TITLE: OPTICAL CHARACTER RECOGNITION (OCR) USING TESSRACT IN RASPBERRY PI

INTRODUCTION

In the running world, there is growing demand for the software systems to recognize characters in computer system when information is scanned through paper documents as we know that we have number of newspapers and books which are in printed format related to different subjects.
These days there is a huge demand in "storing the information available in these paper documents in to a computer storage disk and then later reusing this information by searching process".

Thus our need is to develop character recognition software system to perform Document Image Analysis which transforms documents in paper format to electronic format. For this process there are various techniques in the world.
Among all those techniques we have chosen Optical Character Recognition as main fundamental technique to recognize characters. The conversion of paper documents in to electronic format is an on-going task in many of the organizations particularly in Research and Development (R&D) area, in large business enterprises, in government institutions, so on.

Optical character Recognition has become one of the most successful applications of technology in the field of pattern recognition and artificial intelligence. Many commercial systems for performing OCR exist for a variety of applications, although the machines are still not able to Complete with human reading capabilities. Optical character recognition is needed when the information should be readable both too human and to a machine. Both hand written and printed character may be recognized. Optical character recognition is performed off line after the writing or printing has been completed ,as opposed to on line recognition where the computer recognizes the character as they are drawn.OCR method for recognizing documents either printed or handwritten without any knowledge of the font.

MOTIVATION

Optical character recognition (OCR) is motivated by the need to efficiently and accurately digitize printed or handwritten text. OCR technology enables the conversion of scanned documents or images into editable, searchable, and shareable digital files.
OCR is particularly useful in situations where large volumes of text need to be processed quickly and accurately, such as in data entry, archiving, and document management. By automating the process of digitizing text, OCR can save time and reduce errors associated with manual data entry.

OCR also has applications in fields such as education, where it can be used to digitize books, journals, and other printed materials, making them more accessible to students and researchers.
Overall, OCR technology has the potential to greatly enhance the efficiency and productivity of many industries by enabling the seamless integration of paper-based information into digital workflows.

OBJECTIVES AND SCOPE

Tesseract is an open-source OCR engine developed by Google. The primary objective of OCR using Tesseract is to recognize and convert images of printed or handwritten text into machine-readable and editable text format. Here are some specific objectives of OCR using Tesseract:
1. Simplify texts with various backgrounds for the visually challenged.
2. Translating various fonts/handwriting scripts into a readable font. The primary objective of OCR using Tesseract is to achieve high accuracy in text recognition. Tesseract has been trained on a vast dataset of fonts, language scripts, and text styles, which makes it capable of recognizing text with high accuracy.
3. Convert a physical paper document, or an image into an accessible electronic version with text. Tesseract is highly flexible and customizable, allowing users to adjust the OCR engine's parameters and settings to suit their needs. The objective of OCR using Tesseract is to provide a versatile and customizable OCR solution.
4. Clean segmentation of the foreground text from background.
5.Integration with Other Applications: Tesseract can be integrated with other applications and programming languages, such as Python, Java, and C++, to perform OCR tasks automatically. The objective of OCR using Tesseract is to provide an OCR solution that can be integrated into various applications and workflows.

To sum up, developed models is able to detect & extract text from images. However, accuracy is maximum for tesseract engine moreover output is saved in text file. Tesseract performs well when document images follow the next guidelines:
Clean segmentation of the foreground text from background, horizontally aligned and scaled appropriately high-quality image without blurriness and noise. In order to improve its accuracy, our future approach will be based on gathering more enhanced techniques from various sources and improving the proposed image processing algorithm.

DESCRIPTION OF THE PROJECT WORK

5.1 HARDWARE REQUIREMENTS

The following table shows the hardware used in this project:

Table 5.1

Component	Quantity
Raspberry Pi 4 module	1

5.2 SOFTWARE REQUIREMENTS

The following table shows the software used for this project:

Table 5.2

Project	Platform
OCR	Python, with Pillow and Open CV packages

5.3 BLOCK DIAGRAM

The following figure shows the flow of process from input to output:

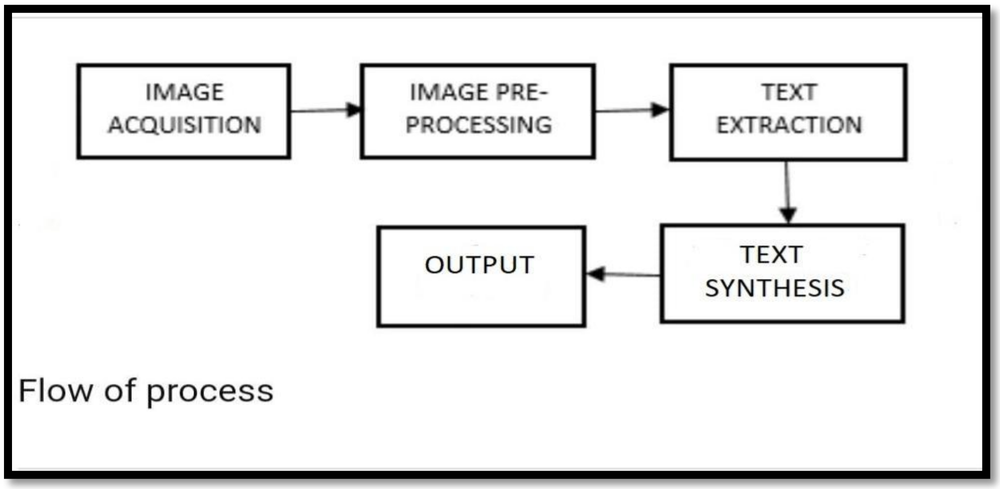


Figure 5.1 Block diagram

5.3 WORKING OF THE PROJECT

In this project, we first read an input image from Python Image Library (PIL). The text on the image is recognized using pytessarct and displayed on the shell of Raspberry Pi 4.

Python OCR is a technology that recognizes and pulls out text in images like scanned documents and photos using Python. It can be completed using the open-source OCR engine Tesseract. We can do this in Python using a few lines of code. One of the most common OCR tools that are used is the Tesseract.

Pytesseract or Python-tesseract is an OCR tool for python that also serves as a wrapper for the Tesseract-OCR Engine. It can read and recognize text in images and is commonly used in python ocr image to text use cases.

1. The Pillow package is used to open the desired image and save it under a variable name.
2. Then we use the image_to_sting method from the pytesseract package to detect any text from the image and save it as a string in the variable text.
3. Finally we print the value of text to check the results.

Since numbers cannot be recognized, people normally use OpenCV to remove noise from the program and then configure the Tesseract OCR engine based on the image to get better results. The Tesseract OCR engine will then analyze the image and return the recognized text.

The recognized text can then be used for various purposes, such as text-to-speech conversion, data analysis, or further processing.

RESULTS

In this chapter we discuss the results obtained for various test images, including handwritten scripts to test the response of the said code.

6.1 INPUTS

The input images in order as per the code are shown below:

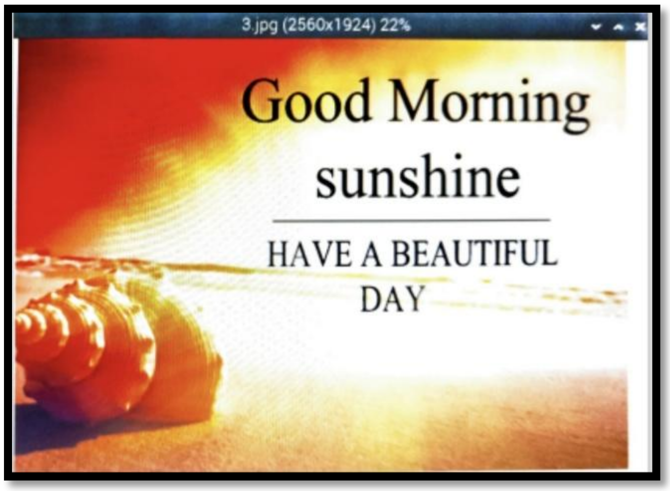
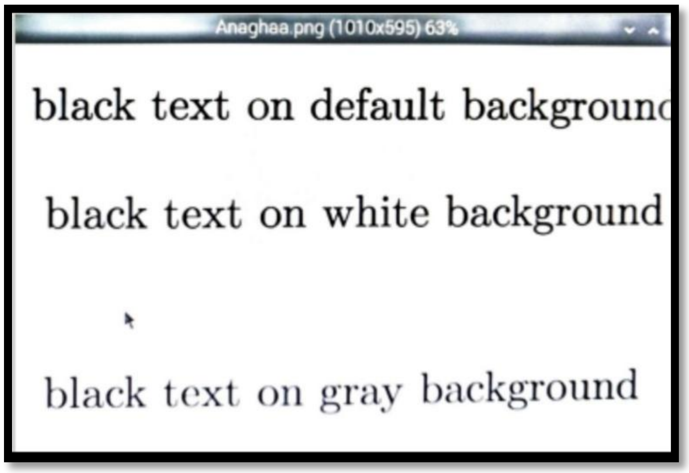


Figure 6.1 jpg

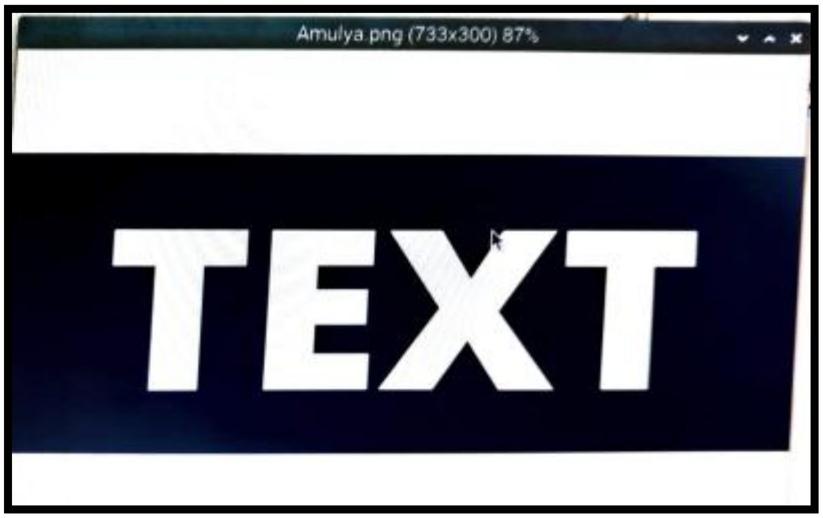


black text on default backgroun

black text on white background

black text on gray background

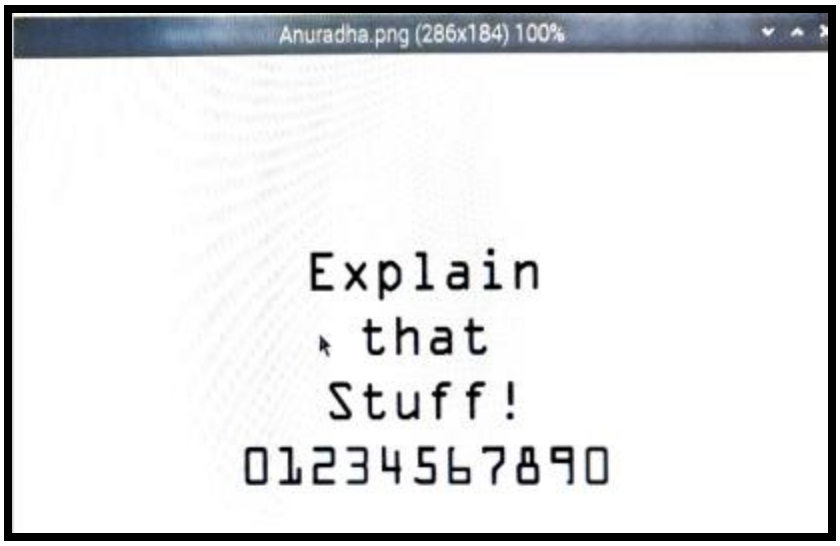
Figure 6.2 Anaghaa.png



Amulya.png (733x300) 87%

TEXT

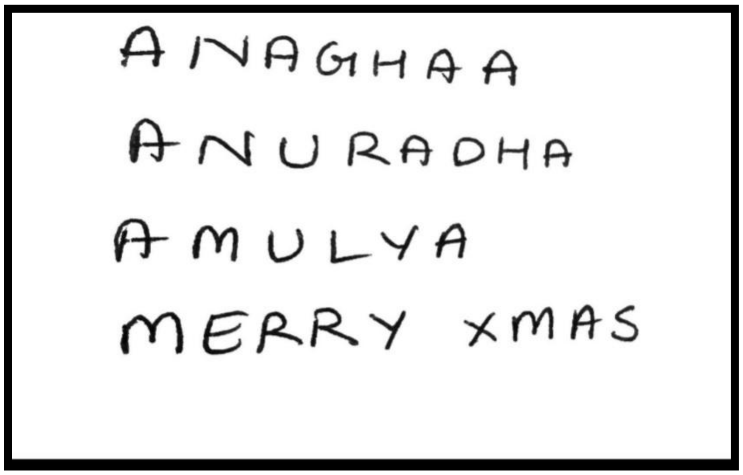
Figure 6.3 Amulya.png



Anuradha.png (286x184) 100%

Explain
that
Stuff!
01234567890

Figure 6.4 Anuradha.png

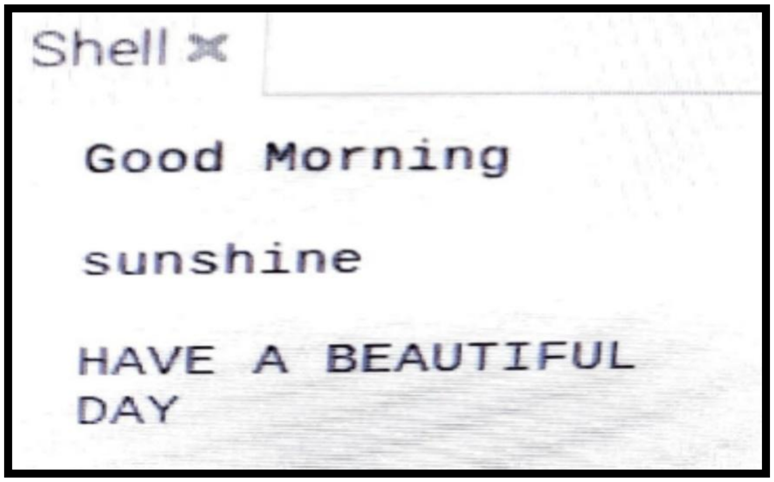


ANAGHAA
ANURADHA
AMULYA
MERRY XMAS

Figure 6.5 Input 5

6.2 OUTPUTS

We obtained the following outputs for the aforementioned images input:



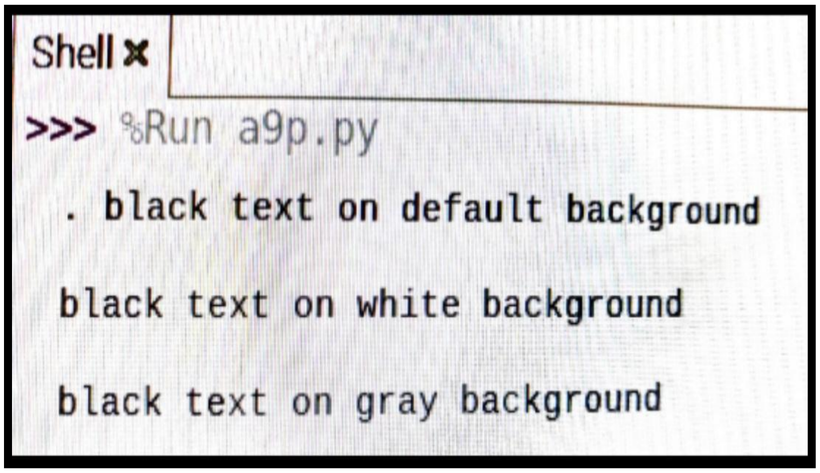
```
Shell

Good Morning

sunshine

HAVE A BEAUTIFUL
DAY
```

Figure 6.6 Output for 3.jpg

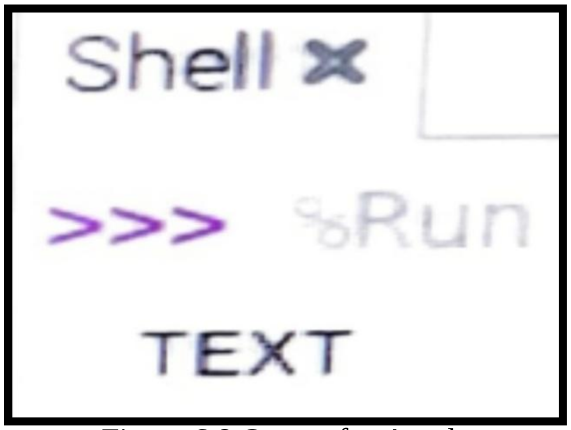


```
Shell
>>> %Run a9p.py
 . black text on default background

black text on white background

black text on gray background
```

Figure 6.7 Output for Anaghaa.png

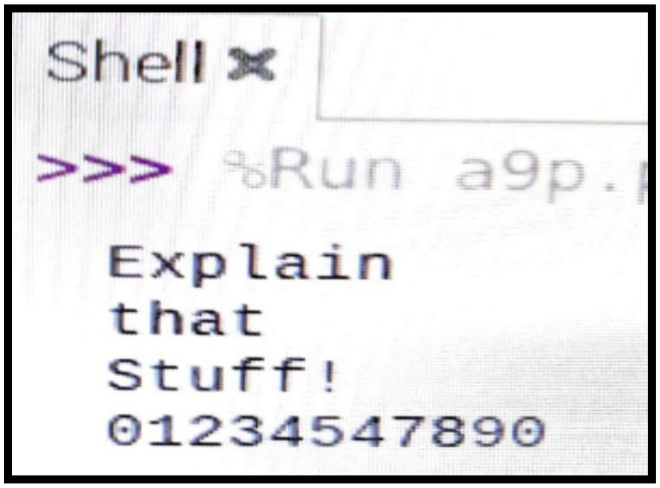


```
Shell

>>> %Run

    TEXT
```

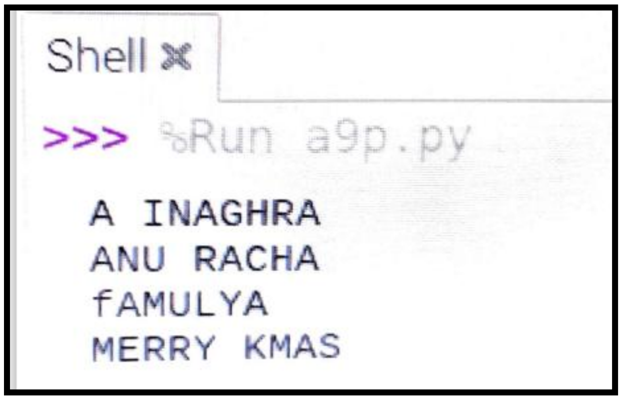
Figure 6.8 Output for Amulya.png



```
Shell
>>> %Run a9p.

   Explain
   that
   Stuff!
   01234547890
```

Figure 6.9 Output for Anuradha.png



```
Shell
>>> %Run a9p.py

   A INAGHRA
   ANU RACHA
   fAMULYA
   MERRY KMAS
```

Figure 6.10 Output for input 5

We observed that the output for handwritten text as shown in Fig 6.5, that the code cannot fully interpret the text as our handwriting might have minute errors. This is clearly shown in Fig 6.10

CONCLUSION

Just as deep learning has impacted nearly every facet of computer vision, the same is true for character recognition and handwriting recognition.

Tesseract performs well when document images follow the next guidelines:

1. Clean segmentation of the foreground text from background
2. Horizontally aligned and scaled appropriately
3. High-quality image without blurriness and noise and shows promising scope for development of the same.