**BLIND ASSISTANCE SPEECH ENGINE**

***Control System***

***Course Code: ECE2010***

***Slot: F1+TF1***

***By***

# *18BEC2008: DAYA SHANKAR SINGH*

# *18BEC2014: RAHUL MAHESH AWARI*

# *18BEC2016: SWAPNIL SHEKHAR*

# *18BEC2029: SAMRIDDH SINGH*

# 

**Under the guidance of**

**Prof. Rajesh R**

**School of Electronics Engineering**

**Vellore Institute of Technology**

**Vellore**



**Fall Semester 2020**

# Declaration by Authors

*This is to declare that this report has been written by us as part of our coursework. No part of the report is plagiarized from other sources. All information included from other sources have been duly acknowledged. We aver that if any part of the report is found to be plagiarized, we shall take full responsibility for it.*

DAYA SHANKAR SINGH 18BEC2008

RAHUL MAHESH AWARI 18BEC2014

SWAPNIL SHEKHAR 18BEC2016

SAMRIDDH SINGH 18BEC2029

## 

Place: Vellore

Date: 2nd November, 2020

## Acknowledgements

We would like to thank our professor Prof. Rajesh R whom we are highly indebted for giving us this opportunity to perform this project and for his help, and guidance throughout this project. We would like to express our gratitude to VIT University and the School of Electronics and Communication Engineering (SENSE). We would like to extend our gratitude to the Dean of SENSE and this prestigious University in supporting us and giving us the opportunity to carry out our studies at this University. And finally, we would like to thank our parents and friends for supporting and helping us throughout our project.

## Abstract

Visual impairment is one of the biggest limitations for humanity, especially in this day and age when information is communicated a lot by text messages (electronic and paper based) rather than voice. The device we have proposed aims to help people with visual impairment. In this project, we developed a device that converts an image’s text to speech.

The basic framework is an embedded system that captures an image, extracts only the region of interest (i.e. region of the image that contains text) and converts that text to speech. It is implemented using a mobile or laptop camera.

The captured image undergoes a series of image pre-processing steps to locate only that part of the image that contains the text and removes the background. Two tools are used convert the new image (which contains only the text) to speech. They are OCR (Optical Character Recognition) software and TTS (Text-to-Speech) engines. The audio output is heard through the audio jack using speakers or earphones.

# TABLE OF CONTENTS

**TITLE PAGE NO.**

**Declaration by Authors 2**

**Acknowledgement 3**

**Abstract 4**

## Table of Contents 5

1. Introduction **6**

**2.** Problem Statement**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_7**

**3.** Literature Survey \_\_\_\_\_\_8

* 1. Scope of the Project**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 8**

**4.** System Model and Block Diagram**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ 9**

**5.** Methodology\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**12**

**5.1** MSER using MATLAB**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_12**

* 1. OCR using PYTHON**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_17**

**5.3** gTTS\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**21**

**5.4** pyTTS**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_24**

**6.** Performance Analysis**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_25**

**7.** Results and Discussions**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_26**

**7.1** Output and Simulation \_**26**

**8.** Conclusion**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_29**

* 1. Future Implementation**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_29**

1. List of references**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_30**

**Introduction**

## In our planet of 7.4 billion humans, 285 million are visually impaired out of whom 39 million people are completely blind, i.e. have no vision at all, and 246 million have mild or severe visual impairment (WHO, 2011). It has been predicted that by the year 2020, these numbers will rise to 75 million blind and 200 million people with visual impairment.



**https://rb.gy/gh0xe9**

**Fig. 1**

As reading is of prime importance in the daily routine (text being present everywhere from newspapers, commercial products, sign-boards, digital screens etc.) of mankind, visually impaired people face a lot of difficulties. Our device assists the visually impaired by reading out the text to them.

There have been numerous advances in this area to help visually impaired to read without much difficulties. The existing technologies use a similar approach as mentioned in this report, but they have certain drawbacks.

Firstly, the input images taken in previous works have no complex background, i.e. the test inputs are printed on a plain white sheet. It is easy to convert such images to text without pre-processing, but such an approach will not be useful in a real-time system.

Also, in methods that use segmentation of characters for recognition, the characters will be read out as individual letter and not a complete word. This gives an undesirable audio output to the user. For our project, we wanted the device to be able to detect the text from any complex background and read it efficiently.

Inspired by the methodology used by Apps such as “Cam Scanner”, we assumed that in any complex background, the text will most likely be enclosed in a box, like example billboards, screens etc. By being able to detect a region enclosing four points, we assume that this is the required region containing the text. This is done using warping and cropping.

The new image obtained then undergoes edge detection and a boundary is then drawn over the letters. This gives it more definition. The image is then processed by the OCR and gTTS to give audio output.

**Problem Statement**

We have made an **OCR based Text to Speech** converter which will help the visually impaired or blind people to get access to all the books and other text materials which further can be used in an App or Web Services.

Our project is designed for people with mild or moderate visual impairment by providing the capability to listen to the text. It can also act as a learning aid for people suffering from dyslexia or other learning disabilities that involve difficulty in reading or interpreting words and letters. We wish to enable these people to be independent and self-reliant as they will no longer need assistance to understand printed text

## Literature Survey

The initial focus of our project was to create a software application which will help visually impaired people to provide them the capability to listen to text easily.

So, we went ahead and explore various languages and platform to select the one which will give us the most suitable and accurate result. We went through various research paper and documents on **OCR** (Optical character recognition) and conversion of text to speech and vice-versa and see what we can adapt in our project to make it efficient and affordable.

We went through many websites to compare various algorithm, packages and libraries to compare which is the best among all and how to adapt or apply that in our project. We had many problems in running and importing the packages into our system. We went through various videos and Stack Overflow to solve the errors in our codes.

We concluded our literature survey by looking through various reports and research paper to understand the working background of various algorithm and packages. We will be mentioning all research paper and report details in our reference section.

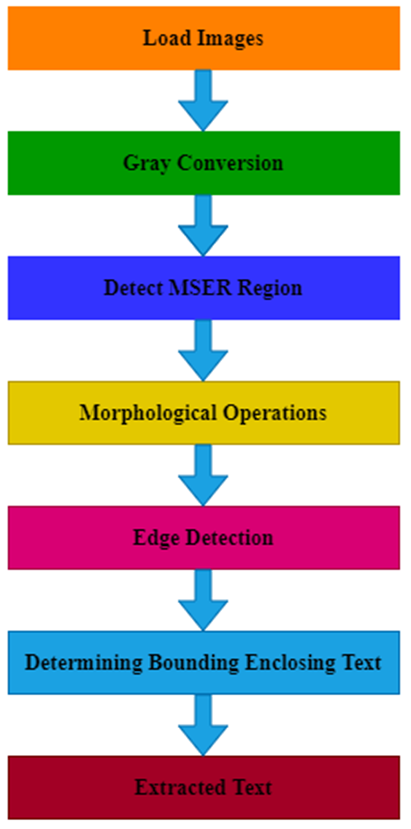
## 3.1 Scope of the Project

We have made OCR based Text to Speechconverter which will help the visually impaired or blind people to get access to all the books and other text materials which further can be used in an App or Web Services.

This technique further implemented using best datasets of different languages and using Deep learning we can exactly recognize the text which are in the images, pdf format or any format. And also, for reverse implementation that is Speech recognition which is used everywhere like Google, Amazon Alexa, and also speech-based command. This technology was further implemented in defense purpose like fighter plane pilot having a mounted helmet which include this feature instead of manually controlling the system.

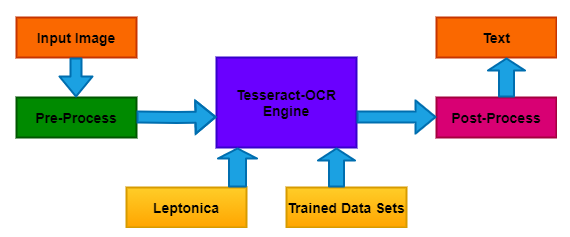
**System Model and Block Diagram**

**4.1 MSER Flowchart**

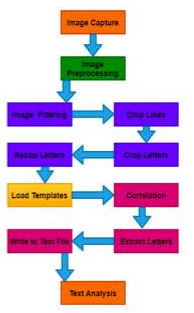


**Fig. 2-4.1**

**4.2 OCR Flowchart**

****

**Fig. 3-4.2**

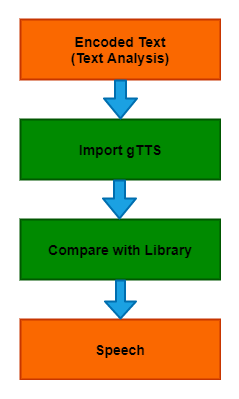
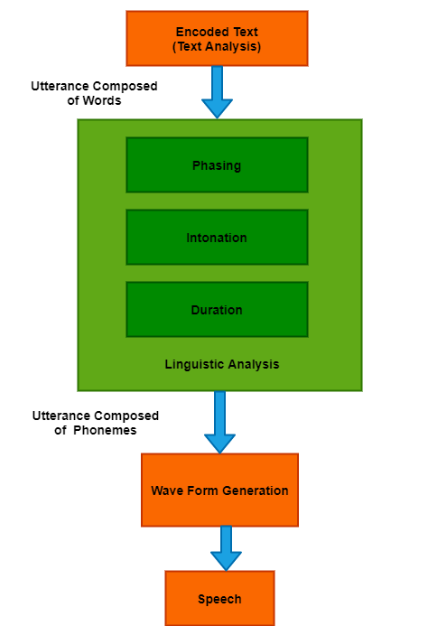


We will do this by taking input of PDF or any other form of document to **OCR Tools** and the praised text will be in in the form of **JSON** which is user readable format and will work as Audio Description using Text to Speech and for reading live books, templates and images.

Also, we used **gTTS** which is a python library which help us to processing and tokenizing the speech input with automatic retrieval of supported language and finally convert them Speech to Text.

**Fig. 4-4.2**

**4.3 Text to Speech**

**Fig. 5-4.3**

Here, we have used two methods is used to convert the text to speech. We have used Python text to speech (pyTTS) and Google text to speech (gTTS). The working of both is same. Firstly synthesized speech can be created by concatenating pieces of recorded speech that are stored in a [database](https://en.wikipedia.org/wiki/Database).

Then it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. Then it assigns [phonetic transcriptions](https://en.wikipedia.org/wiki/Phonetic_transcription) to each word, and divides and marks the text into [prosodic units](https://en.wikipedia.org/wiki/Prosody_(linguistics)), like [phrases](https://en.wikipedia.org/wiki/Phrase), [clauses](https://en.wikipedia.org/wiki/Clause), and [sentences](https://en.wikipedia.org/wiki/Sentence_(linguistics)). At last back-end converts the symbolic linguistic representation into sound.

**Methodology**

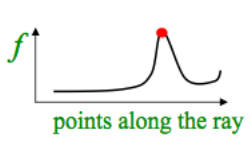
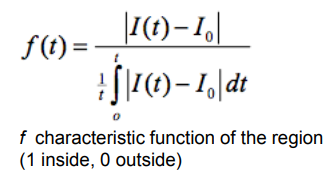
## 5.1 MSER using MATLAB

In **Fig. 2-4.1** MSER is a method for blob detection in an image. This algorithm extracts from an image a number of co-variant regions called Maximally Stable Extremal Regions. MSER feature detector works well for text because consistent color and high contrast of text leads to stable intensity. An MSER is a stable connected component of some gray-level sets of the images. For region detection invariance transformations that should be considered are illumination changes, translation, rotation, scale and full affine transform. This technique was applied in MATLAB.

Affine Invariant Intensity Extrema based---

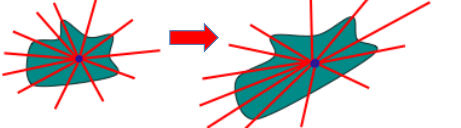
**5.1.1 Algorithm:**

* Start form local intensity extremum point.
* Go in every direction until the point of extremum of some function f.
* Compute geometric moments of orders up to 2 for this region.
* Replace the region with ellipse.
  + 1. **Detecting extremal region:**
* Detect anchor points. It detected at multiple scales are local extremes of intensity.
* Explore image around rays from each anchor point. Go every ray starting from this point until an extremum of function f is reached.

**Fig. 6-5.1.2**

* All points create some irregular-shaped region.



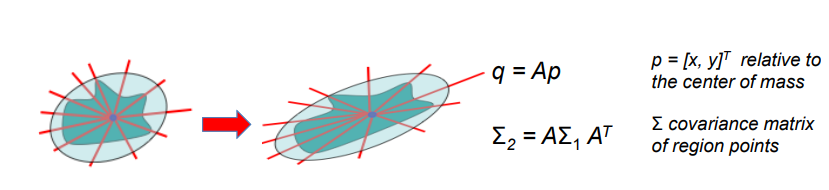
**Fig. 7-5.1.2**

**5.1.3 Approximating the regions:**

* Being function f the characteristic function of the region, moments up to 2nd order allow to approximate the region with an ellipse.



* The ellipse of an affine-transformed region corresponds to the ellipse of the original region under the same transformation.



**Fig. 8-5.1.3**

* As regions are illumination-based warp ellipse to circle for affine invariance.

**Step-1:**

In this step the MSER algorithm works for finding text regions to find all the regions within the image and plot the results. There are many non-text regions detected alongside the text which are removed in third step. To detect text regions MSER first convert the color image into gray scale image.



**Fig. 9-5.1**

**Step-2:**

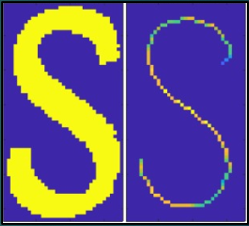
Remove Non-Text Regions Based on Basic Geometric Properties. The MSER algorithm chooses most of the text and it also detects many other stable regions in the image that are not text. We can use a rule-based approach to detect and remove non-text regions present in the image. The non-text regions from image are filter by using geometric properties.



**Fig. 10-5.1**

**Step-3:**

Remove Non-Text Regions Based on Stroke Width Variation. Another common metric used to make decision between text and non-text is stroke width. The stroke width variation properties are measure of the width of the curves and lines that make up a character. Text regions have little stroke width variation, whereas no text regions have larger stroke width variations.



**Fig. 11-5.1**

**Step-4:**

Merge Text Regions. At this point, all the detection results are composed of individual text characters. To use these results for recognition tasks, such as OCR, the individual text characters must be merged into words or text lines. Enables words exactly matching with more meaningful information instead of comparing individual characters. Merging of individual text into lines or words there is one technique that is first find neighboring text then form a bounding box region. To find neighboring region, expand the bounding boxes computed earlier with region props. This makes the bounding boxes of neighboring text regions overlap such that text regions that are part of the same word or text line form a chain of overlapping bounding boxes.

Now, overlapping bounding boxes can be merged together to form a single bounding box around individual words or text lines. To do so, compute the overlap ratio between all bounding box pairs. This quantifies the distance between all pairs of text regions so that it is possible to find groups of neighboring text regions by looking for non-zero overlap ratios. Once the pair-wise ratios are computed, use a graph to find all the text regions “connected” by a non-zero overlap ratio. To do so, use bboxOverlapRatio () to compute pair-wise overlap ratio for all expanded bounding boxes, then use graph to find all the connected regions.



**Fig. 12-5.1**

**Step-5:**

After detecting the text regions, use the OCR function to recognize the text within each bounding box. Without first finding the text regions, the output of the OCR function would be considerably noisier.



**Fig. 15-5.1**

**Codes Screenshots:**



**Fig. 16-5.1**



**Fig. 17-5.1**



**Fig. 18-5.1**



**Fig. 19-5.1**

## 5.2 OCR using PYTHON

In **Fig. 3-4.2** and **Fig. 4-4.2** OCR is an electronic conversion of the typed, handwritten or printed text images into machine encoded text. Mainly OCR are divided into 2 main systems.

**Matrix Matching / Pattern Matching:**

It identified image-based files as the equivalent plain text character when an image corresponds to one of these selected bitmaps within a certain degree of likeness. It is simpler and easy to apply.

**Feature Extraction / ICR:**

It searches for common elements, like open spaces, closed forms, lines-diagonals intersecting etc. depend on precise matching to set templates.

We will do this by taking input of PDF or any other form of document to OCR Tools and the praised text will be in in the form of JSON which is user readable format and will work as Audio Description using Text to Speech and for reading live books, templates and images. Also, we used gTTS which is a python library which help us to processing and tokenizing the speech input with automatic retrieval of supported language and finally convert them Speech to Text. So, we implement OCR in Python which detect the image-text into speech.

**5.2.1 Pre-Processing:**

This is done to boost the chances of recognition of a text-images. Technique used:

**De-Skew: -**

* If the document was not correctly aligned when scanned, it may need to be titled a few degrees clockwise or counterclockwise to create text-lines completely horizontal or vertical.

**Despeckle: -**

* Remove positive and negative spots, smoothing edges.

**Binarization: -**

* Convert an image to black and white (binary image 2 colors only used). The binarization task is conducted as an easy and accurate way to distinguish text from background.

**Line Removal: -**

* Cleans up non-glyph boxes and lines.

**Layout Analysis / Zoning: -**

* Identifies columns, paragraphs, captions etc. as blocks. (Multi-columns layout tables)

**Line and Word Detection: -**

* Establish word and character shapes baseline, divide words when required.

**Script Recognition: -**

* In multiple language documents, the script may transform the word level and therefore script identification is vital before the relevant OCR to manage the particular script.

**Segmentation / Character Isolation: -**

* Various character linked by image artifacts should be divided, single characters broken into several artifacts-based pieces should be linked.

**Normalization: -**

* Normalize the aspect ratio and scale.

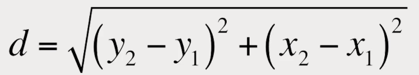
****

**Fig. 20-5.2.1**

* + 1. **Feature Extraction:**

There are two main methods for extracting features in OCR. The algorithm for feature detection defines a character by evaluating its lines and strokes. Pattern recognition works by identifying the entire character. We can recognize a line of text by searching or white pixel rows that have black pixels in between. Similarly, where character starts and ends.

Next convert the image of the character into a binary matrix where white pixels are 0s and black pixels are 1s. Then, by using the distance formula, we can find the distance from the center of the matrix to the farthest1.



We can create a circle of that radius and split it up into more granular sections. At this stage the algorithm compares each subsection to a database of the matrices representing characters with different fonts to identify the characters it has most in common statistically. It makes easy to bring printed media into digital world by doing this for every line and character.

* + 1. **Post-Processing:**

OCR accuracy can be improved if the output is limited by a lexicon (a list of words permitted in a document). This method can be less efficient if the document contains words that are not in the lexicon, like proper nouns. To improve this, we are using OCR library which are open source which is used to control the segmentation of the characters. The output stream can be a single string or a character’s file.

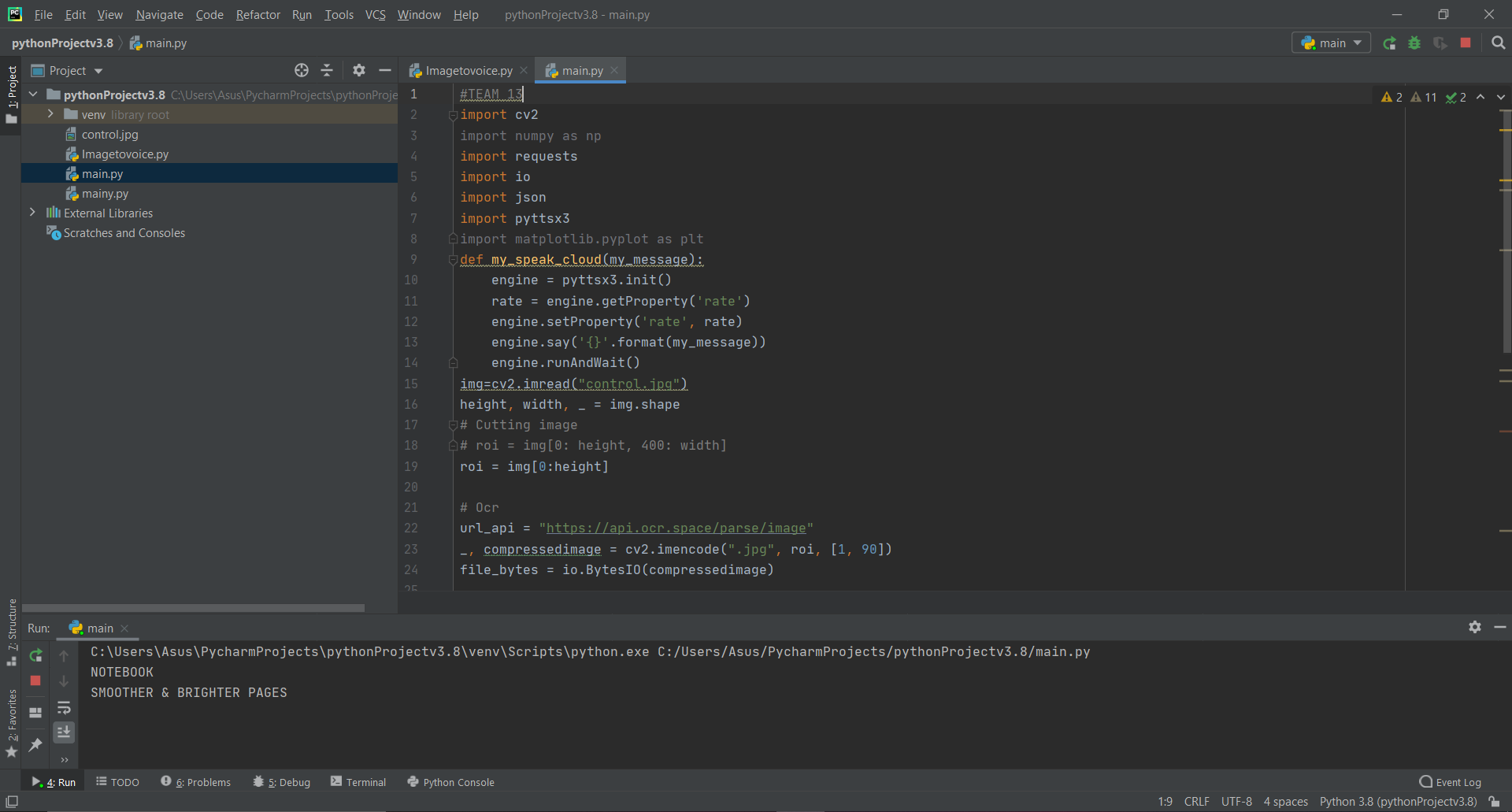
* + 1. **Error Correction:**

The ‘nearest neighbor’ can use frequencies for co-occurrence to correct mistakes by nothing that some words has been together.

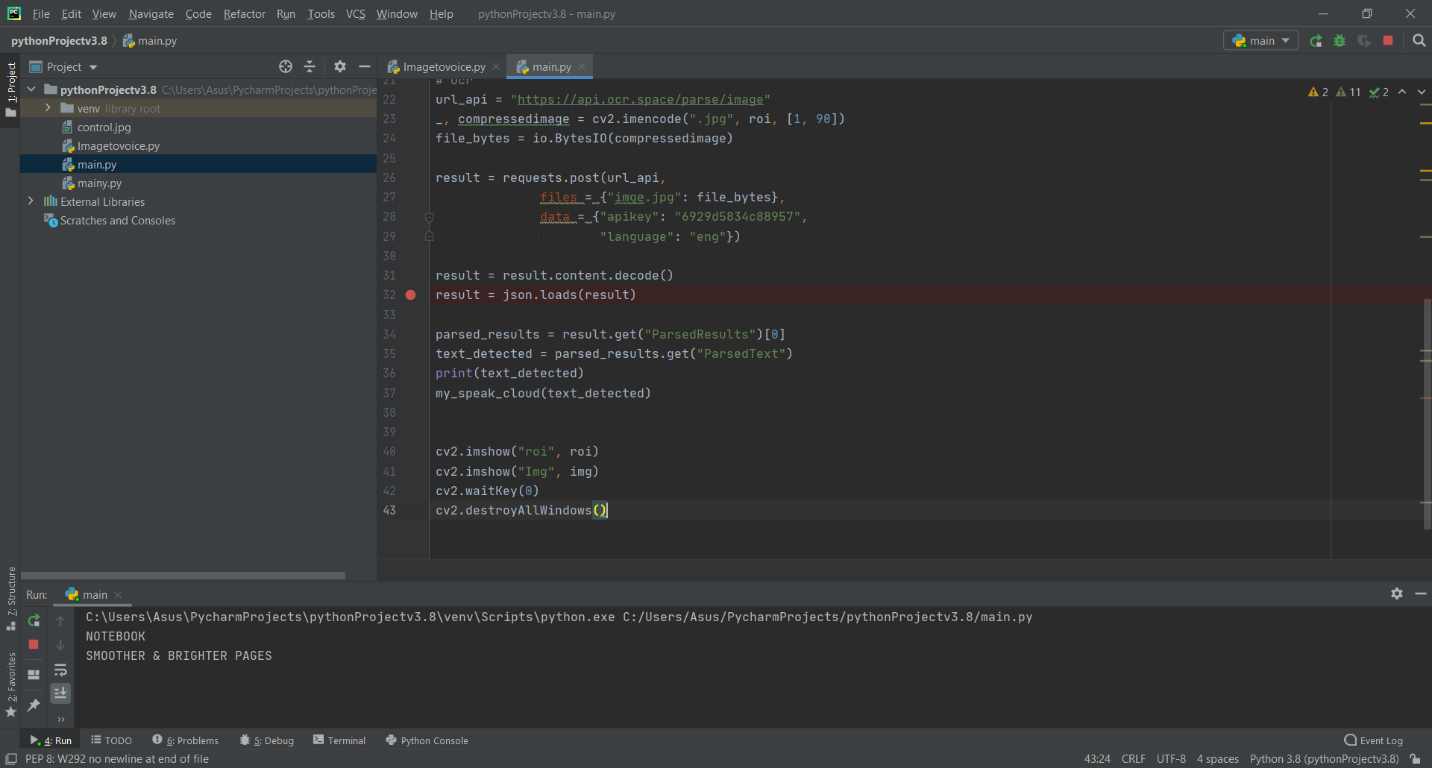


**Fig. 21-5.2.4**

**Codes Screenshots:**

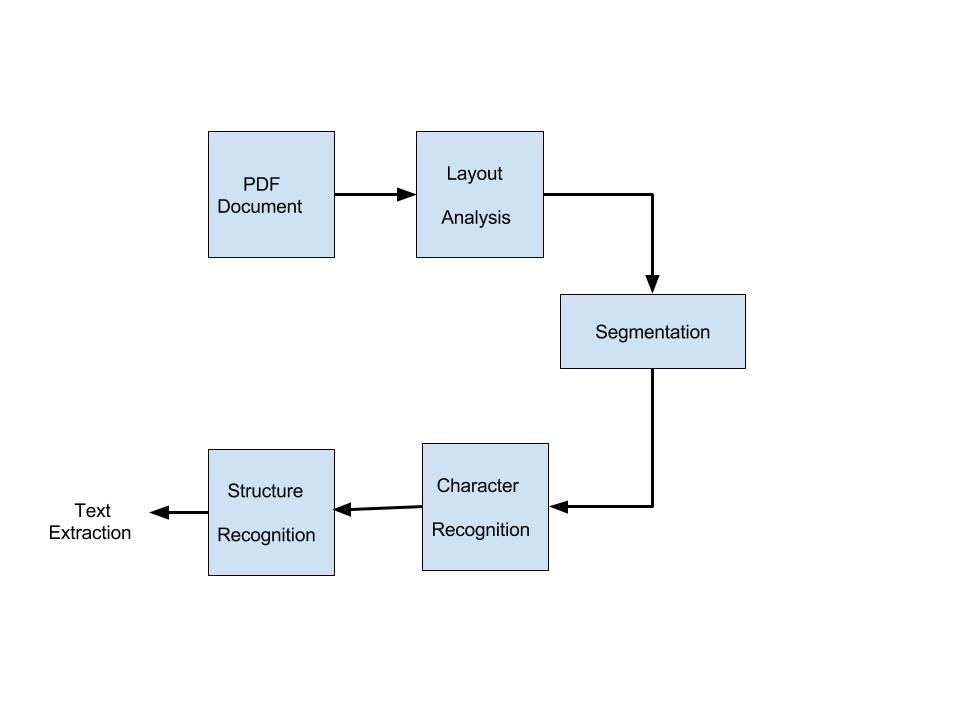


**Fig. 21-5.2**



**Fig. 22-5.2**

## 5.3 gTTS



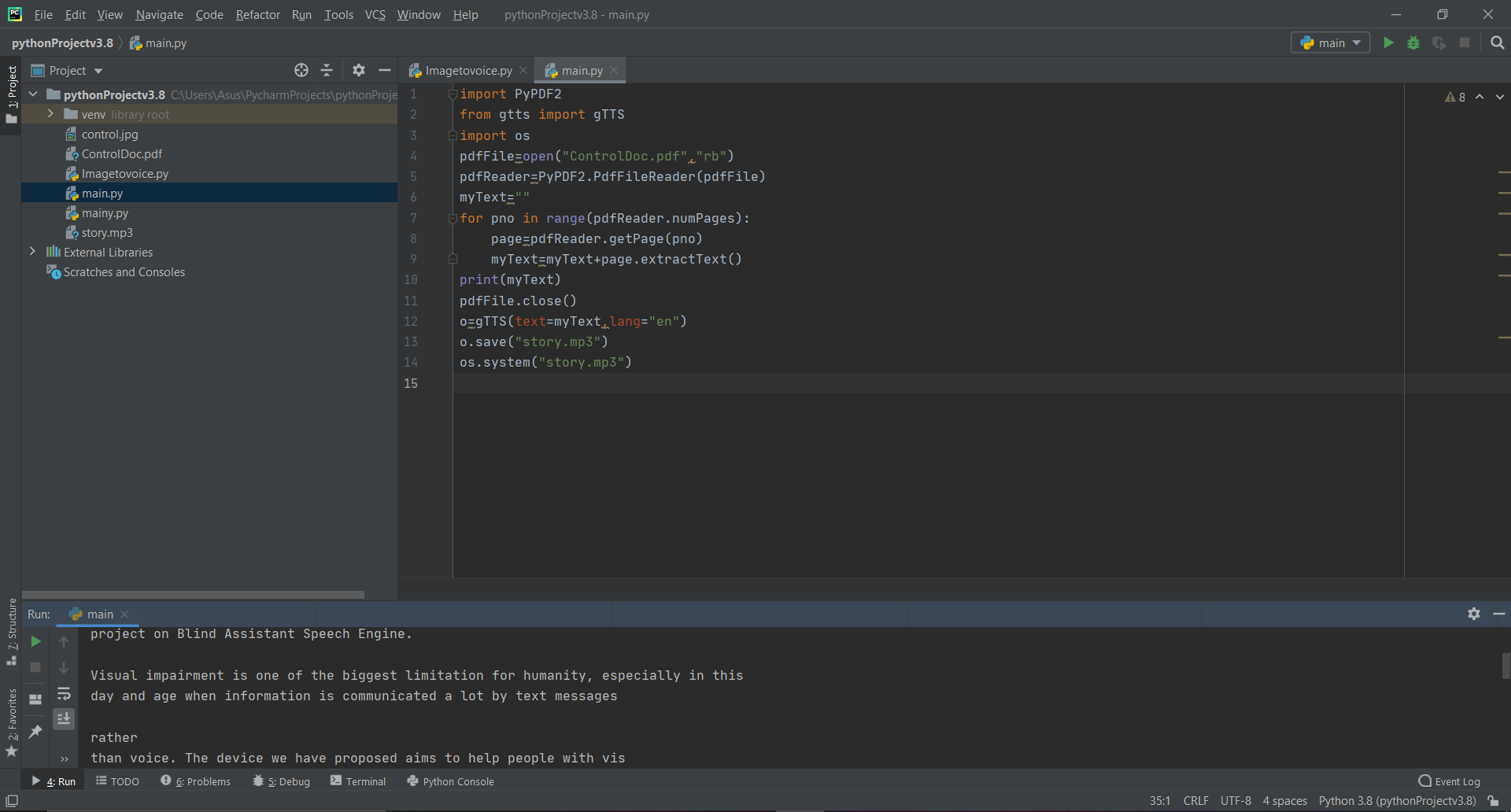
**Fig. 23-5.3**

In **Fig. 5-4.3** Here, we have used two methods is used to convert the text to speech. We have used Python text to speech (pyTTS) and Google text to speech (gTTS). The working of both is same. Firstly synthesized speech can be created by concatenating pieces of recorded speech that are stored in a [database](https://en.wikipedia.org/wiki/Database). Then it converts raw text containing symbols like numbers and abbreviations into the equivalent of written-out words. Then it assigns [phonetic transcriptions](https://en.wikipedia.org/wiki/Phonetic_transcription) to each word, and divides and marks the text into [prosodic units](https://en.wikipedia.org/wiki/Prosody_(linguistics)), like [phrases](https://en.wikipedia.org/wiki/Phrase), [clauses](https://en.wikipedia.org/wiki/Clause), and [sentences](https://en.wikipedia.org/wiki/Sentence_(linguistics)). At last back-end converts the symbolic linguistic representation into sound.

After detecting characters through OCR, the text is in encoded form. So, we have to convert that encoded text into speech using gTTS and pyTTS. PyPDF2 is an [open-source](https://en.wikipedia.org/wiki/Open-source_license) [command-line](https://en.wikipedia.org/wiki/Command-line) utility for converting [PDF](https://en.wikipedia.org/wiki/PDF) files to [plain text](https://en.wikipedia.org/wiki/Plain_text) files ,i.e. extracting text data from PDF-encapsulated files.

In this the characters are first extracted from pdf by the same method as OCR and then it is converted into text. With the help of gTTS (Google text to speech), the converted text is converted into sound waveforms and is spoken as words considering all linguistic spaces and characters. We are also creating the sound file of the same and saving it for future use.

**Codes Screenshots:**



**Fig. 24-5.3**

**Convert Pdf text into Speech and printed in terminal**

**5.3.1 Speech Recognition:**

We have tried to implement additional feature of speech to text conversion. For this we have used the speech recognition python library.

**Digital Sampling: -**

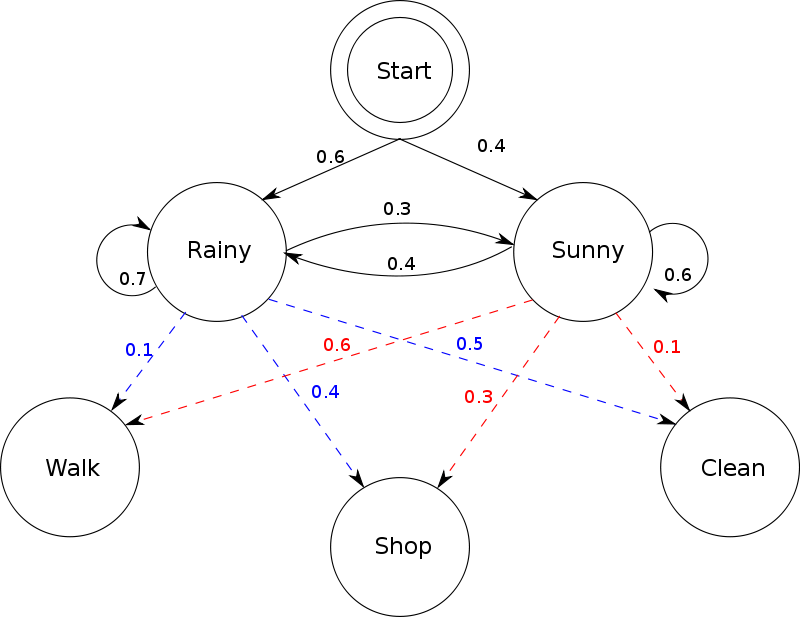
* When you speak, you create vibrations in the air. The **ADC** translate this analog wave into digital data that the computer can understand.
* To do this, it samples or digitizes, the sound by taking precise measurements of the wave at the frequent intervals.
* The system filters the digitized sound to remove unwanted noise, and sometimes to separate to separate it into different bands of frequency.

**Acoustic Model: -**

* Next the signal is divided into small segments as short as a few hundredth of a second or even thousandths in the case of plosive consonants sounds – consonant stops produced the obstructing airflow in the vocal tract – like “**p**” or “**t**”.
* The program then matches these segments to known phonemes in the appropriate languages.
* A phoneme is the smallest element of a language – a representation of the sounds we make and put together to form meaningful expression.

**Hidden Markov Model (HMM): -**

* In this model, each phoneme is like a link in a chain, and the complete chain is a word.
* The chain branches off in the different directions as the program attempts to match the digital sound with the phoneme that’s most likely to come next.,
* During this process, the program assigns a probability score to each phoneme, based on its built-in dictionary and user training.

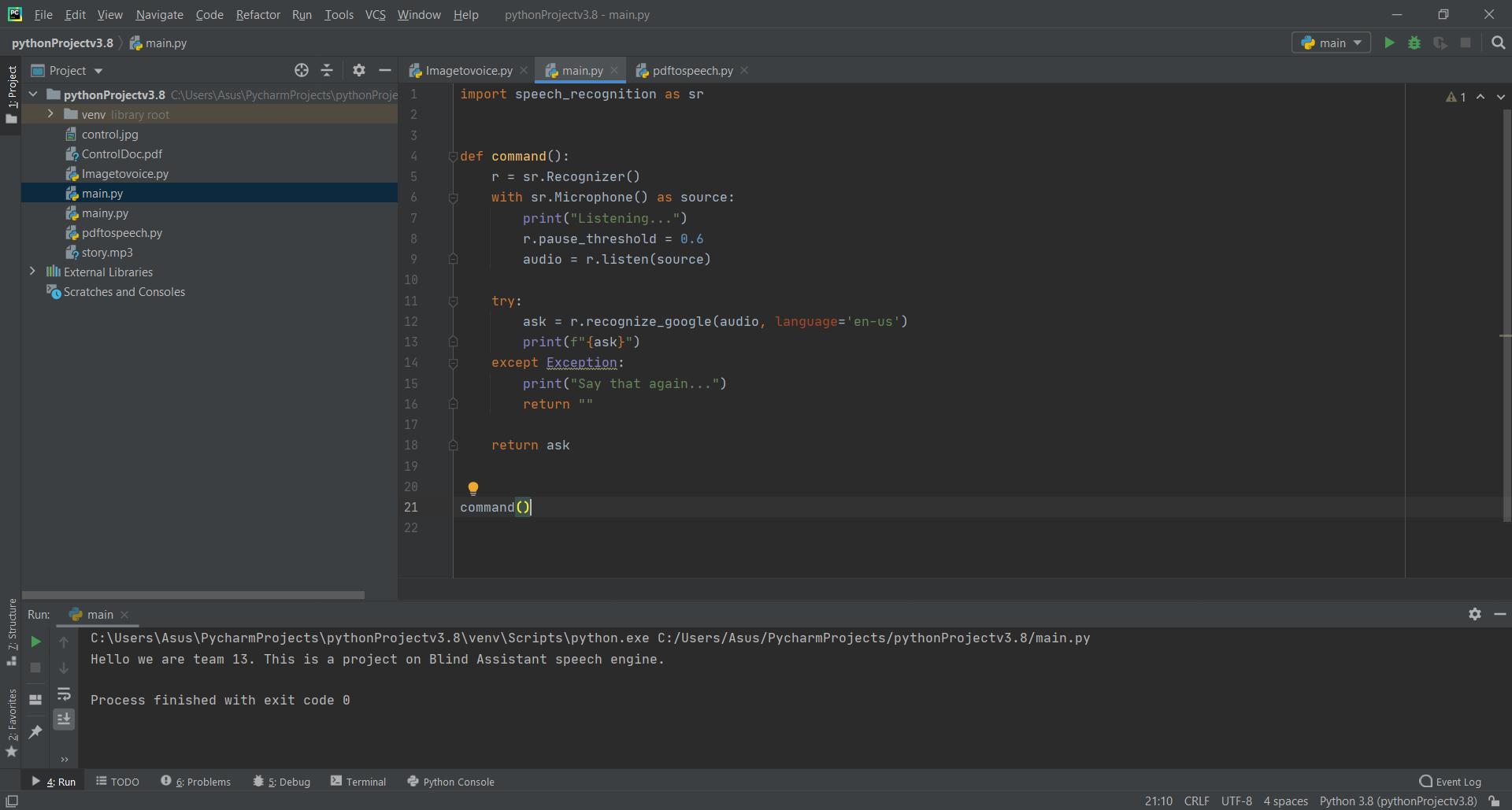


**Fig. 25-5.3.1**

**Program Training: -**

* The process is more complicated for phrases and sentences – the system has to figure out where each word stops and starts.
* The statistical systems need lots of exemplary training data to reach their optimal performance.
* Sometimes on the order of thousands of hours of human-transcribed speech and hundreds of megabytes of text.
* The training data are used to create acoustic models of words, word lists and multi-word probability networks.
* The details can make the difference between a well performing system and a poorly performing system – even when using the same basic algorithm.

**Codes Screenshots:**



**Fig. 26-5.3.1**

**Speech Input given in microphone gets printed in terminal**

## 5.4 pyTTS

## 

A python script which was used for implementing Speech to Text process. It works same way as gTTS work. Only difference was that their library.

**Performance Analysis**

For designing Text to Speech, we use different algorithm such as OCR and MSER and for reverse implementation we used gTTS and pyTTS. So, one was done on MATLAB and other was done on Python and also used different datasets for matching the characters. While using different platform the number of recognize text was different.

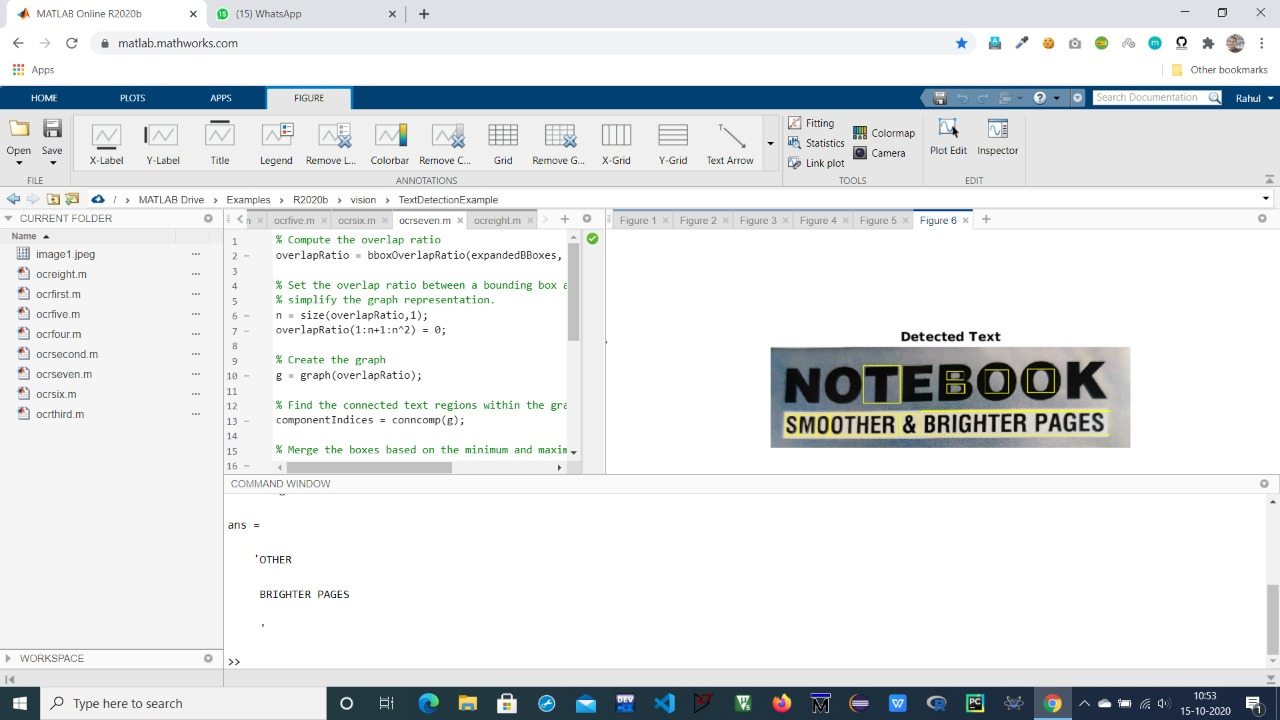
|  |  |
| --- | --- |
| **MATLAB** | **PYTHON** |
| **Recognized 20 characters out of 30 characters.** | **Recognized 20 characters out of 30 characters.** |
| **Accuracy ~ 66.667%** | **Accuracy ~ 100%** |
| **Data Loss ~ 33.333%** | **Data Loss ~ 0%** |

**Table 1-6**

## Results and Discussion

## 7.1 Output and Simulation

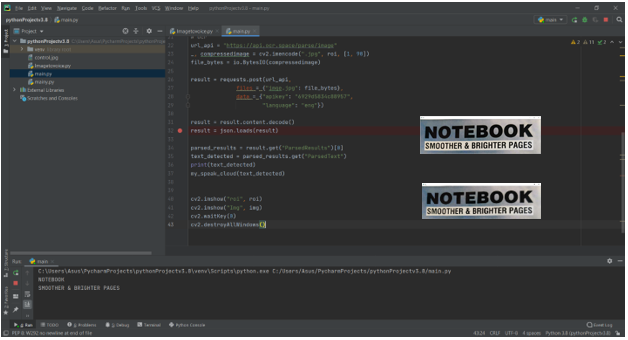
**MSER using MATLAB:**



**Fig. 27-7.1**

**Only some of the text gets detected and gets printed in terminal**

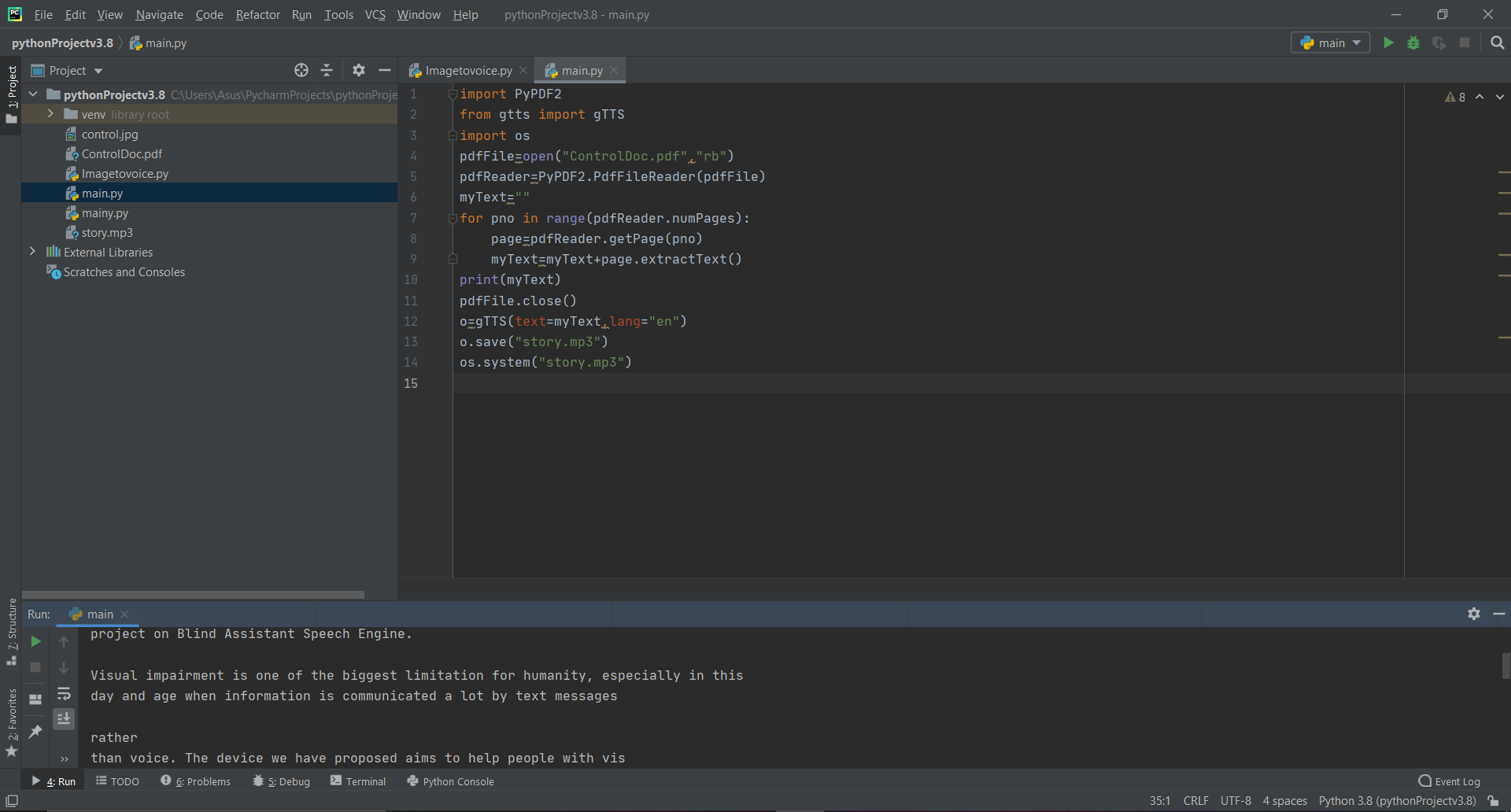
**OCR using PYTHON:**



**Fig. 28-7.1**

**All the text gets detected and gets printed in terminal**

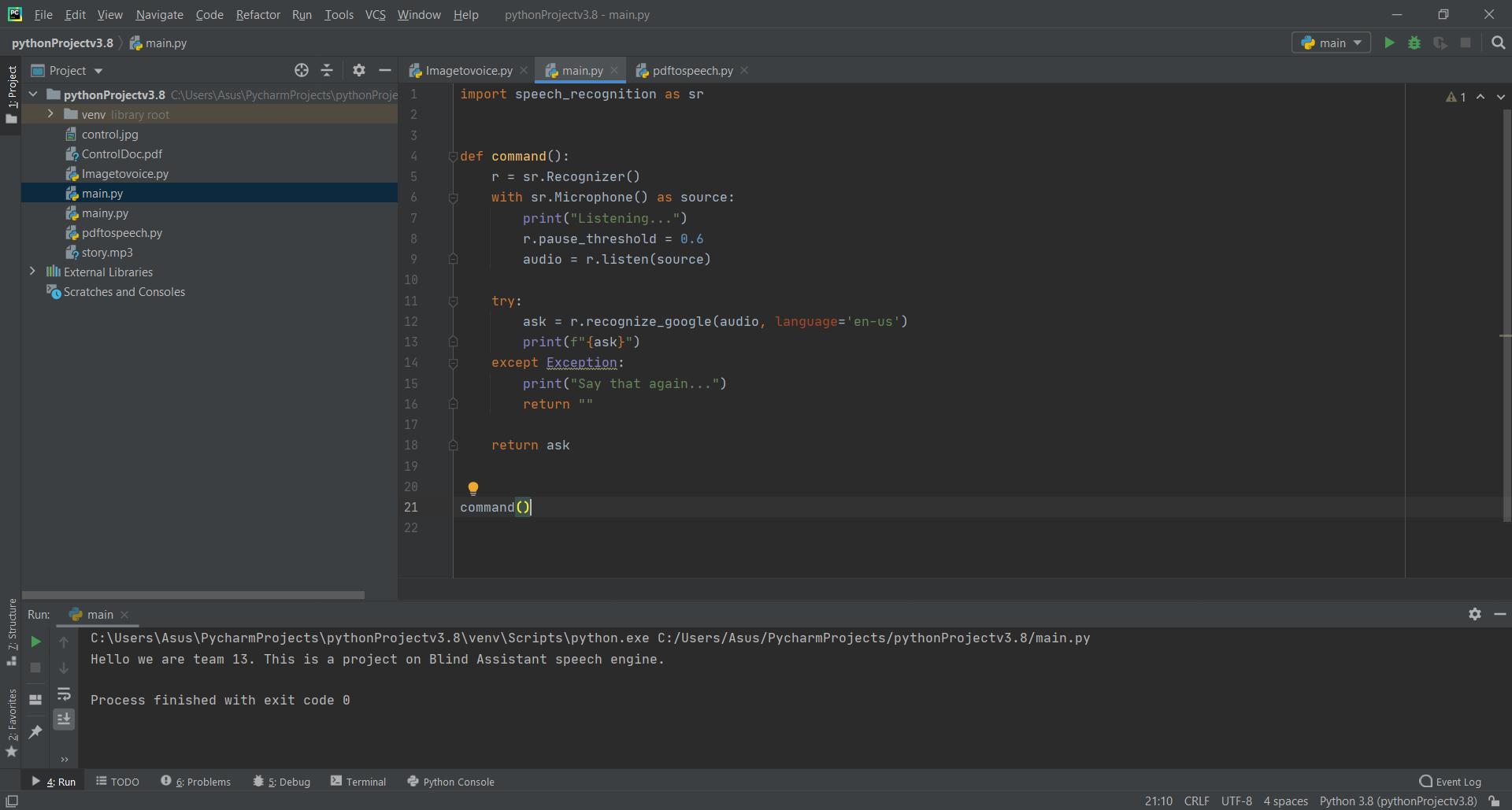
**gTTS – convert pdf text into Speech:**



**Fig. 29-7.1**

**Convert pdf text given into speech and gets printed in terminal**

**Speech to Text:**



**Fig. 30-7.1**

**Input speech given from microphone gets converted into text and gets printed into terminal**

**Conclusion**

Here, we used 2 different algorithm which was implemented in different platform one was MATLAB and other was PYTHON. And we can see in output that using PYTHON that means OCR algorithm which detects maximum number of characters from images as compared to MSER algorithm which was implemented in MATLB.

The difference in output was due to datasets difference which we are used while implementing the algorithm. In OCR there was a greater number of datasets present which increase the probability of matching the character which are present in input sample.

## 8.1 Future Implementation

This technique further implemented using best datasets of different languages and using Deep learning we can exactly recognize the text which are in the images, pdf format or any format. And also, for reverse implementation that is Speech recognition which is used everywhere like Google, Amazon Alexa, and also speech-based command. This technology was further implemented in defense purpose like fighter plane pilot having a mounted helmet which include this feature instead of manually controlling the system.

**Video Link:**

The videos of our complete project have been made and uploaded on google drive mentioned.

https://drive.google.com/drive/folders/1nKnDLoav\_Hp555Ph6WrKqNWlYj8XiK0q?usp=sharing

## List of References

1. L. Neumann and J. Matas, “Efficient Scene text localization and recognition with local character refinement,” Proceedings of the International Conference on Document Analysis and Recognition, ICDAR, vol. 2015- Novem, pp. 746–750, 2015.
2. L. M. Bergasa and J. J. Yebes, “Text Location in Complex Images,” International Conference on Pattern Recognition”, no. Icpr, pp. 617– 620, 2012.
3. https://www.researchgate.net/publication/235956427\_Optical\_Character\_Recognition\_by\_Open\_source\_OCR\_Tool\_Tesseract\_A\_Case\_Study
4. Geetanjali Adlinge, Shashikala kashid, Tejasvini, Virendrakumar Dhote, “Text Extraction form image using MSER approach”, IRJET vol. 2016- May.
5. Yong, Tien & Azad, Saiful & Rahman, Mohammed & Zamli, Kamal & Rabby, Gollam. (2018). A Highly Accurate PDF-To-Text Conversion System for Academic Papers Using Natural Language Processing Approach. Advanced Science Letters. 24. 7844-7849. 10.1166/asl.2018.13029.
6. Das, Prerana & Acharjee, Kakali & Das, Pranab & Prasad, Vijay. (2015). VOICE RECOGNITION SYSTEM: SPEECH-TO-TEXT. Journal of Applied and Fundamental Sciences. 1. 2395-5562.