VOICE AIDED IMAGE PROCESSING FOR BLIND PEOPLE – A COMPUTER VISION APPROACH

A PROJECT REPORT

Submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

in

COMPUTER SCIENCE AND ENGINEERING

BY

G.V.S. Ravi Kiran Varma
20331A0554

Devarakonda Partha Sai
20331A0545

Bangaari Eswari Sai
21335A0501

Botsa Yaswanth
20331A0527

Under the Supervision of
Mrs. B. Aruna Kumari
Associate Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAHARAJ VIJAYARAM GAJAPATHI RAJ COLLEGE OF ENGINEERING

(Autonomous)

(Approved by AICTE, New Delhi, and permanently affiliated to JNTUGV, Vizianagaram), Listed u/s 2(f) & 12(B) of UGC Act 1956.

Vijayaram Nagar Campus, Chintalavalasa,Vizianagaram-535005, Andhra Pradesh

APRIL, 2024

CERTIFICATE

This is to certify that the project report entitled "**VOICE AIDED IMAGE PROCESSING FOR BLIND PEOPLE – A COMPUTER VISION APPROACH**" being submitted by G.V.S. Ravi Kiran Varma (20331A0554), D. Partha Sai (20331A0545), B. Eswari Sai (21335A0501), B. Yaswanth (20331A0527) in partial fulfillment for the award of the degree of "**Bachelor of Technology" in Computer Science and Engineering** is a record of bonafide work done by them under my supervision during the academic year 2023-2024.

Mrs. B. Aruna Kumari
Associate Professor,
Supervisor,
Department of CSE,
MVGR College of Engineering(A),
Vizianagaram.

Dr. T. Pavan Kumar
Associate Professor,
Head of the Department,
Department of CSE,
MVGR College of Engineering(A),
Vizianagaram.

External Examiner

DECLARATION

We hereby declare that the work done on the dissertation entitled "**VOICE AIDED IMAGE PROCESSING FOR BLIND PEOPLE – A COMPUTER VISION APPROACH**" has been carried out by us and submitted in partial fulfilment for the award of credits in Bachelor of Technology in Computer Science and Engineering of MVGR College of Engineering (Autonomous) and affiliated to Jawaharlal Nehru Technological University (Kakinada). The various contents incorporated in the dissertation have not been submitted for the award of any degree of any other institution or university.

ACKNOWLEDGEMENTS

We express our sincere gratitude to **Mrs. B. Aruna Kumari** for her invaluable guidance and support as our mentor throughout the project. Her unwavering commitment to excellence and constructive feedback motivated us to achieve our project goals. We are greatly indebted to her for her exceptional guidance.

Additionally, we extend our thanks to **Prof. P.S. Sitharama Raju (Director)**, **Prof. Ramakrishnan Ramesh (Principal)**, and **Dr. T. Pavan Kumar (Head of the Department)** for their unwavering support and assistance, which were instrumental in the successful completion of the project. We also acknowledge the dedicated assistance provided by all the staff members in the Department of Computer Science & Engineering. Finally, we appreciate the contributions of all those who directly or indirectly contributed to the successful execution of this endeavour.

G.V.S. Ravi Kiran Varma	**(20331A0554)**
Devarakonda Partha Sai	**(20331A0545)**
Bangaari Eswari Sai	**(21335A0501)**
Botsa Yaswanth	**(20331A0527)**

LAST MILE EXPERIENCE (LME)

PROJECT TITLE

VOICE AIDED IMAGE PROCESSING FOR BLIND PEOPLE

BATCH NUMBER – 9A

BATCH SIZE – 4

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

Name: G.V.S. Ravi Kiran Varma
Email: ravivarma25052@gmail.com
Contact Number: 6303948186

Name: Devarakonda Partha Sai
Email: devarakondaparthasai@gmail.com
Contact Number: 9390925653

Name: Bangaari Eswari Sai
Email: eswarisai4371@gmail.com
Contact Number: 7993789589

Name: Botsa Yaswanth
Email: yaswanthyassu9726@gmail.com
Contact Number: 8367259633

Project Supervisor

Name: Mrs. B. Aruna Kumari
Designation: Associate Professor
Email: arunasrinivas@mvgrce.edu.in
Contact Number: 9849672453

Project Objectives

1. Create a user-friendly GUI prototype for an IoT device specifically designed for deaf-blind individuals.
2. Implement features in the device to capture text from images, translate it into various languages, and summarize the content.
3. Enable the device to transcribe real-time speech in any language and display it on the GUI.
4. Integrate features that make education more accessible for deaf-blind individuals.

Project Outcomes

1. Creation of a Multifunctional Device
2. Bridging the Communication Gap
3. Self-Education for Illiterates
4. Content Summarization

Domain of Specialisation

Computer Vision

How your solution helping the domains?

Our Computer Vision project utilizes image recognition and optical character recognition (OCR) to extract text from images, showcasing practical applications for assistive technologies.

List the Program Outcomes (POs) that are being met by doing the project work

PO3 - Design / Development of Solutions	We designed and developed a user-friendly GUI prototype for an IoT device, demonstrating the ability to design solutions for complex engineering problems.
PO4 - Conduct Investigations of Complex Problems	Our project investigates a complex problem enhancing accessibility for deaf-blind individuals. It uses research-based knowledge to analyze and interpret data.
PO6 - The Engineer and Society	Our project caters to the specific needs of deaf-blind individuals, demonstrating an understanding of the societal impact of engineering solutions.
PO9 - Individual and Teamwork	The project was done as a team, it shows the ability to function effectively as an individual and in multidisciplinary teams.

End Users of Your Solution

Physically Challenged Individuals – Deaf and Blind

ABSTRACT

In the modern world, technology has become an integral part of our lives. However, its benefits are not equally accessible to everyone. People with visual and hearing impairments often face challenges in accessing and utilizing these technologies. This project aims to bridge this gap and make technology more inclusive.

The system proposed in this project uses advanced algorithms and techniques to convert text into speech and vice versa. For visually impaired individuals, the system can read out loud the text from books, signs, or digital screens, enabling them to understand their surroundings better. This feature can significantly enhance their learning experience and make navigation in unfamiliar environments safer and easier.

For individuals with hearing impairments, the system can convert spoken words into written text. This feature can be particularly useful in classroom settings, where hearing-impaired students can read what the teacher is saying in real-time. It can also facilitate better communication with others, as they can read the transcribed text of the spoken conversation.

Moreover, the system is designed to be user-friendly and easy to operate, making it a practical tool for people of all ages. It can be integrated into smartphones, computers, and other digital devices, making it readily accessible whenever and wherever it’s needed.

By making learning and communication easier for people with visual and hearing impairments, this project hopes to foster a more inclusive society where everyone has equal opportunities to learn, grow, and thrive. The ultimate goal is to empower these individuals and help them lead more independent and fulfilling lives.

CONTENTS

List of Abbreviations

CI	-	Cognitive Impairment
GUI	-	Graphical User Interface
IoT	-	Internet of things
AI	-	Artificial intelligence
DEF	-	Deaf Enabled Foundation
HIV	-	Human Immuno Deficiency Virus
CV	-	Computer Vision
DL	-	Deep Learning
IA	-	Image Acquisition
IR	-	Image Restoration
IS	-	Image Segmentation
OCR	-	Optical Character Recognition
ROI	-	Region of Intrest
AT	-	Amazon Tessaract
Pyttsx3	-	Python Text to Speech X3
gTTS	-	Google Text to Speech
PIL	-	Python Image Library
PDF	-	Portable Document Format
AJAX	-	Asynchronous JavaScript And XML
HTTP	-	Hyper Text Transfer Protocol
LLMs	-	Large Language Models
GPT	-	Generative Pre-trained Transformer
BERT	-	Bidirectional Encoder Representations from Transformers
LC	-	LangChain
SDK	-	Software Development Kit
MP3	-	MPEG Audio Layer-3
MAC	-	Media Access Control Address
API	-	Application Programming Interfce
WAVE	-	Waveform Audio File Format
PCM	-	Pulse-Code Modulation
NLP	-	Natural Lanaguage Processing
RTPT	-	Real Time Speech Transcription

List of Figures

CHAPTER 1
INTRODUCTION

The field of education for deaf-blind children is limited by the number of children involved and the number of people actively interested in the problem. There are relatively few individuals with the double handicap. Deaf-blind- ness occurs most often in adults. Accidents, severe diseases and sickness cause deafness or blindness in these already having one handicap or it can result in both handicaps occur ring in one person. Inheritance, rubella during the first three months of pregnancy, birth injuries, and premature births are additional contributing causes of deaf-blindness in children.

Most deaf-blind people are adults because the double handicap occurs more often in the later years of life. Many deaf-blind children are feeble-minded and are placed in institutions for custodial care. The remaining number of children with whom schools and education are directly concerned represent a small percentage of the total.

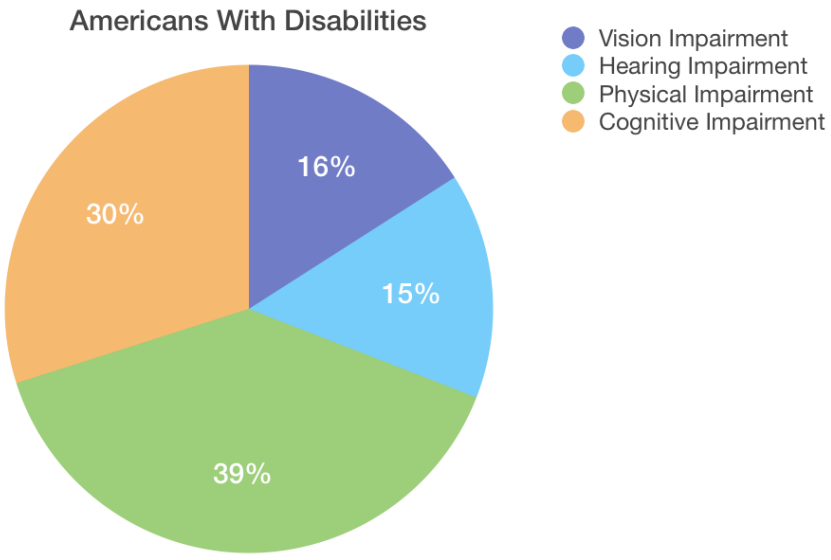


Figure 1.1: Pie chart of 2010 US census report on Americans with Disabilities

This consideration is important because overlooking it means potentially overlooking a large swath of your user base. According to the 2010 US census report on Americans with disabilities, approximately 50.8 million people in the United States have a disability of some sort: 8.1 million people have difficulty seeing, 7.6 million have difficulty hearing, 19.9 million have a physical disability, and 15.2 million have learning or cognitive disabilities like dyslexia or more serious developmental disabilities.

Adults with deaf blindness were less likely to have attended school as children than adults with other disabilities and adults without disabilities in each dataset. After accounting for age and gender, this finding was still statistically significant in all datasets, except for Mexico and Uruguay, where there was no significant difference between adults with deaf blindness and those with other disabilities.

Considering that age of onset is likely to have been after the completion of education for many adults with deaf blindness, this may be due to adults with lower socio-economic status being less likely to attend school or seek healthcare for functional limitations related to ageing.

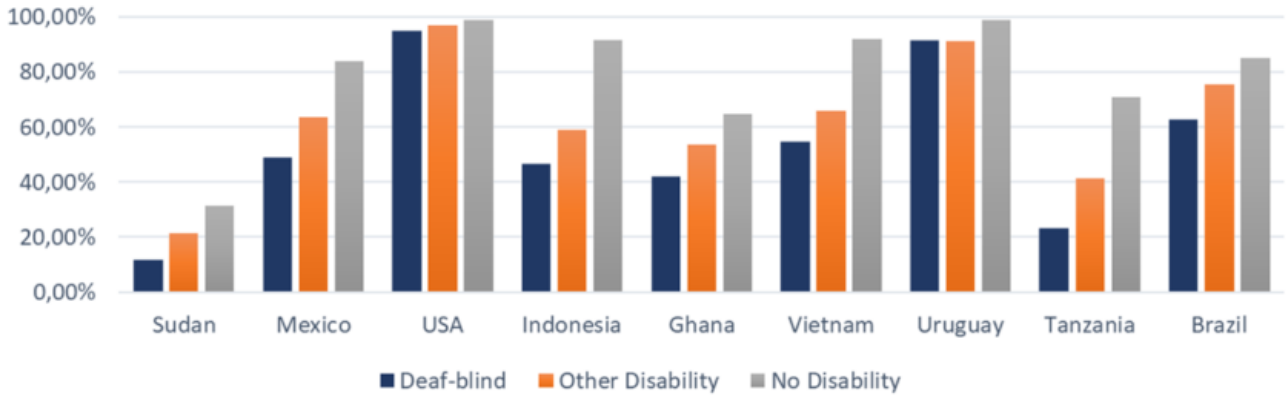


Figure 1.2: Bar diagram showing Adults with deaf-blindness

1.1 Identification of seriousness of the problem

The number of students with visual impairment being educated in general education classrooms is increasing worldwide. Inclusion is every student's right, not a privilege.

General education teachers' attitudes toward including visually impaired students vary. Some teachers are positive, while others may feel unprepared. Teacher training and effective pedagogical strategies are crucial.

Although visually impaired students often study at or above their grade level, they face challenges in participating fully in classroom activities. Key subjects like Mathematics, Science, and Physical Education are areas of concern.

Deaf pupils in England struggle at every stage of school. Before secondary school (age 11), fewer than half (43%) of deaf students reach the expected standard for reading, writing, and math (compared to 74% of other children).

Overall, deaf adults in the United States attain lower levels of education than their hearing peers. In 2017, 83.7% of deaf adults successfully completed high school, compared to 89.4% of hearing adults.

Approximately 2.21% of the total population in India is disabled, with 19% being visually impaired. Inclusion increases self-confidence and helps visually impaired children integrate into society.

Major causes of visual impairment in children depend on the local context but may include cataract, Vitamin A deficiency, retinopathy of prematurity, pediatric glaucoma, amblyopia, and uncorrected refractive error.

1.2 Problem definition

As deaf blindness in children and young adults is rare, most educational professionals receive little, if any, training, or support to work with students with deaf blindness. Learners with deaf blindness are also a very heterogeneous group, so teaching and learning strategies may vary greatly between individuals. This underscores the importance of making education more accessible and inclusive for all.

1.3 Objective

With the help of our algorithm, we tackle the independence and mobility issues faced by visually impaired individuals, as well as the communication barriers encountered by deaf individuals. These challenges are further compounded by the lack of access to quality education, which restricts their opportunities and potential.

1.4 Our Solution

This project involves the development of an inclusive technology in the form of an user friendly GUI prototype for an IoT device, designed specifically for deaf-blind individuals. The device captures text from images, translates it into various languages, and summarizes the content, making information more accessible.

For those with hearing impairments, the device transcribes speech in any language and displays it on the screen in real-time. Each button on the device is labeled with both Braille and normal text, ensuring its usability for everyone.

By integrating these features, the project aims to make education more accessible for deaf-blind individuals, fostering a more inclusive learning environment and empowering these individuals to lead more independent and fulfilling lives.

1.5 Existing models

1.5.1 For Deaf People

- AI-based solutions for Deaf people: This project by MIT Solve trains machines with artificial intelligence to do sign language, improving the experience of communicating with people who don't know sign language. Another example is LetsTalkSign, developed by an Indian startup called DeepVisionTech.AI, which is an AI-powered, device-agnostic solution that enables easy two-way communication for people with hearing/speech impairment.
- FEELIT: This project promotes social and work inclusion for deaf and hard-of-hearing people by creating a training curriculum for them to become travel agents. The consortium will create an online community where users can share knowledge and feedback. It is also developing a VR game for hearing persons to understand what it's like to be a visitor with hearing impairments.
- Deaf Enabled Foundation: This foundation organizes various activities such as Leadership Training, Personality Development programmes and Seminars, Interpreting Services, Sign Language Seminars and Deaf Education Methodology Seminars for Schools, HIV Awareness Workshops in urban areas, Deaf Women Empowerment, Cultural programmes, Matrimonial services and Counseling for family members of the Deaf.

1.5.2 For Blind People

- Technology for the Blind: There are several technologies developed for the blind, such as a ring-like device that can recognize text and read it aloud, a touchscreen capable of creating figures and Braille, and 3D printing of children's books.
- Smart Tech Developments: Microsoft has developed several technologies for people who are blind or have low vision, such as Seeing AI, an app designed to help people with low vision or who are blind, and Microsoft Soundscape, which builds a detailed audio map that relates what's taking place around a person with visual impairment.
- Activities for Blind and Low-Vision People: DIY sensory bags, stress balls, blankets, and scrapbooks are some activities they can do or even a garden terrarium.

CHAPTER 2
LITERATURE SURVEY

2.1 Introduction

Visually impaired and deaf students face unique educational challenges that often necessitate specialized support and accommodations. These challenges, if not properly understood and addressed, can hinder their academic progress and social integration. This literature survey delves into the educational problems encountered by visually impaired and deaf students. It scrutinizes existing research, identifies common themes, and discusses potential solutions aimed at promoting inclusive education.

2.2 Educational Challenges for Visually Impaired Students

Visually impaired students grapple with numerous obstacles in accessing educational materials and participating fully in classroom activities. Research indicates that these students often have insufficient access to learning materials in accessible formats, such as braille or large print. This is compounded by limited teacher training in inclusive teaching methods, which can result in these students not receiving the support they need. Social stigma and misconceptions about visual impairment can also contribute to their educational difficulties (Arditi, 2020).

Furthermore, visually impaired students often face challenges related to navigation within school environments. The lack of assistive technologies, such as screen readers or magnification software, can further hinder their academic progress (Hersh, Johnson, & Hirsh, 2021). To address these challenges, strategies such as providing braille materials, implementing tactile graphics, and promoting inclusive classroom environments have been proposed (Reinhard & Spaulding, 2019).

2.3 Educational Challenges for Deaf Students

Deaf students encounter distinct educational challenges, primarily related to communication barriers and access to information. Limited access to sign language interpretation services can result in these students missing out on crucial classroom discussions. Inadequate teacher training in deaf education and a lack of awareness about deaf culture can further contribute to the educational disparities faced by deaf students (Marschark & Spencer, 2020).

Moreover, the absence of appropriate assistive technologies, such as hearing aids or cochlear implants, and insufficient support for developing literacy skills can exacerbate these challenges (Hauser, O'Hearn, McKee, & Steider, 2017). To address these issues, interventions such as providing sign language interpreters, utilizing captioning services, and fostering deaf-friendly learning environments have been proposed (Marschark & Knoors, 2012).

2.4 Intersectionality and Additional Considerations

It is crucial to recognize the intersectionality of challenges faced by students who are both visually impaired and deaf. These individuals often encounter compounded barriers that require multifaceted interventions (Fernandes & Fox, 2018). Moreover, considerations such as cultural and linguistic diversity within these communities must be taken into account when designing educational interventions (Tran & Nguyen, 2020). Collaborative efforts involving educators, policymakers, families, and disability advocates are essential for addressing the complex educational needs of visually impaired and deaf students (Ferrell, 2019). These efforts should aim to create an inclusive and supportive educational environment where all students, regardless of their sensory abilities, can thrive.

CHAPTER 3
THEORETICAL BACKGROUND

3.1 Computer Vision

3.1.1 What is Computer Vision

Computer vision is a field of computer science that focuses on enabling computers to identify and understand objects and people in images and videos. Like other types of AI, computer vision seeks to perform and automate tasks that replicate human capabilities. In this case, computer vision seeks to replicate both the way humans see, and the way humans make sense of what they see.

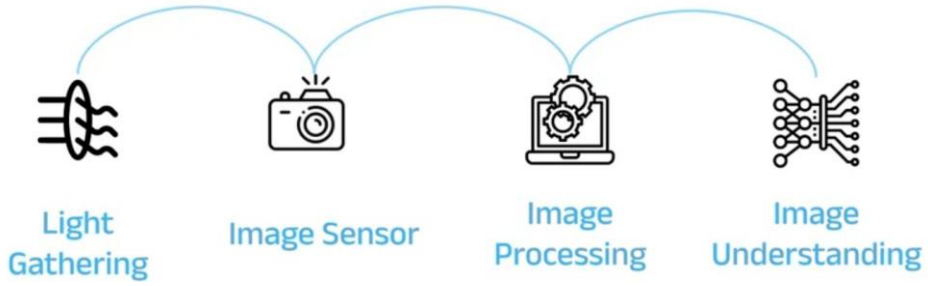


Figure 3.1.1: Process of Vision for Computers

The range of practical applications for computer vision technology makes it a central component of many modern innovations and solutions. Computer vision can be run in the cloud or on premises.

3.1.2 How computer vision works

Computer vision applications use input from sensing devices, artificial intelligence, machine learning, and deep learning to replicate the way the human vision system works. Computer vision applications run on algorithms that are trained on massive amounts of visual data or images in the cloud. They recognize patterns in this visual data and use those patterns to determine the content of other images.

3.1.3 Deep learning V/S computer vision

Modern computer vision applications are shifting away from statistical methods for analyzing images and increasingly relying on what is known as deep learning. With deep learning, a computer vision application runs on a type of algorithm called a neural network, which allows it deliver even more accurate analyses of images.

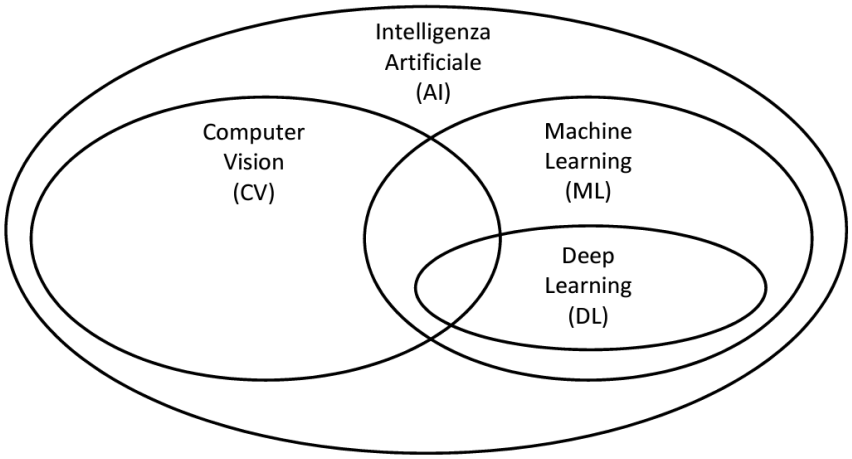


Figure 3.1.2: Relation between Computer Vision and Deep Learning

In addition, deep learning allows a computer vision program to retain the information from each image it analyzes—so it gets more and more accurate the more it is used.

3.2 OpenCV

3.2.1 What is OpenCV

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV. This OpenCV tutorial will help you learn the Image-processing from Basics to Advance, like operations on Images, Videos using a huge set of Opencv-programs and projects.

3.2.2 OpenCV Functionalities

- Image/Video and Processing: OpenCV can read, write, & manipulate images and videos.
- Object/Feature Detection: It can identify objects and features within images.
- Geometry-based Computer Vision: OpenCV understands the 3D structure from 2D images and performs tasks like camera calibration and video stabilization.
- Computational Photography: It enhances images and videos using advanced techniques.
- Machine Learning & Clustering: OpenCV includes machine learning algorithms for tasks like object classification and data clustering.
- CUDA Acceleration: It utilizes NVIDIA's CUDA architecture for faster processing of certain operations.

3.3 Image Processing

3.3.1 What Is an Image

An image is a two-dimensional function, denoted as F(x, y), where x and y are spatial coordinates. The intensity of the image at any point (x, y) represents its amplitude.

When x, y, and intensity values are finite, we call it a digital image.

A digital image is represented by a two-dimensional array arranged in rows and columns, with each element called a pixel.

3.3.2 Types of Images

- Binary Image: Contains only two-pixel values (0 and 1), representing black and white.
- Black and White Image: Consists of only black and white colors.

3.3.3 What is Image Processing

Image processing is the process of transforming an image into a digital form and performing certain operations to extract useful information from it. Image processing involves manipulating digital images using algorithms and mathematical models. Its primary goals are to enhance image quality, extract meaningful information, and automate image-based tasks.

3.3.4 Basic Steps in Digital Image Processing

- Image Acquisition: Capturing an image using a digital camera, scanner, or importing an existing image into a computer.
- Image Enhancement: Improving visual quality by adjusting contrast, reducing noise, and removing artifacts.

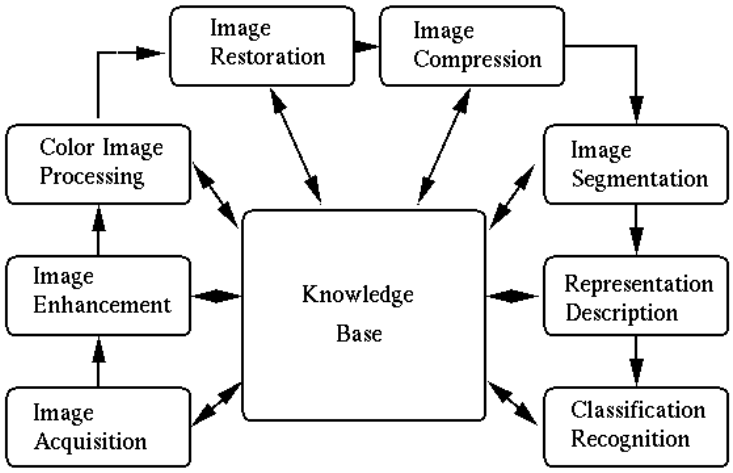


Figure 3.3: Fundamental Steps in Digital Image Processing

- Image Restoration: Removing degradation (e.g., blurring, noise) from an image.
- Image Segmentation: Dividing an image into regions corresponding to specific objects or features.

- Image Representation and Description: Representing an image for computer analysis and describing its features concisely.
- Image Analysis: Extracting information from an image, such as object recognition and feature quantification.
- Image Synthesis and Compression: Generating new images or compressing existing ones for storage and transmission.

3.4 Optical Character Recognition (OCR)

3.4.1 What is OCR

Optical Character Recognition (OCR) is a field of machine learning that specializes in recognizing characters within images like scanned documents, printed books, or photos. It's also referred to as text recognition or text extraction. OCR uses machine-learning-based techniques to extract printed or handwritten text from images such as posters, street signs, and product labels, as well as from documents like articles, reports, forms, and invoices. The text is typically extracted as words, text lines, and paragraphs or text blocks, enabling access to a digital version of the scanned text.

3.4.2 Why OCR

Optical character recognition (OCR) technology is a business solution for automating data extraction from printed or written text from a scanned document or image file and then converting the text into a machine-readable form to be used for data processing like editing or searching.

3.4.3 What is Tesseract OCR

Tesseract OCR is an optical character reading engine developed by HP laboratories in 1985 and open sourced in 2005. Since 2006 it is developed by Google.

Tesseract has Unicode (UTF-8) support and can recognize more than 100 languages "out of the box" and thus can be used for building different language scanning software also. Latest Tesseract version is Tesseract 4.

It adds a new neural net (LSTM) based OCR engine which is focused on line recognition but also still supports the legacy Tesseract OCR engine which works by recognizing character patterns.

3.4.4 How OCR works

- Preprocessing of the Image: This is the first step in the OCR process. The goal of preprocessing is to improve the image quality by reducing noise and simplifying the image.

This can involve several operations such as binarization (converting the image to black and white), noise removal, skew correction, and scaling.

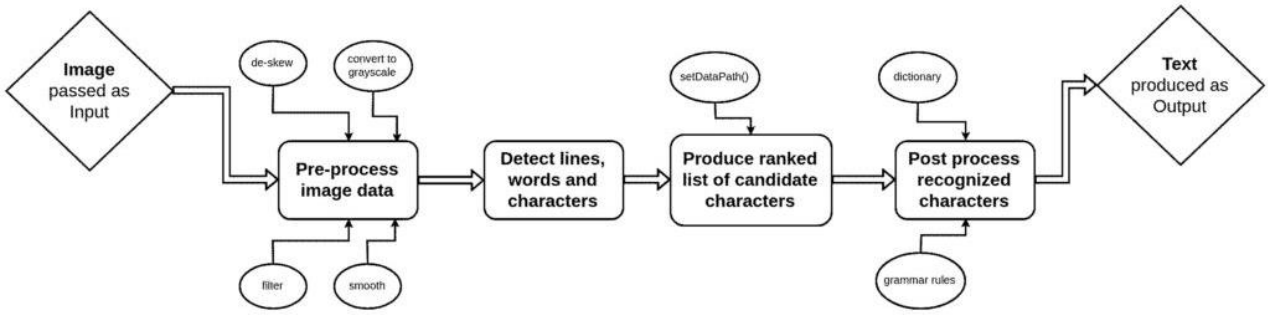


Figure 3.4: General Working of OCR

- Text Localization: This step involves identifying and isolating blocks of text in the image. Advanced techniques for text localization can handle a variety of challenges such as different fonts, sizes, and orientations.
- Character Segmentation: Once the blocks of text have been identified, the next step is to separate the individual characters. This can be a challenging task, especially with different writing styles and fonts.
- Character Recognition: This is the core of OCR. In this step, each individual character is recognized and converted into an ASCII equivalent. This is typically done using machine learning techniques, where a model is trained on a large dataset of characters and then used to recognize individual characters.
- Post Processing: The final step in the OCR process is post-processing. This can involve checking the recognized text against a dictionary to correct any recognition errors, and formatting the output text.

3.5 OCR pytesseract

3.5.1 What is Pytesseract

Pytesseract is a Python wrapper for Google's Tesseract-OCR Engine. It recognizes and reads the text embedded in images. Pytesseract is a widely-used Optical Character Recognition (OCR) library for Python applications. Its primary role is to extract text from images and documents, making it accessible and usable for various text analysis and data processing tasks. Tesseract pre-processes the input image first in order to improve its quality. After that, it examines the page's arrangement/orientation to determine text blocks, paragraphs, and characters. It is also useful as a stand-alone invocation script to tesseract, as it can read all

image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.

3.5.2 Three-stage process used in Tesseract

- Text Detection: The first step is text detection in the image. Tesseract uses OpenCV's EAST text detection model to detect the presence of text in an image. This is a crucial step as it identifies all the possible areas in the image where the text could be present.
- Text Localization: Once the text is detected, the next step is to extract the text Region of Interest (ROI) from the image. This is done using basic image cropping/NumPy array slicing. The goal here is to isolate each word or character for the next step.
- Character Recognition (OCR): After the text regions have been localized and extracted, these regions are then passed into Tesseract to OCR the text. This is the final step where Tesseract recognizes and reads the text embedded in the images.

3.6 easyOCR

3.6.1 What is easyOCR

easyOCR is a Python computer language Optical Character Recognition (OCR) module that is both flexible and easy to use. It is a general OCR that can read both natural scene text and dense text in document. OCR technology is useful for a variety of tasks, including data entry automation and image analysis. It enables computers to identify and extract text from photographs or scanned documents.

easyOCR optical character recognition library reads short texts (such as serial numbers, part numbers, and dates). It uses font files (pre-defined OCR-A, OCR-B and Semi standard fonts, or other learned fonts) with a template matching algorithm that can recognize even badly printed, broken, or connected characters of any size.

easyOCR is created by the company named Jaided AI company. It supports 42+ languages for detection purposes. It is a python package that holds PyTorch as a backend handler.

3.6.2 easyOCR over Pytesseract

- EasyOCR and Pytesseract are both Optical Character Recognition (OCR) tools. OCR is a technology used to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera, into editable and searchable data.
- EasyOCR is a ready-to-use OCR with 80+ supported languages and all popular writing scripts including Latin, Chinese, Arabic, Devanagari, Cyrillic, etc. It's a Python library that uses deep learning for OCR. It's designed to handle short texts, such as serial numbers, part numbers, and dates.

- On the other hand, Pytesseract is a Python wrapper for Google's Tesseract-OCR Engine. It is used to recognize and read the text embedded in images. Pytesseract is useful as a stand-alone invocation script to tesseract, as it can read all image types supported by the Pillow and Leptonica imaging libraries, including jpeg, png, gif, bmp, tiff, and others.

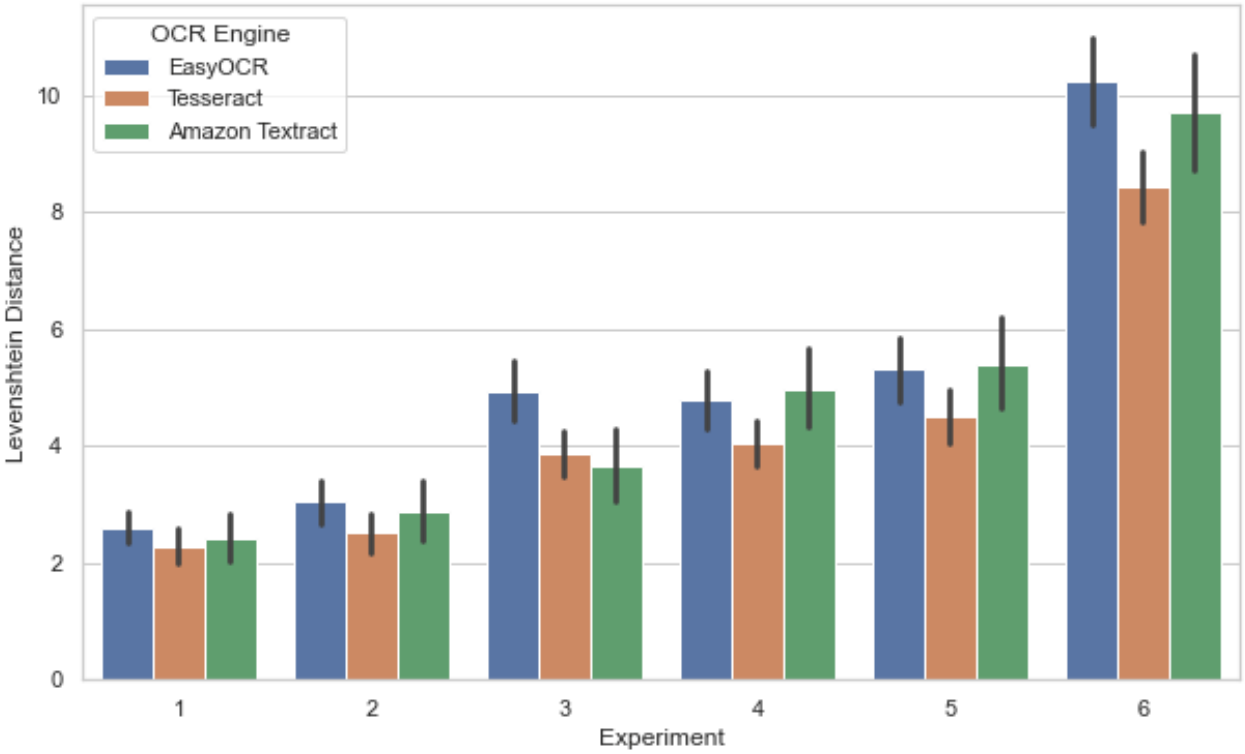


Figure 3.6: Statistics between Various OCR Engines over Experimentation

- The choice between EasyOCR and Pytesseract depends on your specific use case. If your document is alphabet-heavy, you may give Pytesseract higher weights. On the other hand, if your document contains a lot of numbers, you may favor EasyOCR. If you want the most accurate results, a hybrid process may be considered.

3.7 Voice Assistance

3.7.1 What is pyttsx3

Pyttsx3 is a Python library designed for text-to-speech conversion. Unlike some other libraries, Pyttsx3 operates entirely offline, making it convenient for various applications where internet connectivity might be limited or unavailable. Moreover, it's compatible with both Python 2 and Python 3, ensuring broad usability across different versions of the language.

3.7.2 How pyttsx3 works

The pyttsx3.init(driverName=None, debug=False) method is used to initialize the text-to-speech engine. It takes two optional parameters: driverName, which is the name of the requested driver (if not provided, it selects a driver that works on the current platform), and debug, which, if set to True, prints debug information to stderr. The default value is False.

Once the engine is initialized, you can add text to be spoken using the engine.say(text, name=None) method. This method takes the text to be spoken and an optional name to associate with this speech command.

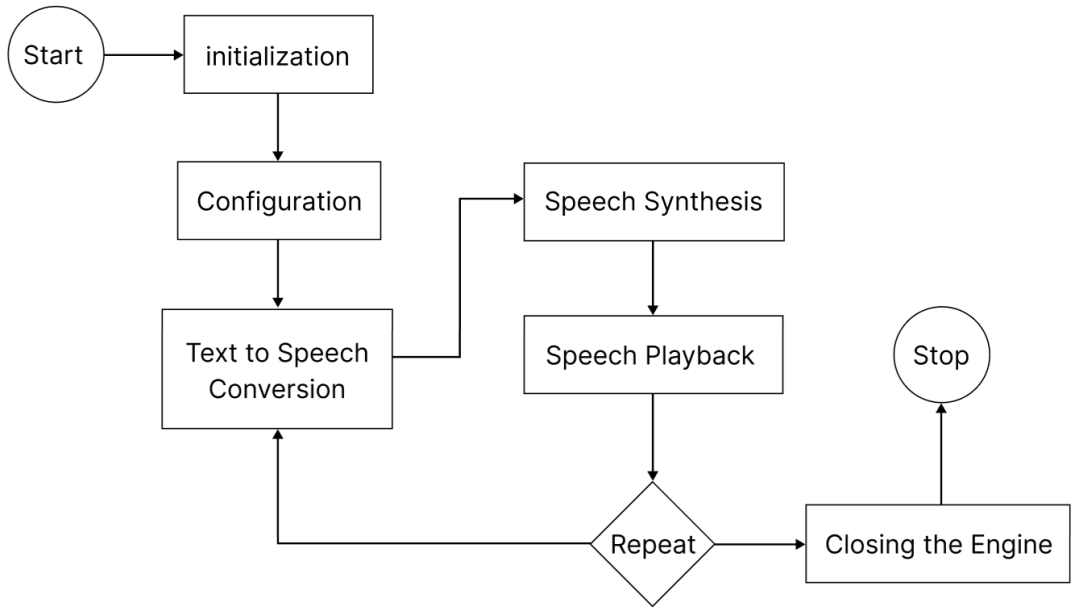


Figure 3.7: Basic Work Flow of Pyttsx3 module

After adding text to the queue, you need to tell the engine to start speaking. This is done using the engine.runAndWait() method, which processes all currently queued commands and blocks until all commands queued before this call are emptied from the queue.

The pyttsx3 module also allows you to customize the speech output. You can get the current value of an engine property using the engine.getProperty(name) method, where name is the name of the property (‘rate’, ‘volume’, or ‘voices’). You can set the value of an engine property using the engine.setProperty(name, value) method, where name is the name of the property (‘rate’, ‘volume’, or ‘voice’), and value is the new value for the property.

To stop the engine from speaking using the engine.stop() method. This method empties the command queue and stops the current utterance.

CHAPTER 4
APPROACH DESCRIPTION

4.1 Text Capturing

4.1.1 Image Acquisition

The process commences with the camera being activated, allowing a continuous stream of images to appear on the graphical user interface (GUI). Upon user interaction, the system captures the current image, marking the initiation of further processing. This real-time image acquisition ensures to provid up-to-date visual data for analysis and interpretation.

4.1.2 Text Extraction using EasyOCR

Following image capture, the EasyOCR reader extracts text from the captured image. Through iterative analysis, the system identifies the predominant language within the text, laying the foundation for subsequent operations. EasyOCR's robust text extraction capabilities ensure accurate retrieval of textual content, regardless of variations in fonts, sizes, or backgrounds, thus enhancing the overall reliability of the system.

4.1.3 Translation of Text

Upon user request, the recognized text can be translated into the desired language using the googletrans module Translator. This functionality enhances accessibility and facilitates cross-lingual communication. The Translator will return an object that contains src (the source language), dest (the destination language), origin (the original text), and text (the translated text).

4.1.4 Summarization of Text

In cases where the text exceeds a predefined length threshold, an automatic summarization process is initiated. The Prompt Template is a feature of the LangChain library that allows you to create a template for generating prompts for language models. Where we gave input_variables i.e., Text and Language for the Prompt Template, and the summarized text was returned to the further processing.

4.1.5 Audio Generation

Subsequently, whether translated or summarized, the processed text is transformed into audible speech using the gTTS module. This enables users to access content through auditory means, enhancing usability and accessibility. By converting text into speech, gTTS enables individuals with visual impairments or those preferring audio-based content consumption to access the information effortlessly, thereby promoting inclusivity and user satisfaction.

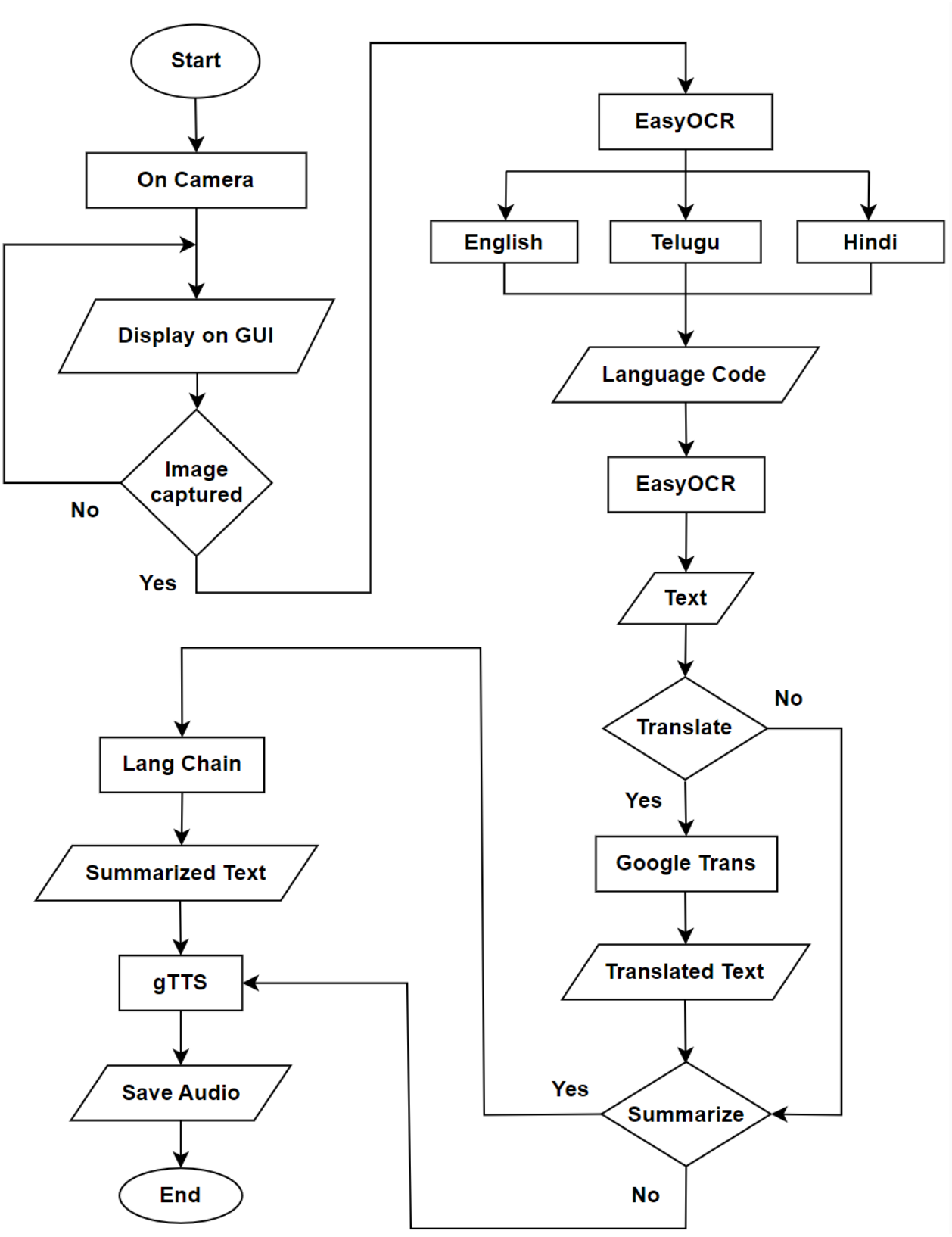


Figure 4.1: Flow Diagram for Audio to Text Capturing

4.2 Voice Capturing

4.2.1 Initiating the Process

The process begins when the microphone is turned on, prompting the PyAudio Python module to segment the incoming audio stream. Each segment is processed to form a unified frame. PyAudio's algorithms handle this stream manipulation, ensuring the capture of the spoken content effectively.

4.2.2 Transcription and Analysis

Next, the SpeechRecognition module takes over, employing advanced algorithms to transcribe the audio into written text. This transcription continues until the microphone is turned off, ensuring all spoken content is captured accurately.

4.2.3 Multilingual Enhancement

The transcribed text undergoes further processing with Googletrans Translator, allowing for translation into different languages. This feature ensures inclusivity and accessibility, as the text can be understood by speakers of various languages.

4.2.4 Integration into the GUI

Finally, the transcribed text is displayed in a designated area of the Graphical User Interface (GUI), providing users with easy access. This user-friendly interface promotes seamless interaction and understanding of the transcribed content, facilitating effective communication.

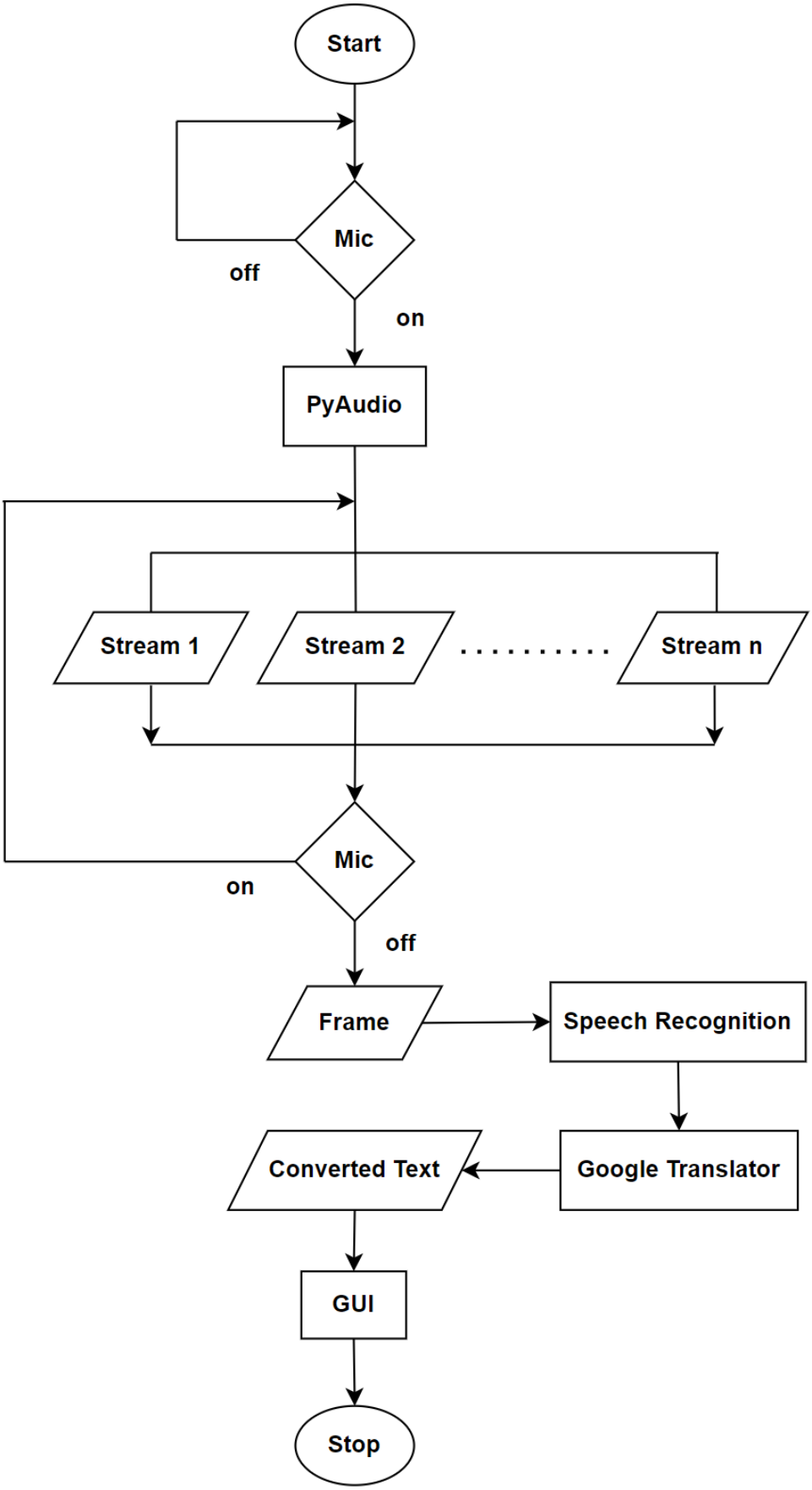


Figure 4.2: Flow Diagram for Text Capturing from Images

CHAPTER 5
TEXT CAPTURING

5.1 Live Image Capturing

5.1.1 What is PIL

The Python Imaging Library (PIL) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different images file formats. Image and ImageTk are two modules from PIL (or Pillow, which is a fork of PIL with user-friendly features).

- Image: This module provides a class with the same name which is used to represent a PIL image. The module also provides many functions to manipulate images. Image.fromarray() is used to convert raw pixel data into a PIL Image object. It is necessary because the video frames captured by OpenCV are in a raw format that cannot be directly used with Tkinter.
- ImageTk: This module contains support to create and modify Tkinter BitmapImage and PhotoImage objects. ImageTk.PhotoImage() is used to convert the PIL Image object into a format that can be used as a photo image with Tkinter. This is necessary because Tkinter's PhotoImage class does not support the raw image format used by OpenCV.

5.1.2 Process of Image Capturing

The application starts by capturing video from the default camera. Each frame of the video is captured one by one. Each captured frame is an image, and it's initially in a format that's not compatible with Tkinter, the library used to create the graphical user interface. So, the frame needs to be converted into a format that Tkinter can work with. This is where PIL comes in.

The captured frame is first converted from one color space to another. This is because the format used by the video capture source and the format required by Tkinter are different. Once the frame is in the correct color space, it's converted into a format that can be used with Tkinter.

The converted frame, which is now an image in a format that Tkinter can work with, is then displayed on a canvas in the application window. This process repeats for each frame in the video, giving the illusion of a continuous video stream.

This process repeats every 10 milliseconds, giving the illusion of a continuous video stream. The application is designed to stop processing if it fails to read a frame, which could happen if the video ends or there's an issue with the video source. It also ensures that the image object used to display the frame on the canvas is not prematurely deleted.

5.2 Text Capturing

5.2.1 What is easyOCR

EasyOCR is a Python library that provides an easy-to-use interface for Optical Character Recognition (OCR). OCR is a technology used to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera, into editable and searchable data.

EasyOCR stands out for its support of more than 80 languages and various writing scripts including Latin, Chinese, Arabic, Devanagari, Cyrillic, and others. This makes it a versatile tool for multilingual OCR tasks.

The library works by taking an image file or an image path as input and returns a list of bounding box locations and recognized text along with their confidence levels. This can be useful in a variety of applications, such as automating data entry processes, digitizing printed documents, or developing assistive technologies for visually impaired individuals.

EasyOCR is also integrated into Huggingface Spaces using Gradio, which allows you to try out its capabilities through a web demo.

5.2.2 Language Identification

The application is designed to determine the language of the text present in an image. It does this by using the EasyOCR library to recognize text in the image for a set of predefined languages. For each language, it calculates an average confidence score based on the recognition results.

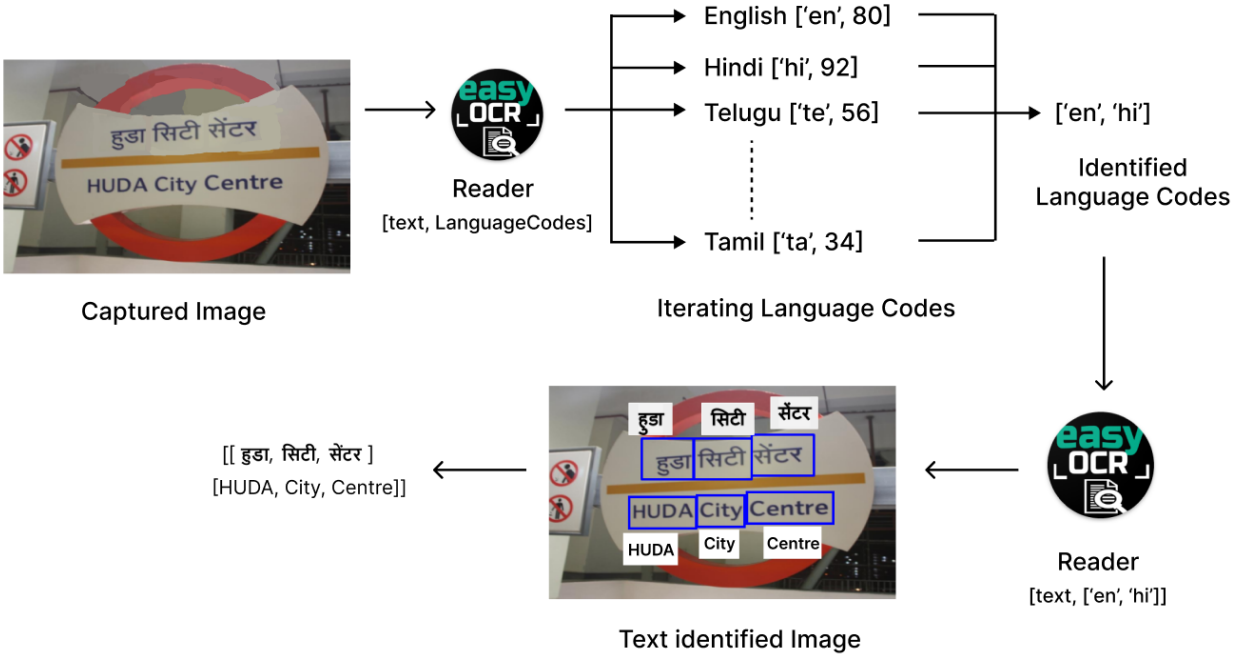


Figure 5.2: Text identification Using easyOCR

The language with the highest average confidence score is considered the language of the text. If the identified language is not English, it returns a list containing both the English language code and the identified language code. If no text was identified in the image, the function returns False. This process allows the script to handle text in multiple languages.

5.3 Text Translation

5.3.1 What is googletrans

The googletrans library is a free and unlimited Python interface to Google Translate's AJAX API. It allows developers to implement language translation features in their Python applications. The library can automatically detect the language of the input text, translate a batch of strings in a single method call, and supports HTTP/2. It uses the same servers that Google Translate uses, making it fast and reliable.

The main functions provided by googletrans are translate for translating text and detect for detecting the language of a given text. The translate method is used to translate text from one language to another, and if the source language is not given, Google Translate attempts to detect the source language.

5.3.2 Prompting the required Language

After identifying the language of the text in the image, it uses an OCR reader to extract the text from the image. The application speak out the identified language and prompts the user to translate the text into any other language. The text, original text language and required translated language were send to the translate method in googletrans.Translator class.

5.3.3 Translate method

It is used for translating text from one language to another. It accepts several parameters including the text to be translated, the source language (src), the destination language (dest), and optional parameters like service_urls, user_agent, and proxies. If the source language is not specified, the method automatically detects it.

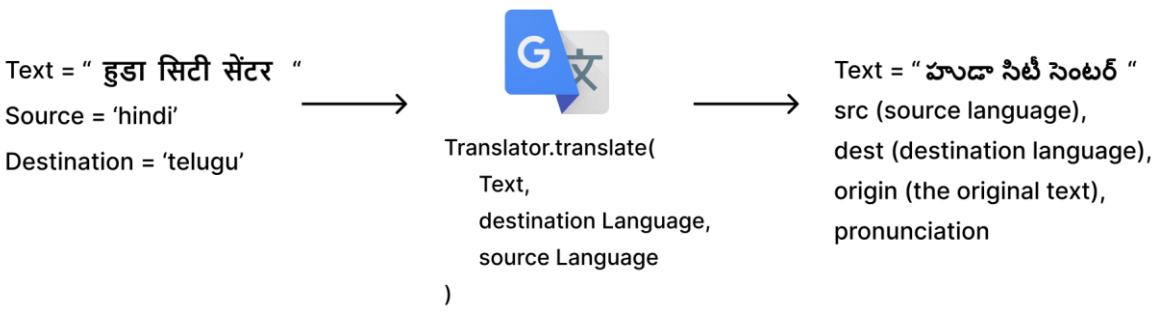


Figure 5.3: Text Translation Using googletrans

The method sends a request to the Google Translate servers with the text and the languages, and the servers return the translated text. The method can translate a single string or a list of strings for bulk translations.

The translate method returns a Translated object, which includes the original text, the translated text, the source language, and the destination language. The Translated object has several attributes including src (the source language), dest (the destination language), origin (the original text), text (the translated text), and pronunciation (the pronunciation of the translated text).

5.4 Summarization

5.4.1 what is LLM

Large Language Models (LLMs) are a type of artificial intelligence algorithm that applies neural network techniques with lots of parameters to process and understand human languages or text. They are highly efficient in capturing the complex entity relationships in the text at hand and can generate the text using the semantic and syntactic of that language in which we wish to do so. Examples of such LLM models are Chat GPT by OpenAI, BERT (Bidirectional Encoder Representations from Transformers) by Google, etc.

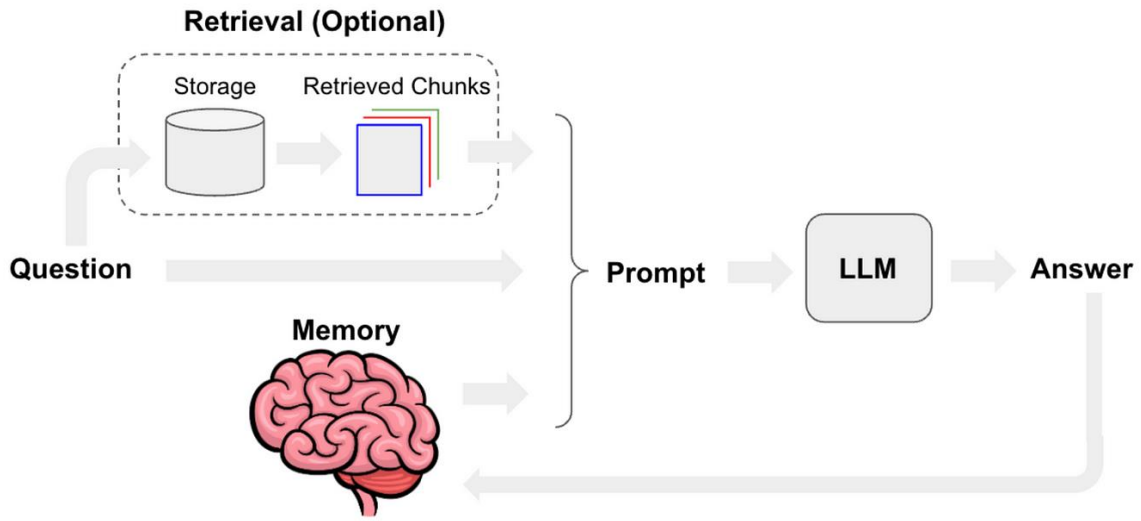


Figure 5.4.1: Working of LLM

LLMs operate on the principles of deep learning, leveraging neural network architectures to process and understand human languages. These models are trained on vast datasets using self-supervised learning techniques. The core of their functionality lies in the intricate patterns and relationships they learn from diverse language data during training.

5.4.2 What is Lang Chain

LangChain is an open-source framework that simplifies the creation of applications using Large Language Models (LLMs). LangChain is related to OpenAI as it provides integrations for OpenAI through their SDK, offering a convenient way to interact with OpenAI models. It provides a standard interface for chains, integrations with other tools, and end-to-end chains for common applications. It includes components like LLM Wrappers, Prompt Template, and Indexes for relevant information retrieval.

5.4.3 OpenAI Model

The OpenAI model we used is an instance of the OpenAI's GPT-3.5 Turbo model, which is a powerful language model capable of understanding and generating natural language text. The model is initialized with specific parameters including temperature, openai_api_key, and model. The temperature parameter controls the randomness of the model's output, with lower values making the output more deterministic. The openai_api_key is your unique key for accessing the OpenAI API. The model parameter specifies the particular model you want to use, in this case, "gpt-3.5-turbo-instruct".

5.4.4 Prompt Template

The PromptTemplate is a feature of the LangChain library that allows you to create a template for generating prompts for language models. The template is a string that can contain placeholders, which will be replaced with actual values when the template is formatted. The PromptTemplate is initialized with a template and input_variables. The template is a string that defines the structure of the prompt and can contain placeholders.

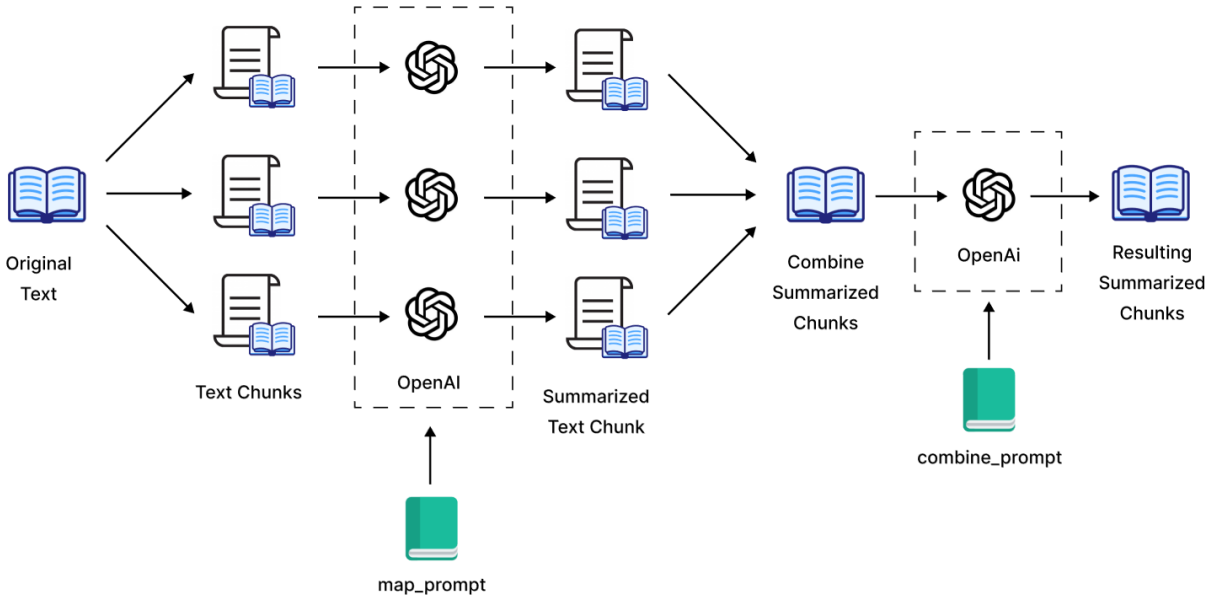


Figure 5.4.2: Work Flow of Summarization of Text

We had defined a template string "give the text in {x} by summarizing the below text, within 50 to 60 words. text: {y}.". This template string contains two placeholders, {x}, and {y}, which are intended to be replaced with a language and a text respectively. The input_variables parameter is a list of these placeholder names. When you call prompt.format(x=lang, y=text), it replaces {x} with the value of lang and {y} with the value of text, creating a prompt that requests a summary of the given text in the specified language. This formatted prompt is then passed to the model, an instance of the OpenAI model, which generates a summarized version of the text in the specified language. This approach allows for dynamic control of the model's behavior based on the specific requirements of your application.

5.5 Vocalization of Text

5.5.1 What is GTTS

The gTTS (Google Text-to-Speech) library in Python interfaces with Google Translate's text-to-speech API, allowing you to convert text into speech in a variety of languages. The library also provides customizable text pre-processors for pronunciation corrections.

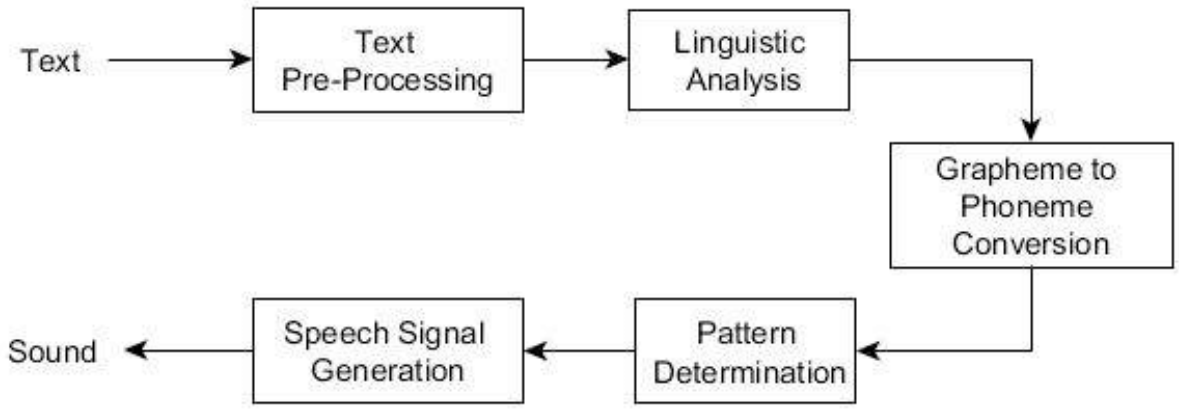


Figure 5.5: gTTS converting Text to Speech process

The main function provided by gTTS is gTTS(), which takes a string of text and converts it into speech. The speech can then be saved as an MP3 file. While this library is free and unlimited, it could be blocked at any time since it uses the Google Translate speech functionality and is different from Google Cloud Text-to-Speech.

5.5.2 Speech Conversion

The gTTS function is a key component of the Google Text-to-Speech (gTTS) library in Python. It interfaces with Google's Text-to-Speech API and is used to convert text into speech. The

function accepts several parameters including the text to be converted into speech, the language in which the text should be read, and the speed of the speech.

If the slow parameter is set to True, the speech will be slower. The function also allows for automatic language detection if the language is not specified.

5.5.3 Saving the Audio

Upon calling the gTTS function with the appropriate parameters, it returns an instance of the gTTS class. This instance represents the speech audio for the given text. The speech audio can then be saved as an MP3 file using the save method of the gTTS instance. For example, if speech is an instance of the gTTS class, you can save the speech audio into an MP3 file named ‘output.mp3’ using speech.save('output.mp3'). Once the audio file is saved, it can be played using any standard audio playback software or library. In this way, the gTTS function provides a simple and effective way to convert text into speech.

CHAPTER 6

VOICE CAPTURING

6.1 Multi-Threading

6.1.1 What is Threading

Threading is a technique in programming where a single set of code can be used by several processors at different stages of execution. This allows the program to perform multiple operations concurrently, making optimal use of available processing power and enabling tasks to run in the background while the main program continues to run.

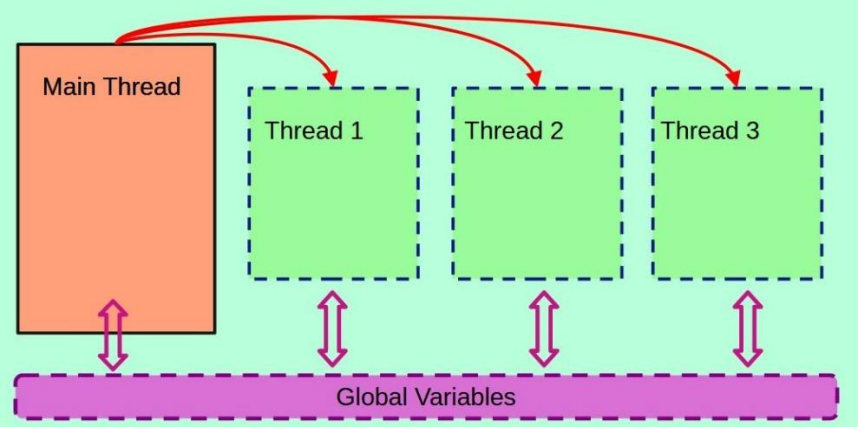


Figure 6.1: Multi-threading of a process

In Python, threading can be implemented using the threading module, which provides a way to create and manage threads. For I/O-bound tasks or for improving responsiveness in GUI applications, threading can be very beneficial.

6.1.2 Problems encountered without using Threads

- Blocking Main Thread: The main thread of the program would be blocked while the recording is happening. This means that the program would not be able to do anything else, like updating the user interface or responding to user input, until the recording is finished.
- Lack of Responsiveness: Your application could become unresponsive. For instance, if you are recording audio in a GUI application, the GUI could freeze and not respond to user interactions like button clicks.
- Difficulty in Control: Controlling the start and stop of the recording could become more complex. With threading, you can easily start the recording in one thread and stop it from another (like from a main thread responding to a user's stop command).
- Performance Issues: In some cases, not using threads could lead to performance issues. If your application has other tasks that can be done in parallel with the recording, then not

using threads could prevent these tasks from being done concurrently, which could slow down your application.

- Design Limitations: Not using threads could limit the design of your application. For example, if you want to add features that require concurrent execution (like real-time audio processing), then not using threads could make these features difficult or impossible to implement.

6.2 Pyaudio

6.2.1 What is Pyaudio

PyAudio is a Python library that provides bindings for PortAudio, a cross-platform audio I/O library. It allows developers to record and play audio across multiple platforms. PyAudio is also open source and distributed under the MIT License, making it free to use and modify.

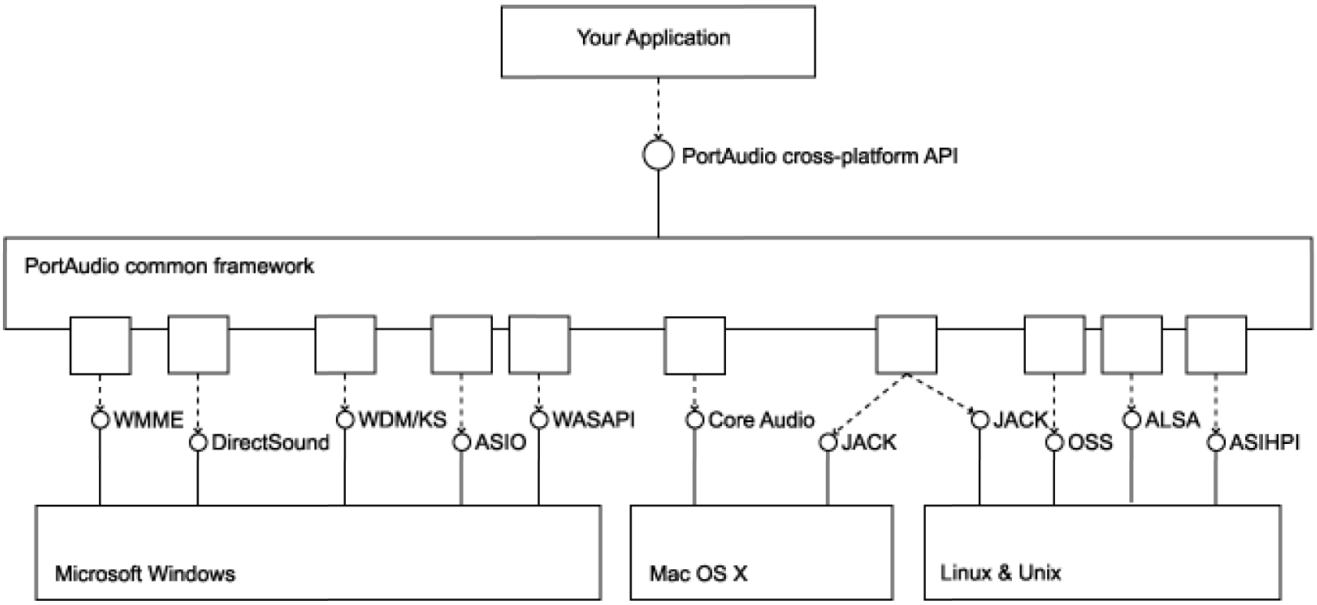


Figure 6.2: Pyaudio Supported Operating Systems

Using PyAudio involves creating a new PyAudio instance, opening an audio input or output stream with the desired properties, reading from or writing to the stream, and then closing the stream to free up system resources.

6.2.2 Recording the Audio Stream

Audio recording process, managing the capture, storage, and initial processing of the audio data is designed to run in a separate thread, allowing the main program to remain responsive and control the recording process as needed.

Setting up the audio stream: An instance of pyaudio.PyAudio is created. This instance is used to open an audio input stream with the open method. The stream is set up with the specified format, channels, and rate as follows:

- format: This is the format of the audio data. pyaudio.paInt16 is a common format that represents audio data as 16-bit integers.
- channels: This specifies the number of channels to use. 1 means mono sound, while 2 would mean stereo sound.
- rate: This is the sample rate, which is the number of samples of audio carried per second. A common value is 44100 Hz, which is also the standard for audio CDs. A higher sample rate can capture more detail in the audio signal, but also requires more data to store.
- input: This is set to True to indicate that the stream is for input (recording). If it were output (for playing audio), this would be False.
- frames_per_buffer: This specifies the number of frames per buffer. The "buffer" here refers to a temporary storage area in memory where audio data is kept before it is processed. Each 'chunk' of audio data that the program reads from the input stream will contain this many frames. A smaller chunk size means the program reads data more frequently, which can make the recording smoother but might increase CPU usage.

At this point, the audio stream is open and ready to receive audio data from the microphone. The data can be read from the stream in chunks (or "frames") and processed as needed.

6.3 Audio formatting

6.3.1 What is Wave Library

The wave module in Python is an interface to the Waveform Audio File Format (WAVE), commonly known as WAV due to its filename extension, which is a common format for storing audio data on PCs. The module supports audio parameters such as the number of channels, sample width (bit depth), frame rate (sample rate), and number of frames. It supports only uncompressed Pulse-Code Modulation (PCM) encoded wave files, a standard form of digital audio in computers and digital communication systems. it is compatible with the Pyaudio library, making it easy to save the recorded audio data to a WAV file.

6.3.2 Saving the Audio Stream

The process begins by opening a new .wav file in write mode. on setting the audio parameters such as the number of channels, sample width, and frame rate for the .wav file, it writes the audio frames, which are passed as an argument, to the .wav file.

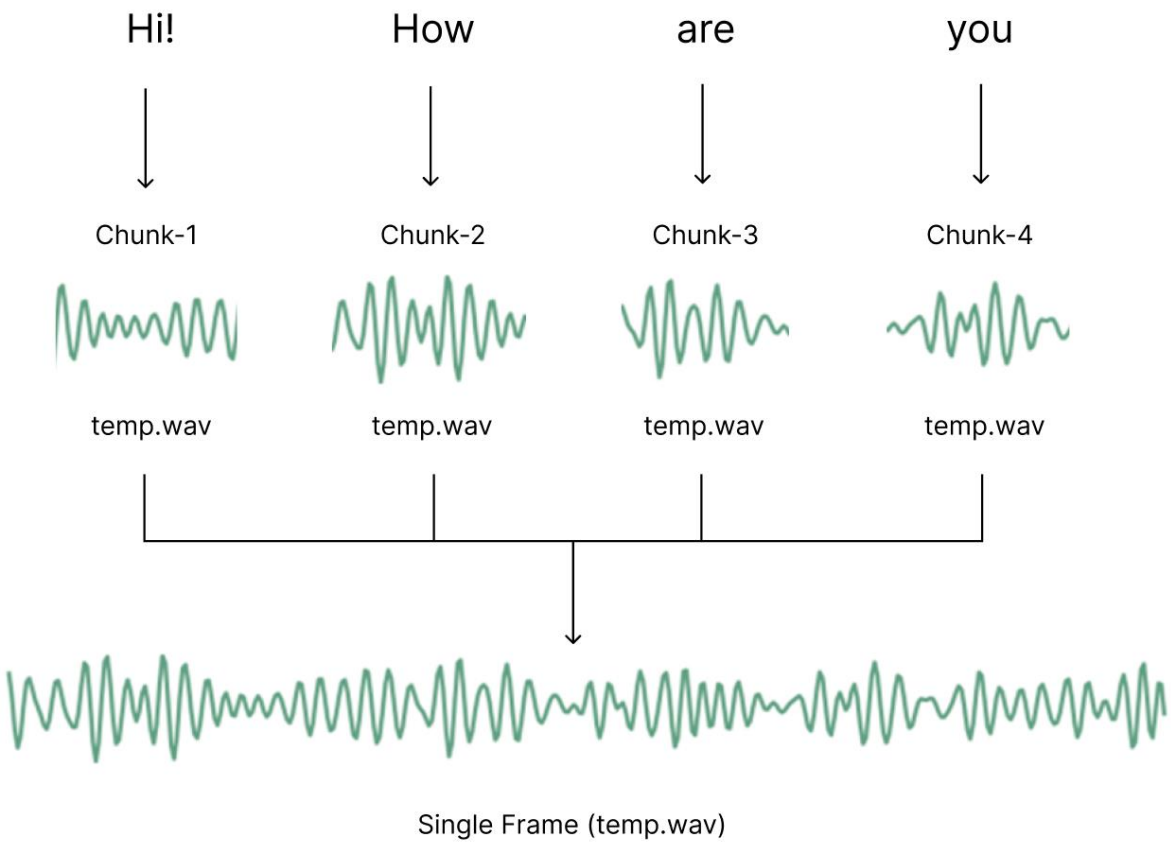


Figure 6.3: Capturing process of the Audio

These frames are the chunks of audio data recorded from the microphone. The frames are joined together into a single byte object before being written to the file. Once all the frames are written, the .wav file is closed.

6.4 Recognizing the Speech

6.4.1 What is speech_recognization module

The speech_recognition module in Python is a powerful library that provides the ability to convert spoken language into written text. It supports multiple speech recognition engines and APIs, both online and offline, including Google Speech Recognition, Google Cloud Speech API, Microsoft Bing Voice Recognition, and IBM Speech to Text. The module can process audio data from both audio files and microphones, and it has built-in capabilities for handling ambient noise. However, the accuracy of the speech recognition can be influenced by factors such as the quality of the input audio, background noise, and the speaker's accent.

6.4.2 Capturing Text

The recorded audio will be converted into text using the speech_recognition module in Python. It starts by opening the .wav file containing the recorded audio data and reads the entire file

into an AudioData object. This object is then passed to the recognize_google method of the Recognizer object, which sends the audio data to the Google Web Speech API for processing.

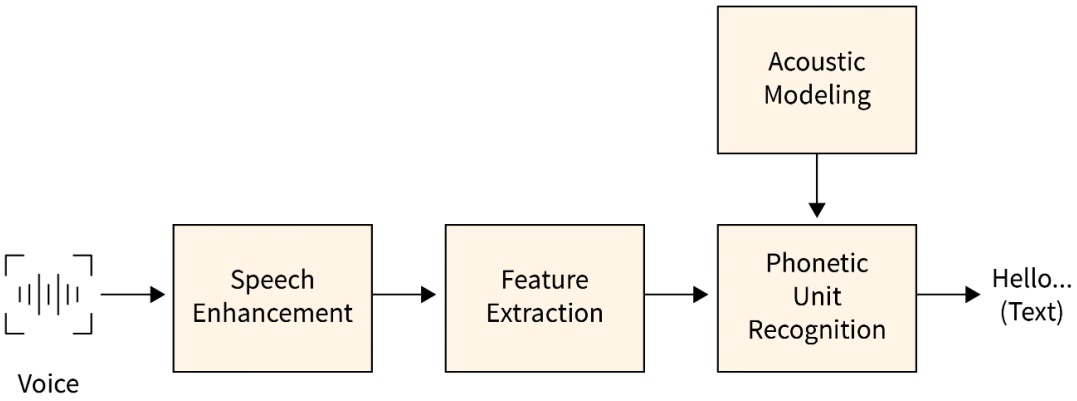


Figure 6.4: Flow Diagram of Speech Recognition

The API uses machine learning algorithms to convert the speech in the audio data into text. If the speech recognition is successful, the recognized text is printed on the console and displayed on a GUI label. If the audio cannot be understood or if there's an error connecting to the speech recognition service, appropriate error messages are displayed. The function concludes by deleting the temporary audio file.

CHAPTER 7
GRAPHICAL USER INTERFACE

7.1 What is GUI

A Graphical User Interface (GUI) is a type of user interface that allows users to interact with electronic devices through graphical icons and visual indicators. It's an alternative to text-based interfaces, like command-line interfaces (CLIs), which require commands to be typed on a keyboard. GUIs are more user-friendly, especially for those not familiar with the device's command language. In Python, GUIs can be created using libraries such as Tkinter, PyQt, or wxPython. These libraries provide tools to create various GUI elements like windows, buttons, menus, text boxes, etc. For instance, Tkinter, a standard Python interface to the Tk GUI toolkit, is widely used for developing GUIs in Python.

7.2 What is Tkinter

Tkinter is a standard Python interface to the Tk GUI toolkit and is the most used library for creating Graphical User Interfaces (GUIs) in Python. It allows developers to create windows, buttons, text boxes, and other GUI elements.

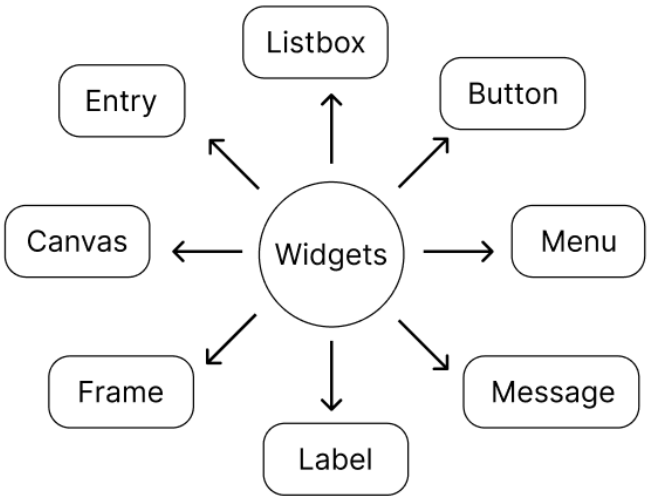


Figure 7.2: Python tkinter Widgets

You can create a GUI application by importing the Tkinter module, creating the main window, adding widgets to the window, applying event triggers on the widgets, and running the application using the mainloop() method. Tkinter also provides geometric configuration of the widgets, which can organize the widgets in the parent windows using geometry manager classes like pack(), grid(), and place().

7.3 Integrating with our Project

We have developed a prototype IoT device with a Graphical User Interface (GUI) designed to assist deaf-blind individuals for educational purposes. This device is built using Python, leveraging the Tkinter library for the GUI. The device is equipped with functionalities that capture text from images, translate and summarize it. This is achieved by integrating Python libraries such as Optical Character Recognition (OCR) for text extraction from images, and Natural Language Processing (NLP) libraries for translation and summarization.

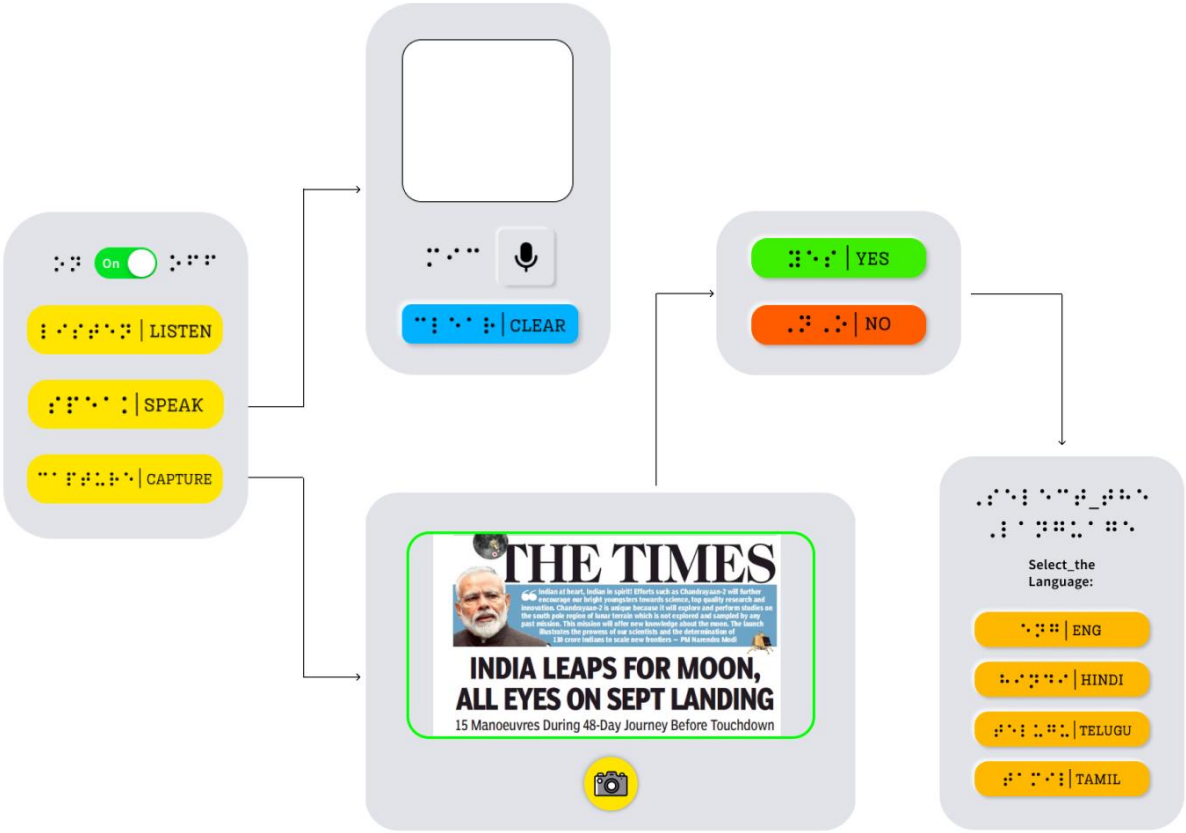


Figure 7.3: GUI elements in our Project

For individuals with hearing impairment, the device is designed to display speech in any language on the text area of the device's screen. This is accomplished by integrating Speech-to-Text algorithms in Python. Each button on the device is labeled with both Braille and standard text to ensure easy identification and accessibility.

CHAPTER 8
RESULTS AND CONCLUSIONS

8.1 Results

- Communication: The real-time transcription of speech and the ability to translate and summarize text from images can facilitate better understanding and interaction with the world around them.
- Real-Time Speech Transcription: The real-time speech transcription feature was assessed for its speed and accuracy in converting spoken language into text, proving reliable across various languages and aiding deaf-blind individuals in engaging conversations effectively.
- Education: The device can make educational content more accessible, thereby promoting inclusive education. It can be particularly beneficial in e-learning settings, where digital content can be easily converted into a format that deaf-blind individuals can understand.
- Braille and Text Labeling: The inclusion of Braille and text labeling on device buttons significantly improved accessibility. Users reported that this feature enhanced their ability to interact with the device independently and efficiently.
- Independence: By making information more accessible, the device can empower deaf-blind individuals to perform tasks independently, reducing reliance on others.
- Social Impact: By addressing the specific needs of deaf-blind individuals, your project could significantly improve their quality of life and social inclusion. It could also raise awareness about the challenges faced by this community and inspire more initiatives aimed at inclusivity.

8.2 Conclusion

- In conclusion, our project has culminated in the creation of a user-friendly graphical interface, the device offers features such as image text recognition, multilingual translation, and real-time speech transcription, all aimed at enhancing accessibility and communication.
- Our primary aim has been to improve educational accessibility and promote independence among deaf-blind individuals. By providing tools to access information and engage with their surroundings more effectively, we believe our device can contribute to a more inclusive learning environment and empower individuals to lead fulfilling lives.
- Moving forward, we are committed to ongoing refinement and collaboration with the deaf-blind community to further optimize the device's functionality and ensure its seamless integration into daily life, thereby advancing the broader goal of equitable access to technology for all individuals.

REFERENCES

[1] https://wfdb.eu/wfdb-report-2018/deafblindness-and-education/ - Global Report 2018, Persons with Deaf blindness, and Education

[2] https://icevi.org/wp-content/uploads/2020/06/GEM-Background-Report-Children-with-Visual-Impairments-in-sub-Saharan-Africa-2020.pdf - Education of Children With Visual Impairments In Sub-Saharan Africa: Challenges And Opportunities

[3] Ahmed, Sara S., Abeer O. Abd El-Basit, Ayat K. Hosny, Martina M. Wahba, Samah A. Saber, and Khaled A. Ali. "Assistive Technology for the Visually Impaired Using Computer Vision and Image Processing." In International Conference on Advanced Intelligent Systems and Informatics, pp. 287-297. Cham: Springer International Publishing, 2022.

[4] Karmel, A., Sharma, A. and Garg, D., 2019. IoT based assistive device for deaf, dumb and blind people. Procedia Computer Science, 165, pp.259-269.

[5] AlSaid, Hawra, Lina AlKhatib, Aqeela AlOraidh, Shoaa AlHaidar, and Abul Bashar. "Deep learning assisted smart glasses as educational aid for visually challenged students." In 2019 2nd International Conference on new Trends in Computing Sciences (ICTCS), pp. 1-6. IEEE, 2019.

[6] Hu, Z., He, X., Lin, Y., Zhao, Y., Liang, Z., & Wu, J. (2023, December). Intelligent Point Reader Based on Machine Vision and Speech Processing. In 2023 IEEE International Conference on Electrical, Automation and Computer Engineering (ICEACE) (pp. 975-983). IEEE.

[7] Patil, K., Kharat, A., Chaudhary, P., Bidgar, S., & Gavhane, R. (2021, March). Guidance system for visually impaired people. In 2021 International conference on artificial intelligence and smart systems (ICAIS) (pp. 988-993). IEEE.

[8] https://www.techtarget.com/whatis/definition/large-language-model-LLM - understanding How LLM algorithms are trained on massive data sets, capable of understanding natural language and performing various tasks.

[9] https://scholarworks.umt.edu/cgi/viewcontent.cgi?article=7064&context=etd - Methods and problems in educating the deaf and blind and a Methods and problems in educating the deaf and blind and a proposed study proposed study.

[10] https://plusqa.com/2018/05/17/accessibility-testing/ - Today is the 7th annual Global Accessibility Awareness Day—a “holiday” borne of a single tweet from Joe Devon back in 2011.

Appendix: A- Packages, Tools used & Working Process

Python Programming language

Python is a high-level Interpreter based programming language used especially for general-purpose programming. Python features a dynamic type of system and supports automatic memory management.

It supports multiple programming paradigms, including object-oriented, functional and Procedural and also has its a large and comprehensive standard library. Python is of two versions. They are Python 2 and Python 3.

This project uses the latest version of Python, i.e., Python 3. This python language uses different types of memory management techniques such as reference counting and a cycle-detecting garbage collector for memory management. One of its features is late binding (dynamic name resolution), which binds method and variable names during program execution. Python's offers a design that supports some of things that are used for functional programming in the Lisp tradition. It has vast usage of functions for faster results such as filter, map, split, list comprehensions, dictionaries, sets and expressions. The standard library of python language has two modules like itertools and functools that implement functional tools taken from Standard machine learning.

Libraries

OpenCV

OpenCV is a huge open-source library for computer vision, machine learning, and image processing. OpenCV supports a wide variety of programming languages like Python, C++, Java, etc. It can process images and videos to identify objects, faces, or even the handwriting of a human. When it is integrated with various libraries, such as Numpy which is a highly optimized library for numerical operations, then the number of weapons increases in your Arsenal i.e whatever operations one can do in Numpy can be combined with OpenCV. This OpenCV tutorial will help you learn the Image-processing from Basics to Advance, like operations on Images, Videos using a huge set of Opencv-programs and projects.

pyttsx3

Pyttsx3 is a Python library designed for text-to-speech conversion. Unlike some other libraries, Pyttsx3 operates entirely offline, making it convenient for various applications where internet connectivity might be limited or unavailable. Moreover, it's compatible with both Python 2 and Python 3, ensuring broad usability across different versions of the language.

PIL

The Python Imaging Library (PIL) is a free library for the Python programming language that adds support for opening, manipulating, and saving many different images file formats. Image and ImageTk are two modules from PIL (or Pillow, which is a fork of PIL with user-friendly features).

easyOCR

EasyOCR is a Python library that provides an easy-to-use interface for Optical Character Recognition (OCR). OCR is a technology used to convert different types of documents, such as scanned paper documents, PDF files or images captured by a digital camera, into editable and searchable data.

EasyOCR stands out for its support of more than 80 languages and various writing scripts including Latin, Chinese, Arabic, Devanagari, Cyrillic, and others. This makes it a versatile tool for multilingual OCR tasks.

The library works by taking an image file or an image path as input and returns a list of bounding box locations and recognized text along with their confidence levels. This can be useful in a variety of applications, such as automating data entry processes, digitizing printed documents, or developing assistive technologies for visually impaired individuals.

EasyOCR is also integrated into Huggingface Spaces using Gradio, which allows you to try out its capabilities through a web demo.

googletrans

The googletrans library is a free and unlimited Python interface to Google Translate's AJAX API. It allows developers to implement language translation features in their Python applications. The library can automatically detect the language of the input text, translate a batch of strings in a single method call, and supports HTTP/2. It uses the same servers that Google Translate uses, making it fast and reliable.

The main functions provided by googletrans are translate for translating text and detect for detecting the language of a given text. The translate method is used to translate text from one language to another, and if the source language is not given, Google Translate attempts to detect the source language.

Lang Chain

LangChain is an open-source framework that simplifies the creation of applications using Large Language Models (LLMs). LangChain is related to OpenAI as it provides integrations for OpenAI through their SDK, offering a convenient way to interact with OpenAI models. It provides a standard interface for chains, integrations with other tools, and end-to-end chains

for common applications. It includes components like LLM Wrappers, Prompt Template, and Indexes for relevant information retrieval.

GTTS

The gTTS (Google Text-to-Speech) library in Python interfaces with Google Translate's text-to-speech API, allowing you to convert text into speech in a variety of languages. The library also provides customizable text pre-processors for pronunciation corrections.

The main function provided by gTTS is gTTS(), which takes a string of text and converts it into speech. The speech can then be saved as an MP3 file. While this library is free and unlimited, it could be blocked at any time since it uses the Google Translate speech functionality and is different from Google Cloud Text-to-Speech.

Threads

In Python's threading module, threads enable concurrent execution of tasks within a process, managed by the operating system's scheduler and sharing the same memory space. The Thread class creates individual threads, while methods like start() initiate thread execution and join() blocks until a thread finishes. is_alive() checks a thread's status, and enumerate() lists all active threads. Additionally, synchronization primitives like locks, semaphores, and condition variables prevent race conditions. Despite threading's utility for I/O-bound tasks, Python's Global Interpreter Lock limits its effectiveness for CPU-bound operations, which may be better suited for multiprocessing or asynchronous programming.

Pyaudio

PyAudio is a Python library that provides bindings for PortAudio, a cross-platform audio I/O library. It allows developers to record and play audio across multiple platforms. PyAudio is also open source and distributed under the MIT License, making it free to use and modify.

Using PyAudio involves creating a new PyAudio instance, opening an audio input or output stream with the desired properties, reading from or writing to the stream, and then closing the stream to free up system resources.

Wave

The wave module in Python is an interface to the Waveform Audio File Format (WAVE), commonly known as WAV due to its filename extension, which is a common format for storing audio data on PCs. The module supports audio parameters such as the number of channels, sample width (bit depth), frame rate (sample rate), and number of frames. It supports only uncompressed Pulse-Code Modulation (PCM) encoded wave files, a standard form of digital audio in computers and digital communication systems. it is compatible with the Pyaudio library, making it easy to save the recorded audio data to a WAV file.

speech_recognization

The speech_recognition module in Python is a powerful library that provides the ability to convert spoken language into written text. It supports multiple speech recognition engines and APIs, both online and offline, including Google Speech Recognition, Google Cloud Speech API, Microsoft Bing Voice Recognition, and IBM Speech to Text. The module can process audio data from both audio files and microphones, and it has built-in capabilities for handling ambient noise. However, the accuracy of the speech recognition can be influenced by factors such as the quality of the input audio, background noise, and the speaker's accent.

Tkinter

Tkinter is a standard Python interface to the Tk GUI toolkit and is the most used library for creating Graphical User Interfaces (GUIs) in Python. It allows developers to create windows, buttons, text boxes, and other GUI elements.

You can create a GUI application by importing the Tkinter module, creating the main window, adding widgets to the window, applying event triggers on the widgets, and running the application using the mainloop() method. Tkinter also provides geometric configuration of the widgets, which can organize the widgets in the parent windows using geometry manager classes like pack(), grid(), and place().

Appendix: B

Sample Source Code with Execution

❖ Image Processing:

```
def identifyLang(img):
    percentages = []
    codes = ['en','te','hi']
    for code in codes:
        reader = easyocr.Reader([code],gpu=False)
        result = reader.readtext(img)
        pb = 0
        count = 0
        for items in result:
            _,_,score = items
            pb += score
            count +=1
        if count==0:
            break
        percentages.append([code, pb/count])
    if len(percentages) == 0:
        return False
    max_pair = max(percentages, key=lambda x: x[1])
    langCode = max_pair[0]
    if langCode == 'en':
        return [langCode]
    else:
        return ['en',langCode]
def I2T(img_path):
    langCodes = identifyLang(img_path)
    if not langCodes:
```

```
        time.sleep(.1)
        print('No text identified...!')
        return
    reader = easyocr.Reader(langCodes)
    print(langCodes)
    result = reader.readtext(img_path)
    text = ' '.join([text[1] for text in result])
    print("The text was :\n" + text)
    ex = translator.detect(text)
    detected_lang = LANGUAGES[ex.lang]
    print("\nThe text was in: " + detected_lang)
    time.sleep(.1)


def T2T(window, message, srcLang, destLang):
        trans = Translator()
        t = trans.translate(message, dest=destLang, src=srcLang)
        window.destroy()
        translated(t.text, destLang)
def summarize(text, lang):
    prompt = PromptTemplate(
        template = "give the text in {x} by summarizing the below text, within 50 to 60
words. text: {y}.",
        input_variables = ["x", "y"]
    )
    query = prompt.format(x=lang, y=text)
    translated_text = model(prompt=query)
    return translated_text
def T2S(text,lan):
    length = len(text.split())
    if length > 50:
        print(f"\nThe text contains {length} words.")
```

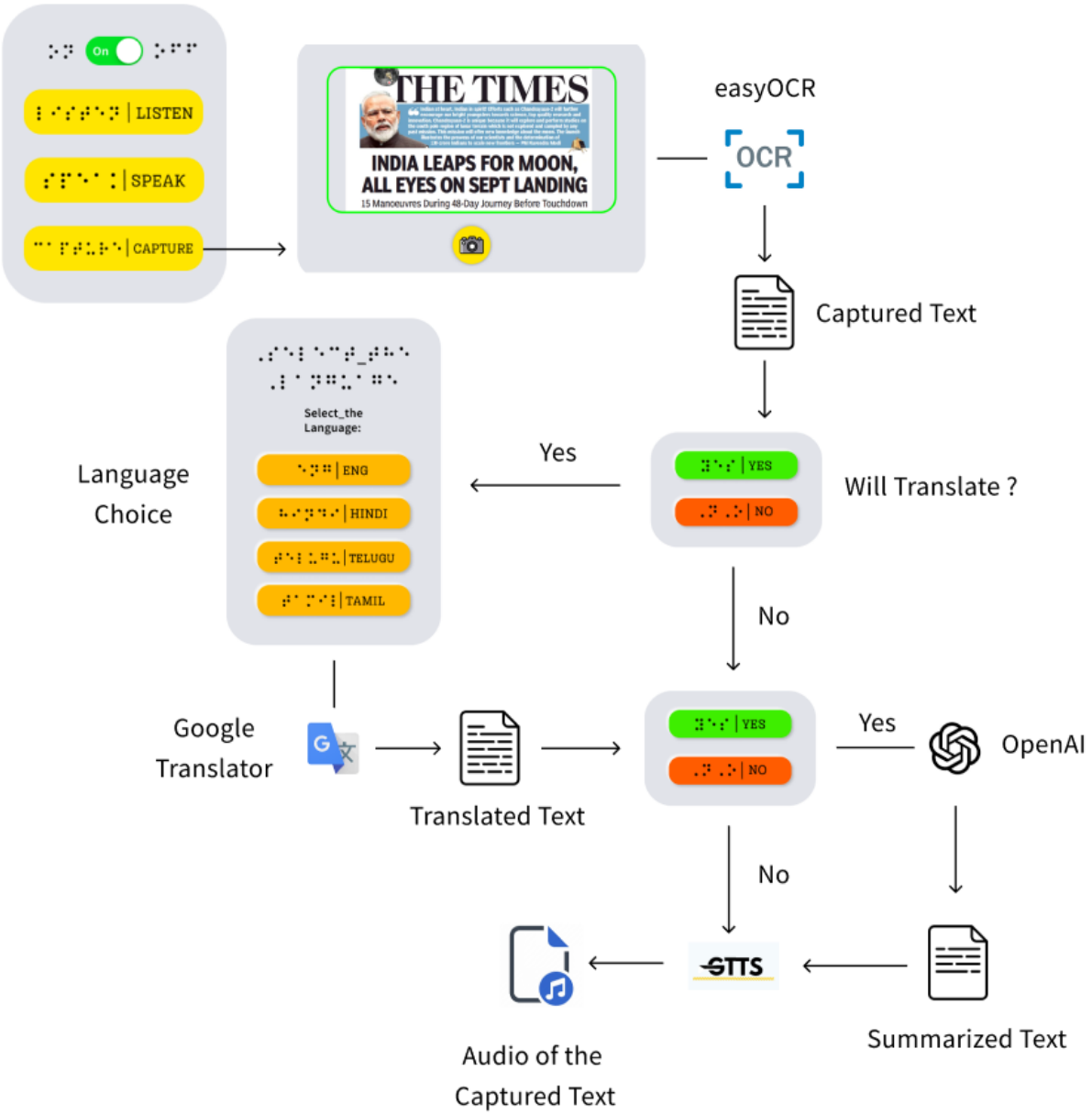
```
    def willSummarize(result,text):
        if result:
            text = summarize(text, LANGUAGES[lan])
            print("\nThe summarized text is : ",text)
        audio = gTTS(text=text, lang=lan, slow=False)
        audio.save("audio.mp3")
        os.system("start audio.mp3")
    else:
        audio = gTTS(text=text, lang=lan, slow=False)
        audio.save("audio.mp3")
        os.system("start audio.mp3")
```

❖ Voice Capturing:

```
def record_audio():
    CHUNK = 1024
    FORMAT = pyaudio.paInt16
    CHANNELS = 1
    RATE = 44100
    FILE_NAME = "temp.wav"
    p = pyaudio.PyAudio()
    stream = p.open(
        format=FORMAT,
        channels=CHANNELS,
        rate=RATE,
        input=True,
        frames_per_buffer=CHUNK
    )
    frames = []
    while not stop_recording_event.is_set():
        frames.append(stream.read(CHUNK))
```

```
    stream.stop_stream()
    stream.close()
    p.terminate()
    save_audio(frames, FILE_NAME)
    process_audio(FILE_NAME)
def save_audio(frames, file_name):
    wf = wave.open(file_name, 'wb')
    wf.setnchannels(1)
    wf.setsampwidth(pyaudio.PyAudio().get_sample_size(pyaudio.paInt16))
    wf.setframerate(44100)
    wf.writeframes(b".join(frames))
    wf.close()
def process_audio(file_name):
    recognizer = sr.Recognizer()
    try:
        with sr.AudioFile(file_name) as source:
            audio_data = recognizer.record(source)
            text = recognizer.recognize_google(audio_data)
            ex = translator.detect(text)
            translated_text = translator.translate(text, dest=ex.lang) # type: ignore
            entry.delete("1.0", tk.END)
            entry.insert("1.0", translated_text.text) # type: ignore
            print("Recognized Text:", translated_text.text) # type: ignore
    except sr.UnknownValueError:
        print("Google Speech Recognition could not understand audio")
    except sr.RequestError as e:
        print(f"Could not request results from Google Speech Recognition service; {e}")
    finally:
        os.remove(file_name)
```

Image Capture Output:



Voice Capture Output: