided because it affects the storage areas of the system. Then in the input and output screen design, the design should be made user friendly. The menu should be precise and compact.

5.2 UML DIAGRAMS

Introduction to UML

UML is the language used in the information technology industries versions of blue print. It is a method for describing the system architecture in details using this blue print. It becomes much easier to build or maintain a system and to ensure that the system we hold up to requirement changes.

Definition: UML is general purpose visual modeling language that is used to

- Specify
- Visualize
- Construct
- Document

- The artifacts of the system
- Reasons to model
- To communicate the desired structure and behavior of the system.
- To visualize and control the system architecture.
- To better understand the system and expose opportunities for specification and reuse.
- To manage task

5.3 APPLICATIONS OF UML

The UML is intended primarily for software intensive systems. It has been used effectively for such domain as

1. Aerospace
2. Enterprise information systems
3. Banking and financial services
4. Telecommunications
5. Transportation
6. Defense
7. Retail
8. Medical Electronics
9. Scientific
10. Distributed Web-based services

5.4 A CONCEPTUAL MODEL OF THE UML

Model is a blue simplification of reality. A model provides the blueprints of a system. We build models so that we can better understand the system we are developing. We build models of complex systems because we cannot comprehend such a system in its entirety Modelling helps us to visualize a system as it is or as we want it to be. They give us a template that guides us in constructing System and documents the decisions we have made. Principles of Modelling: The choice of what models to create has a profound influence on how a problem is attacked and how a solution is shaped. Every model may be expressed at different levels of precision. The best models

are connected to reality. No single model is sufficient. Every nontrivial system best approached through a small set of nearly independent models.

5.5 BUILDING BLOCKS OF THE UML

The three kinds of building blocks in UML are

- Things
- Relationships
- Diagrams

Things in UML

Things are the most important building blocks of UML. Things can be:

1.Structural

2.Behavioural

3.Grouping

4.Annotational

STRUCTURAL THINGS: The Structural things define the static part of the model. They represent physical and conceptual elements. Following are the brief descriptions of the structural things.

Class: Class represents set of objects having similar responsibilities

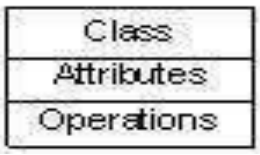


Fig. 5.1(a)

Interface: Interface defines set of operations which specify the responsibility of a class

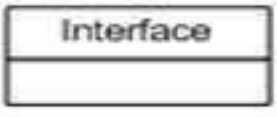


Fig. 5.1(b)

Collaboration: Collaboration defines interaction between elements

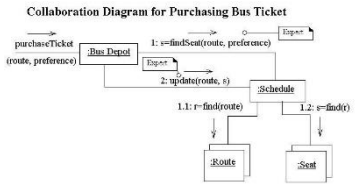


Fig. 5.1(c)

Use case: Use case represents set of actions performed by a system for a specific goal

Fig. 5.1(d)

Component: Component describes physical part of a system

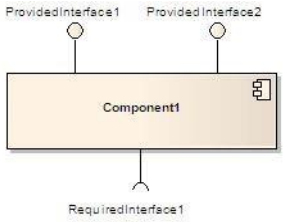


Fig. 5.1(d)

Node: A node can be defined as a physical element that exists at run time

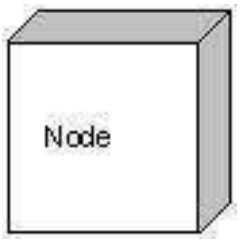


Fig. 5.1(e)

BEHAVIORAL THINGS: A behavioral thing consists of the dynamic parts of UML models. Following are the behavioral things:

Interaction: Interaction is defined as a behavior that consists of a group of messages exchanged among elements to accomplish a specific task.

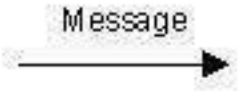


Fig. 5.1(f)

State machine: State machine is useful when the state of an object in its life cycle is important. It defines the sequence of states an object goes through in response to events. Events are external factors responsible for state change

Fig. 5.1(g)

GROUPING THINGS: Grouping things can be defined as a mechanism to group elements of a UML model together. There is only one grouping thing available:

Package: Package is the only one grouping thing available for gathering structural and behavioral things

Fig. 5.1(h)

ANNOTATIONAL THINGS: Annotational things can be defined as a mechanism to capture remarks, descriptions, and comments of UML model elements. Note is the only one Annotational thing available

Note: A note is used to render comments, constraints etc. of an UML element

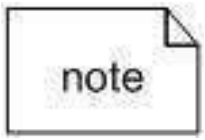


Fig. 5.1(i)

Relationships

Relationship is another most important building block of UML. It shows how elements are associated with each other and this association describes the functionality of an application. There are four kinds of relationships available:

DEPENDENCY: Dependency is a relationship between two things in which change in one element also affects the other one.

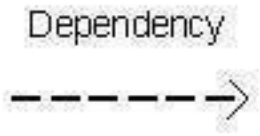


Fig. 5.1(j)

Association: Association is basically a set of links that connects elements of an UML model. It also describes how many objects are taking part in that relationship.

Fig. 5.1(k)

Generalization: Generalization can be defined as a relationship which connects a specialized element with a generalized element. It basically describes inheritance relationship in the world of objects.

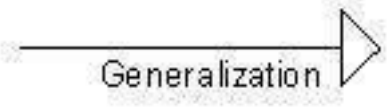


Fig. 5.1(l)

Realization: Realization can be defined as a relationship in which two elements are connected. One element describes some responsibility which is not implemented and the other one implements them.

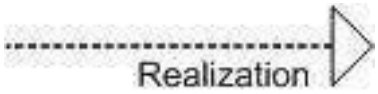


Fig. 5.1(m)

Diagrams

ACTIVITY DIAGRAM:

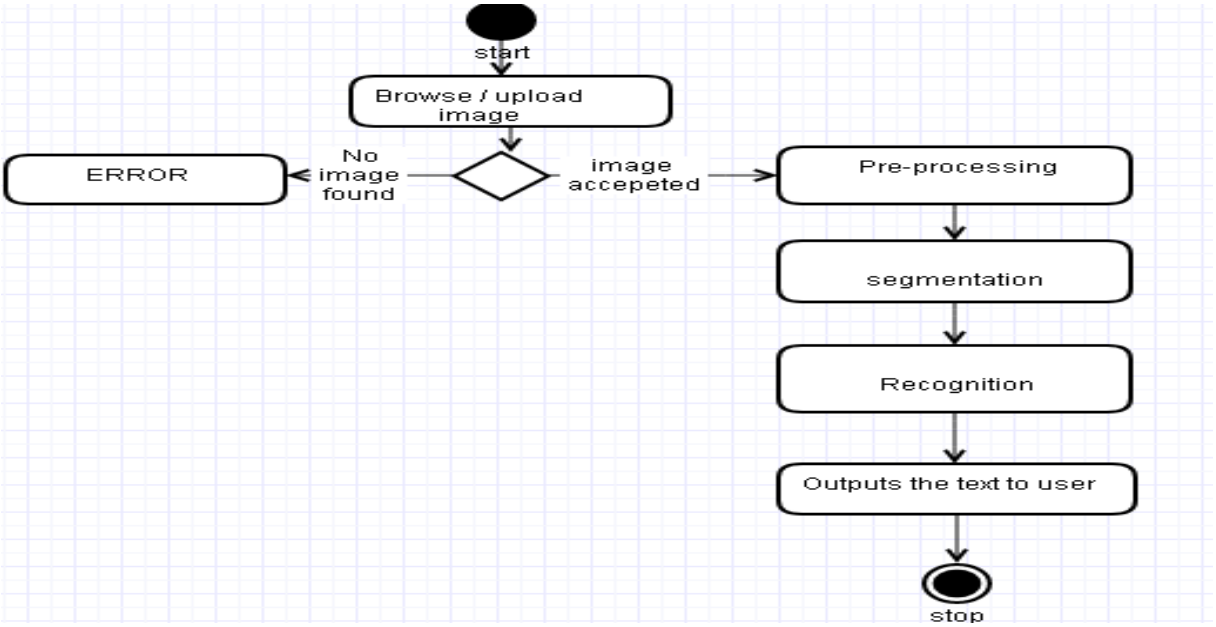


Fig. 5.2

USECASE DIAGRAM:

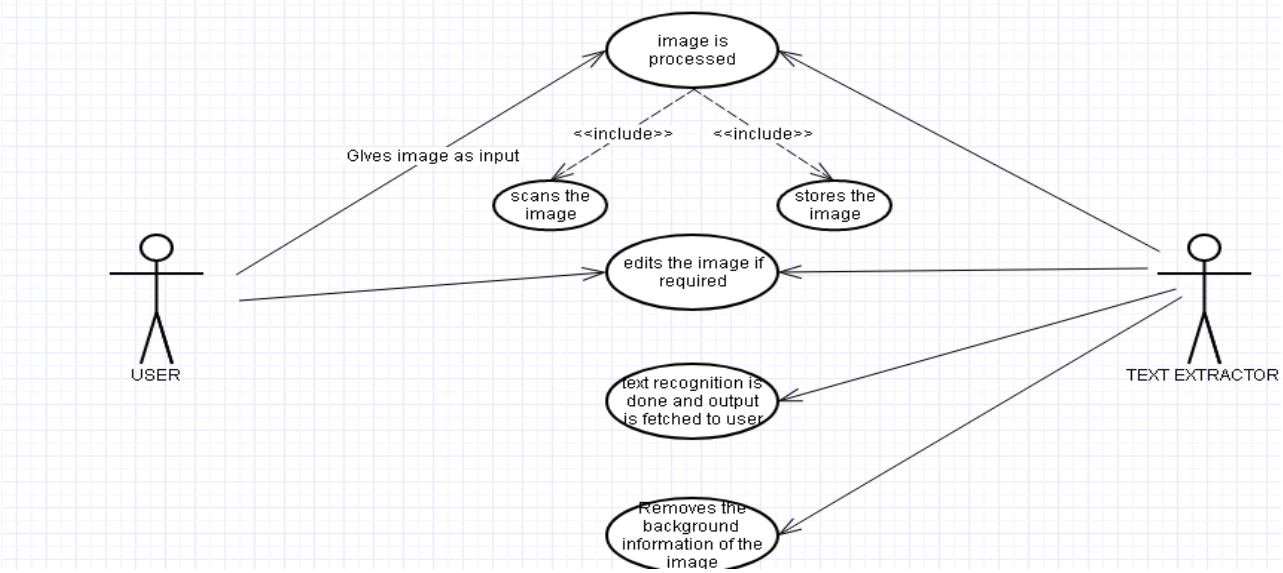


Fig. 5.3

SEQUENCE DIAGRAM**:**

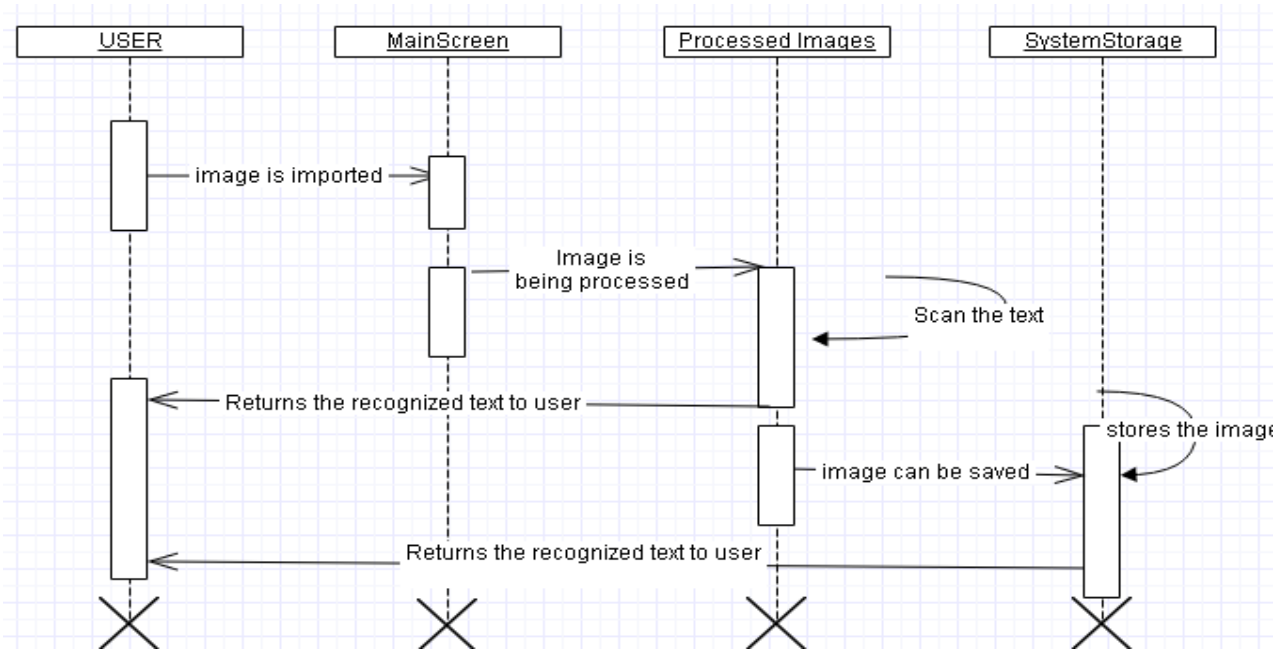


Fig. 5.4

CLASS DIAGRAM**:**

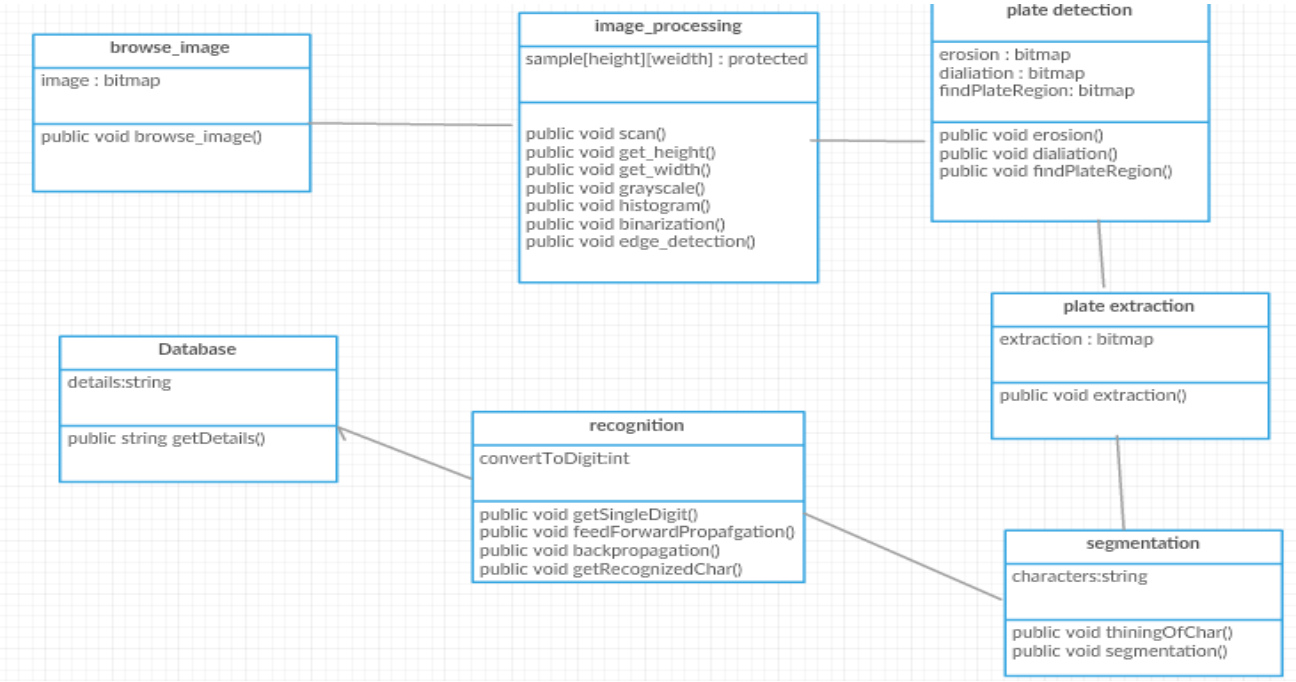


Fig. 5.5

BLOCK DIAGRAM:

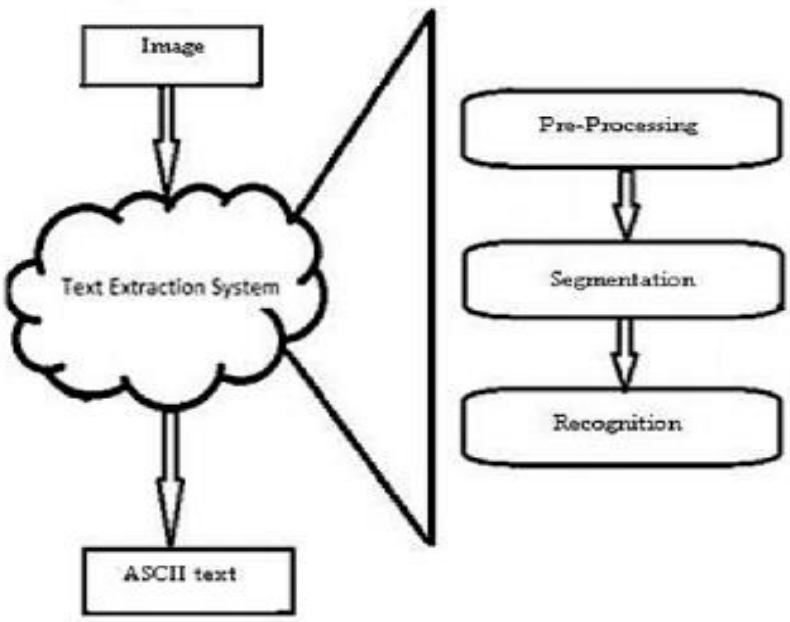


Fig. 5.6

ARCHITECTURE:

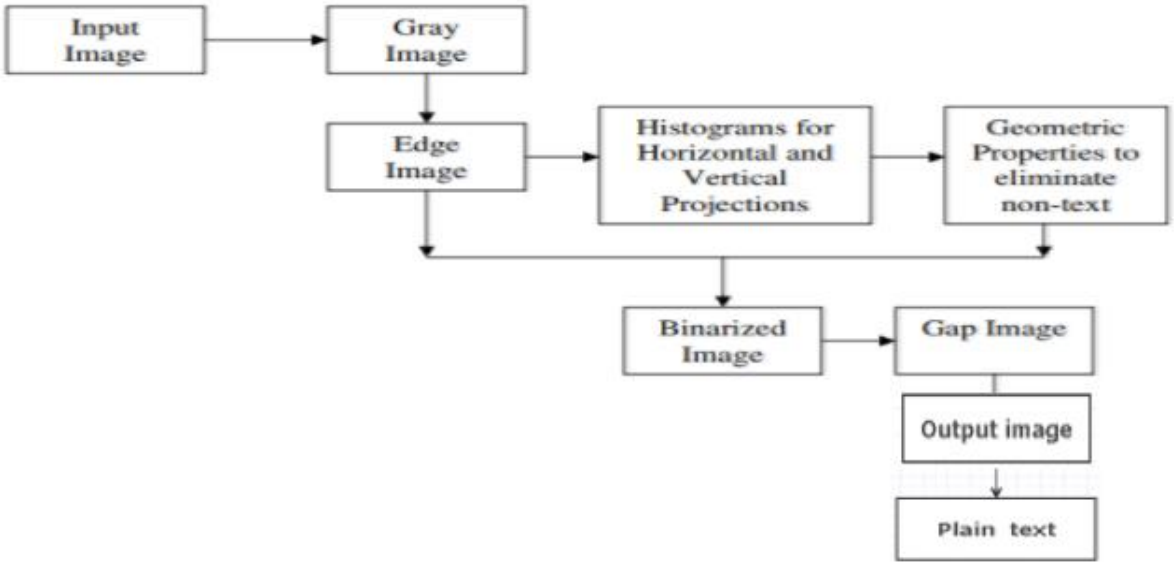


Fig. 5.7

6. IMPLEMENTATION

6.1 ALGORITHM FOR EDGE-BASED TEXT REGION EXTRACTION [5]

The basic steps of the edge-based text extraction algorithm are given below and diagrammed in Figure The details are explained in the following sections.

1.Create a Gaussian pyramid by convolving the input image with a Gaussian kernel and successively down-sample each direction by half. (Levels: 4)

2. Create directional kernels to detect edges at 0, 45, 90 and 135 orientations.

3. Convolve each image in the Gaussian pyramid with each orientation filter.

4. Combine the results of step 3 to create the Feature Map.

5. Dilate the resultant image using a sufficiently large structuring element (7x7 [1]) to cluster candidate text regions together.

6. Create final output image with text in white pixels against a plain black background.

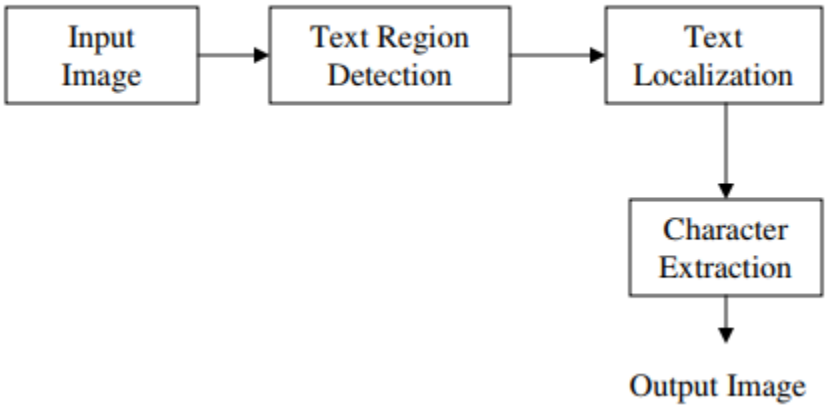


Figure 6.1. Basic block diagram for edge-based text detection

As given in [5], the procedure for extracting a text region from an image can be broadly classified into three basic steps: (1) detection of the text region in the image, (2) localization of the region, and (3) creating the extracted output character image

6.1.1 DETECTION

This section corresponds to Steps 1 to 4 of 3.1. Given an input image, the region with a possibility of text in the image is detected [5]. A Gaussian pyramid is created by successively filtering the input image with a Gaussian kernel of size 3x3 and down sampling the image in each direction by

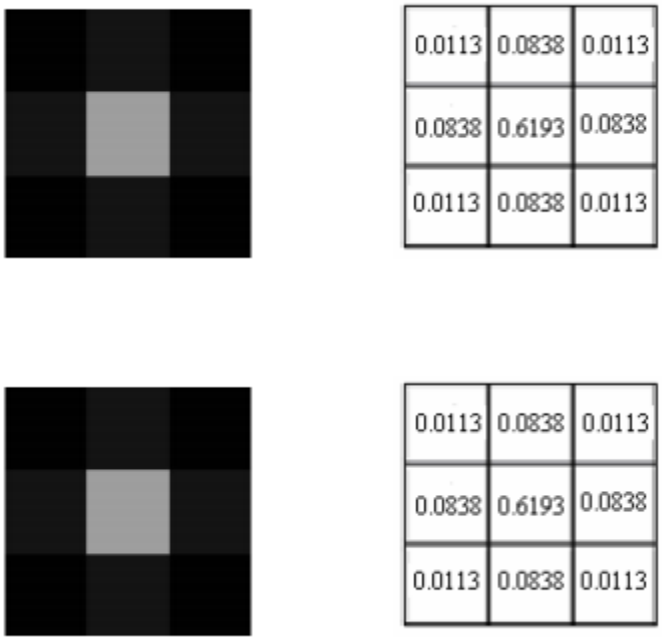


Figure 6.2. Default filter returned by the fspecial Gaussian function.

Size [3 3], Sigma 0.5

half. Down sampling refers to the process whereby an image is resized to a lower resolution from its original resolution. A Gaussian filter of size 3x3 will be used as shown in Figure 6.2. Each level in the pyramid corresponds to the input image at a different resolution. A sample Gaussian pyramid with 4 levels of resolution is shown in Figure 6.3. These images are next convolved with directional filters at different orientation kernels for edge detection in the horizontal (0°), vertical (90°) and diagonal (45°, 135°) directions. The kernels used are shown in Figure 6.5.

Figure 6.3. Sample Gaussian pyramid with 4 levels

Figure 6.4. Each resolution image resized to original image size

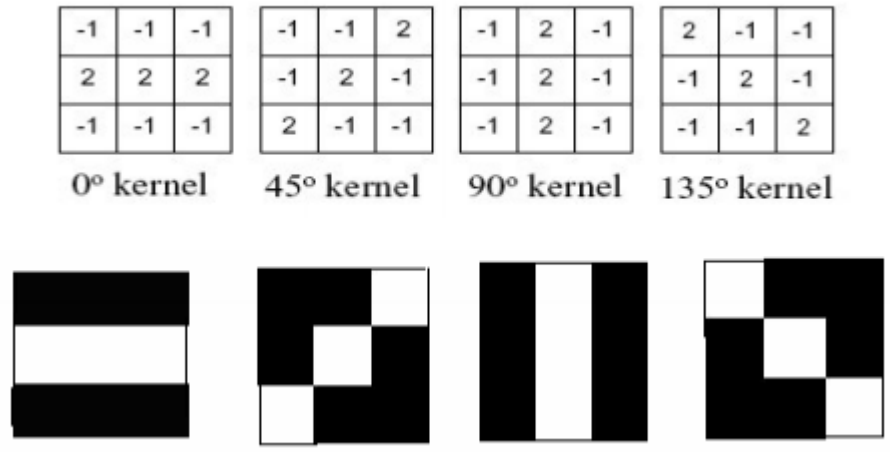


Figure 6.5. The directional kernels [1]

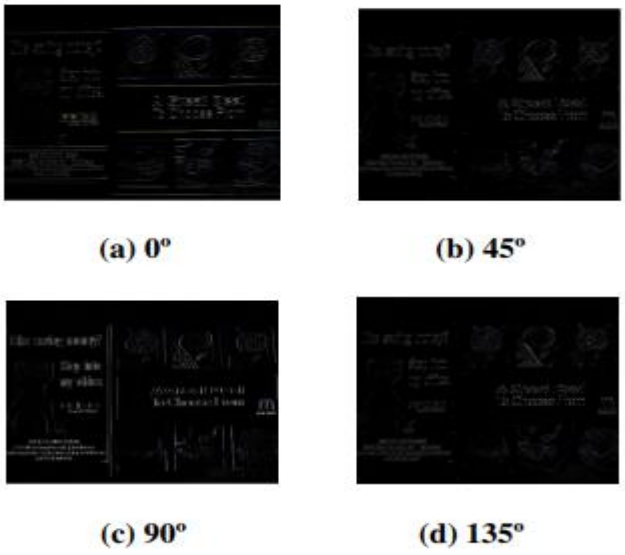


Figure 6.6. Sample image from Figure 6.3 after convolution with each directional kernel Note how the edge information in each direction is highlighted.

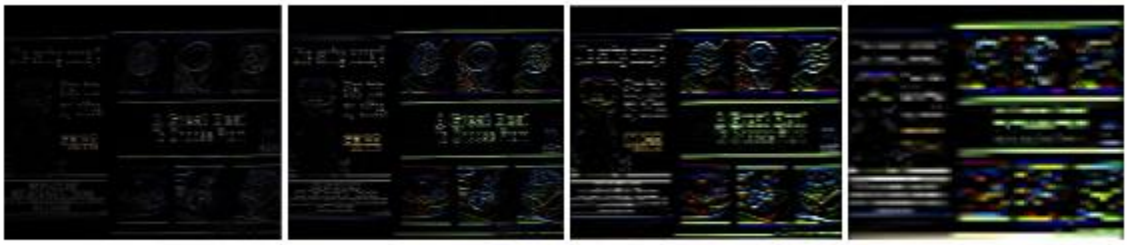
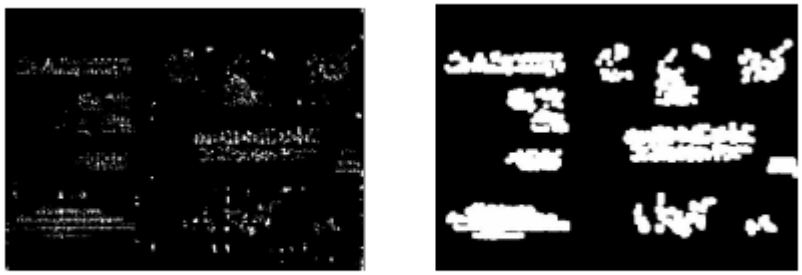


Figure 6.7. Sample resized image of the pyramid after convolution with 0º kernel

After convolving the image with the orientation kernels, a feature map is created. A weighting factor is associated with each pixel to classify it as a candidate or noncandidate for text region. A pixel is a candidate for text if it is highlighted in all of the edge maps created by the directional filters. Thus, the feature map is a combination of all 13 edge maps at different scales and orientations with the highest weighted pixels present in the resultant map.

6.1.2 LOCALIZATION

This section corresponds to Step 5. The process of localization involves further enhancing the text regions by eliminating non-text regions [9]. One of the properties of text is that usually all characters appear close to each other in the image, thus forming a cluster. By using a morphological dilation operation, these possible text pixels can be clustered together, eliminating pixels that are far from the candidate text regions. Dilation is an operation which expands or enhances the region of interest, using a structural element of the required shape and/or size. The process of dilation is carried out using a very large structuring element in order to enhance the regions which lie close to each other. In this algorithm, a structuring element of size [7x7] has been used [5]. Figure 6.8 below shows the result before and after dilation.



(a) (b)

Figure 6.8. (a) Before dilation (b) After dilation

The resultant image after dilation may consist of some non-text regions or noise which needs to be eliminated. An area-based filtering is carried out to eliminate noise blobs present in the image. According to [7], only those regions in the final image are retained which have an area greater than or equal to 1/20 of the maximum area region.

6.1.3. CHARACTER EXTRACTION

This section corresponds to Step 6 of 3.1. The common OCR systems available require the input image to be such that the characters can be easily parsed and recognized. The text and background should be monochrome and background-to-text contrast should be high [5]. Thus, this process generates an output image with white text against a black background [1,4]. A sample test

image [1,2] and its resultant output image from the edge-based text detection algorithm are shown in Figures 6.9(a) and 6.9(b) below.

Figure 6.9. (a) Original Image [1,2] (b) Result

6.2 ALGORITHM FOR CONNECTED COMPONENT BASED TEXT REGION EXTRACTION [3]

The basic steps of the connected-component text extraction algorithm are given below, and diagrammed in Figure 6.10. The details are discussed in the following sections.

1. Convert the input image to YUV color space. The luminance(Y) value is used for further processing.

The output is a gray image.

2. Convert the gray image to an edge image.

3. Compute the horizontal and vertical projection profiles of candidate text regions using a histogram with an appropriate threshold value.

4. Use geometric properties of text such as width to height ratio of characters to eliminate possible non text regions.

5. Binarize the edge image enhancing only the text regions against a plain black background.

6. Create the Gap Image (as explained in the next section) using the gap-filling process and use this as a reference to further eliminate non-text regions from the output.

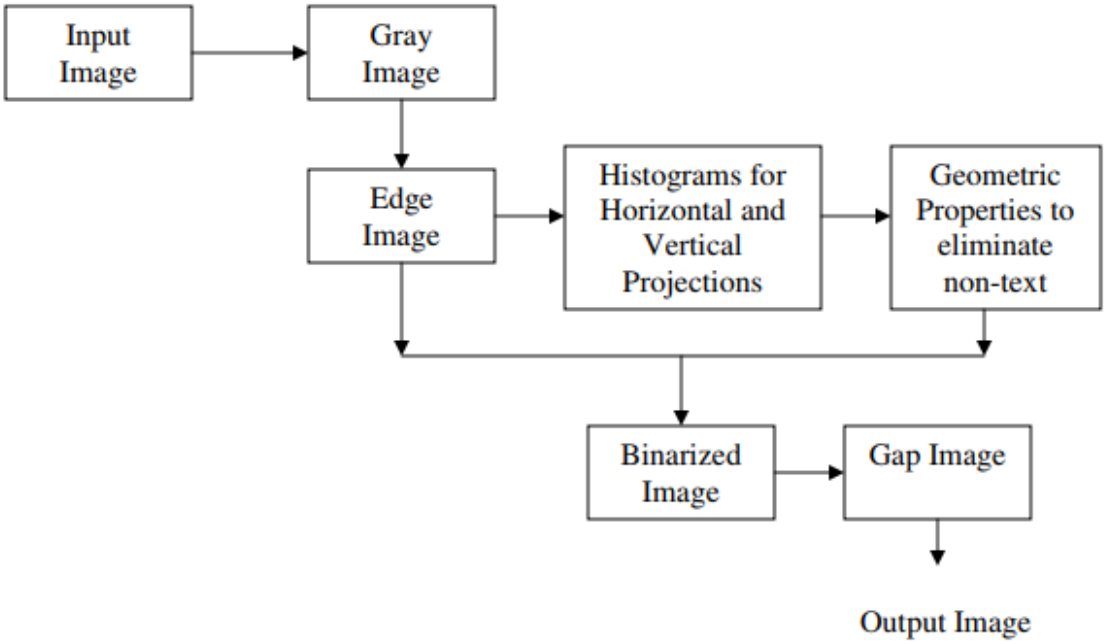


Figure 6.10. Basic Block diagram for Connected Component based text extraction.

6.2.1 PRE-PROCESSING

This section corresponds to Step 1. The input image is pre-processed to facilitate easier detection of text regions. As proposed in [10], the image is converted to the YUV color space (luminance + chrominance), and only the luminance(Y) channel is used for further processing. The conversion is done using the MATLAB function 'rgb2ycbcr' which takes the input RGB image and converts it into the corresponding YUV image. The individual channels can be extracted from this new image. The Y channel refers to brightness or intensity of the image whereas the U and the V channels refer to the actual color information [8]. Since text present in an image has more contrast with its background, by using only the Y channel, the image can be converted to a grayscale Input Image Gray Image Edge Image Histograms for Horizontal and Vertical Projections Geometric Properties to eliminate non-text Binarized Image Gap Image 17 image with only the brightness / contrast information present. Figure 6.11(2, 3, 4) show the Y, U and V channels respectively for an input test image [1,2] in (1).

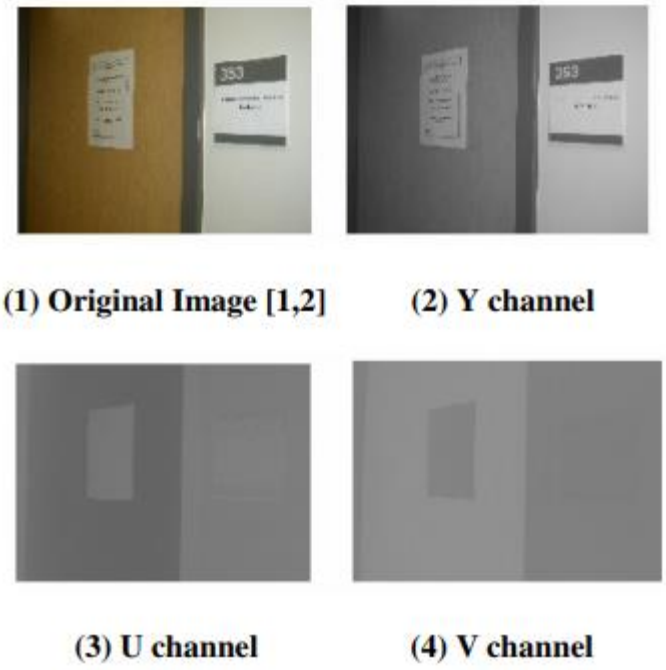


Figure 6.11. YUV channels for test image (1)

6.2.2 DETECTION OF EDGES

This section corresponds to Step 2. In this process, the connected-component based approach is used to make possible text regions stand out as compared to non-text regions. Every pixel in the edge image is assigned a weight with respect to its neighbors in each direction. As depicted in Figure 6.12, this weight value is the maximum value between the pixel and its neighbors in the left (L), upper (U) and upper-right (UR) directions [3]. The algorithm proposed in [3] uses these three neighbor values to detect edges in horizontal, vertical and diagonal directions. The resultant edge image obtained is sharpened in order 18 to increase contrast between the detected edges and its background, making easier to extract text regions. Figure 6.13 below shows the sharpened edge image for the Y Channel gray image G from Figure 6.11, obtained by the algorithm proposed in [3].

The algorithm for computing the edge image E, as proposed in [3] is as follows:

1. Assign left, upper, upperRight to 0.

2. For all the pixels in the gray image G(x,y) do

a. left = (G(x,y) – G(x-1,y))

b. upper = (G(x,y) – G(x,y-1))

c. upperRight = (G(x,y)-G(x+1,y-1))

d. E(x,y) = max(left, upper, upperRight)

3. Sharpen the image E by convolving it with a sharpening filter.

W (x,y) = max(L,U,UR)

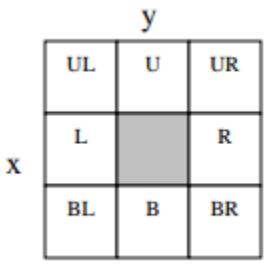


Figure 6.12. Weight for pixel (x,y)

Figure 6.13. Sharpened Edge Image

6.2.3. LOCALIZATION

This section corresponds to Step 3. In this step, the horizontal and vertical projection profiles for the candidate text regions are analyzed [9]. The sharpened edge image is considered as the input intensity image for computing the projection profiles, with white candidate text regions against a black background. The vertical projection profile shows the sum of pixels present in each column of the intensity or the sharpened image. Similarly, the horizontal projection profile shows the sum of pixels present in each row of the intensity image. These projection profiles are essentially

histograms where each bin is a count of the total number of pixels present in each row or column. The vertical and horizontal projection profiles for the sharpened edge image from Figure 6.13, are shown in Figure 6.14 (a) and (b) respectively.

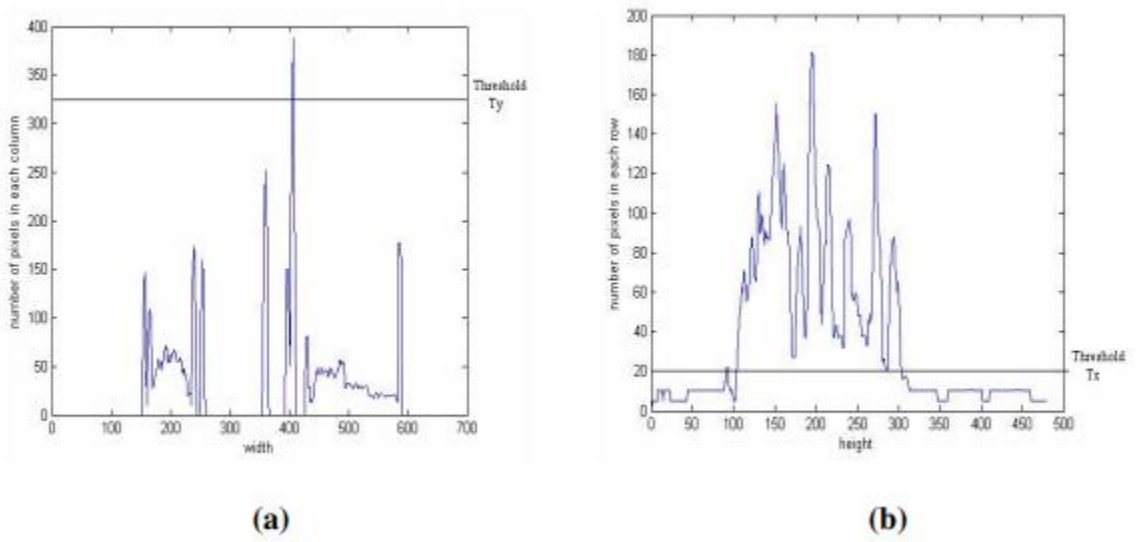


Figure 6.14. (a) Vertical Projection profile (b) Horizontal Projection profile for Sharpened image in Figure 6.13.

Candidate text regions are segmented based on adaptive threshold values, Ty and Tx are calculated for the vertical and horizontal projections respectively. Only regions that fall within the threshold limits are considered as candidates for text. The value of threshold Ty is selected to eliminate possible non-text regions such as doors, window edges etc. that have a strong vertical orientation. Similarly, the value of threshold Tx is selected to eliminate regions which might be non-text or long edges in the horizontal orientation.

$$Tx = \frac{\text{Mean(Horizontal projection profile)}}{20} \quad (3)$$

$$Ty = \text{Mean(Vertical projection profile)} + \frac{\text{Max (Vertical projection profile)}}{10} \quad (4)$$

6.2.4 ENHANCEMENT AND GAP FILLING

This section corresponds to Steps 4 to 6. The geometric ratio between the width and the height of the text characters is considered to eliminate possible non-text regions. This ratio value will be defined after experimenting on different kinds of images to get an average value. In this project, regions with minor to major axis ratio less than 10 are considered as candidate text regions for further processing. Next a gap image will be created which will be used as a reference to refine the localization of the detected text regions [3]. If a pixel in the binary edge image created is surrounded by black (background) pixels in the vertical, horizontal and diagonal directions, this pixel is also 21 substituted with the background value. This process is known as gap filling. An example of extracted text using this technique is shown in Figure 6.15.

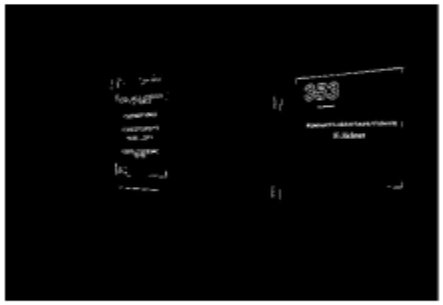


Figure 6.15. Result obtained by connected component-based text detection algorithm for test image in Figure 6.11 (1)

6.3 TEST CASES & OUTPUTS

Original image

GRAY Conversion

Binarized

Dilated

Threshold

Segmentation

After classification of the Gap image final text will be yielded as follows and an audio output is generated after text extraction

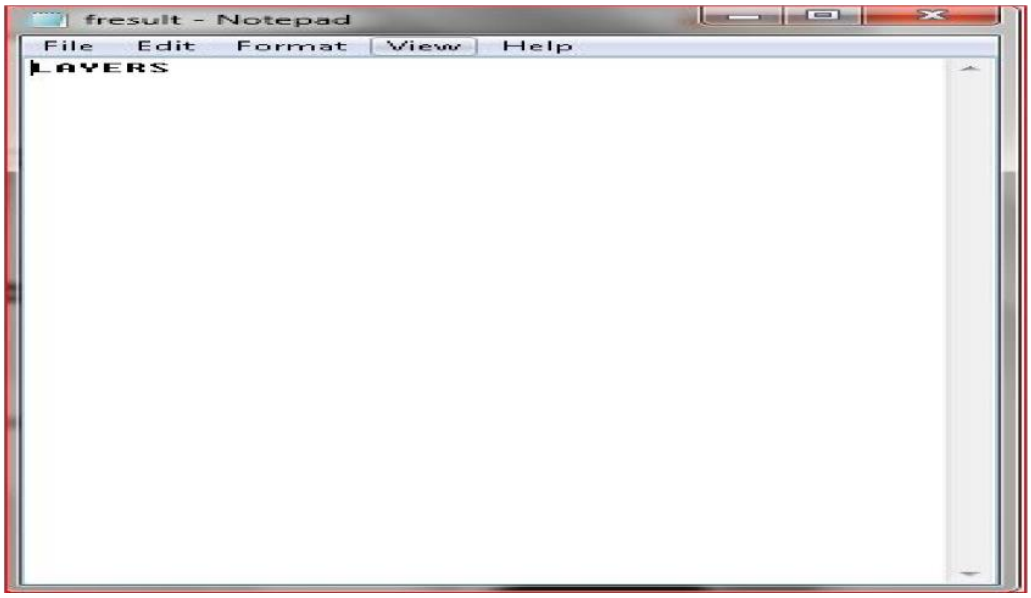


Fig. 6.16 Implementation Steps

Test cases : Success

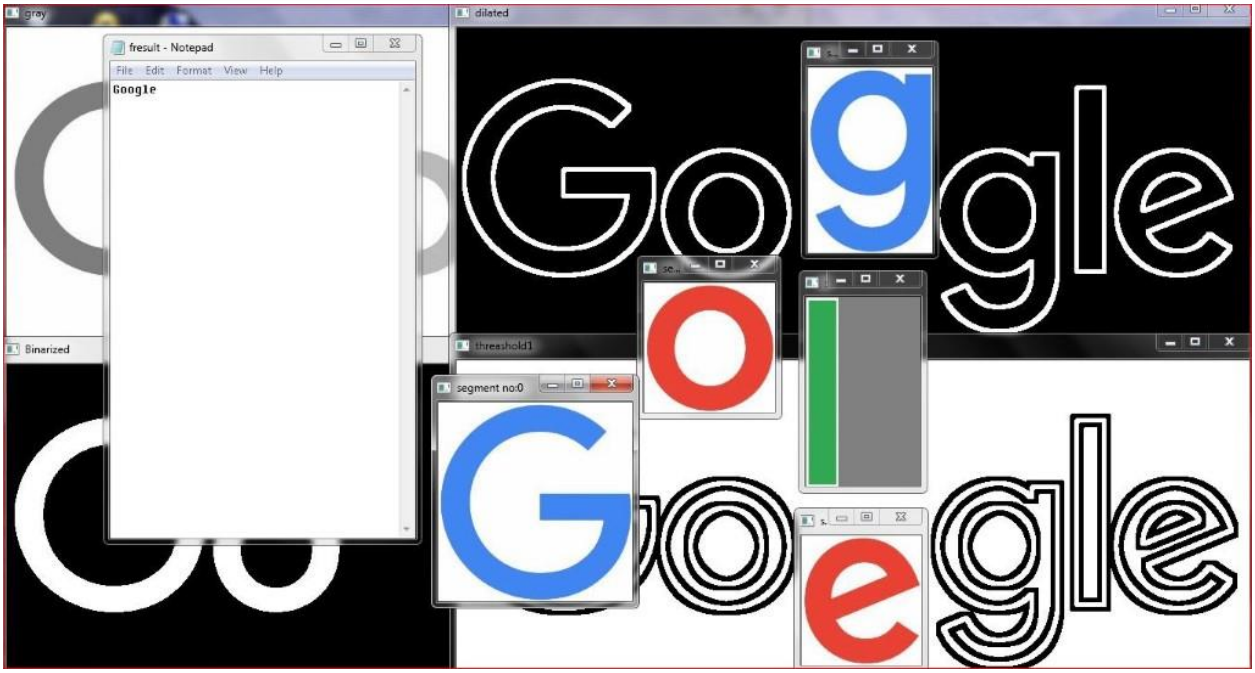


Fig. 6.17(a)


4
CREATE FACEBOOK TEXT POST
With Colored Background

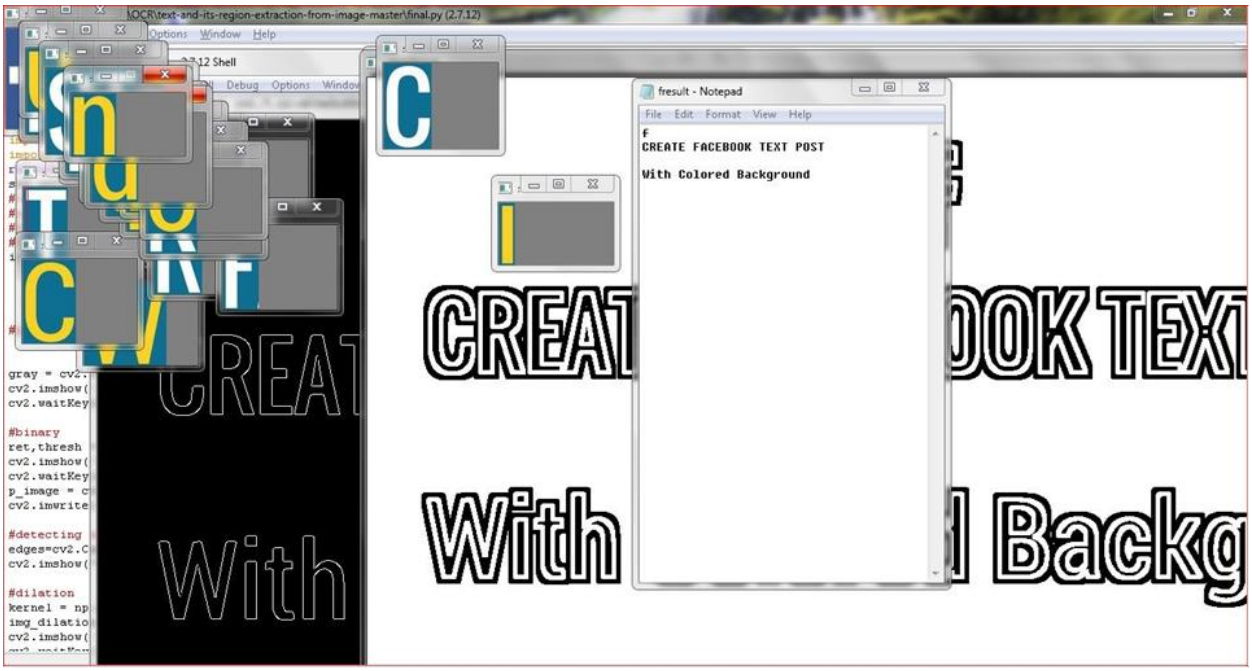

fresult - Notepad
File Edit Format View Help
f
CREATE FACEBOOK TEXT POST
With Colored Background

Fig. 6.17(b)


VNR VJIET


gray
VNR VJIET
Binarized
dilated
threashold1
segment no:0 (Not Responding)
fresult - Notepad
File Edit Format View Help
VNR VJIET

Fig. 6.17(c)


UNCooL

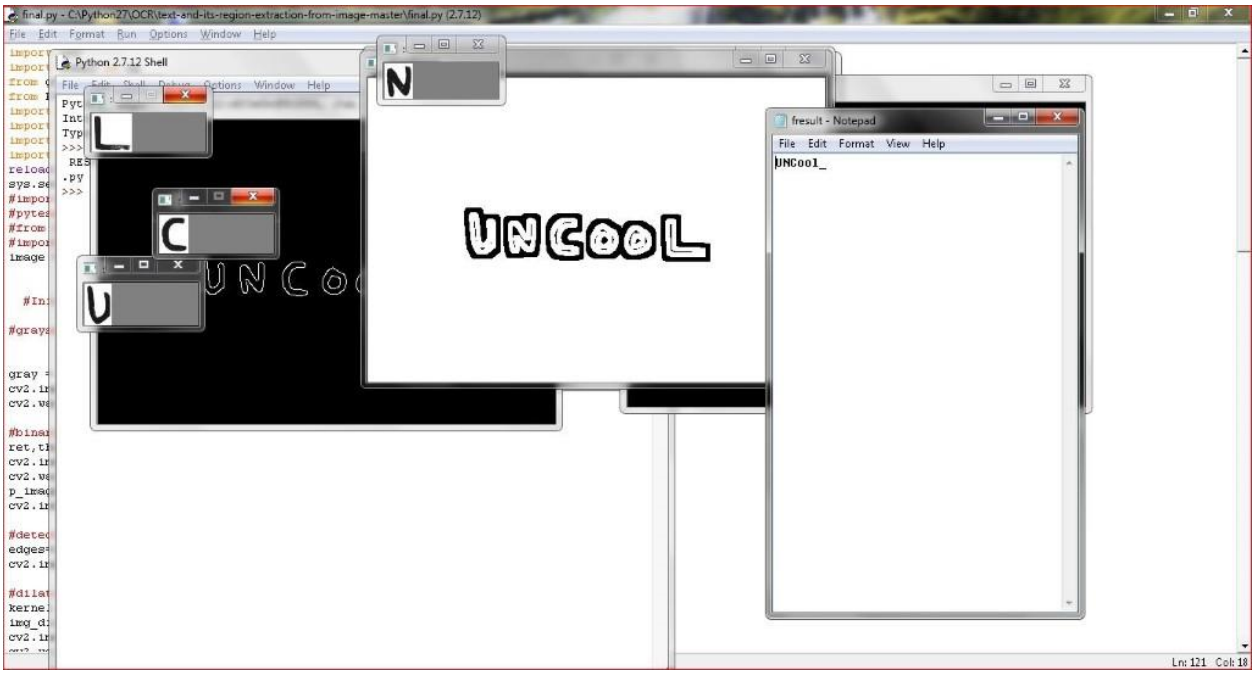

final.py - C:\Python27\OCR\text-and-its-region-extraction-from-image-master\final.py (2.7.12)
File Edit Format Run Options Window Help
Python 2.7.12 Shell
UNCooL
fresult - Notepad
File Edit Format View Help
UNCool_
Ln: 121 Col: 18

Fig. 6.17(d)


34
What
Who
Why
When
Where
How
Where
What
When
Who
Why
segment no:1
When
How
segment no:3
What
Who
Why
fresult - Notepad
File Edit Format View Help
What
Who Why
When Where
Binarized
What
Who
Why
When
Where

Fig. 6.17(e)


49
EMOJI REACTION PROJECT
Turn your reaction into action.

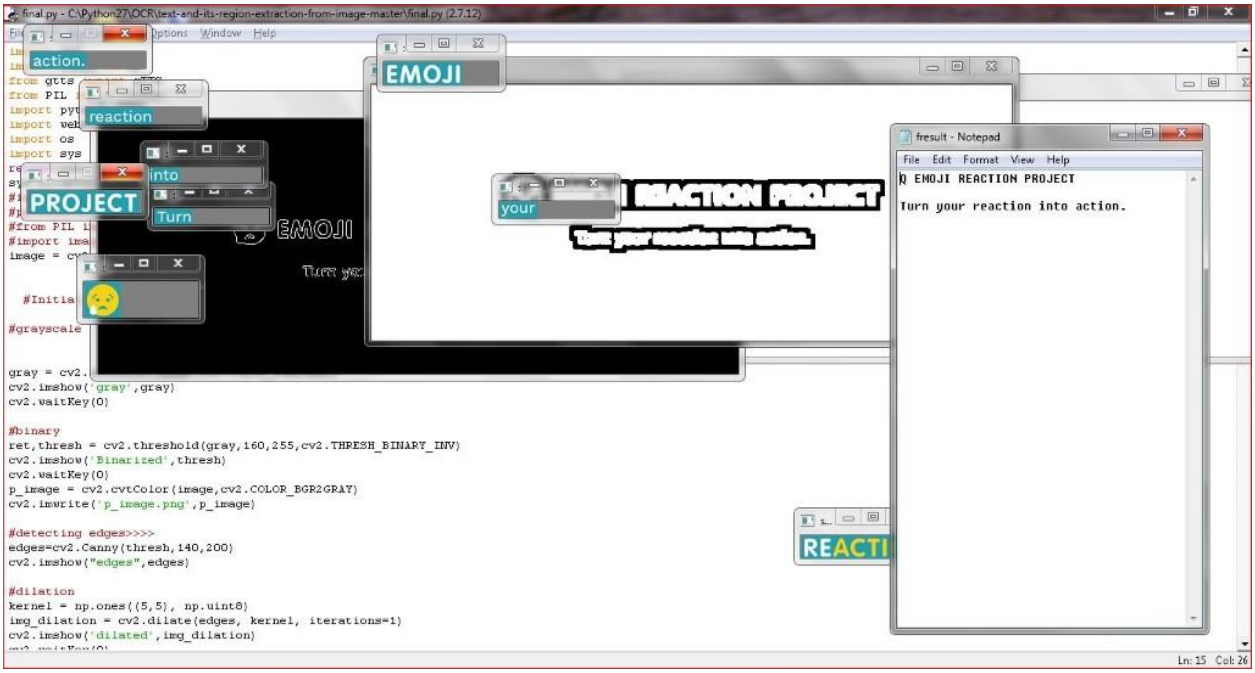

action.
reaction
into
PROJECT
Turn
EMOJI
EMOJI
your
fresult - Notepad
File Edit Format View Help
EMOJI REACTION PROJECT
Turn your reaction into action.
#grayscale
gray = cv2.
cv2.imshow('gray',gray)
cv2.waitKey(0)
#binary
ret,thresh = cv2.threshold(gray,160,255,cv2.THRESH_BINARY_INV)
cv2.imshow('Binarized',thresh)
cv2.waitKey(0)
p_image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
cv2.imwrite('p_image.png',p_image)
#detecting edges>>>>
edges=cv2.Canny(thresh,140,200)
cv2.imshow("edges",edges)
#dilation
kernel = np.ones((5,5), np.uint8)
img_dilation = cv2.dilate(edges, kernel, iterations=1)
cv2.imshow('dilated',img_dilation)

Fig. 6.17(f)


develop(CODE);
test(CODE);
be(AWESOME);

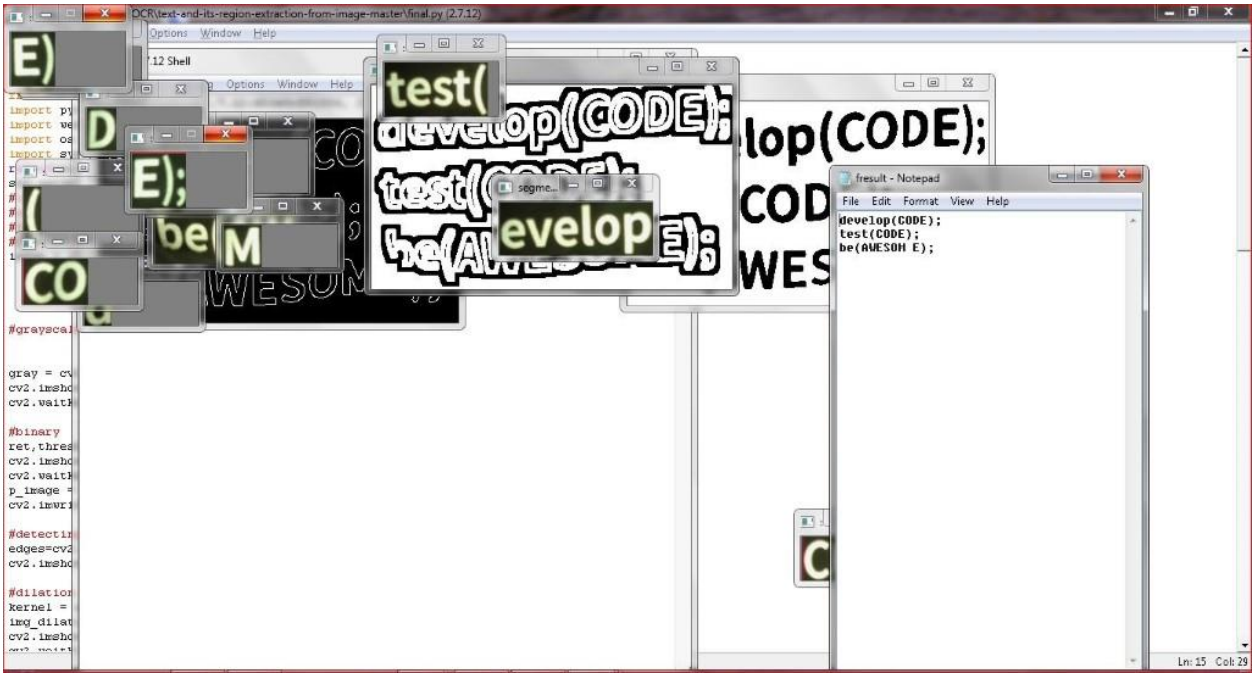

develop(CODE);
test(CODE);
be(AWESOH E);

Fig. 6.17(g)

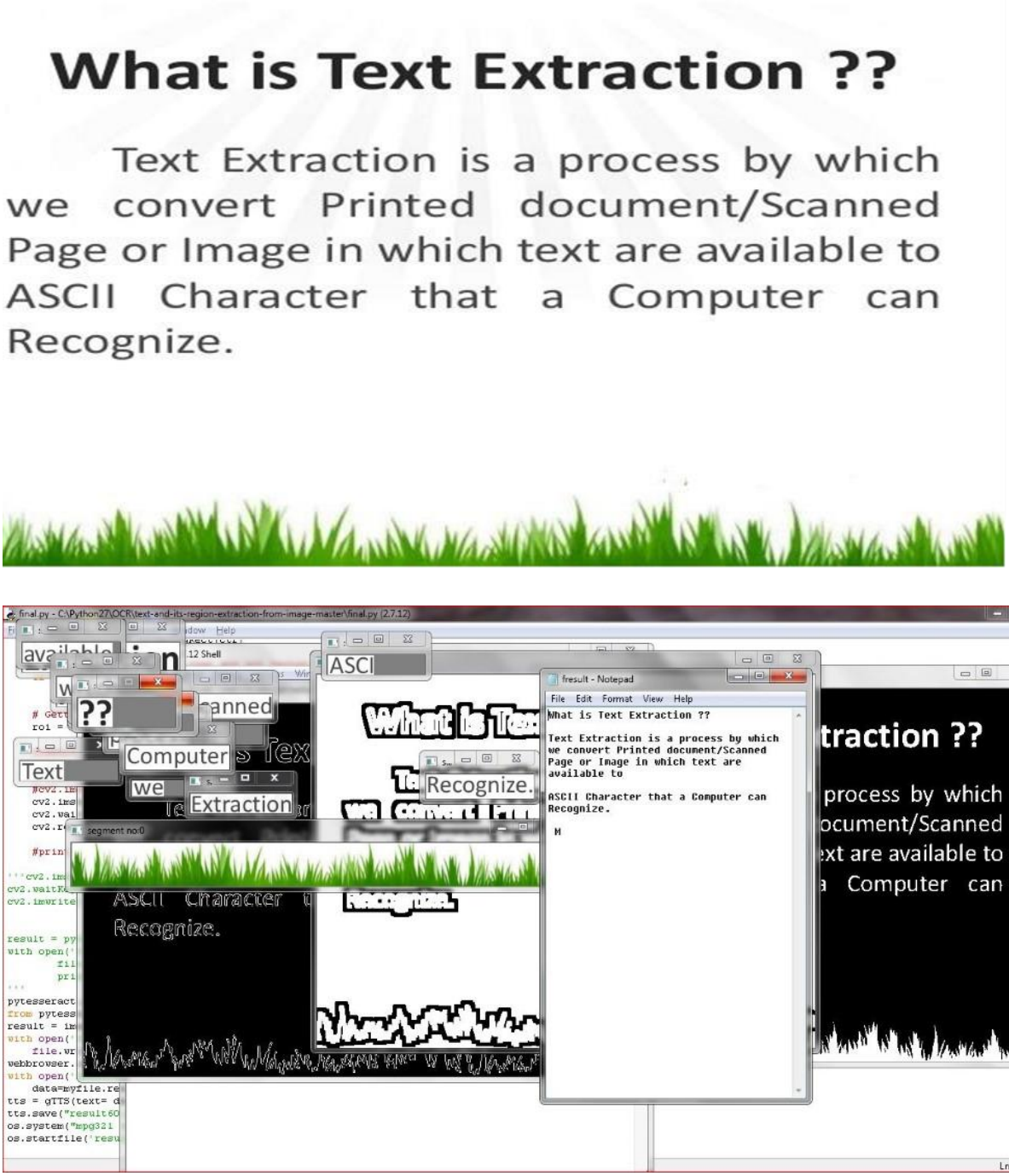

What is Text Extraction ??
Text Extraction is a process by which we convert Printed document/Scanned Page or Image in which text are available to ASCII Character that a Computer can Recognize.
fresult - Notepad
File Edit Format View Help
What is Text Extraction ??
Text Extraction is a process by which we convert Printed document/Scanned Page or Image in which text are available to
ASCII Character that a Computer can Recognize.

Fig. 6.17(h)


MOUNTAINS

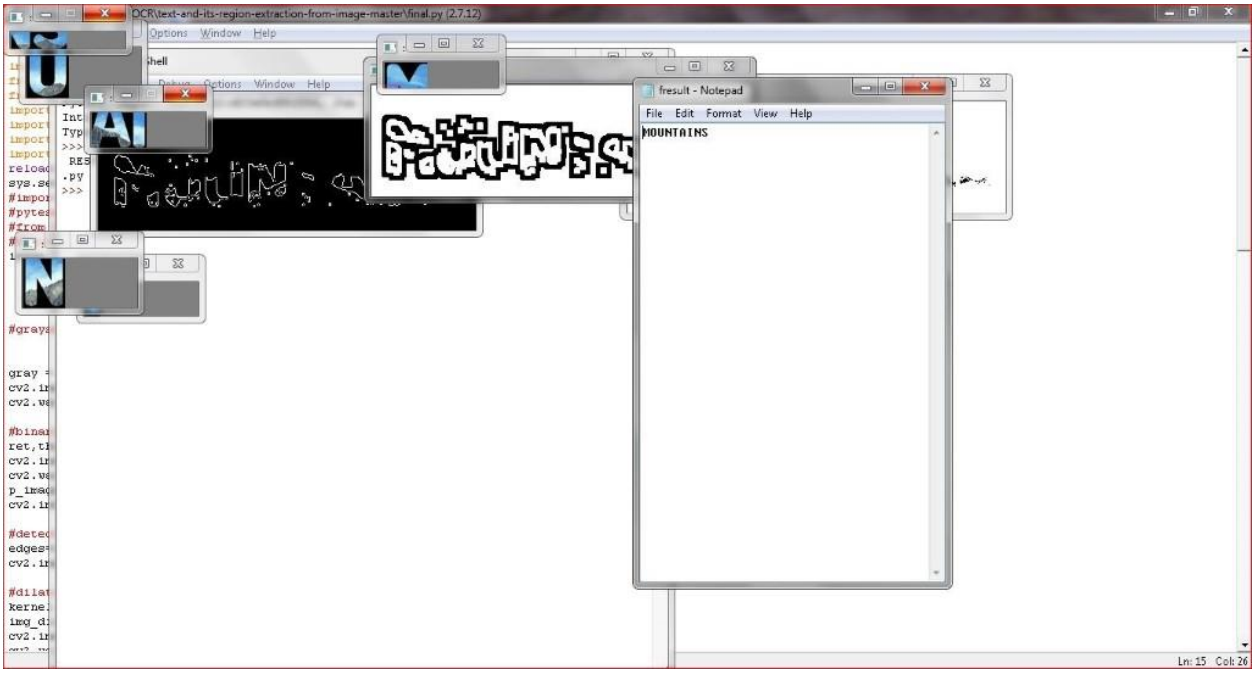

fresult - Notepad
File Edit Format View Help
MOUNTAINS

Fig. 6.17(i)

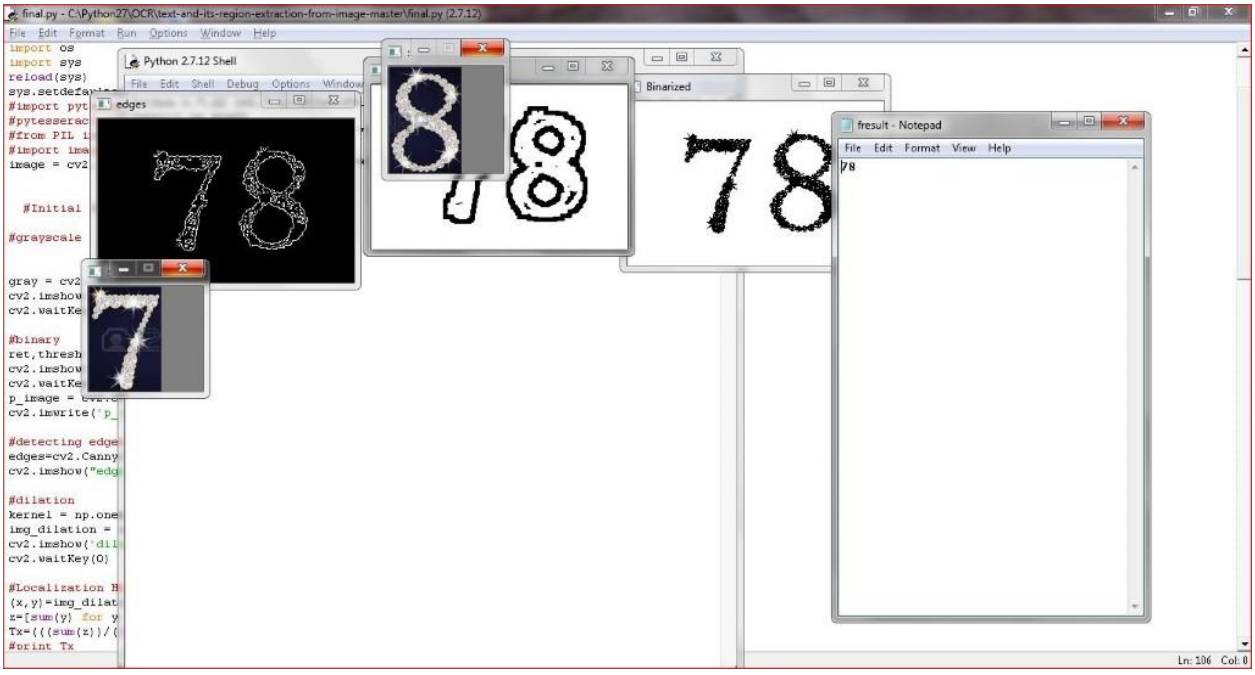

final.py - C:\Python27\OCR\text-and-its-region-extraction-from-image-master\final.py (2.7.12)
Python 2.7.12 Shell
edges
Binarized
fresult - Notepad
78

Fig. 6.17(j)

115


NETFLIX

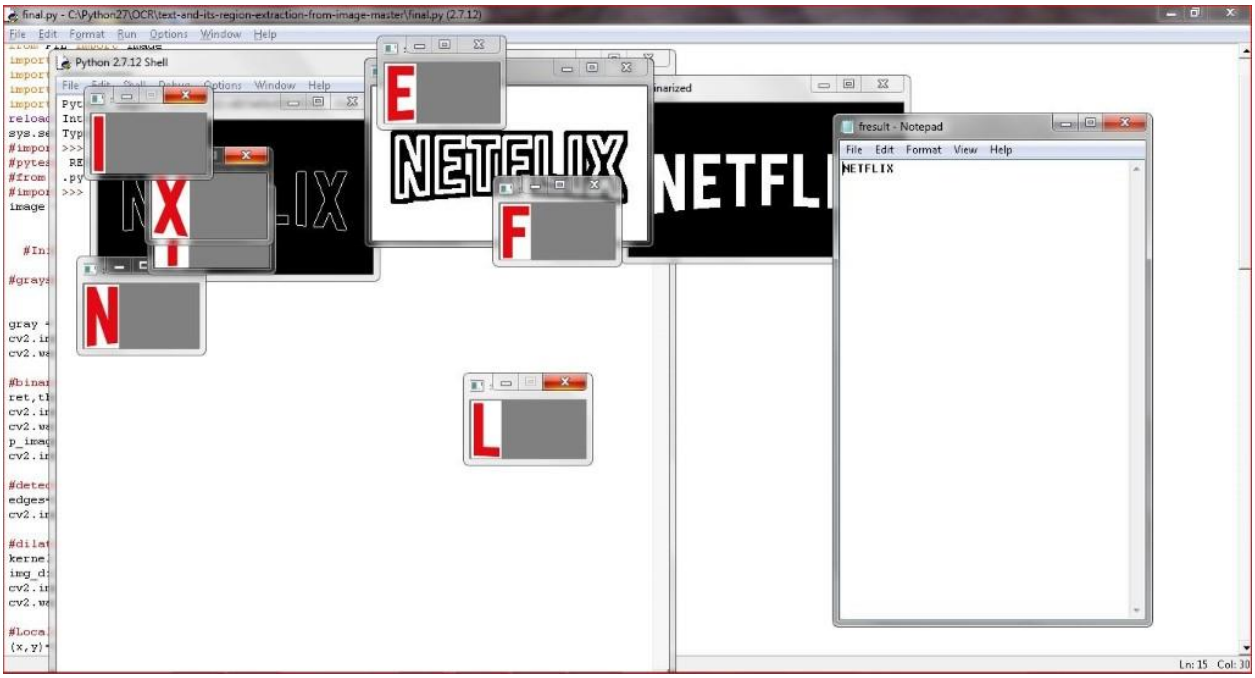

final.py - C:\Python27\OCR\text-and-its-region-extraction-from-image-master\final.py (2.7.12)
Python 2.7.12 Shell
fresult - Notepad
NETFLIX

Fig. 6.17(k)


143
AXIS BANK
10% instant discount using
Axis Bank Credit & Debit cards

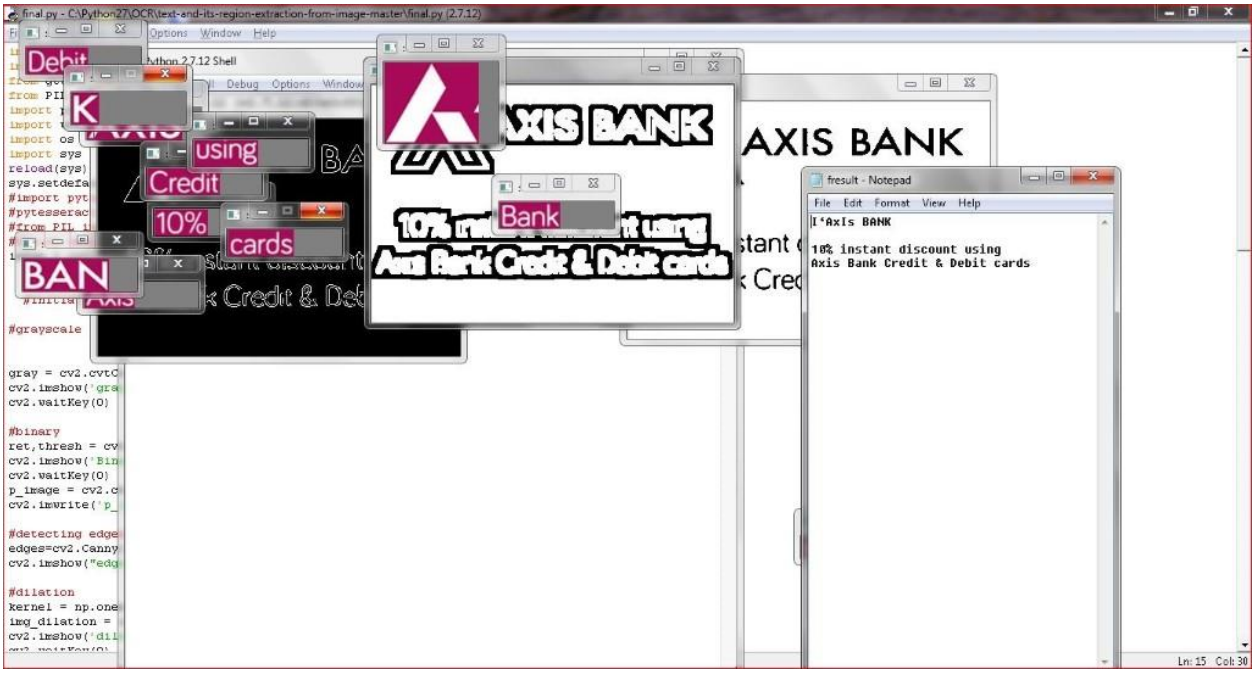

AXIS BANK
AXIS BANK
fresult - Notepad
File Edit Format View Help
I'AxIs BANK
10% instant discount using
Axis Bank Credit & Debit cards
Debit
using
Credit
10%
cards
Bank

Fig. 6.17(l)

ɜst cases : Failure

EMINƎM

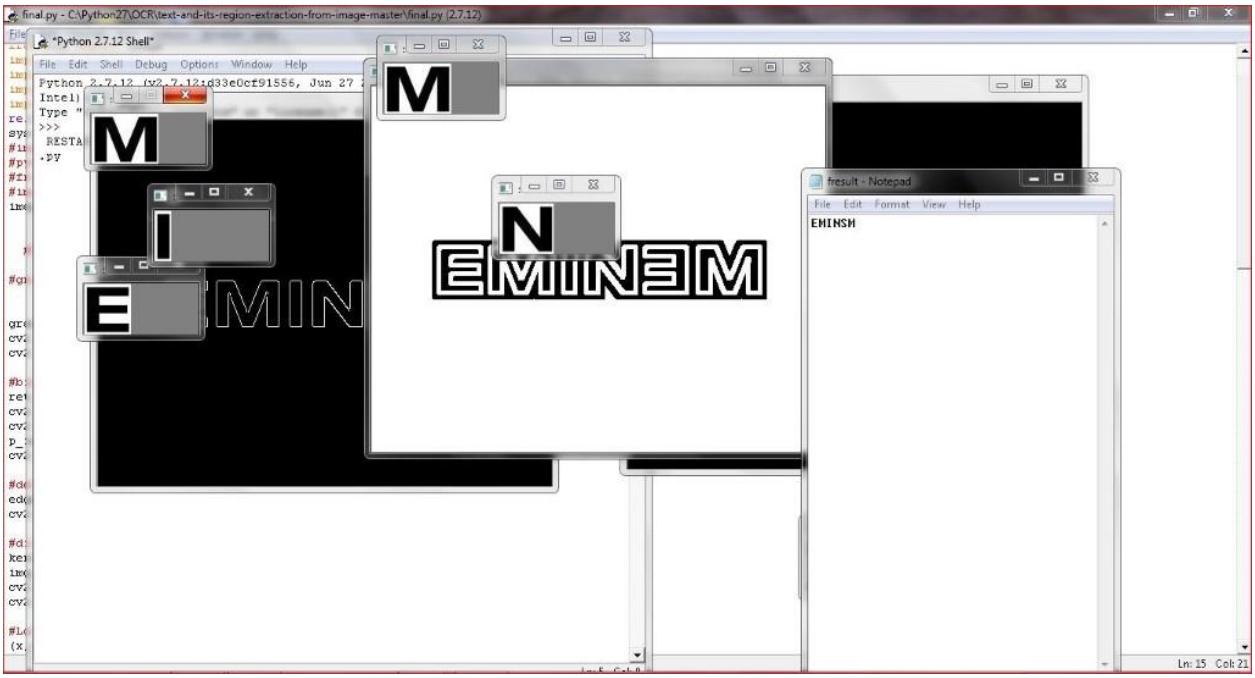


Fig. 6.17(m)


final.py - C:\Python27\OCR\text-and-its-region-extraction-from-image-master\final.py (2.7.12)
File Edit Format Run Options Window Help
Python 2.7.12 Shell
File Edit Shell Debug Options Window
edges
threshold1
segment no:0
Precision
Extraction solutions
fresult - Notepad
File Edit Format View Help

Fig. 6.17(n)


17
It's
FRIDAY

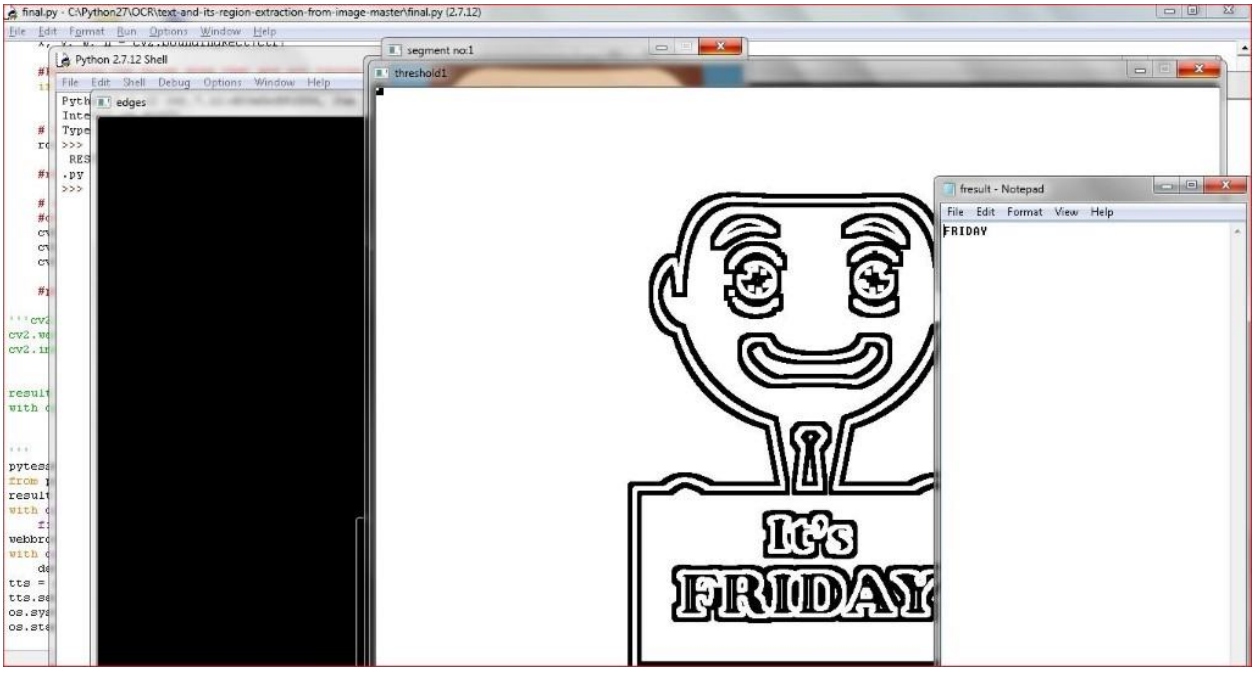

final.py - C:\Python27\OCR\text-and-its-region-extraction-from-image-master\final.py (2.7.12)
Python 2.7.12 Shell
segment no:1
threshold1
edges
fresult - Notepad
FRIDAY

Fig. 6.17(o)


manifest:file-entry
type
file
packages
manifest:algorithm
within
document
manifest:start-key-generation
entry
contained
signature
contains
urn:oasis:names:tc:opendocument:xmlns:manifest:1.0
name
compressed
see
media
part
attribute
defined
OASIS
schema
root
relative
xmlenc-core
algorithms
start
may
child
algorithm
elements
specified
metadata
value
Committee
digital
manifest:full-path
Blowfish
manifest:encryption-data
encryption
encrypted
path
support
OpenDocument
specifies
usable
IRI
manifest:version
used
conforming
values
Schema
Zip
W3C
directory
specification
references
format
documents
class
manifest:key-derivation
Version
manifest:algorithm-name
package
attributes
element
Package
files
manifest
following
data
key
XML
contain

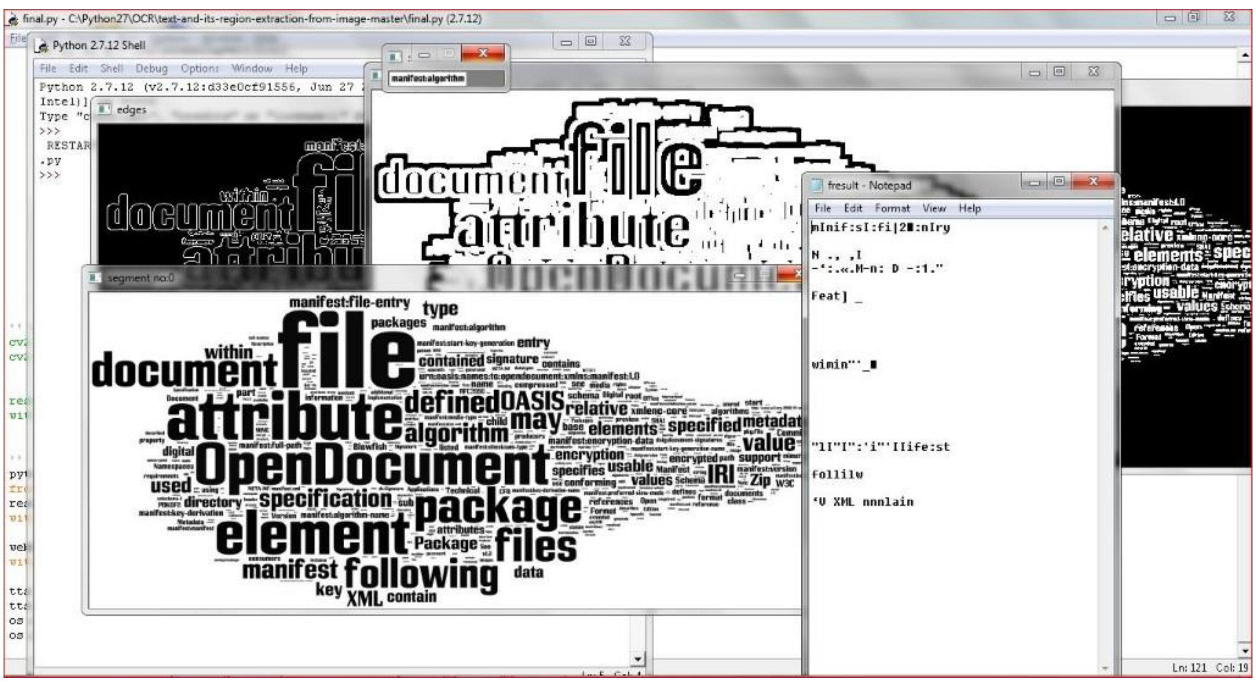

final.py - C:\Python27\OCR\text-and-its-region-extraction-from-image-master\final.py (2.7.12)
Python 2.7.12 Shell
File Edit Shell Debug Options Window Help
edges
segment no:0
fresult - Notepad
File Edit Format View Help
Ln: 121 Col: 19

Fig. 6.17(p)


Office
PowerPoint
Opening: Nisha concrete -97.ppt (0%)

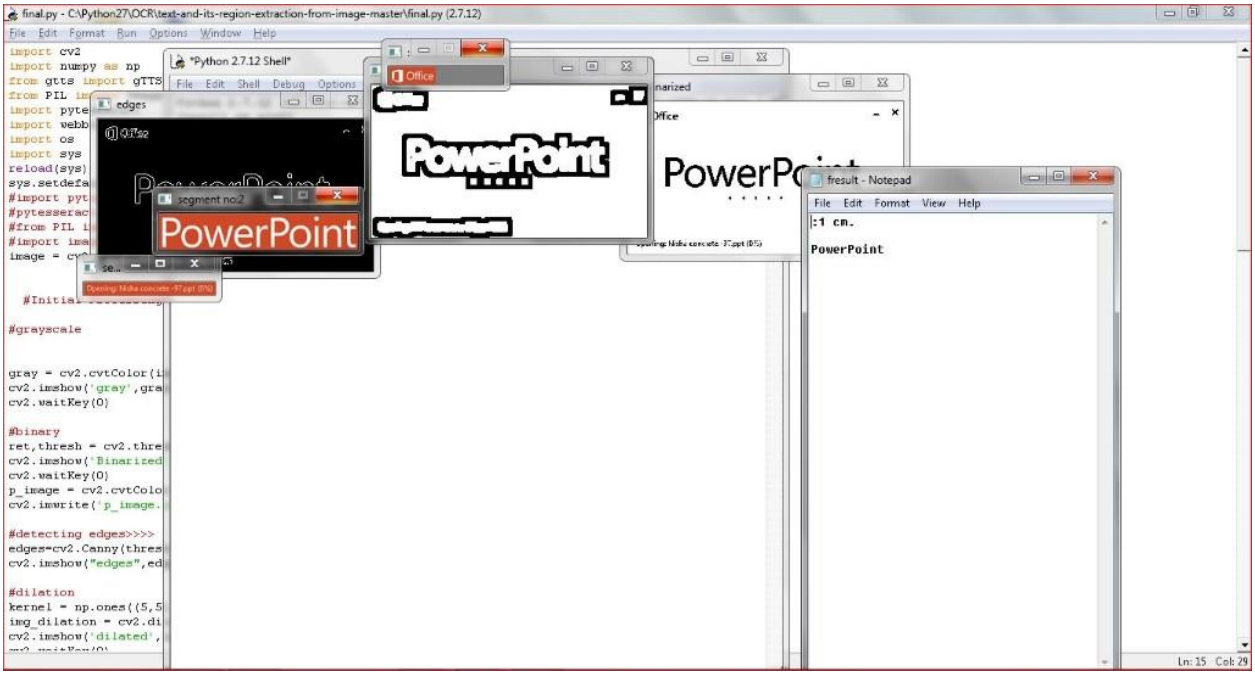

final.py - C:\Python27\OCR\text-and-its-region-extraction-from-image-master\final.py (2.7.12)
File Edit Format Run Options Window Help
Python 2.7.12 Shell
File Edit Shell Debug Options
edges
segment no:2
PowerPoint
Office
PowerPoint
fresult - Notepad
File Edit Format View Help
:1 cm.
PowerPoint
Ln: 15 Col: 29

Fig. 6.17(q)

Test Case_id	Expected Output	Actual Output	Online OCR	Status
1	Google	Google	e	Success
7	EMINƎM	EMINSM	EMIN3M	Partially Success
12	Precision extract solutions	-----	-----	Failure
13	VNR VJIET	VNR VJIET	VNR VJ 1 ET	Success
17	It’s Friday	Friday	ti* It's FRIDAY	Partially Success
26	Uncool_	Uncool_	UNCooL	Success
34	What Who Why When Where How	What Who Why When Where	-----	Partially Success
64	MOUNTAINS	MOUNTAINS	MOUNTAINS	Success
66	78	78	78	Success
115	NETFLIX	NETFLIX	-----	Success

Table 6.1 Classification Status of the images

We have in total 146 test cases. Out of 146, 62 test cases have successfully classified with 100% accuracy, 32 items have partially classified with over 95% accuracy, 4 test cases are classified but incorrectly classified and rest 48 test cases are failure with no text recognition. On comparing with the existing OCR, we are able to have slight improvement and accuracy on all those 146 test cases.

7.TESTING

7.1 INTRODUCTION

Testing is an integral part of developing any software project, to check how well the developed software performs and what it`s limitations are.

7.2 TYPES OF TESTING

7.2.1 SYSTEM TESTING

System Testing (ST) is a black box testing technique performed to evaluate the complete system's compliance against specified requirements. In System testing, the functionalities of the system are tested from an end-to-end perspective. System Testing is usually carried out by a team that is independent of the development team in order to measure the quality of the system unbiased. It includes both functional and Nonfunctional testing.

7.2.2 TESTING METHODOLOGIES AND STRATEGIES

A testing methodology is a tool or method used to test an application. As you listed, some methodologies include monkey testing, automated UI testing, regression testing, and so forth. Some might argue that testing techniques such as pairwise-combinatorial interdependence modeling or model-based testing are also methodologies. A testing strategy, on the other hand, is a holistic view to how you will test a product -- it's the approach you will take, the tools (and methodologies) you will use to deliver the highest possible quality at the end of a project. In software quality, the test strategy consists of a myriad of methodologies, activities, and staffing solutions. The strategy overall sets the acceptable bar and calls out how the test team will achieve that bar. It is the sum of all the inputs, in an organized plan. Testing methodologies are the different approaches you will take to testing.

7.2.3 SOFTWARE TESTING – LEVELS

There are different levels during the process of testing. In this chapter, a brief description is provided about these levels. Levels of testing include different methodologies that can be used while conducting software testing. The main levels of software testing are:

Functional Testing

Non-functional Testing

Functional Testing

This is a type of black-box testing that is based on the specifications of the software that is to be tested. The application is tested by providing input and then the results are examined that need to conform to the functionality it was intended for. Functional testing of software is conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements.

There are five steps that are involved while testing an application for functionality.

Testing activities: Different levels of testing are used in this testing process; each level of testing aims to test different aspects of the system. The basic levels are:

Unit testing

Integration testing

System testing

Acceptance testing

Unit Testing

Unit testing focuses on the building blocks of the software system, that is, objects and sub system. There are three motivations behind focusing on components.
First, unit testing reduces the complexity of the overall tests activities, allowing us to focus on smaller units of the system. Second, unit testing makes it easier to pinpoint and correct faults given that few components are involved in this test. Third, Unit testing allows parallelism in the testing activities, that is, each component can be tested independently of one another. Hence the goal is to test the internal logic of the module.

Integration Testing:

In the integration testing, many test modules are combined into subsystems, which are then tested. The goal here is to see if the modules can be integrated properly, the emphasis being on testing module interaction. After structural testing and functional testing, we get error free modules. These modules are to be integrated to get the required results of the system. After checking a module, another module is tested and is integrated with the previous module. After the integration, the test cases are generated and the results are tested.

System Testing:

In system testing the entire software is tested. The reference document for this process is the requirement document and the goal is to see whether the software meets its requirements. The system was tested for various test cases with various inputs.

Acceptance Testing:

Acceptance testing is sometimes performed with realistic data of the client to demonstrate that the software is working satisfactory. Testing here focus on the external behavior of the system, the internal logic of the program is not emphasized. In acceptance testing the system is tested for various inputs.

Non-Functional Testing

This section is based upon testing an application from its non-functional attributes. Non-functional testing involves testing software from the requirements which are non-functional in nature but important such as performance, security, user interface, etc Some of the important and commonly used non-functional testing types are discussed below.

Performance testing:

It is mostly used to identify any bottlenecks or performance issues rather than finding bugs in software. There are different causes that contribute in lowering the performance of software:

Network delay

Client-side processing

Database transaction processing

Load balancing between servers

Data rendering

Performance testing is considered as one of the important and mandatory testing type in terms of the following aspects:

Speed (i.e. Response Time, data rendering and accessing)

Capacity

Stability

Scalability

Load testing:

It is a process of testing the behavior of software by applying maximum load in terms of software accessing and manipulating large input data. It can be done at both normal and peak load conditions. This type of testing identifies the maximum capacity of software and its behavior at peak time.

Usability testing:

Usability testing is a black-box technique and is used to identify any error(s) and improvements in the software by observing the users through their usage and operation. According to Nielsen, usability can be defined in terms of five factors, i.e. efficiency of use, learn-ability, memory-ability, errors/safety, and satisfaction. According to him, the usability of a product will be good and the system is usable if it possesses the above factors.

7.3 TEST PLAN

A test plan outlines the strategy that will be used to test an application, the resources that will be used, the test environment in which testing will be performed and the limitations of the testing and the schedule of testing activities. Typically, the Quality Assurance Team Lead will be responsible for writing a Test Plan. A test plan includes the following:

Introduction to the Test Plan document

Assumptions while testing the application

List of test cases included in testing the application

List of features to be tested

What sort of approach to use while testing the software

List of deliverables that need to be tested

The resources allocated for testing the application

Any risks involved during the testing process

A schedule of tasks and milestones to be achieved

7.4 TEST SCENARIO

It is a one-line statement that notifies what area in the application will be tested. Test scenarios are used to ensure that all process flows are tested from end to end. A particular area of an application can have as little as one test scenario to a few hundred scenarios depending on the magnitude and complexity of the application. The terms 'test scenario' and 'test cases' are used interchangeably; however, a test scenario has several steps, whereas a test case has a single step. Viewed from this perspective, test scenarios are test cases, but they include several test cases and the sequence that they should be executed. Apart from this, each test is dependent on the output from the previous test.

7.5 TEST CASE

Test cases involve a set of steps, conditions, and inputs that can be used while performing testing tasks. The main intent of this activity is to ensure whether software passes or fails in terms of its functionality and other aspects. There are many types of test cases such as functional, negative, error, logical test cases, physical test cases, UI test cases, etc. Furthermore, test cases are written to keep track of the testing coverage of software. Generally, there are no formal templates that can

be used during test case writing. However, the following components are always available and included in every test case:

- Test case ID
- Product module
- Product version
- Revision history
- Purpose
- Assumptions
- Pre-conditions
- Steps
- Expected outcome
- Actual outcome
- Post-conditions

Many test cases can be derived from a single test scenario. In addition, sometimes multiple test cases are written for software which are collectively known as test suites.

7.5.1 POSITIVE TEST CASE

Positive Testing is testing process where the system validated against the valid input data. In this testing tester always check for only valid set of values and check if a application behaves as expected with its expected inputs. The main intention of this testing is to check whether software application not showing error when not supposed to & showing error when supposed to. Such testing is to be carried out keeping positive point of view & only execute the positive scenario. Positive Testing always tries to prove that a given product and project always meets the requirements and specifications. Under Positive testing is test the normal day to day life scenarios and checks the expected behavior of application.

7.5.2 NEGATIVE TEST CASE

Negative Testing is testing process where the system validated against the invalid input data. A negative test checks if an application behaves as expected with its negative inputs. The main intention of this testing is to check whether software application not showing error when supposed to & showing error when not supposed to. Such testing is to be carried out keeping negative point of view & only execute the test cases for only invalid set of input data. Negative testing is a testing process to identify the inputs where system is not designed or un-handled inputs by providing different invalid. The main reason behind Negative testing is to check the stability of the software application against the influences of different variety of incorrect validation data set.

8. CONCLUSION & FUTURE SCOPE

8.1 CONCLUSION

This project is just a short version of the project we are presenting. This project can be taken further, and then can be used for different purposes. Recently most of the text extraction algorithm uses a direct technique where the whole image is feed to the machine and tries to extract the text from that which is not efficient because it need separate image segmentation for most of the different images.

But, what we are trying to do is first of all extract each text segment using connected component concept i.e., Contours and then feed each segment to the separate algorithm, which will treat each segment of text as a separate image. Doing this we will get efficient output, because applying initial processing on a small segment of the image which is already processed is more accurate than applying it on a whole image.

8.2 FUTURE SCOPE

In future we can try to improve the accuracy rate of text detection and text recognition by improving the implementation. Also enhance this application, which will support for multi-language text extraction and its audio conversion. Also, we can delve further into the implementation of Neural Networks and come up with methods to increase our accuracy levels. Last, but not the least, we can develop a GUI which shall enable greater User usability and popularity by embedding it with Raspberry-Pi which brings IOT into the picture.

9. REFERENCES

[1] Fanfeng Zeng, Guofeng Zhang, Jin Jiang, "Text Image with Complex Background Filtering Method Based on Harris Comer-point Detection", Journal of Software, vol. 8, no. 8, pp. 1827-1834, 2013

[2] T. Pavlidis, J. Zhou, "Page segmentation and classification", Computer Vision Graphics and Image Processing, vol. 56, no. 6, pp. 484-496, 1992.

[3] D. Wang, S. N. Srihari, "Classification of newspaper image blocks using texture analysis", Computer Vision Graphics and Image Processing, vol. 47, pp. 327-352, 1989

[4] P. Tofani and R. Kasturi, "Segmentation of text from color map images," in Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on, vol. 1, aug 1998, pp. 945 -947 vol.1.

[5] Z. Lu, "Detection of text region from digital engineering drawings," IEEE Trans. Pattern Anal. Machine Intell., vol. 20, pp. 431–439, Apr.1998

[6] L. O. Gorman, R. Kasturi, "Document Image Analysis" in , Los Alamitos, California, USA:IEEE Computer Society Press, 1995.

[7] B. Epshtein, E. Ofek, and Y. Wexler, "Detecting text in natural scenes with stroke width transform," in Proc. IEEE Conf. Comput.Vis. Pattern Recog., 2010, pp. 2963–2970.

[8] L. Neumann and J. Matas, "On combining multiple segmentations in scene text recognition," in Proc. 12th Int. Conf. Document Anal.Recog., Aug. 2013, pp. 523–527.

[9] L. Neumann and J. Matas, "A method for text localization and rec-ognition in real-world images," in Proc. 10th Asian Conf. Comput. Vis., Nov. 2010, pp. 2067–2078.

[10] X.-C. Yin, X. Yin, K. Huang, and H.-W. Hao , "Robust text detection in natural scene images," IEEE Trans. Pattern Anal. Mach. Intell., vol. 36, no. 5, pp. 970–983, May 2014.

Links:

http://opencvpython.blogspot.in/2012/06/hi-this-article-is-tutorial-which-try.html

https://docs.opencv.org/3.3.1/dd/d49/tutorial_py_contour_features.html

http://www.academia.edu/33514973/An_Approach_To_Extract_Text_Regions_From_Scene_Image

http://hanzratech.in/2015/01/21/adaptive-thresholding.html