TEXT DETECTION AND RECOGNITION USING MACHINE LEARNING

A Project Report Submitted By

MANOJ S BHAT	**4NM20EC409**
P VIGNESH PRABHU	**4NM20EC411**
SAMPATH KUMAR	**4NM20EC413**
SHIVA KUMAR	**4NM20EC415**

Under the Guidance of

Ms. SHUBHA B

Assistant Professor

in partial fulfillment of the requirements for the award of the Degree of

Bachelor of Engineering

in

Electronics and Communication Engineering

from

Visvesvaraya Technological University, Belagavi

NITTE EDUCATION TRUST | NMAM INSTITUTE OF TECHNOLOGY

Nitte-574110, Karnataka, India

May-2023

NITTE EDUCATION TRUST | N.M.A.M. INSTITUTE OF TECHNOLOGY
(An Autonomous Institution affiliated to Visvesvaraya Technological University, Belagavi)
Nitte – 574 110, Karnataka, India
ISO 9001:2015 Certified, Accredited with 'A' Grade by NAAC
☎:08258 - 281039 – 281263, Fax: 08258 – 281265

Department of Electronics and Communication Engineering

Certificate

Certified that the project work entitled

*"**Text Detection And Recognition Using Machine Learning**"*

is a bonafide work carried out by

MANOJ S BHAT (4NM20EC409), P VIGNESH PRABHU (4NM20EC411), SAMPATH KUMAR (4NM20EC413) & SHIVA KUMAR (4NM20EC415)

*in partial fulfillment of the requirements for the award of **Bachelor of Engineering Degree in Electronics and Communication Engineering***

*prescribed by **Visvesvaraya Technological University, Belagavi***

during the year 2022-2023.

It is certified that all corrections/suggestions indicated for Internal Assessment have been incorporated in the report deposited in the departmental library.

The project report has been approved as it satisfies the academic requirements in respect of the project work prescribed for the Bachelor of Engineering Degree.

Signature of the Guide **Signature of the HOD**

Signature of the Principal

Semester End Viva Voce Examination

	Name of the Examiners	Signature With Date
1.	_____	_____
2.	_____	_____

Abstract

The proposed system for text detection using machine learning offers several benefits over traditional methods. By utilizing color conversion techniques and edge detection algorithms, the system can accurately detect text even in challenging situations, such as low lighting or complex backgrounds. The system's use of Optical Character Recognition (OCR) and machine learning pattern recognition algorithms such as Tesseract and OpenCV enhances its accuracy and enables it to handle different types of text, fonts, and languages. OCR is a technology that recognizes text within images and converts it into machine-readable text. Moreover, the system's ability to convert text to speech output has various practical applications in fields such as assistive technology, where it can help individuals with visual impairments access text-based information. The system's real-time capabilities also make it useful in security and surveillance applications, where it can automatically detect and read license plates or text on signs and provide alerts to operators. This can be particularly useful in law enforcement and border security applications.

Overall, the proposed system for text detection using machine learning has significant practical applications and can enhance data processing and analysis in various fields. Further optimization and development of the system can improve its performance and make it an essential tool for industries that rely on text-based data processing and analysis, such as publishing, marketing, and finance. The proposed system has the potential to revolutionize text detection and recognition technology and can offer benefits in various industries and fields.

Acknowledgement

Our project would not have been successful without the encouragement, guidance, and support by various personalities. First and foremost, we would like to express our sincere gratitude towards our project guide **Ms. Shubha B**, Assistant Professor, Department of Electronics and Communication Engineering, N. M. A. M. Institute of Technology, Nitte, for his/her guidance, encouragement, and inspiration throughout the project work.

We extend our gratitude to **Dr. K V S S S S Sairam**, Professor and Head, Department of Electronics and Communication Engineering, N. M. A. M. Institute of Technology, Nitte, for his encouragement and for providing necessary facilities in the department.

We wish to acknowledge the support of **Dr. Niranjan N. Chiplunkar,** Principal, N. M. A. M. Institute of Technology, Nitte, for providing a motivational academic environment.

We would like to express our sincere thanks to all the teaching and non-teaching staff of the Department of Electronics and Communication Engineering, N. M. A. M. Institute of Technology, Nitte, for their patience and help during our project work.

Finally, we would like to thank all our friends and well-wishers who have helped us whenever needed and supported us.

Manoj S Bhat
P Vignesh Prabhu
Sampath Kumar
Shiva Kumar

Table of Contents

Abstract **i**

Acknowledgement **iii**

Table of Contents **v**

List of Figures **vii**

1 Introduction **1**
1.1 Aim . . . 1
1.2 Objective/s . . . 2
1.3 Problem Formulation . . . 2
1.4 Proposed Method/Technique . . . 2
1.5 Methodology . . . 3
1.6 Literature Survey . . . 3
1.7 Organization of the Report . . . 5

2 System Architecture **7**
2.1 Block diagram . . . 7
2.1.1 Raspberry Pi 4B . . . 8
2.1.2 Power Supply . . . 8
2.1.3 Camera . . . 8
2.1.4 Touch LCD Display . . . 8
2.1.5 Speaker . . . 9
2.2 Flow Chart . . . 9
2.2.1 Real-time input . . . 10
2.2.2 Pre-processing . . . 10
2.2.3 Feature extraction . . . 10
2.2.4 Training . . . 10
2.2.5 Text Detection . . . 10
2.2.6 Post-processing . . . 11
2.2.7 Text Recognition . . . 11
2.2.8 TTS conversion . . . 11

3 Software/Hardware description **13**
3.1 Software Description . . . 13
3.1.1 Integrated Development Environment . . . 13

3.1.2 Python 13
3.1.3 Open CV 14
3.1.4 Tesseract 14
3.1.5 Tkinter 15
3.1.6 PIL 15
3.1.7 Pyttsx 16
3.2 Hardware Description 17
3.2.1 Raspberry Pi 4 Model B 17
3.2.2 5 MP Camera Module 18
3.2.3 LCD Display 19
3.2.4 Speaker 20

4 Methodology **21**
4.1 Preprocessing techniques 21
4.1.1 Gaussian Blur 21
4.1.2 Adaptive Thresholding 22
4.1.3 Invert 23
4.1.4 Dilation 24
4.1.5 Grey scaling 25
4.2 Text Processing-NLTK 25

5 Algorithm **27**
5.1 Preprocessing 27
5.2 Text detection and Reorganization 27
5.3 Text-To-Speech 29

6 Results **31**

7 Conclusion and Future scope **35**
7.1 Conclusion 35
7.2 Future Scope 35

Bibliography **37**

List of Figures

2.1 Block Diagram Of The System 7
2.2 Flow Chart Of The System 9

3.1 Raspberry Pi 4 Model B 17
3.2 Camera Module 18
3.3 LCD Display 19
3.4 Speaker 20

5.1 Text Detection and Reorganization Flow Chart 28
5.2 Text-To-Speech Flow Chart 30

6.1 System model 31
6.2 Captured image 32
6.3 Before the detection 32
6.4 Detected output 33

Chapter 1
Introduction

The ability to extract and recognize text from images and videos has become increasingly important in today's society, with numerous practical applications in fields such as document analysis, security, and image processing. To address this need, this engineering project focuses on developing a system for text detection and recognition using machine learning algorithms. The proposed system utilizes color conversion techniques, edge recognition algorithms, and text area and geometrical properties localization to enhance the accuracy of text detection even with complex backgrounds. Optical Character Recognition (OCR) and machine learning pattern recognition algorithms such as Tesseract and OpenCV are used to extract and identify text from images and videos. The system's ability to convert text to speech output also has potential applications in assistive technology, allowing individuals with visual impairments to access text-based information.

Overall, the development of a system for text detection and recognition using machine learning algorithms has significant practical implications in numerous industries and fields. The proposed system's accuracy and efficiency make it a valuable tool for data processing and analysis. Further optimization and development of the system can enhance its performance and make it applicable in various industries and fields, making it a promising area of research and development for the future.

1.1 Aim

This project aims to develop an accurate and efficient system for text detection and recognition using machine learning techniques. The system will be designed to process images containing text, identify the location of the text within the image, and accurately recognize the text using machine learning algorithms. The ultimate goal is to create a system that can be used for various applications, including document digitization, text extraction from images, and real-time text recognition in videos. The system should be able to handle various languages and font types and achieve high accuracy and speed in text detection and recognition.

1.2 Objective/s

- To develop an algorithm for detecting text regions in images using machine learning techniques.
- To implement a method for recognizing text within those regions using OCR technology.
- To optimize the performance of text detection and recognition.
- To test the system on a variety of real-world images to evaluate its effectiveness and accuracy.
- To explore and compare different machine learning techniques for text detection and recognition and determine their relative strengths and weaknesses.
- To create a user-friendly interface for the system.

1.3 Problem Formulation

The problem formation involves developing a system that can accurately detect and recognize text from images. This requires addressing various challenges such as varying fonts, sizes, and orientations of the text, as well as dealing with noise and other artifacts in the images. Additionally, the system should be able to perform these tasks in real time and with high accuracy. The use of machine learning techniques such as Convolutional Neural Networks (CNNs) and OCR algorithms can aid in achieving these goals. The goal of the project is to develop a robust text detection and recognition system that can be applied in various real-world scenarios.

1.4 Proposed Method/Technique

The proposed method in the "text detection and recognition using machine learning" project is to use OpenCV for image processing and Pytesseract for OCR. The code provided captures images from a camera feed and saves them as PNG files. The captured images are then processed using OpenCV functions such as resizing, rotation, adaptive thresholding, and dilation to prepare them for OCR. The Pytesseract library is then used to perform OCR and extract text from the processed images. Finally, the extracted text can be output to the user via a Graphical User Interface (GUI) or converted to speech using the Pyttsx3 library.

1.5 Methodology

- Capturing the Image: A camera feed is shown on the screen using OpenCV, and the user captures an image by pressing the "Capture" button. The captured image is saved as a PNG file on the disk.

- Text Detection: The captured image is read using OpenCV, and pre-processing techniques such as Gaussian blur, adaptive thresholding, inversion, and dilation are applied to enhance the text. The pre-processed image is passed to Pytesseract for OCR to extract the text from the image.

- Text-to-Speech: The extracted text is converted to speech using the pyttsx3 library.

- Displaying the Output: The extracted text is displayed on the screen using the tkinter Text widget. The user can also play the text as sound by pressing the "Sound" button.

- Exiting the Application: The application can be closed by pressing the "Exit" button, which releases the camera and destroys the GUI window.

1.6 Literature Survey

Huang et al., [1]. proposed a method for detecting and recognizing text from scenes by using two algorithms Maximally Stable Extremal Regions (MSER) and Support Vector Machine (SVM). Unlike text identification at the end, they distribute recognition issues in the detection and recognition. In the detection phase, the system extracted the attached components using MSER and color cluster to extract possible per text. Next, for obtaining connected components and non-text areas are filtered using Visual Saliency. Lastly, word lines can be obtained by text line generation. In the identification phase, the system uses vertical projection to divide words and then recognize the characters with the SVM-based framework. Experimental results have been speculated with better performance than traditional text detection and identification methods.

S L Wasankar et al., [2], Proposed an algorithm that first detects the frame having the text from live video streaming. The character is at its best view which means no broken point in a single character and no merge between groups of characters. The image processor tallies the images and burns the main microcontroller accordingly. This technology can prove to be immensely important to various human activities in day-to-day life.

R. Ani et al., [3], Developed a smart glasses-based system called Smart Specs that helps visually impaired individuals read printed text using Text-to-Speech (TTS) technology. The system uses optical character recognition (OCR) to detect and recognize text from the captured images and then converts it into speech using TTS. The system also provides additional features such as speech feedback, object detection, and facial recognition to enhance the user experience.

Q. Ye et al., [4], Discusses various techniques and concepts used for text detection and recognition in images. The survey covers several methods, including connected component analysis, sliding window-based approaches, and deep learning-based methods. The authors also discuss the challenges associated with text detection and recognition in natural scene images and provide a comprehensive overview of the field.

C. Jeeva et al., [5], proposed an Intelligent Image Text Reader system that uses Easy OCR, NRCLex, and NLTK techniques. The system employs image pre-processing techniques, text extraction algorithms, and machine learning models to extract and recognize text from images. The proposed system was evaluated on multiple datasets and achieved high accuracy in text recognition. Overall, the paper demonstrates the effectiveness of combining different techniques and tools for text detection and recognition in images.

Surana et al., [6], Provides a comprehensive review of the research on text extraction and detection from images using machine learning techniques. The authors discuss the challenges involved in text extraction and detection and review various approaches, such as feature extraction, classification, and segmentation. They also discuss the recent advancements in deep learning techniques for text detection and extraction, such as convolutional neural networks (CNNs) and Recurrent Neural Networks (RNNs). Overall, the paper is a valuable resource for researchers working on text detection and recognition using machine learning

R. Fernandes et al., [7], Presented a machine learning-based approach to the recognition of handwritten Kannada scripts. The authors utilized a dataset and evaluate their model, which consisted of preprocessing techniques such as binarization and thinning, followed by feature extraction using a Histogram of Oriented Gradients (HOG) and Local Binary Patterns (LBP). The model was trained using a Support Vector Machine (SVM) classifier. The paper provides insight into the effectiveness of machine learning techniques in Kannada script recognition.

M.R. Islam et al., [8], proposed a method for text detection and recognition using enhanced Maximally Stable Extremal Regions (MSER) detection and a novel OCR technique. The authors use MSER for text detection and enhance this process by considering the size, color, and edge information of the regions. They then employ a novel OCR technique that utilizes a dictionary-based approach for character

recognition. The proposed method is evaluated on two benchmark datasets, and the results demonstrate its effectiveness compared to other state-of-the-art methods.

V. Padmapriya et al., [9], discussed a study on text recognition and obstacle detection techniques. The authors investigate the use of deep learning-based methods for text recognition and obstacle detection in images. Specifically, they use a Convolutional Neural Network (CNN) for text recognition and a combination of CNN and a Histogram of Oriented Gradients (HOG) for obstacle detection. The proposed method is evaluated on a dataset of real-world images, and the results demonstrate its effectiveness for both tasks.

1.7 Organization of the Report

- Chapter 1 gives a brief introduction to the project
- Chapter 2 explains the general system architecture of the project
- Chapter 3 outlines the software/hardware description of the project.
- Chapter 4 provides details of the methodology of the project.
- Chapter 5 presents the algorithms used in the project.
- Chapter 6 presents the results and analysis of the project.
- Chapter 7 gives the concluding remarks.

Chapter 2
System Architecture

2.1 Block diagram

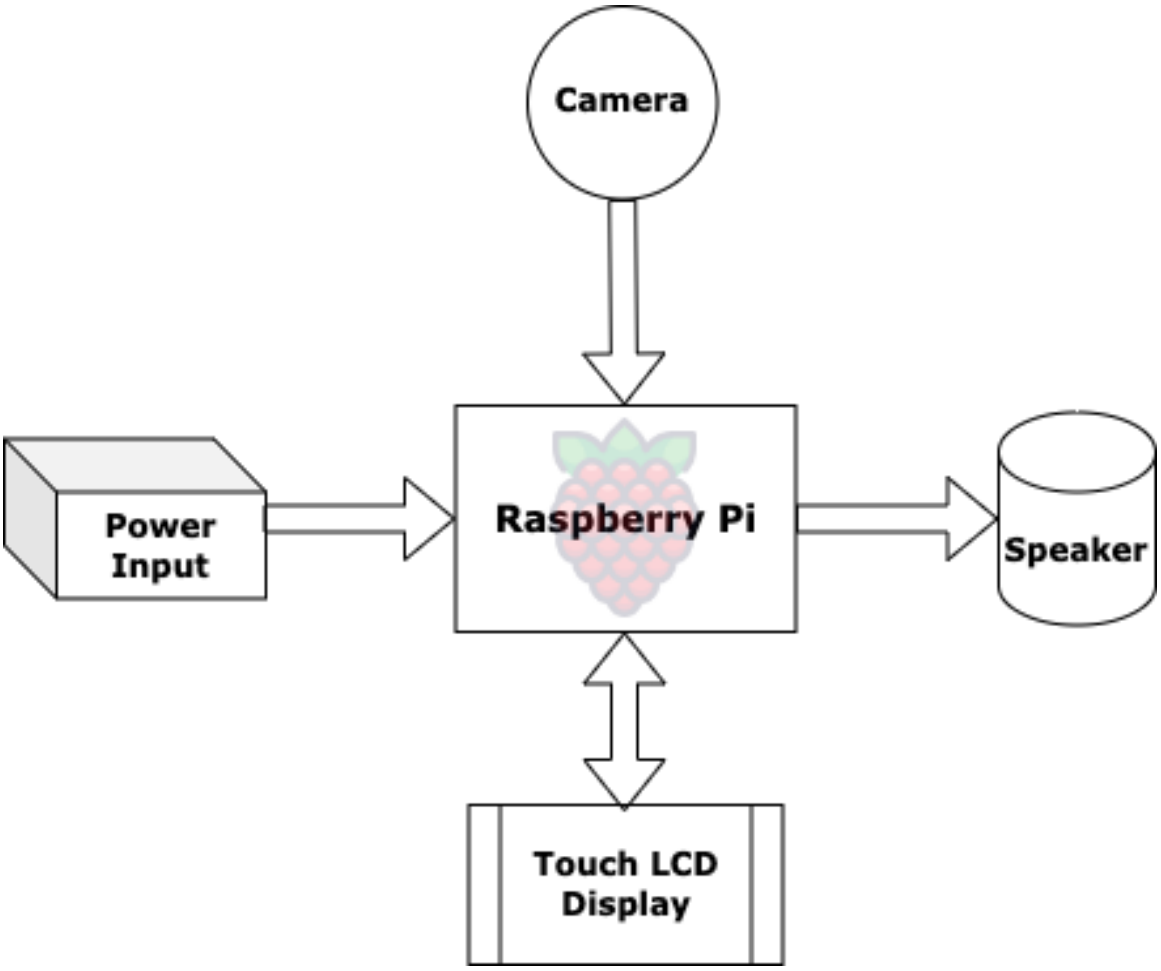


Figure 2.1: Block Diagram Of The System

Figure 2.1 shows Block Diagram with several interconnected components to achieve its objective. It includes a camera module, graphical user interface (GUI) module, text detection module, text recognition module, and text-to-speech module. The camera module captures the image of the text, which is displayed on the GUI module. The text detection module detects the text regions in the image captured by the camera. The text recognition module then recognizes the text in the detected regions using the Pytesseract library. Finally, the text-to-speech module converts the recognized text into speech using the pyttsx3 library. The system is designed to be interactive, allowing the user to capture an image, recognize the text

in the captured image, and have the text read aloud. The system is programmed using the Python programming language and uses several external libraries, including OpenCV, tkinter, and Pillow.

2.1.1 Raspberry Pi 4B

Raspberry Pi 4B is a single-board computer that has various components such as a processor, RAM, USB ports, Ethernet ports, and microSD card slots. It is designed for projects that require a small and affordable computer. Raspberry Pi 4B offers improved processing power, faster networking, and better multimedia support compared to its previous models. It is powered by a 5.1V DC power supply through a USB-C port and requires a minimum of 3A output current. Raspberry Pi 4B has various input/output pins that allow it to interact with other electronic components [10]. It is a versatile and customizable platform for various projects, including machine learning.

2.1.2 Power Supply

The Raspberry Pi 4B requires a 5.1V DC power supply with at least 3A output current via a USB-C port. Using a poor-quality USB-C cable may cause voltage drops and power fluctuations, leading to instability and data loss. Power input can be influenced by connected peripherals, so considering their power requirements are important. A stable power supply is crucial for the efficient operation of the Raspberry Pi, particularly in demanding tasks like machine learning.

2.1.3 Camera

The Raspberry Pi 4 Model B can be connected to a camera module. The 5MP camera module features an Omni Vision OV5647 sensor, a 5-element lens, and a built-in IR filter. It is capable of capturing 1080p video at 30 fps and still images at a resolution of 2592 x 1944 pixels. The camera module connects to the Raspberry Pi 4B via a ribbon cable and is powered by the Raspberry Pi itself. It is compatible with both the official Raspberry Pi camera software and third-party software like OpenCV [10]. A camera module is a useful tool for various projects, including computer vision and surveillance applications.

2.1.4 Touch LCD Display

The Raspberry Pi 4B supports a 3.5-inch touch LCD that can be connected directly to the GPIO pins. The display features a resolution of 480x320 pixels and

supports touch input via a resistive touch panel. It requires a power supply of 5v and consumes approximately 50mA of current [10]. The display can be used to display GUIs, images, videos, and other visual content in projects that require a compact and portable display.

2.1.5 Speaker

For audio output, the Raspberry Pi 4B has a 3.5mm audio jack that can be used to connect speakers or headphones. The audio output is stereo and supports a maximum resolution of 16 bits at 48 kHz. Alternatively, the Raspberry Pi also supports audio output over HDMI, which can be used to connect to a monitor or TV with built-in speakers. Additionally, it is possible to connect external USB or Bluetooth speakers or headphones to the Raspberry Pi for audio output.

2.2 Flow Chart

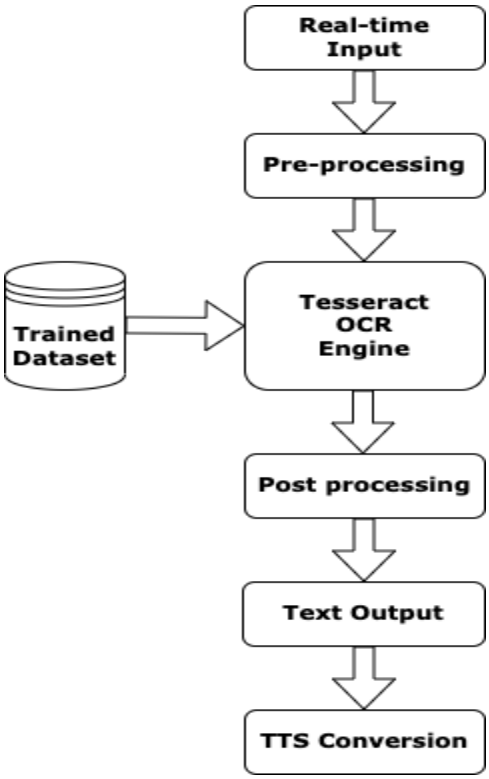


Figure 2.2: Flow Chart Of The System

Figure 2.2 refers to the flow chart of the model which consists of the following contents, the different steps of the flow chart are as follows:

2.2.1 Real-time input

Real-time input refers to the data input that is being received and processed as it is generated. In the context of text detection using machine learning, real-time input could be any text image or video that is captured by a camera or uploaded from a source. It is crucial to have real-time input to enable the system to detect text accurately and quickly.

2.2.2 Pre-processing

Pre-processing is the first step in the machine learning pipeline. It involves cleaning and transforming the raw data into a format that can be used for analysis. In the context of text detection using machine learning, pre-processing techniques such as image scaling, color conversion, and noise reduction are used to improve the quality of the input image. These techniques can help to reduce the noise and make the text easier to detect [9].

2.2.3 Feature extraction

Feature extraction is the process of extracting relevant features from the pre-processed input data that can be used for training a machine learning model. In the context of text detection using machine learning, feature extraction techniques such as edge detection, morphological operations, and connected component analysis can be used to extract relevant features such as text regions, shapes, and sizes [6].

2.2.4 Training

Training is the process of using labeled data to teach a machine-learning model how to recognize specific patterns or features in the input data. In the context of text detection using machine learning, a training dataset consisting of labeled text images is used to train a machine learning model to recognize text regions in new images. The trained model can then be used to detect text regions in real-time input images.

2.2.5 Text Detection

Text detection refers to the process of identifying the regions of an image that contain text. In the context of text detection using machine learning, the trained machine learning model is used to detect text regions in real-time input images. The text regions can be highlighted or marked in some way to help users identify the text more easily.

2.2.6 Post-processing

Post-processing is the process of refining the output of the text detection system to improve the accuracy and quality of the detected text regions. In the context of text detection using machine learning, post-processing techniques such as non-maximum suppression, bounding box refinement, and text region merging can be used to improve the accuracy of the detected text regions [1].

2.2.7 Text Recognition

Text recognition is the process of identifying the text content of the detected text regions. In the context of text detection using machine learning, text recognition techniques such as optical character recognition (OCR) can be used to extract the text content from the detected text regions.

2.2.8 TTS conversion

Text-to-speech (TTS) conversion is the process of converting text into spoken audio. In the context of text detection using machine learning, TTS conversion can be used to read out the text content of the detected text regions to users who may have difficulty reading or seeing the text on the screen [3]. TTS conversion is particularly useful for applications such as assistive technology for people with visual impairments.

Chapter 3
Software/Hardware description

The Software and the hardware requirements and their description are described below.

3.1 Software Description

3.1.1 Integrated Development Environment

On a Raspberry Pi, Thonny is a well-liked Python Integrated Development Environment (IDE). It has an easy-to-use user interface that makes it simple to create, debug, and execute code and is made for beginners. The Raspbian operating system, which is the official operating system of the Raspberry Pi, comes pre-installed with Thonny. This implies that clients can begin utilizing Thonny straight away, without the need to download and introduce extra programming.

In addition, Thonny has several features that make it ideal for novice developers, such as a code editor that shows syntax errors in real-time, an interactive debugger that lets users move through their code and see the results of each step and a straightforward interface for managing Python packages and libraries. Furthermore, Thonny has an inherent help for the GPIO (Universally useful Information/Result) pins on the Raspberry Pi, which permits clients to control outside equipment like LEDs, engines, and sensors. By and large, Thonny is an easy-to-use IDE that is appropriate for fledglings on a Raspberry Pi, with highlights that make it simple to make, test, and troubleshoot Python code.

3.1.2 Python

Python is a high-level, interpreted programming language that was first released in 1991. It emphasizes code readability, simplicity, and ease of use, making it popular among developers of all skill levels. Python supports multiple programming paradigms, including procedural, object-oriented, and functional programming. Its extensive standard library provides developers with a wide range of built-in functions and modules for their programs. Python is used in various applications, including web development, scientific computing, data analysis, artificial intelligence, and machine learning [11].

3.1.3 Open CV

Over the years, OpenCV, a potent library for computer vision applications, has become very well known. Since Intel initially created it in 1999, a sizable development community has been actively maintaining it. The C++-written library offers a wide range of APIs that can be used for a variety of computer vision and image-processing activities. It is cross-platform and works with Android, iOS, Windows, Linux, and Mac OS.

The library is made to be simple to use and offers the user a high level of abstraction. As a result, users can easily do challenging computer vision tasks using only a small amount of code. Object recognition, face detection, picture segmentation, image stitching, motion detection, and video analysis are a few of the most often used OpenCV applications [12].

OpenCV's adaptability is one of its main benefits. It has a variety of uses, including robotics, healthcare, and security. Examples of applications for OpenCV include the detection and tracking of objects in robotics, the identification of faces and the tracking of motion in security applications, and the detection of tumors in medical images.

In general, OpenCV is a strong and adaptable library that is crucial for everyone working in the computer vision industry. Users may find a wealth of information on OpenCV online, including tutorials, code samples, documentation, and more.

3.1.4 Tesseract

Tesseract is an open-source optical character recognition (OCR) engine. It is a popular OCR tool that can read text from scanned documents and photographs and can recognize it in a variety of languages and formats. Tesseract is extremely accurate and trustworthy because it was trained on a substantial character dataset. The Tesseract OCR engine has a Python wrapper called Pytesseract. It enables the usage of Tesseract in Python programs and scripts. A simple-to-use API from Pytesseract enables programmers to extract text from PDF and picture files. Additionally, it is multilingual and can be trained with unique datasets to increase accuracy.

Features of Tesseract: Tesseract supports more than 100 languages, including the majority of those spoken in Europe, Asia, and Africa. Additionally, Pytesseract supports numerous languages, High accuracy: Tesseract has a very high accuracy rate. Python programs may exploit Tesseract's precision with ease thanks to Pytesseract, Tesseract and Pytesseract can be trained on unique datasets to increase accuracy for particular use cases.

Tesseract and Pytesseract are robust OCR solutions that offer high accuracy and language support, in conclusion. They are a popular option for various applications since they are open-source and may be tailored for certain use cases.

3.1.5 Tkinter

A common Python library for building graphical user interfaces (GUIs) is Tkinter. It offers a quick approach to making user-interactive windows, dialogues, buttons, menus, and other GUI elements. The majority of Python installs come with Tkinter, which is compatible with Windows, macOS, and Linux.

Tkinter's foundation is the Tk toolkit, which was created for the Tcl scripting language in the late 1980s. To generate windows, buttons, menus, and other GUI elements, the Tk toolkit offers a collection of widgets (GUI elements). It is simple to construct GUIs in Python thanks to Tkinter, which offers a Python interface to the Tk toolkit [11].

The simplicity of learning and utilizing Tkinter is one of its main advantages. It includes an easy-to-use API that makes it possible to quickly write GUIs with little code. Additionally, Tkinter offers a large selection of widgets, such as buttons, labels, text boxes, checkboxes, radio buttons, scrollbars, and more, that you may use to build your GUIs.

The event-driven programming feature of Tkinter enables your GUI to react to user activities like button clicks, menu selections, and keyboard input. This makes it simple to develop interactive GUIs capable of handling challenging jobs.

Using Tkinter's Canvas widget, you can make your unique widgets in addition to the conventional Tkinter widgets. You can design your unique widgets, such as graphs, charts, and diagrams, using the canvas widget, which offers a drawing surface.

Overall, Tkinter is a strong and adaptable Python library for building GUIs. Tkinter is a wonderful option for developing desktop apps with Python.

3.1.6 PIL

PIL, or the Python Imaging Library, is a well-liked Python image processing package. It enables programmers to access, edit, and save a wide variety of picture file formats. PIL offers a variety of image editing features, such as cropping, rotating, filtering, and image scaling. PIL is perfect for a wide range of projects kinds because of its straightforward and user-friendly interface. Its functionality can be further expanded thanks to a variety of add-on modules, making it a very flexible system [11].

- Image Processing A variety of tools are available through PIL to process and manipulate images. This contains the capacity to resize, crop, rotate, flip, and filter photographs in addition to supporting several image formats.

- Enhancement of photos PIL offers a variety of methods for improving the caliber of photos. In addition to tools for removing noise and sharpening photos, this also offers tools for changing brightness, contrast, and color balance.

- Image Analysis PIL comes with a variety of tools for image analysis. As well as tools for evaluating picture attributes like color histograms, edge detection, and others, this also includes tools for locating features and patterns within images.

- Image File Formats JPEG, PNG, GIF, BMP, and TIFF are just a few of the many image file formats that PIL supports. Additionally, it offers resources for converting photographs between different file formats.

PIL is quite extendable, and there are a variety of add-on modules available that can be used to increase its functionality. This comprises modules for using image filters, segmenting images, and other things. PIL is renowned for these capabilities in addition to being user-friendly and flexible. It is widely utilized in many various fields, such as the creation of websites, scientific research, and image editing software.

3.1.7 Pyttsx

A Python speech synthesis toolkit called Pyttsx makes it easy and effective to produce synthetic speech. It is constructed on top of the open-source Text-to-Speech project's (TTS) speech engine. Pyttsx can leverage platform-specific native speech APIs on Windows, Linux, and macOS in addition to a variety of speech engines, such as eSpeak and Microsoft Speech API [3]. Additionally, Pyttsx offers event-driven programming to regulate speech production. To regulate the voice output in real-time and to track the beginning and end of the speech, you can add event listeners. In conclusion, pyttsx is a strong and adaptable Python library for speech synthesis, supporting a variety of speech engines and event-driven programming. It is simple to use and flexible enough to provide a variety of voice outputs.

3.2 Hardware Description

3.2.1 Raspberry Pi 4 Model B

Figure 3.1: Raspberry Pi 4 Model B

The Raspberry Pi 4B is a powerful single-board computer as shown in Figure 3.1 with a quad-core 64-bit ARM Cortex-A72 CPU, ranging from 1.5GHz to 2GHz. It supports up to 8GB of LPDDR4-3200 SDRAM and features Gigabit Ethernet and dual-band 802.11ac wireless networking with Bluetooth 5.0 and BLE support. The board includes two USB 3.0 and two USB 2.0 ports, dual monitor support via HDMI and two micro-HDMI ports, and 40 GPIO pins for interfacing with sensors, displays, and other devices. It also features hardware-accelerated video playback up to 4Kp60, improved thermal management, and a microSD card slot for storage and operating system installation. Additionally, it supports optional PoE (Power over Ethernet) with a PoE HAT accessory, making it ideal for a variety of applications, including machine learning and computer vision projects [10].

Features :

- Quad-core 64-bit ARM Cortex-A72 CPU, ranging from 1.5GHz to 2GHz
- Supports up to 8GB of LPDDR4-3200 SDRAM
- Gigabit Ethernet and dual-band 802.11ac wireless networking

- Bluetooth 5.0 and BLE support
- Two USB 3.0 and two USB 2.0 ports for connecting peripherals
- Dual monitor support via HDMI and two micro-HDMI ports
- 40 GPIO pins for interfacing with sensors, displays, and other devices
- Hardware-accelerated video playback up to 4Kp60
- Improved thermal management for better performance and stability
- MicroSD card slot for storage and operating system installation
- Optional PoE (Power over Ethernet) support with a PoE HAT accessory

3.2.2 5 MP Camera Module

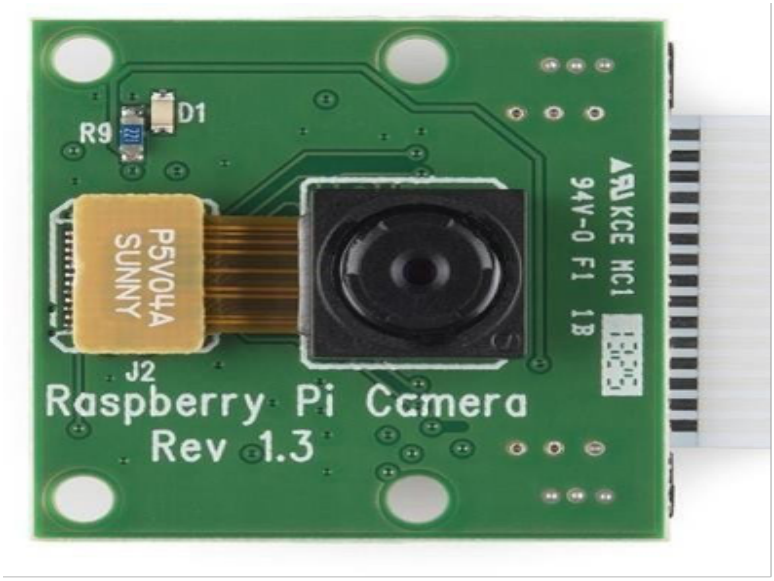


Figure 3.2: Camera Module

Camera module with cable equipped with a flexible cable as shown in Figure 3.2 that can be connected to the raspberry pi. It is Lightweight is very useful in various aspects with very little space allowance. This 5MP camera provides high-definition pictures and can also record videos [10]. The camera is capable of 2592 x 1994 pixels static images, and also capable of 1080p30, 720p60, and 640 x 480p60/90 video.

Features :

- Fully compatible with both Model A, Model B and B++.
- Still picture resolution: 2592 x 1994.

- Supports 1080p @ 30fps, 720p @ 60gps, and 640 x 480p 60/90 Video recording.
- 15-Pin MIPI Camera Serial Interface plugged directly into Raspberry Pi board.
- It weighs 3gm.

3.2.3 LCD Display

Figure 3.3: LCD Display

LCD communicates using the I2C protocol. LCD Display as shown in Figure 3.3 has a resolution of 320×480 resolution. It will be connected to the GPIO pins of Raspberry Pi 4 and data transfer will take place through GPIP pins. It supports multiple operating modes: 3-wire, SPI, 4-wire SPI, and I2C [10].

Features:

- 320×480 resolution.
- Resistive touch control.
- Supports any revision of Raspberry Pi (directly pluggable)
- Compatible with Raspberry Pi A, B, A+, B+, 2B, 3B, 3B+, and 4B versions.
- Drivers provided (works with your own Raspbian/Ubuntu directly)The size perfectly fits the Pi.
- Supports Raspbian system, ubuntu system, Kali Linux system.

3.2.4 Speaker

The Raspberry Pi 4B offers a 3.5mm audio jack for connecting speakers as shown in Figure 3.4 or headphones, providing stereo sound output with a maximum resolution of 16-bit/48kHz. HDMI can also be used for audio output, allowing connection to a display or television with built-in speakers. The Raspberry Pi can also be connected to external speakers or headphones via USB or Bluetooth for audio output.

Figure 3.4: Speaker

It's essential to adjust the audio output volume using software controls. Additionally, using USB or Bluetooth headphones requires the installation of drivers for proper functionality [10]. Moreover, the Raspberry Pi's audio output quality is not as high as that of dedicated audio devices due to its primary focus on general-purpose computing.

Chapter 4
Methodology

This is Python code that implements a text recognition application using OpenCV, Pytesseract, tkinter, and NLTK libraries. This application captures an image from a camera and uses the Pytesseract OCR engine to recognize text from the image. We then display the recognized text in her GUI and allow the user to convert it to speech using her pyttsx3.

4.1 Preprocessing techniques

4.1.1 Gaussian Blur

Gaussian Blur is a commonly used image processing technique that can be used for text recognition to remove noise and blur images. It does this by convolving the image with a Gaussian filter kernel and blurring the image to reduce image noise and make the text easier to see and perceive.

The weight that should be assigned to each pixel in the image during the convolution process is specified by the Gaussian filter kernel, which is a matrix of numbers. The size of the kernel and its standard deviation, which influence the spread of the blur, are typically used to characterize it [8].

Text recognition can use Gaussian blur to preprocess images to improve the accuracy of OCR software such as Tesseract. The blurring process helps to smooth the image. This reduces noise, enhances edges, and makes it easier for Tesseract to see and recognize text. Tesseract uses a combination of image processing techniques, including Convolutional Neural Networks (CNN), to perform his OCR. CNN is a class of deep learning algorithms commonly used for image recognition and classification tasks. CNNs work by learning patterns and features in images through a process of training and backpropagation. The two-dimensional Gaussian function is a commonly used mathematical function for describing the probability distribution of a point in a two-dimensional space is given by,

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.1}$$

Where, G(x,y) is the value of the Gaussian function at a particular point (x,y) in the x-y plane. $1/(2\pi\sigma^2)$ is a normalization factor that ensures that the integral of the

Gaussian function over the entire x-y plane is equal to 1. This factor is constant and does not depend on the values of x and y. σ is the standard deviation of the Gaussian function. The variance determines the spread of the Gaussian function and how quickly the values decay as you move away from the center. In image processing and computer vision, the Gaussian function is often used as a smoothing filter to remove noise and blur edges. The standard deviation of the distribution, denoted by σ, controls the spread of the distribution and determines the rate at which the values decay as you move away from the center.

4.1.2 Adaptive Thresholding

To distinguish an image's foreground from background, image processors utilize a technique called adaptive thresholding. Adaptive thresholding is frequently used in text detection and OCR to increase the contrast of the text against the backdrop and boost the precision of text recognition. In this, thresholds are dynamically calculated based on local pixel intensities in the image. This means that the threshold is not constant across the image, but varies with local pixel values. This is useful when the lighting conditions in the image do not match the background [4].

Adaptive thresholding is used by Tesseract OCR to preprocess photos before text recognition. The technique initially blurs the image with a Gaussian function to lessen noise and smooth the text areas. The image is then binarized using adaptive thresholding after being converted to grayscale. This procedure aids in enhancing the contrast of the text areas and eliminates any background noise or shadows that can obstruct OCR. Tesseract OCR uses several image-processing algorithms to partition the text regions and identify the characters in the image after the image has been preprocessed with adaptive thresholding. These methods include character recognition utilizing convolutional neural networks, linked component analysis, and text line identification. The outcome is excellent OCR output that recognizes the text in the image precisely.

Each pixel in the neighborhood contributes equally to computing the threshold value, T. In the Gaussian mean, pixel values farther away from the (x, y)-coordinate center of the region contribute less to the overall calculation of T. The threshold value is calculated by,

$$T = \text{mean}(L) - C \tag{4.2}$$

Where the mean is either the arithmetic or Gaussian mean, L is the local sub-region of the image I(x, y), and C is some constant which can use to fine-tune the threshold value T.

Adaptive thresholding works by calculating a threshold for each pixel based on the surrounding pixels. It's particularly useful for images with uneven lighting or variations in contrast. First, the image is divided into smaller regions or "blocks". A threshold is calculated for each block based on the average intensity of the pixels within the block. This threshold is then applied to all pixels in that block to create a binary image. Each pixel is black or white depending on whether its intensity is above or below the threshold.

4.1.3 Invert

Inverting a binary image means converting all white pixels to black or vice versa, effectively creating a negative image. This is useful for text recognition as it helps improve the contrast between the text and the background. When using Tesseract OCR, reversing the binary image improves OCR accuracy as the text is more visible and recognizable. This is because Tesseract works best when there is a clear contrast between the text and the background [1]. The image gradient magnitude at pixel (x, y) is given by,

$$I'(x, y) = \max(I(x, y)) - I(x, y) \tag{4.3}$$

Where I(x, y) represents the intensity value at pixel (x, y) in the input image I, max(I(x, y)) represents the maximum intensity value in the local neighborhood of pixel (x, y). The size of the neighborhood is typically specified by a parameter known as the window size or kernel size and I'(x, y) represents the gradient magnitude at pixel (x, y), which is equal to the difference between the maximum intensity value in the local neighborhood and the intensity value at pixel (x, y).

Used in image processing to calculate the image gradient, which is a measure of the intensity variation in an image at each point. The equation computes the difference between the maximum intensity value in the local neighborhood of a pixel (x, y) and the intensity value at that pixel itself. This difference represents the magnitude of the gradient at that point.

4.1.4 Dilation

Dilation is a morphological operation used in image processing to extend the boundaries of the foreground pixel region of a binary image. A structuring element is pushed into the binary image, replacing the pixel in the middle of the kernel with the maximum pixel value in the kernel.

In Tesseract OCR, dilation is an image-processing technique used to improve text recognition results. Dilation is a morphological operation applied to binary images. This involves adding pixels to the boundaries of objects in the image. This allows you to fill in small gaps and make objects more cohesive.

Text recognition uses dilation to make regions of text more continuous and easier to see. This is done by applying a structuring element, a small array of pixels that defines the shape of the stretch operation, to the binary image. The structuring element is usually a square or rectangular matrix whose size is determined by the size of the text area in the image [6].

The stretch operation of Tesseract text recognition is applied to the binary image obtained after thresholding. Its main purpose is to fill gaps in the text and connect broken pieces of letters that are close to each other. This creates a more complete and contiguous range of foreground pixels that Tesseract OCR can further analyze and extract text. The dilation of the image is given by,

$$B \oplus A = \{z|(B - z) \cap A \neq \varnothing\} \tag{4.4}$$

Where, B represents the structuring element, which is typically a small binary image used for defining the shape and size of the operation. It is often used to detect features or patterns in an image, A represents the set of points to which the structuring element is applied. It is often a binary image or a binary mask representing an object or a region of interest in an image, and Z represents the translation of the structuring element B to different positions in the space. (B - z) represents the translation of set B to position z.

The binary image's text portions are enlarged during the dilation operation, which can make them more linked and simpler for Tesseract OCR to read. Tesseract OCR may have a harder time identifying specific characters if the structuring element is too large since it may cause the text regions to converge. Tesseract OCR uses dilatation to improve text identification results, but for best results, it must be utilized cautiously and in conjunction with other image processing methods. But if the kernel size is too large, or if the image contains other regions without text, the

augmentation can introduce noise and false positives. Therefore, it is often used in combination with other morphological operations such as erosion and openings to improve the overall performance of text recognition algorithms.

4.1.5 Grey scaling

Converting an image from BGR to grayscale reduces the dimensionality of the image, making it easier to process and useful for text recognition using Tesseract. Also, since grayscale images have only one channel which is intensity, Tesseract can easily identify regions of text in the image. In addition, grayscale images help reduce image noise and make text regions easier to identify. This is because color information can contribute to image noise, and removing color information through a grayscale conversion helps reduce noise [1]. The gray image is given by,

$$I_{gray}(x, y) = 0.2989 \times R(x, y) + 0.5870 \times G(x, y) + 0.1140 \times B(x, y) \tag{4.5}$$

Where, R(x, y), G(x, y), and B(x, y) represent the red, green, and blue color values of the pixel located at (x, y) in the color image. The equation assigns a weight of 0.2989 to the red color value, 0.5870 to the green color value, and 0.1140 to the blue color value. These weights are chosen to match the relative luminance of the three primary colors that are perceived by the human eye. The resulting grayscale value at pixel (x, y) is the weighted sum of the color values at that pixel.

4.2 Text Processing-NLTK

NLTK (Natural Language Toolkit) is a Python library that provides a wide range of tools for natural language processing tasks such as tokenization, stemming, tagging, parsing, and semantic reasoning. NLTK can be used to preprocess and analyze text data to extract meaningful features and metadata. This can include tasks such as sentence segmentation, part-of-speech tagging, and named entity recognition. NLTK also provides access to a wide range of pre-trained models and corpora for training machine learning models for natural language tasks. This can be useful for training models for text detection and recognition tasks, such as training a machine learning model to recognize specific patterns or entities in text data [5].

NLTK is a software library designed to aid in Natural Language Processing (NLP) tasks, such as sentiment analysis, topic modeling, and language translation.

It provides a comprehensive set of tools to manipulate and analyze human language data, including tokenization, stemming, lemmatization, and part-of-speech tagging. NLTK also comes with pre-built corpora, including the Brown Corpus, which contains over a million words of English text, and the Treebank Corpus, which is annotated with phrase structure and part-of-speech tags. Additionally, NLTK offers machine learning algorithms that can be trained on custom datasets for tasks such as text classification and named entity recognition [5].

Chapter 5
Algorithm

Developing a GUI for detecting and extracting text from live camera feed using the Tesseract OCR engine. The application also allows the user to capture an image and perform OCR on the captured image. The application is built using the OpenCV library for handling the camera feed and image processing, a tkinter for the GUI, and a pytesseract for performing OCR. Additionally, the application uses the NLTK library to filter out non-English words and the pyttsx3 library to convert the extracted text to speech. Upon running the application, a GUI window is opened, which displays the live camera feed. The user can capture an image by clicking the "Capture" button, which saves the image to a file and performs OCR on the image to extract text. The extracted text is filtered to remove non-English words and is displayed in the output text box. The following are general algorithms for Text detection [11].

5.1 Preprocessing

To increase the algorithm's accuracy for text detection, preprocessing is a crucial step. The input image is improved and cleaned using several approaches before being fed into the text detection algorithm. Image scaling, noise removal, contrast correction, and binarization are all part of the preprocessing stage. To normalize the size of the input image to the necessary dimensions for the word detection technique, image resizing is done. The image's undesired noise is removed using noise-removal techniques like median filtering, Gaussian filtering, or bilateral filtering. By altering the brightness and contrast of the image, contrast adjustment is used to make the text in the image more visible.

5.2 Text detection and Reorganization

Text Detection and Reorganization is a process used in computer vision to identify and extract text from images, and then organize the recognized text into a structured format. The algorithm involves preprocessing the input image to enhance quality and remove noise, followed by text detection to identify text regions. The text is then extracted and recognized using a text recognition algorithm. The recognized text is organized into a structured format, with further reorganization

and summarization using natural language processing. The output is the organized text in the desired format. Specific techniques used may vary depending on project requirements.

The general flow chart involved in a typical text detection and reorganization algorithm is shown in figure 5.1

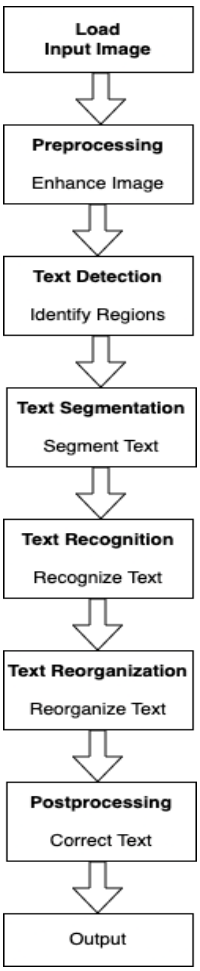


Figure 5.1: Text Detection and Reorganization Flow Chart

The steps involved in the flow chart of text detection and reorganization algorithm are as follows:

- **Pre-processing:** The input image is Pre-processed to enhance the text regions and remove any noise or unwanted elements. This involves techniques such as image filtering, thresholding, and binarization.

- **Text Detection:** The pre-processed image is analyzed to identify regions that contain text. This can be done using various techniques such as connected component analysis, edge detection, or machine learning algorithms.

- **Text Segmentation:** Once the text regions have been identified, the algorithm needs to segment the text lines and characters from the rest of the image. Involves techniques such as clustering, contour detection, or projection profile analysis.

- **Text Recognition:** The segmented text lines and characters are then recognized using OCR techniques or other machine-learning algorithms.

- **Text Reorganization:** Once the text has been recognized, the algorithm needs to reorganize the text into a coherent structure such as paragraphs, tables, or columns. The techniques such as layout analysis, semantic parsing, or natural language processing are used.

- **Post-processing:** Finally, the output text is post-processed to correct any errors, improve the formatting, and enhance the readability. The techniques such as spell-checking, punctuation correction, or font normalization are involved in this step.

5.3 Text-To-Speech

TTS is a technology that converts written text into spoken words. The algorithm for TTS involves several steps. First, the text is preprocessed to segment it into smaller units such as words or phrases. Next, a linguistic analysis is performed to determine the phonetic transcription of the text. Then, the phonetic transcription is converted into a digital signal using a speech synthesis algorithm, which produces an audio output that can be played through a speaker or headphones [3]. The quality of the synthesized speech depends on the accuracy of the linguistic analysis and the effectiveness of the speech synthesis algorithm. TTS is widely used in various applications such as navigation systems, virtual assistants, and audiobooks.

The text-to-speech algorithm's general flow chart is shown in figure 5.2. The algorithm of the text-to-speech involves many steps which are as follows:

- **Load input text:** Load the input text that needs to be converted to speech. This is the first step of the TTS algorithm where the input text is loaded into the system that needs to be converted to speech. The input text can be written text or spoken input.

- **Text preprocessing:** Preprocess the input text by tokenizing it, tagging parts of speech, and analyzing pronunciation to prepare it for acoustic modeling. In this step, the input text is pre-processed to make it suitable for acoustic

modeling. This includes tasks such as tokenization, part-of-speech tagging, and analyzing pronunciation.

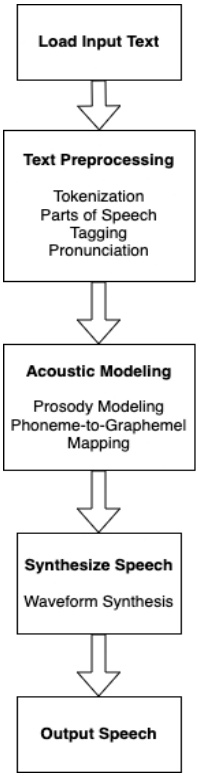


Figure 5.2: Text-To-Speech Flow Chart

- **Acoustic modeling:** Develop a model that maps the pre-processed text into acoustic features such as prosody and phoneme-to-grapheme mapping. Acoustic modeling is the process of mapping the pre-processed text into acoustic features such as prosody and phoneme-to-grapheme mapping. Various statistical models and algorithms are used in this stage.

- **Synthesize speech:** Use of waveform synthesis techniques to generate the speech signal from the acoustic features generated in the previous step. In this step, the synthesized speech signal is generated by combining the acoustic features generated in the previous step with the appropriate prosodic and timing information. Waveform synthesis techniques such as concatenative synthesis or parametric synthesis can be used.

- **Output speech:** The synthesized speech signal is output as an audio file or played through a speaker. This is the final step where the synthesized speech signal is output as an audio file or played through a speaker. This can be in the form of a computer-generated voice, a human voice, or a mix of both.

Chapter 6
Results

Using machine learning techniques, text detection and recognition seek to create an efficient system that can identify and recognize text in photos and videos. Preprocessing the input data, training a machine learning model, and assessing the model's performance are all part of the project. The ultimate goal is to create a robust and accurate text detection and recognition system that can be applied to various real-world applications, such as automated document processing, image and video search, and accessibility tools for the visually impaired.

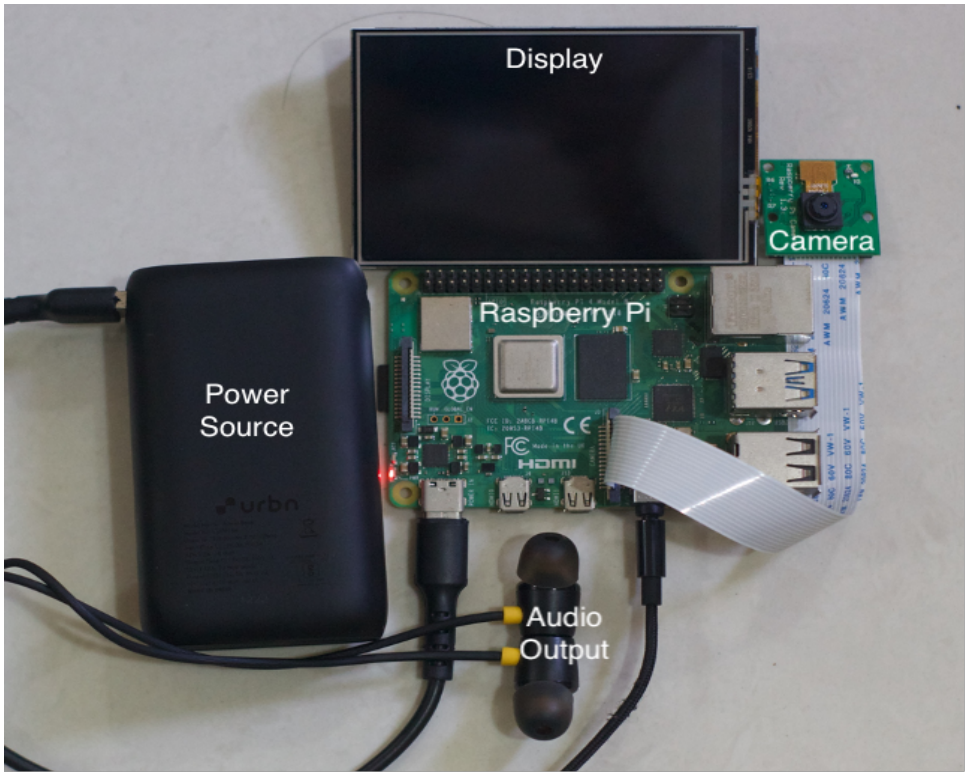
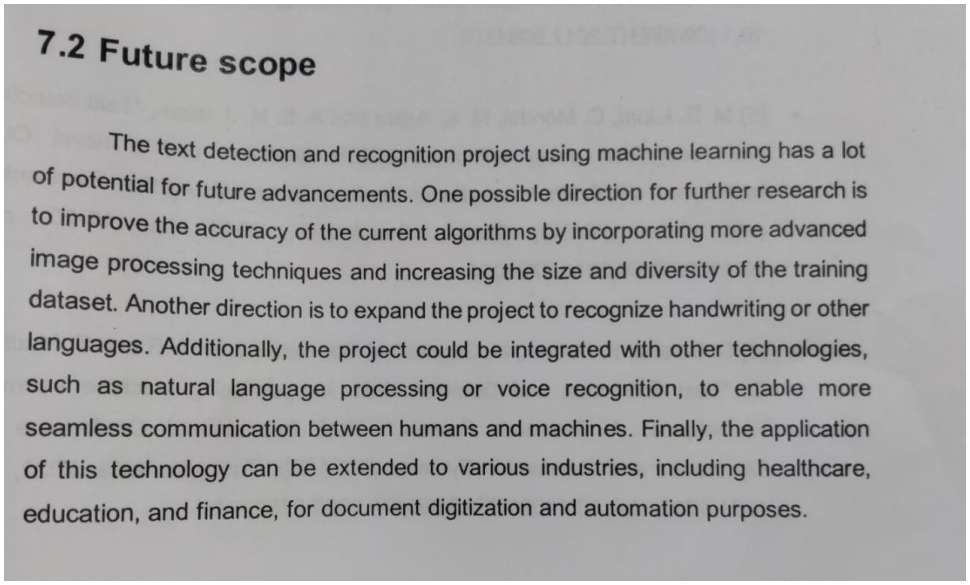


Figure 6.1: System model

The system model is shown in figure 6.1. A photo of input is captured through the camera by triggering the capture button. This will make the system take a photo of the frame at that instance, as shown in figure 6.3, and the input image that the camera has taken at that moment is shown in figure 6.2.



7.2 Future scope

The text detection and recognition project using machine learning has a lot of potential for future advancements. One possible direction for further research is to improve the accuracy of the current algorithms by incorporating more advanced image processing techniques and increasing the size and diversity of the training dataset. Another direction is to expand the project to recognize handwriting or other languages. Additionally, the project could be integrated with other technologies, such as natural language processing or voice recognition, to enable more seamless communication between humans and machines. Finally, the application of this technology can be extended to various industries, including healthcare, education, and finance, for document digitization and automation purposes.

Figure 6.2: Captured image

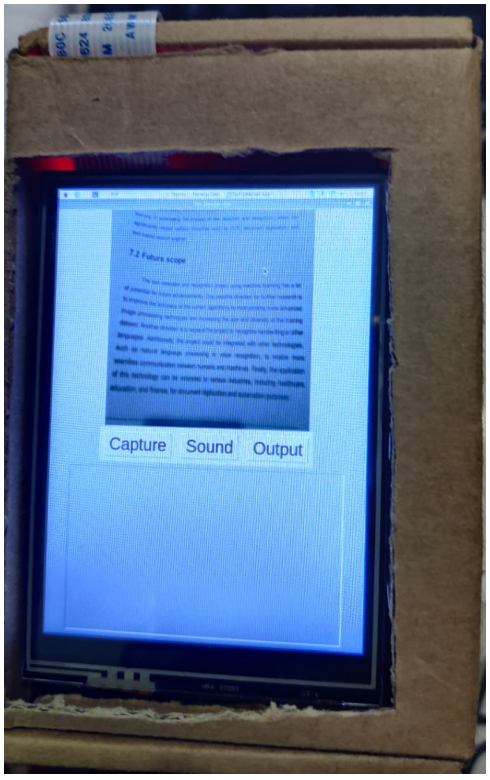


Figure 6.3: Before the detection

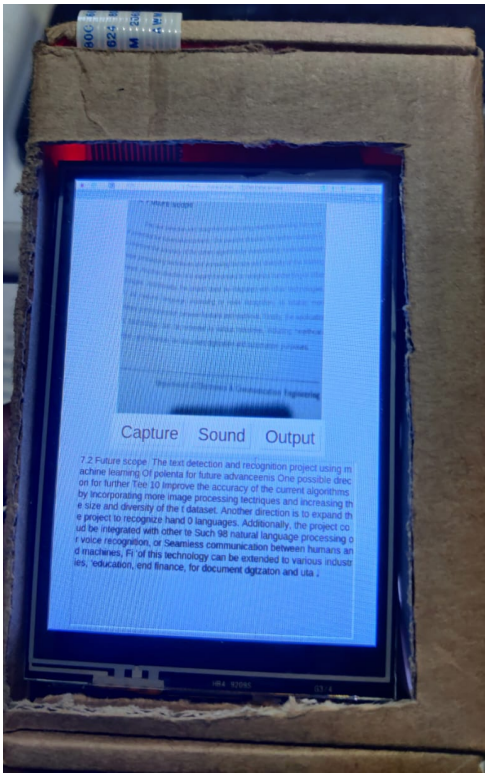


Figure 6.4: Detected output

After the processing of the input image, the detected output will be displayed on the screen as shown in figure 6.4. After the text detection when the button "sound" is pressed, the system converts the detected text to audio output using the text-to-speech conversion method, it is played on a connected audio output device like a speaker or an earphone.

Chapter 7
Conclusion and Future scope

7.1 Conclusion

Text detection and recognition using machine learning is a viable approach for automating text information extraction from images. Preprocessing techniques such as image resizing, binarization, and noise removal can significantly improve the accuracy of text detection algorithms. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) are effective machine learning models for text recognition tasks. The accuracy of the text detection and recognition system can be further improved by using a combination of these models and optimizing their hyperparameters. Overall, this project demonstrates the potential of machine learning in automating the process of text detection and recognition, which can significantly impact various industries such as OCR, document digitization, and text-based search engines.

7.2 Future Scope

The text detection and recognition project using machine learning has a lot of potential for future advancements. One possible direction for further research is to improve the accuracy of the current algorithms by incorporating more advanced image processing techniques and increasing the size and diversity of the training dataset. Another direction is to expand the project to recognize handwriting or other languages. Additionally, the project could be integrated with other technologies, such as natural language processing or voice recognition, to enable more seamless communication between humans and machines. Finally, the application of this technology can be extended to various industries, including healthcare, education, and finance, for document digitization and automation purposes.

Bibliography

[1] X. Huang, T. Shen, R. Wang, and C. Gao, "Text detection and recognition in natural scene images," in *2015 International Conference on Estimation, Detection and Information Fusion (ICEDIF)*. IEEE, 2015, pp. 44–49.

[2] S. L. Wasankar, H. Mahajan, D. Deshmukh, and H. Munot, "Machine learning with text recognition," in *2010 IEEE International Conference on Computational Intelligence and Computing Research*. IEEE, 2010, pp. 1–5.

[3] R. Ani, E. Maria, J. J. Joyce, V. Sakkaravarthy, and M. A. Raja, "Smart specs: Voice assisted text reading system for visually impaired persons using tts method," in *2017 International Conference on Innovations in Green Energy and Healthcare Technologies (IGEHT)*. IEEE, 2017, pp. 1–6.

[4] Q. Ye and D. Doermann, "Text detection and recognition in imagery: A survey," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 37, no. 7, pp. 1480–1500, 2015.

[5] C. Jeeva, T. Porselvi, B. Krithika, R. Shreya, G. S. Priyaa, and K. Sivasankari, "Intelligent image text reader using easy ocr, nrclex & nltk," in *2022 International Conference on Power, Energy, Control and Transmission Systems (ICPECTS)*, 2022, pp. 1–6.

[6] S. Surana, K. Pathak, M. Gagnani, V. Shrivastava, M. T. R, and S. M. G, "Text extraction and detection from images using machine learning techniques: A research review," in *2022 International Conference on Electronics and Renewable Systems (ICEARS)*, 2022, pp. 1201–1207.

[7] R. Fernandes and A. P. Rodrigues, "Kannada handwritten script recognition using machine learning techniques," in *2019 IEEE International Conference on Distributed Computing, VLSI, Electrical Circuits and Robotics (DISCOVER)*, 2019, pp. 1–6.

[8] M. R. Islam, C. Mondal, M. K. Azam, and A. S. M. J. Islam, "Text detection and recognition using enhanced mser detection and a novel ocr technique," in *2016 5th International Conference on Informatics, Electronics, and Vision (ICIEV)*. IEEE, 2016, pp. 15–20.

[9] V. Padmapriya, R. Archna, V. Lavanya, and C. V. Sri, "A study on text recognition and obstacle detection techniques," in *2020 International Conference on System, Computation, Automation, and Networking (ICSCAN)*, 2020, pp. 1–6.

[10] R. P. Foundation. (2021) Raspberry pi. Accessed: 2022-2023. [Online]. Available: https://www.raspberrypi.org/

[11] P. S. Foundation. (2023) Python. [Online]. Available: https://www.python.org/

[12] OpenCV. (2022) Opencv library. [Online]. Available: https://opencv.org/