SAN FRANCISCO STATE UNIVERSITY

ENGR 844 Project Report

Computer Vision for the Visually Impaired

Name	SFSU ID
Hariharan Venkatramanan	916403493
Poornima Eshwara	916436409

Project Overview

As per World Health Organization survey from August 2014, 285 million people are estimated to be visually impaired worldwide: 39 million are blind and 246 have low vision and about 90% of the worlds visually impaired live in low-income settings.

Our project aims at designing and implementing a cost-effective, easy-to-use, offline product that would help the visually impaired with 2 key features of face and text detection on Raspberry Pi. A comprehensive report of various parameters involved in implementing Computer Vision using is also tabulated.

Block diagram

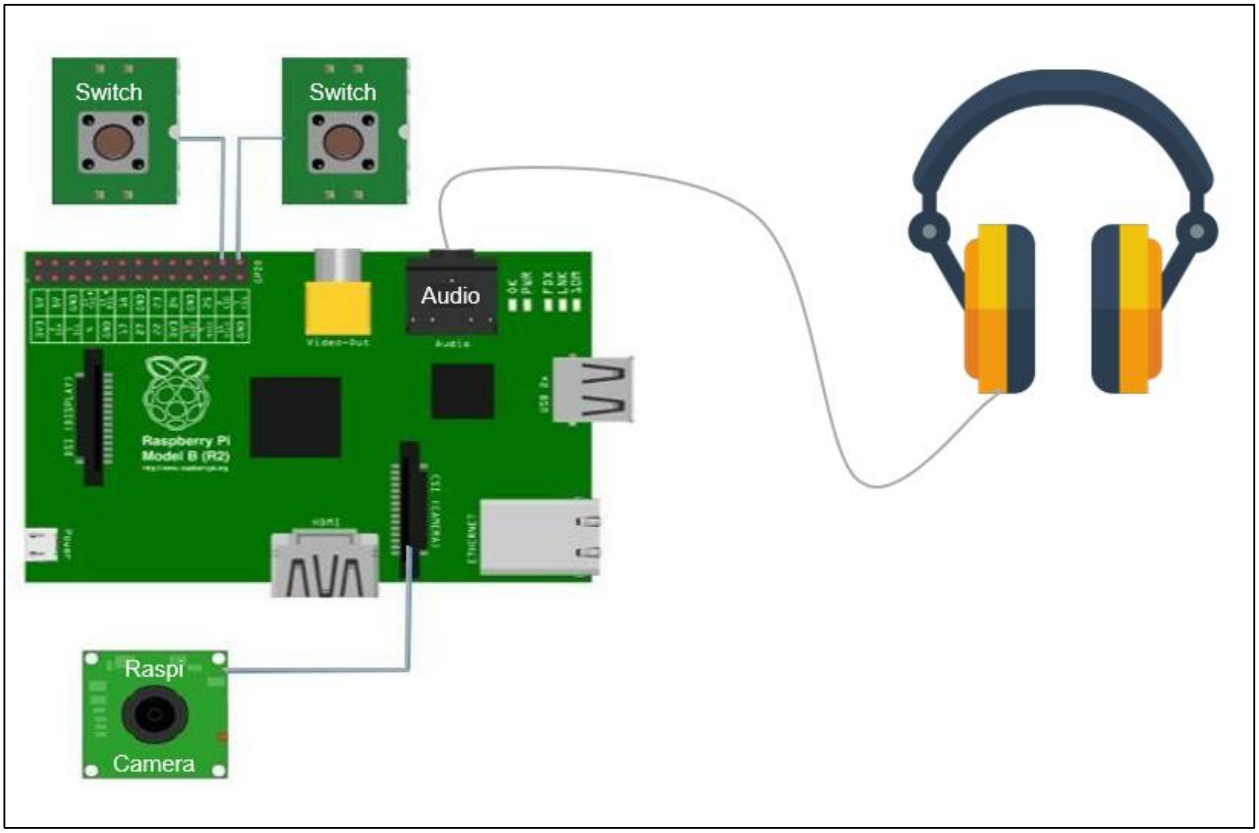

Figure 1 Block Diagram

List of Hardware and Software used

Raspberry Pi-2 B Specifications

- A 900MHz quad-core ARM Cortex-A7 CPU
- 1GB RAM
- 4 USB ports
- 40 GPIO pins

- Full HDMI port
- Ethernet port
- Combined 3.5mm audio jack and composite video
- Camera interface (CSI)
- Display interface (DSI)
- Micro SD card slot
- VideoCore IV 3D graphics core

Pi Camera

The Camera module has a five megapixel fixed-focus camera that supports 1080p30, 720p60 and VGA90 video modes, as well as stills capture. It attaches via a 15cm ribbon cable to the CSI port on the Raspberry Pi. Supports PiCamera Python library.

Software/Libraries

Python: Programming language used.

OpenCV: Library of Programming functions to support Computer Vision.

PiCamera: Python Library to interface Picamera and Raspberry Pi.

Pytesser: Python wrapper to support Tesseract which is an Optical Character Recognition (OCR) engine.

Festival: Library for Text to Speech conversion.

Haar Cascade: Cascade classifier (Training data) used for machine learning.

Design and implementation of individual modules as well as interfaces between modules

- Reading boards Detection - EXIT, STAIRS, ELEVATOR, RESTROOM
- Recognizing Faces and returning the count
- Audio output
- Switch controlled operation
  i. Mode 1 for Face Recognition
  ii. Mode 2 for detecting Text

Execution of the operation is through 3 main program files: Switch.py, Faces.py & Text.py

Switch.py

GPIO pins 20 and 21 are connected to two Push buttons for triggering the operations Face and Text Recognition. When either of the push buttons is pressed, respective program is executed and audio output is generated.

```
GPIO.setmode(GPIO.BCM)

#Push Button to trigger Face Recognition
GPIO.setup(20, GPIO.IN, pull_up_down=GPIO.PUD_UP)

#Push Button to trigger Text Recognition
GPIO.setup(21, GPIO.IN, pull_up_down=GPIO.PUD_UP)

count = 0

while True:
    Button_Faces = GPIO.input(20)
    Button_Text = GPIO.input(21)
    if Button_Faces == False:
        print "Calling the function for face recognition"
        subprocess.call('python Faces.py', shell=True )
        print "Finished calling Faces.py"
    if Button_Text == False:
        print "Calling the function for reading text"
        subprocess.call('python Text.py', shell=True )
        print "Finished calling Text.py"
```

Figure 2 Switch.py

Flowcharts

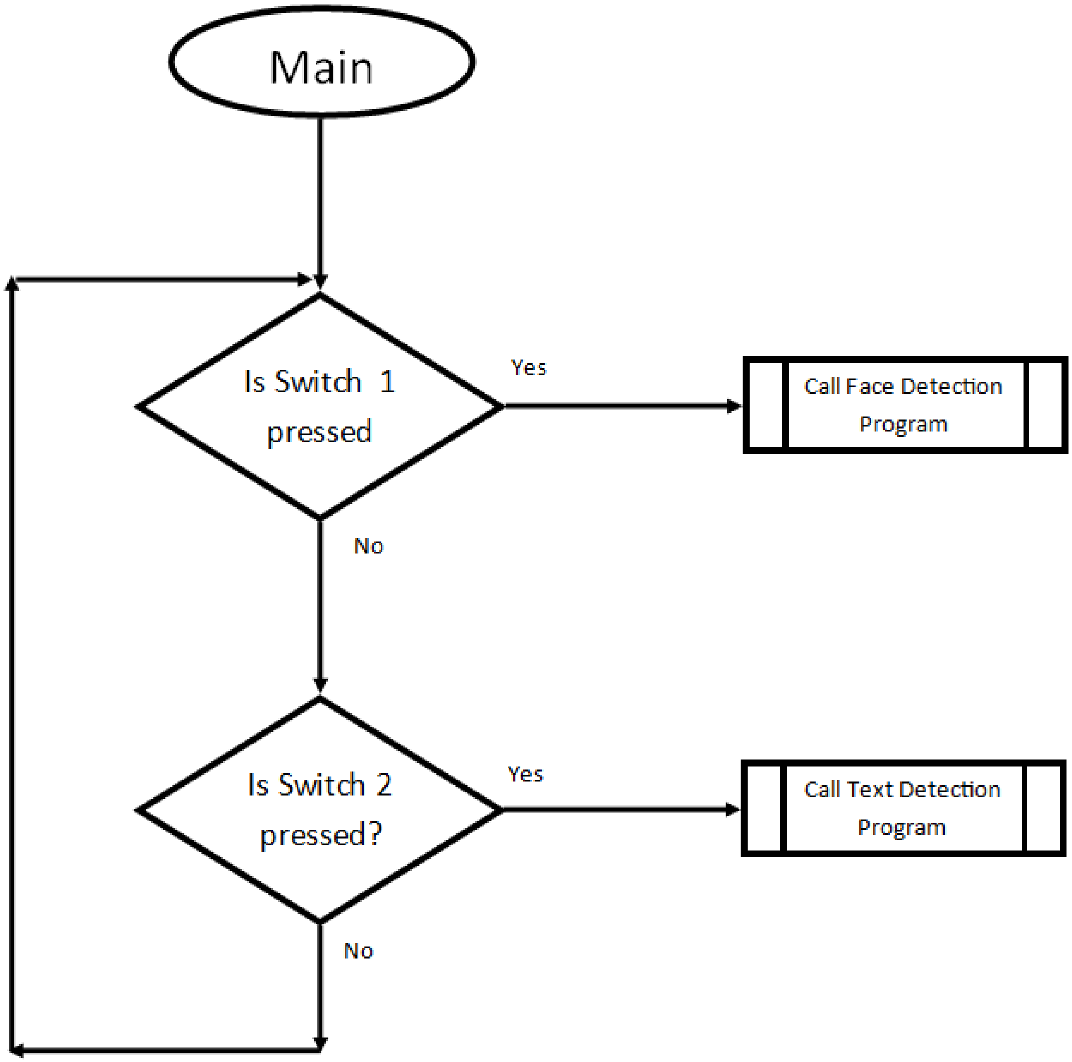

Figure 3 Flowchart: Main

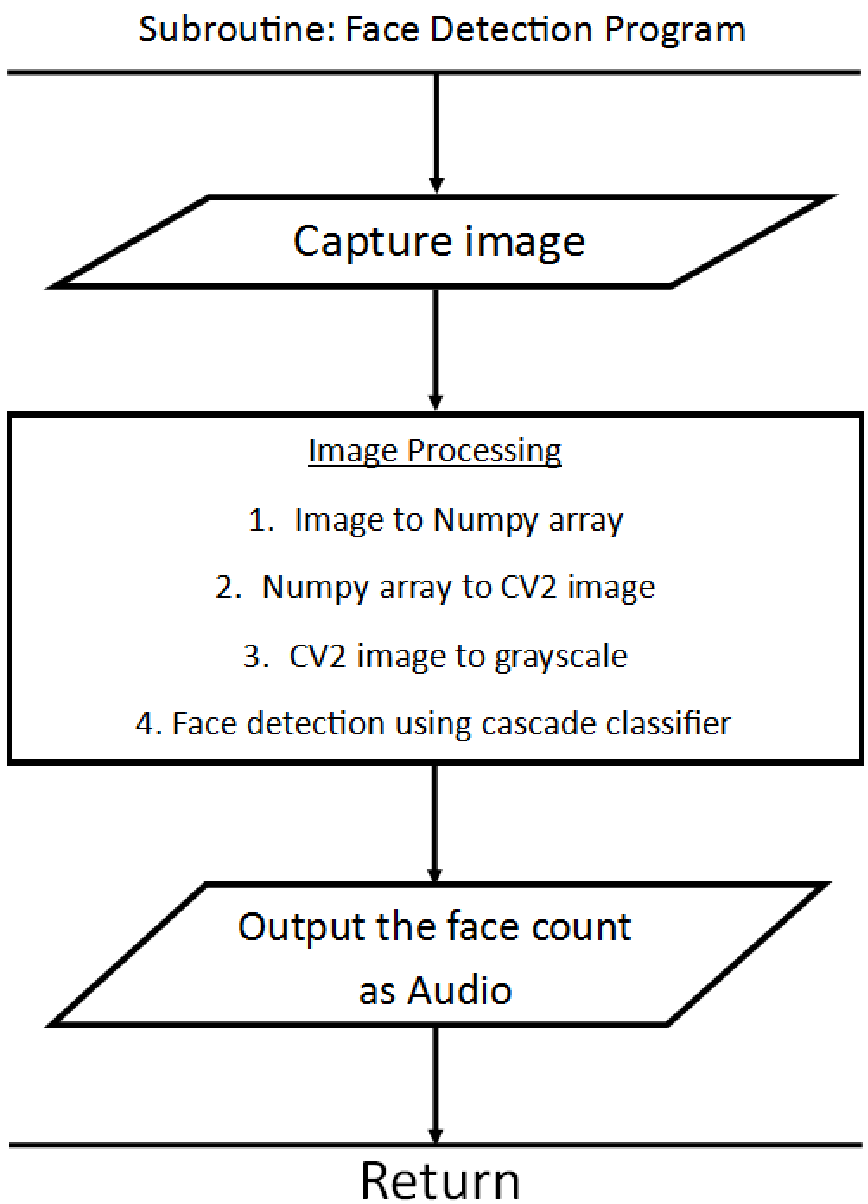

Figure 4 Flowchart Face Recognition Program

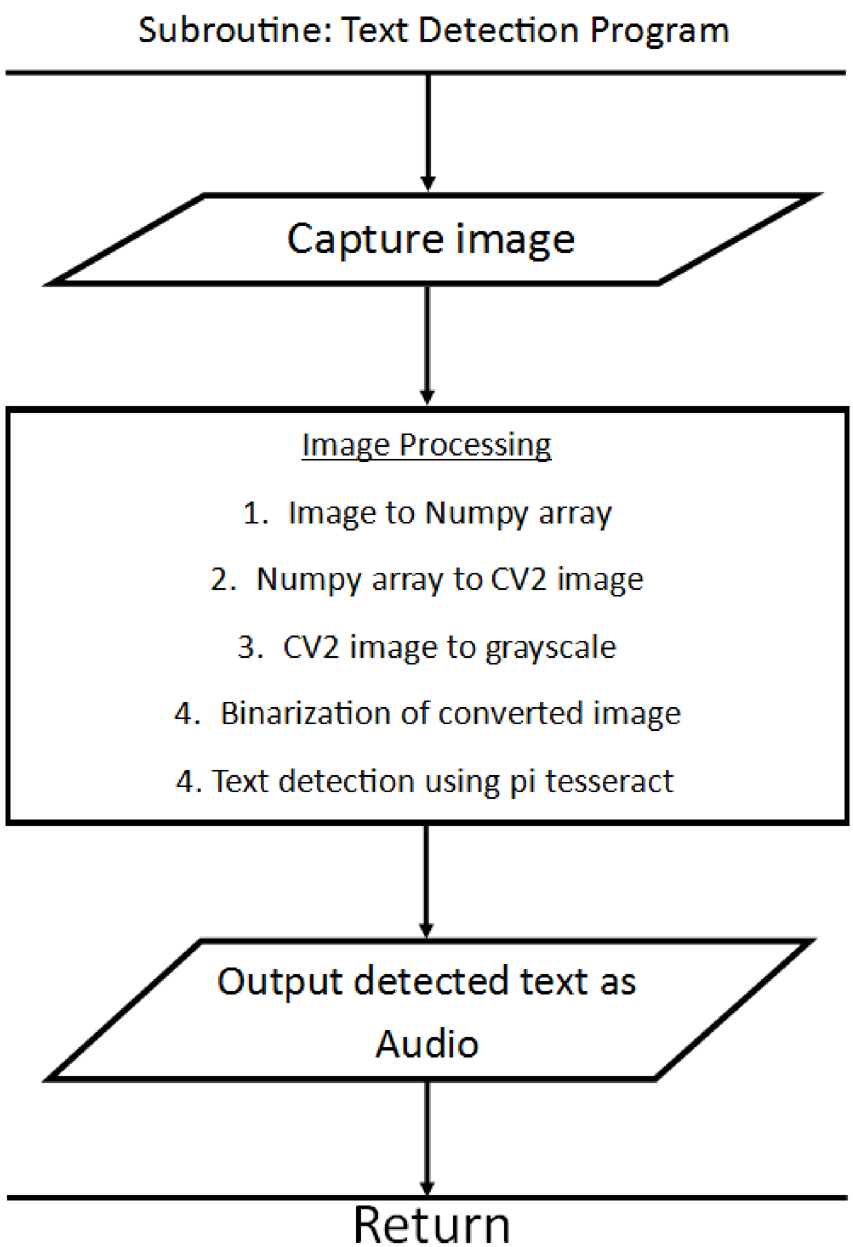

Figure 5 Flowchart Text detection program

Stages of Image Processing for TEXT detection

- ***Image Capture***
  Image is captured from PiCamera on the press of the button and saved as an OpenCV image.
- ***Grayscale Conversion***
  Grayscale conversion is a standard stage of operation in image processing that converts the pixel values from color to gray scale for efficient use of memory and processing speed.
- ***Binarization /Inverse Binarization***
  Binarization is conversion of grayscale image to binary image where each pixel value is mapped to either 0 or 1 depending on its original value compared to a given threshold value. It is useful in identifying text in the images with shadows and non-uniform illumination. In cases where light-colored text is present on a dark background, the inverse binarization is employed. Also depending on the illumination in the surroundings, different value of threshold will yield best result in recognizing the text, this can further be improvised by using adaptive threshold method.
- ***Use of PyTesser for Optical Character Recognition***
  This module will get the binarised image and converts characters in image into text form and saves them in a .txt file.
- ***Text to speech conversion for audio output***
  Festival is used as a text-to-speech engine. It works offline and has minimal processing delay compared to other similar libraries that support text-to- speech

Figure 6.1 Original Image

Figure 6.2 Gray Image

Figure 6.3 Binary Image

Face Recognition

OpenCV libraries are used for implementing computer vision, it's a blend of image processing and machine learning platforms put together to perform given task, face recognition.

Machine learning is a complex process where humungous number of training data are given to a system for it to learn from and process according to the generated training data vectors. These training data are combined into one single XML file called Cascade classifier file.

This cascade classifier files contains image vectors of all the images used for training. For facial recognition, around 3,000 positive

Cascade classifier is generated using thousands positive and negative images which highlights presence /absence of the various important features of the object of interest in an image.

There are various types of cascade classifiers depending on the type of vector values generated. If the vectors are integer, they fall into LBP category, while the Haar Cascade files have float datatypes. Haar files are more accurate but at the cost of extra processing time.

Tabulation of various parameters associated with Computer Vision Unit

The position of the camera

Ideal position to affix the camera such as over the head-attached to a cap, around the neck, waist or affixed to footwear or the wrist.

Around the neck is found to be the ideal position as it is relatively stable than wrist or feet and provide the optimal viewing angle in front of the holder.

Object detection range of the device

Range within which the camera and the sensors can capture the data and process them. Range of operation varies on various factors such as camera resolution, intensity of light, shaky or steady image capturing mode. Refer the Table for a detailed report

Rate at which the person with the device moves

If the person with the device is moving fast, the processing speed might not be able to process the data around and send back accurate results. So the upper limit of the speed at which the device functions effectively has to be determined. Moving too fast would also result in a blurred image disruption the image processing steps. From the moment the image is captured on the press of the button for either TEXT or face recognition, a duration of up to 1- 5 seconds.

Time taken for image processing

Being a real-time system the time-constraint is an important parameter. To determine the time taken by the device to process the image and send back the warning audio message to the person for various cases is tabulated.

Resolution (pixels)	**Response Time (sec)**	**Accuracy in Face Detection**	**Accuracy in different Light Conditions**		**Effects in different conditions**	
			Low Light	**Ambient Light**	**Shaky State**	**Steady State**
1280x960	Slow - ~5	+/- 1 (90%)	Poor	Very Good	Poor	Very Good
640x480	Fair - ~3	+/- 2 (85%)	Poor	Very Good	Poor	Very Good
320x240	Fast - ~2	80%	Very Poor	Good	Poor	Good
160x120	Instant – ~0	50%	Bad	Moderate	Poor	Moderate

Table.1 Tabulation of response time and efficiency for face detection

Results

Face Recognition

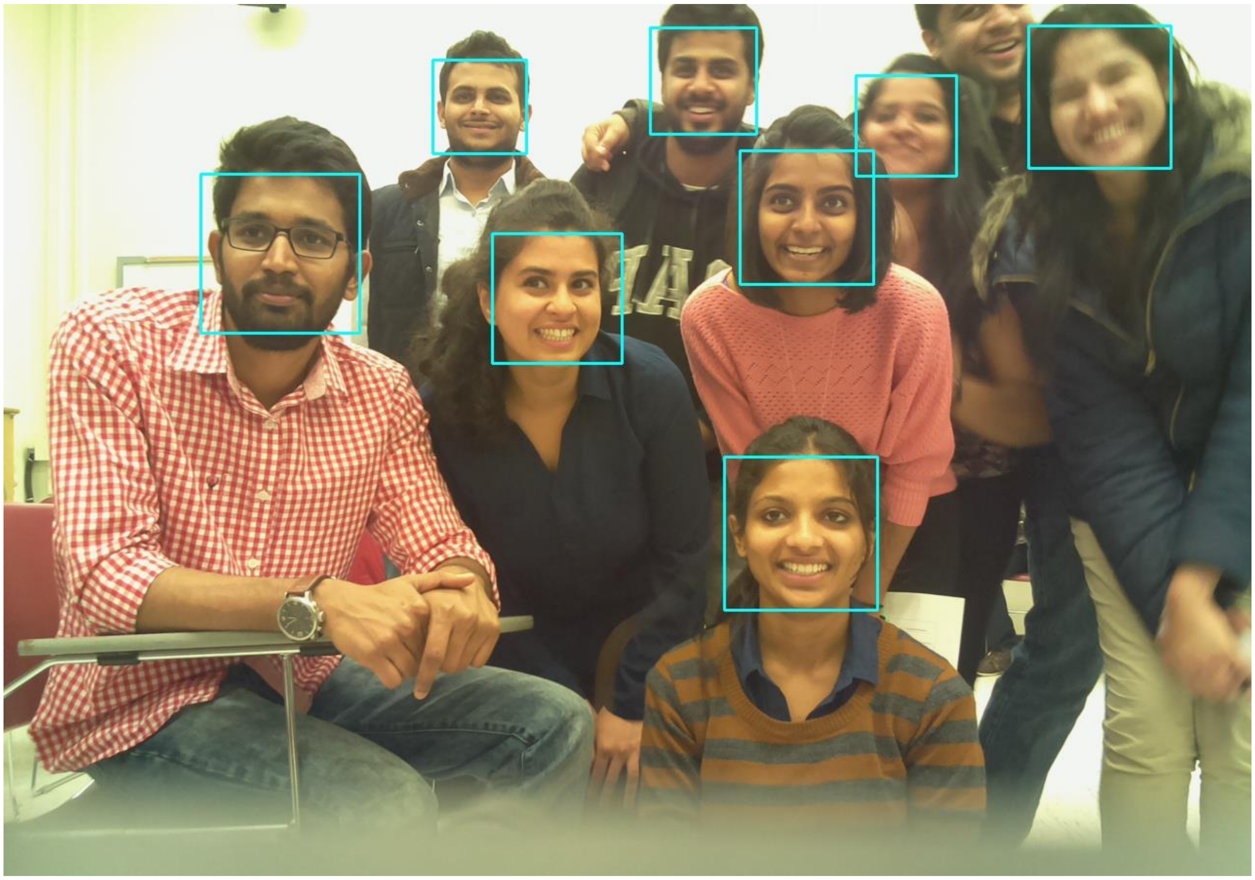

Figure 7 Face detection output

TEXT detection

Figure 8.1 Input Image1

```
Meet 1110
T1va"C Suxu»:s'1I.' 1

Evgluation Kit

inn ».
```

Figure 8.2 Output TEXT file for image1

Figure 8.3Input Image2: Exit

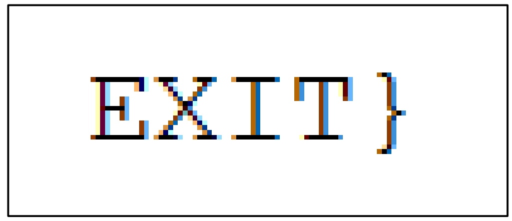

Figure 8.4 Output Text file for Image2 'EXIT'

Scope Extension

- Improving Face detection accuracy by feeding training data for side face.
- Possible improvements for low-light conditions
- Door detection/openings (& Glass doors!)
- Processing multiple Sign Boards together
- Distance detection using camera

References

http://www.who.int/mediacentre/factsheets/fs282/en/

http://users.iit.demokritos.gr/~bgat/cbdar2005.pdf

http://docs.opencv.org/3.1.0/d7/d8b/tutorial_py_face_detection.html#gsc.tab=0

http://machakux.appspot.com/blog/44003/making_speech_with_python

Text Detection in indoor/outdoor Scene images by B.Gatos, I. Pratikakis, K. Kepene and S. J.Perantonis