Indian Institute of Technology
Jodhpur

B.Tech Project

Semester VI

Textual Video to Speech Interface

Author:
Abhay Kumar Singh
(UG201310003)
Deepshi Garg
(UG201313008)

Mentor:
Dr. Gaurav Harit

Abstract

Our aim is the development of an interface to textual information for the visually impaired that uses video, image processing, optical-character-recognition (OCR) and text-to-speech (TTS). The video provides a sequence of low resolution images in which text must be detected, rectified and converted into high resolution rectangular blocks that are capable of being analyzed via off-the-shelf OCR. To achieve this, various problems related to feature detection, mosaicing, binarization, and systems integration were solved in the development of the system.

For getting the image sequences, we will cut out frames at regular interval from the video, then pre-process that image to get a clearer image. After that, using image stiching tool of OpenCV Python, we will be making a single image of the whole text. Thereafter, that image will be given to the OCR (Tesseract), which further will give it's output to the Google Text To Speech engine (gTTS) to make a final audio speech output.

1 Introduction

1.1 Problem Statement

Information from books can be extracted in many ways. But videos provides us a way to make all the recording in a go and later extract required image. These images might not be apt for the OCR to extract all the text from that image because of some noise. Therefore, a still and super resolved image is extracted by image mosaicing and given as input to the OCR.

However, before such a system can be successfully implemented, several problems arising from text identification in images, low resolution sensors, image stabilization, text being warped, and others on the one hand, and practical system integration issues, on the other, have to be solved. We describe here the development of a preliminary prototype device for scene text acquisition and processing. The system consists of a computer, a digital Video Camera, an audio interface.

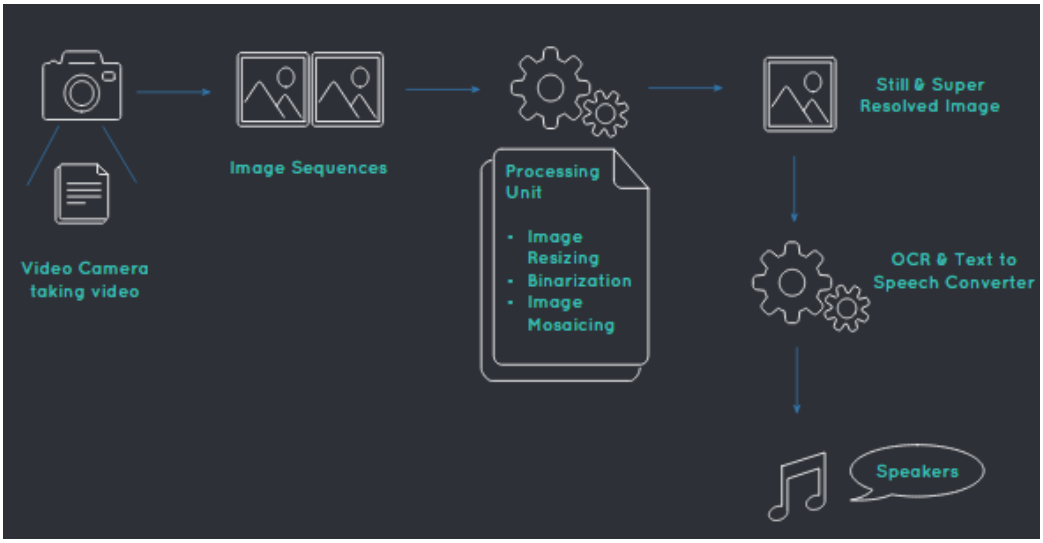


Fig. 1. Schematic Diagram

The camera captures text from the scene, with full control of focus and zoom that depends on orientation and quality of the document video. Video is 'conditioned' before feeding to the OCR, by performing operations such as image mosaicing, binarization, etc. The OCR software recognizes text from still and super-resolved images of whole text blocks, and the recognized text is read back by speech-to-text.

In general, off-the-shelf OCR systems are successful if:

- Document images are binarized and enhanced
- All Text has the same degree of skew and slant
- The text image has sufficient number of pixels per character ≥ 12

To calculate number of frames(patches), it is necessary to determine font-size of text, we then zoom into each patch to obtain the image that satisfy font-size constraint and capture the whole page while it is in-focus. Then, the super-resolved image from the mosaicing algorithm is interpreted by OCR and TTS.

Therefore, we will be making an inerface that will take input a video of texts, then we will process that video to get a sequence of frames. Further, those frames will be stiched together for form a single super resolved image. That image will be given to the OCR tool (Tesseract) as input and it will give a text file as output. That text file will be given to the Google TTS engine (gTTS) which will convert it into a audio speech.

1.2 Motivation and Scope

A very large number of our population suffer from low vision due to old age or any other factor. While this population may legally be classified as blind, they do have some residual vision that can be aided by prostheses and computer processing. In this paper, we describe the development of an interface that can help them to observe and receive textual information available in their environment.

2 Literature survey

- Tesseract is an optical character recognition engine for various operating systems. It is free software, released under the Apache License, Version 2.0, and development has been sponsored by Google since 2006.
- gTTS (Google Text to Speech): a Python interface to Google's Text to Speech API. Create an mp3 file with the gTTS module or gtts-cli command line utility. It allows for unlimited lengths of spoken text by tokenizing long sentences where the speech would naturally pause.
- OpenCV (Open Source Computer Vision) is a library of programming functions mainly aimed at real-time computer vision
- Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine.

3 Technologies Used

- Language : Python for feature implementation, CSS Bootstrap for building User Interface

- For Web Application : Python Flask (micro web framework written in Python based on the Werkzeug toolkit and Jinja2 template engine)
- Environment used : OpenCV (Open Source Computer Vision) Python which is a library of programming functions mainly aimed at real time computer vision

4 Methodologies

- Process the given video by sequencing its frames to form different images of the text. After experimenting on number of videos, we decided to extract every 80th frame from the input video (assuming the video is not too fast). This part of the project majorly used OpenCV python function VideoCapture() which gave output a descriptor that whenever called produces the next frame from the video and a boolean variable signifying success or failure of frame capturing. These all frames are saved in a folder names 'images' with every successive frame named as 'framenumber'.jpg.
- After capturing all the frames and storing it in a folder named 'images', we pre-processes images for better stitching. First, image is binarized based on type of image. Pytesseract OCR itself performs Otsu Binarization, but to improve the results we performed Adaptive Thresholding on the images. Two types of thresholding were considered for Binarization : Simple Threholding and Adaptive Gaussian Thresholding.
  - Simple Thresholding : If pixel value is greater than a threshold value, it is assigned one value (may be white), else it is assigned another value (may be black). The function used is cv2.threshold. First argument is the source image, which should be a grayscale image. Second argument is the threshold value which is used to classify the pixel values. Third argument is the maxVal which represents the value to be given if pixel value is more than (sometimes less than) the threshold value.
  - Adaptive Gaussian Thresholding : It decides how thresholding value is calculated. In case of Adaptive Gaussian Thresholding, threshold value is the weighted sum of neighbourhood values

where weights are a gaussian window.
cv2.ADAPTIVE_THRESH_GAUSSIAN_C is given as the 3rd argument of cv2.adaptiveThreshold() function to perform this binarization.

- After Binarizing all the images, they are resized using 'imutils' library of python. It has a series of convenient functions to make basic image processing functions such as translation, rotation, resizing, skeletonization, displaying Matplotlib images, sorting contours, detecting edges much more easier with OpenCV and Python.

- After getting all the images, images are sequentially sent to stitch() function. This function uses OpenCV Python function SIFT() to obtain keypoints in a particular image. SIFT stands for Scale Invariante Feature Transform. sift.detectAndCompute() function directly finds keypoints and descriptor in a single image. Then FlannBasedMatcher interface is used in order to perform a quick and efficient matching of 2 images by using the FLANN (Fast Approximate Nearest Neighbor Search Library) and if they matches considerably, they are stitched together and then dewarped. The final image is stored as mosaic.jpg

- Python has a library called pytesseract which is basically a python wrapper of Google's OCR tesseract. Text is extracted from the stitched images by giving input the image to image_to_string() function of pytesseract and saving it in a text.txt file.

- Conversion of this text into speech is done using Google Text To Speech engine (gTTS). This module requires Internet connectivity, hence sometimes creates time lag depending upon the speed of internet and amount of text.

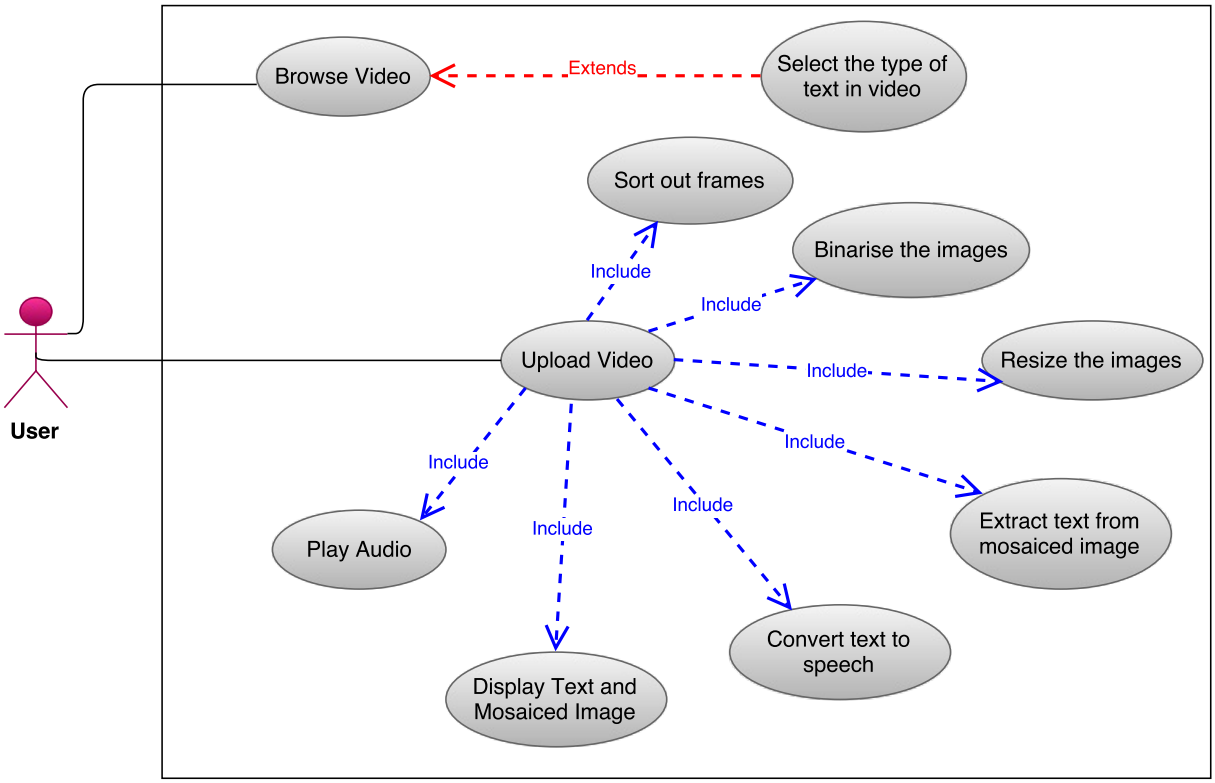
- These all modules are integrated together in one file of python and worked successfully.

- GUI of this software is implemented in Python flask. Flask is a micro web framework written in Python and based on the Werkzeug toolkit and Jinja2 template engine. 'index.html' file is run when application is started which asks for a video as input. After uploading the complete video, the software perform rest of the process in the background

and after it's completion, 'index1.html' file is run which shows the final stitched image with it's text and an option to play the audio. Information is transferred from flask to html using Markup and Jinja template.

- Final Python application is run on local host and tested with different set of inputs. The selected video is uploaded in 'uploads' folder and outcome related to that video is stored in a folder named after video's name.

5 UML Diagrams



' Fig. 2. Use Case Diagram

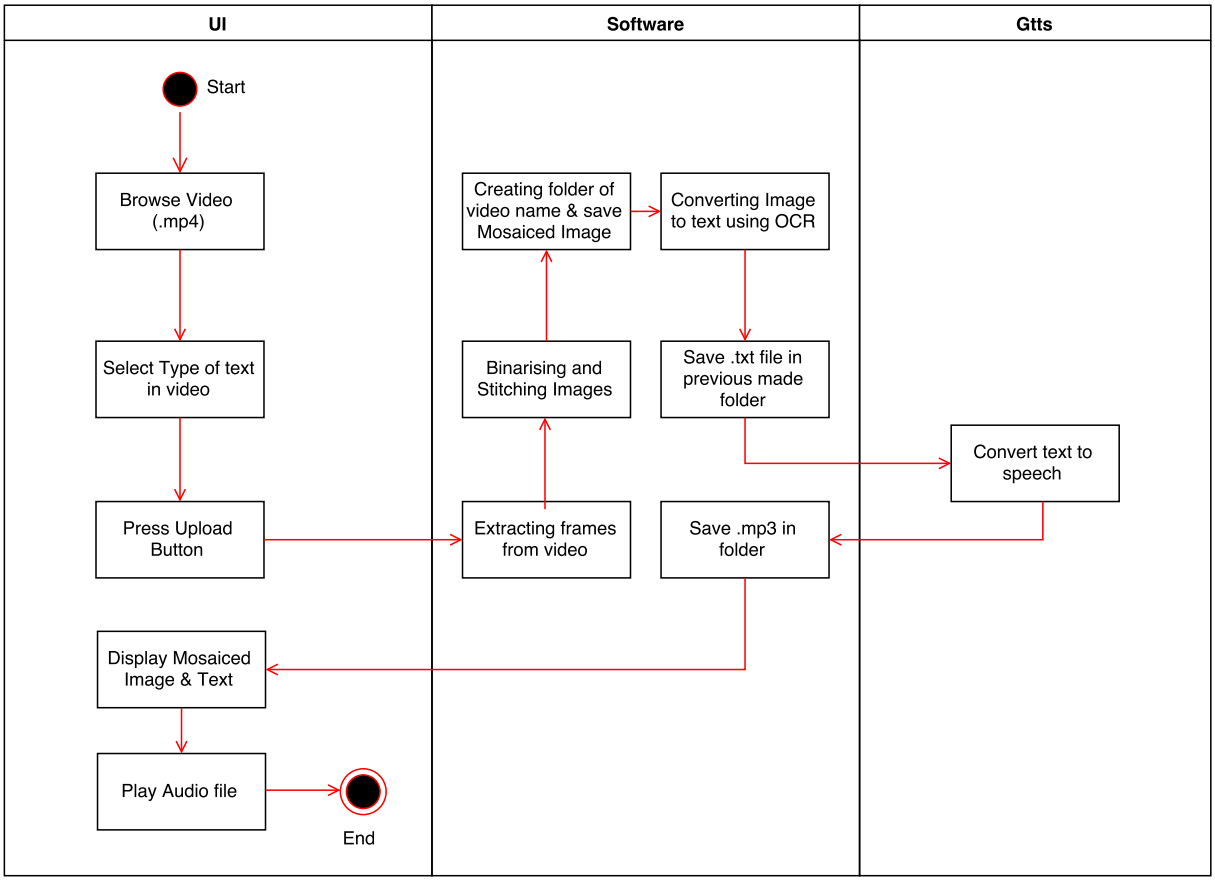


Fig. 3. Activity Diagram

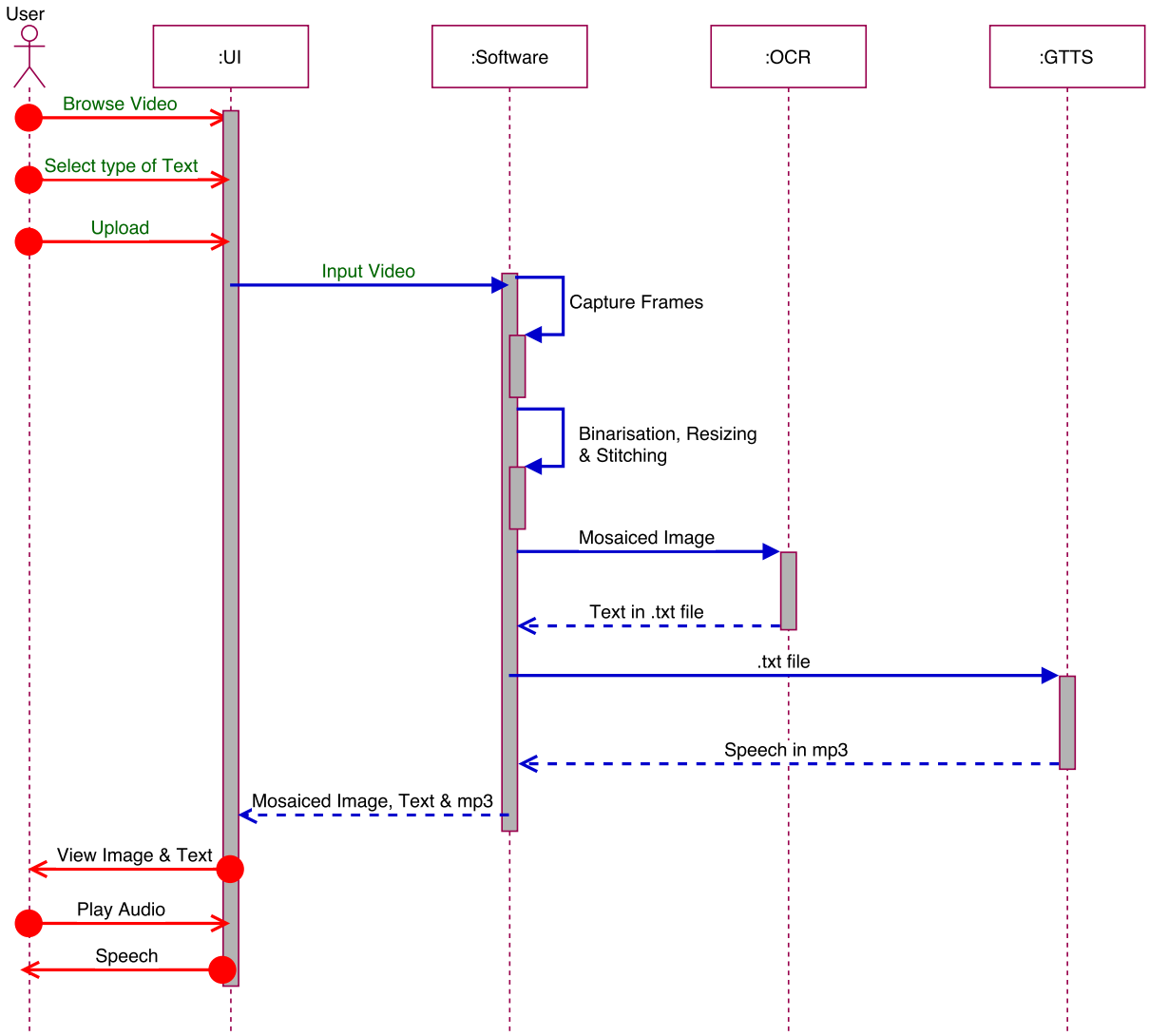


Fig. 4. Sequence Diagram

6 Experiments and Results

- When a video with text of very small font size was taken as input, OCR could not convert it to text efficiently.
- When the font style was chosen to be different from standard book styles, OCR could not recognize it.

- We used many input test cases differentiated in terms of radial distortion and brightness, and the best results were obtained with standard brightness of text on a plain sheet of paper with no distortion.
- When a long video with too many frames was considered, a blurred image after stitching was formed.
- This software gives best results for a video not longer than 15 seconds with text on a plain sheet of paper with no distortion and standard font size and font style.
- With extensive experimentation, we can also state that texts with relatively larger font size and bold font style can be binarised only with simple threshold binarisation technique, whereas for standard book texts, only adaptive gaussian threshold binarisation technique gave desired results.

7 Conclusion

- This software can be used as a reading tool by the visually challenged with no requirement of long term hard disk memory or any other special tools.
- Owing to the limitations of the OCR tool tesseract, only the texts with standard font styles are processed.
- This software accepts videos only in .mp4 format.
- Stitching large number of frames leads to segmentation fault, hence the software expects a video of maximum duration of 20 seconds.
- Text to speech conversion is performed using Google TTS which requires internet connectivity which makes the run time complexity of the software dependent on the speed of the connection.
- Video is expected to be clear and slow. Blurred and shaky videos lead to undesired results by the OCR.

References

[1] *Ali Zandifar, Ramani Duraiswami, Antoine Chahine, Larry S. Davis [Perceptual Interfaces and Reality Laboratory, University of Maryland, USA] [Video Based Interface to Textual Information for the Visually Impaired. Multimodal Interfaces, 2002. Proceedings. Fourth IEEE Conference*]

[2] *Research Paper by Silvio Ferreira, C´eline Thillou* [*From Picture to Speech : an Innovative Application for Embedded Environment.*]

[3] *Image Preprocessing for Improving OCR Accuracy* [*Wojciech Bieniecki, Szymon Grabowski and Wojciech Rozenberg*]

[4] *https://github.com/gali8/Tesseract-OCR-iOS/wiki/Tips-for-Improving-OCR-Results*

[5] *http://www.pyimagesearch.com/2016/01/11/opencv-panorama-stitching/*

[6] *http://flask.pocoo.org/ . [For creating the web application of the software.]*

[7] *http://docs.opencv.org/3.1.0/d0/de3/tutorial*$_{p}y_{i}ntro.htmlgsc.tab = 0.[For learning the basics of OpenCV Python.]$