**“MULTIPLE COLOR DETECTION AND IMAGE TO SPEECH CONVERTOR”**

A Mini Project

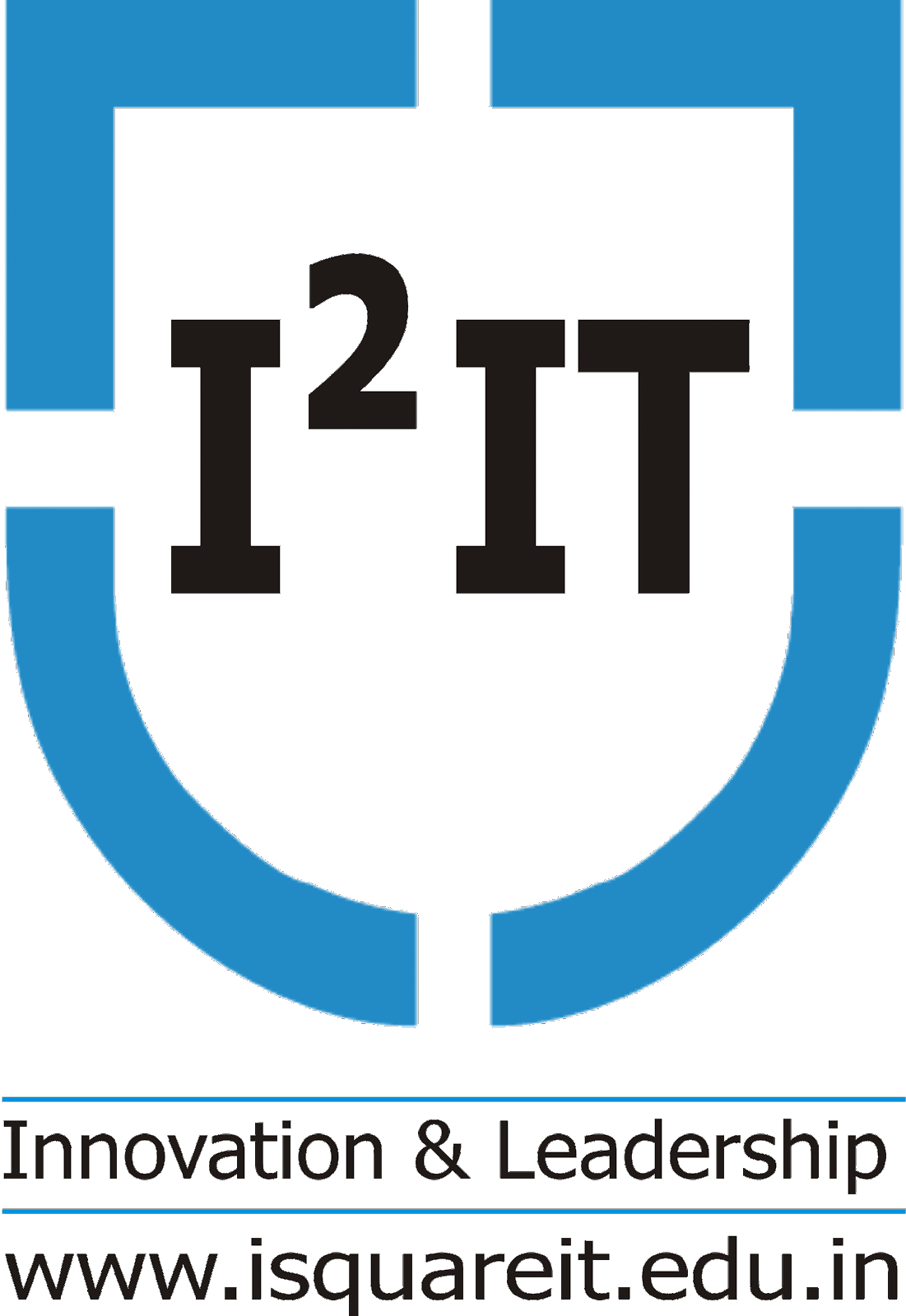
Submitted by

Sadiya Shaikh  (TE068.)

Tejal Jadhav  (TE069.)

Shreyash Tehare(TE070.)

THIRD YEAR COMPUTER ENGINEERING

  
  
 Department of Computer Engineering

Hope Foundation’s

International Institute of Information Technology

Hinjawadi, Pune – 411057

AY 2020-2021

**TABLE OF CONTENTS**

1. PROBLEM STATEMENT                                                                                                1

2. INTRODUCTION                                                                                                              1

3. TECHNOLOGY USED      2

4. METHODOLOGY                                                                                                             3

4.1 ALGORITHMS                                                                                                  3

4.2 SOFTWARE AND HARDWARE DETAILS                                                     7

4.3 LIBRARIES / PACKAGES USED                                                                    7

5. RESULTS                                                                                                                           08

6. CONCLUSION                                                                                                                  10

7.REFERENCES                                                                                                                   11

**1.       PROBLEM STATEMENT:**

Multiple Color Detection And Image To Speech Converter.

**2.       INTRODUCTION :**

**For image to speech:**

Machine replication of human functions like reading is an ancient dream. However, over the last five decades,machine reading has grown from a dream to reality.Today, there are already a few systems that have some promise for coversion use, like portable bar code readers designed to help blind people identify different products in an extensive product database can enable users who are blind to access information about these products through speech and Braille. But a big limitation is that it is very hard for blind users to find the what is written in image.Speech is probably the most efficient medium for communication between humans. To extract the text from image we use optical character recognition technique(OCR) tool for python sponsored by google (Pytesseract (Python-tesseract) ). Optical character recognition has become one of the most successful applications of technology in the field of pattern recognition and artificial intelligence. Character recognition or optical character recognition (OCR) is the process of converting images of machine printed or handwritten text (numerals, letters, and symbols) into a computer format text. Speech synthesis is the artificial synthesis of human speech. PIL adds image processing capabilities to your Python interpreter.gTTS (Google Text-to-Speech), a Python library and CLI tool to interface with Google Translate's text-to-speech API. Write spoken mp3 data to a file, a file-like object (bytestring) for further audio manipulation, or stdout .

**For colour detection:** OpenCV usually captures images and videos in 8-bit, unsigned integer, BGR format. Captured images can be considered as 3 matrices of BLUE, GREEN and RED with integer values ranges from 0 to 255. HSV color space is consists of 3 matrices, 'hue', 'saturation' and 'value'. In OpenCV value range for 'hue', 'saturation' and 'value' are respectively 0-179, 0-255 and 0-255. 'Hue' represents the color, 'saturation' represents the amount to which that respective color is mixed with white and 'value' represents the amount to which that respective color is mixed with black.

RGB color space describes colors in terms of the amount of red, green, and blue present. HSV color space describes colors in terms of the Hue, Saturation, and Value. In situations where color description plays an integral role, Sorting of objects is necessary in industries where products are manufactured on large scale. This process is simplified by automation. Our aim is to classify objects based on colour. Colour based sorting is used in many industries like crayon colour factory, agricultural machineries like rice sorter, beans sorter, peanut sorter etc.

. This can be done with help of an OpenCV. This design system can be implemented in various field for various purposes such as Defence, industrial purposes, games, automation, security, monitoring etc .Even these systems can also play a vital role in field of radar and navigating such as detecting, tracking of a moving coloured object etc Detection of colour play an important role even in the field of medical, such as detection of colour of skin ,identification of a face, recognizing license plate .Major purpose is to determining the colour of packages passing over a conveyor belt and also providing the quality of a particular colour packages by outnumbering. Which is fulfilled by a processing unit i.e. a raspberry pi along with a USB camera for capturing the package ,a display unit for displaying the count and the packaging along with its identified colour . which is done through OpenCV.

**3.  TECHNOLOGIES:**

**Technology used:**

**3.1** **Django framework:**

Django is an open source web application frame work written in Python. The primary goal of Django is to make the development of complex, data-based websites easier. Thus Django emphasizes the reusability and pluggability of components to ensure rapid developments. Django consists of three major parts: model, view and template.

**3.1.1 Model:**

Model is a single, definitive data source which contains the essential field and behavior of the data. Usually one model is one table in the database. Each attribute in the model represents a field of a table in the database. Django provides a set of automatically-generated database application programming interfaces (APIs) for the convenience of users.

**3.1.2 View:**

View is short form of view file. It is a file containing Python function which takes web requests and returns web responses. A response can be HTML content or XML documents or a “404 error” and so on. The logic inside the view function can be arbitrary as long as it returns the desired response. To link the view function with a particular URL we need to use a structure called URLconf which maps URLs to view fucntions.

**3.1.3 Template:**

Django’s template is a simple text file which can generate a text-based format like HTML and XML. The template contains variables and tags. Variables will be replaced by the result when the template is evaluated. Tags control the logic of the template. We also can modify the variables by using filters. For example, a lowercase filter can convert the variable from uppercase into lowercase.

**3.2 Python:**

Python is the language used to build the Django framework. It is a dynamic scripting language similar to Perl and Ruby. The principal author of Python is Guido van Rossum. Python supports dynamic typing and has a garbage collector for automatic memory management. Another important feature of Python is dynamic name solution which binds the names of functions and variables during execution.

**4.METHODOLOGY**

**4.1     ALGORITHM:**

**For colour detection:**

import cv2

import numpy as np

cap = cv2.VideoCapture(0)

cap.set(3,640)

cap.set(4,480)

cap.set(10,100)

while 1:

\_, img = cap.read()

hsv = cv2.cvtColor(img, cv2.COLOR\_BGR2HSV)

# Red color

red\_lower = np.array([136, 87, 111], np.uint8)

red\_upper = np.array([180, 255, 255], np.uint8)

# Blue color

blue\_lower = np.array([110,50,50], np.uint8)

blue\_upper = np.array([130,255,255], np.uint8)

# yellow color

yellow\_lower = np.array([22, 60, 200], np.uint8)

yellow\_upper = np.array([60, 255, 255], np.uint8)

# white color

white\_lower = np.array([0, 0, 200], np.uint8)

white\_upper = np.array([180, 20, 255], np.uint8)

# black color

black\_lower = np.array([0, 0, 0], np.uint8)

black\_upper = np.array([180, 255, 30], np.uint8)

# green color

green\_lower = np.array([50, 50, 50], np.uint8)

green\_upper = np.array([70, 255, 255], np.uint8)

# All color together

red = cv2.inRange(hsv, red\_lower, red\_upper)

blue = cv2.inRange(hsv, blue\_lower, blue\_upper)

yellow = cv2.inRange(hsv, yellow\_lower, yellow\_upper)

white = cv2.inRange(hsv, white\_lower, white\_upper)

black = cv2.inRange(hsv, black\_lower, black\_upper)

green = cv2.inRange(hsv, green\_lower, green\_upper)

# Morphological Transform,Dilation

kernel = np.ones((10, 10), "uint8")

red = cv2.dilate(red, kernel)

res\_red = cv2.bitwise\_and(img, img, mask=red)

blue = cv2.dilate(blue, kernel)

res\_blue = cv2.bitwise\_and(img, img, mask=blue)

yellow = cv2.dilate(yellow, kernel)

res\_yellow = cv2.bitwise\_and(img, img, mask=yellow)

white = cv2.dilate(white, kernel)

res\_white = cv2.bitwise\_and(img, img, mask=white)

black = cv2.dilate(black, kernel)

res\_black = cv2.bitwise\_and(img, img, mask=black)

green = cv2.dilate(green, kernel)

res\_green = cv2.bitwise\_and(img, img, mask=green)

# Tracking red

contours, hierarchy = cv2.findContours(red, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for pic, contour in enumerate(contours):

area = cv2.contourArea(contour)

if area > 300:

x, y, w, h = cv2.boundingRect(contour)

img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 255), 2)

cv2.putText(img, "Red Colour", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 255))

# Tracking blue

contours, hierarchy = cv2.findContours(blue, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for pic, contour in enumerate(contours):

area = cv2.contourArea(contour)

if area > 300:

x, y, w, h = cv2.boundingRect(contour)

img = cv2.rectangle(img, (x, y), (x + w, y + h), (255, 0, 0), 2)

cv2.putText(img, "Blue Colour", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX,0.7, (255, 0, 0),1)

# Tracking yellow

contours, hierarchy = cv2.findContours(yellow, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for pic, contour in enumerate(contours):

area = cv2.contourArea(contour)

if area > 300:

x, y, w, h = cv2.boundingRect(contour)

img = cv2.rectangle(img, (x, y), (x + w, y + h), (70, 255, 255), 2)

cv2.putText(img, "Yellow Colour", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.1, (70, 255, 255))

# Tracking white

contours, hierarchy = cv2.findContours(white, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for pic, contour in enumerate(contours):

area = cv2.contourArea(contour)

if area > 300:

x, y, w, h = cv2.boundingRect(contour)

img = cv2.rectangle(img, (x, y), (x + w, y + h), (255, 255, 255), 2)

cv2.putText(img, "White Colour", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.1, (255, 255, 255))

# Tracking black

contours, hierarchy = cv2.findContours(black, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for pic, contour in enumerate(contours):

area = cv2.contourArea(contour)

if area > 300:

x, y, w, h = cv2.boundingRect(contour)

img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 0, 0), 2)

cv2.putText(img, "Black Colour", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 0, 0))

# Tracking green

contours, hierarchy = cv2.findContours(green, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_SIMPLE)

for pic, contour in enumerate(contours):

area = cv2.contourArea(contour)

if area > 300:

x, y, w, h = cv2.boundingRect(contour)

img = cv2.rectangle(img, (x, y), (x + w, y + h), (0, 255, 0), 2)

cv2.putText(img, "Green Colour", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 0.7, (0, 255, 0))

cv2.imshow("Color Tracking", img)

if cv2.waitKey(10) & 0xFF == ord('q'):

cap.release()

cv2.destroyAllWindows()

break

**For image capturing:**

import cv2

import time

# SET THE COUNTDOWN TIMER

# for simplicity we set it to 3

# We can also take this as input

TIMER = int(5)

# Open the camera

cap = cv2.VideoCapture(0)

cap.set(3,640)

cap.set(4,480)

cap.set(10,100)

while True:

# Read and display each frame

ret, img = cap.read()

cv2.imshow('a', img)

# check for the key pressed

k = cv2.waitKey(125)

# set the key for the countdown

# to begin. Here we set q

# if key pressed is q

if k == ord('q'):

prev = time.time()

while TIMER >= 0:

ret, img = cap.read()

# Display countdown on each frame

# specify the font and draw the

# countdown using puttext

font = cv2.FONT\_HERSHEY\_SIMPLEX

cv2.putText(img, str(TIMER),

(200, 250), font,

7, (0, 255, 255),

4, cv2.LINE\_AA)

cv2.imshow('Taking Image', img)

cv2.waitKey(125)

# current time

cur = time.time()

# Update and keep track of Countdown

# if time elapsed is one second

# than decrese the counter

if cur - prev >= 1:

prev = cur

TIMER = TIMER - 1

else:

ret, img = cap.read()

# Display the clicked frame for 2

# sec.You can increase time in

# waitKey also

cv2.imshow('Taking photo', img)

# time for which image displayed

cv2.waitKey(2000)

# Save the frame

cv2.imwrite('camera.jpg', img)

# HERE we can reset the Countdown timer

# if we want more Capture without closing

# the camera

# Press Esc to exit

elif k == 27:

break

# close the camera

cap.release()

# close all the opened windows

cv2.destroyAllWindows()

**For Image To Speech:**

from gtts import gTTS

import os

import pytesseract as tess

from PIL import Image

tess.pytesseract.tesseract\_cmd = r'C:\Users\sadiya rafiq\AppData\Local\Tesseract-OCR\tesseract.exe'

img = Image.open('camera.jpg')

mytext1 = tess.image\_to\_string(img)

with open('abc.txt', mode='w') as file:

file.write(mytext1)

print(mytext1)

# Language we want to use

language = 'en'

myobj = gTTS(text=mytext1, lang=language, slow=False)

myobj.save("output.mp3")

# Play the converted file

os.system("start output.mp3")

**4.2     SOFTWARE REQUIREMENTS**

The System Environment includes the basic requirements related to the system are software and hardware requirements.

**Hardware Requirements**

* RAM :2GB(or)above
* Graphic card:500MB(or)above
* Hard disk:2TB

**Software Requirements**

* Programming Laguage:Python
* Version:Python3.6/7/8/9
* Library: Numpy ,OpenCV (python -library), Pytesseract(Python-tesseract), Python Imaging Library (PIL), gTTS (Google Text-to-Speech) ,pandas.
* Operating System :window

**4.3     LIBRARIES/PACKAGES USED:**

 there are five modules we will use for this project. To use these modules we have to install the necessary libraries. Library installation is a very easy step using pip. Pip is a package management tool. We will do the installation using the command-line interface. Here is the line to install all 3 libraries at once:

pip install numpy, pandas, opencv-python, Pytesseract, PIL, gTTS

After the installation is completed, we have to import them to our program. Open a new file in your favorite code editor. Here is the code on how to import the installed libraries:

**For colour detection:**

import numpy as np  
import pandas as pd  
import cv2

OpenCv is imported as cv2. And for other libraries, we imported them “as” so that it is easier to call them in the program.

**For image to speech conversion:**

from gtts import gTTS

import os

import pytesseract as tess

from PIL import Image

Perfect! Now, we can move to our next step, where we will define the image we want to use to test our color recognizer application and Image to speech conversion.

**5.    RESULT:**

Home page :



Color Output :

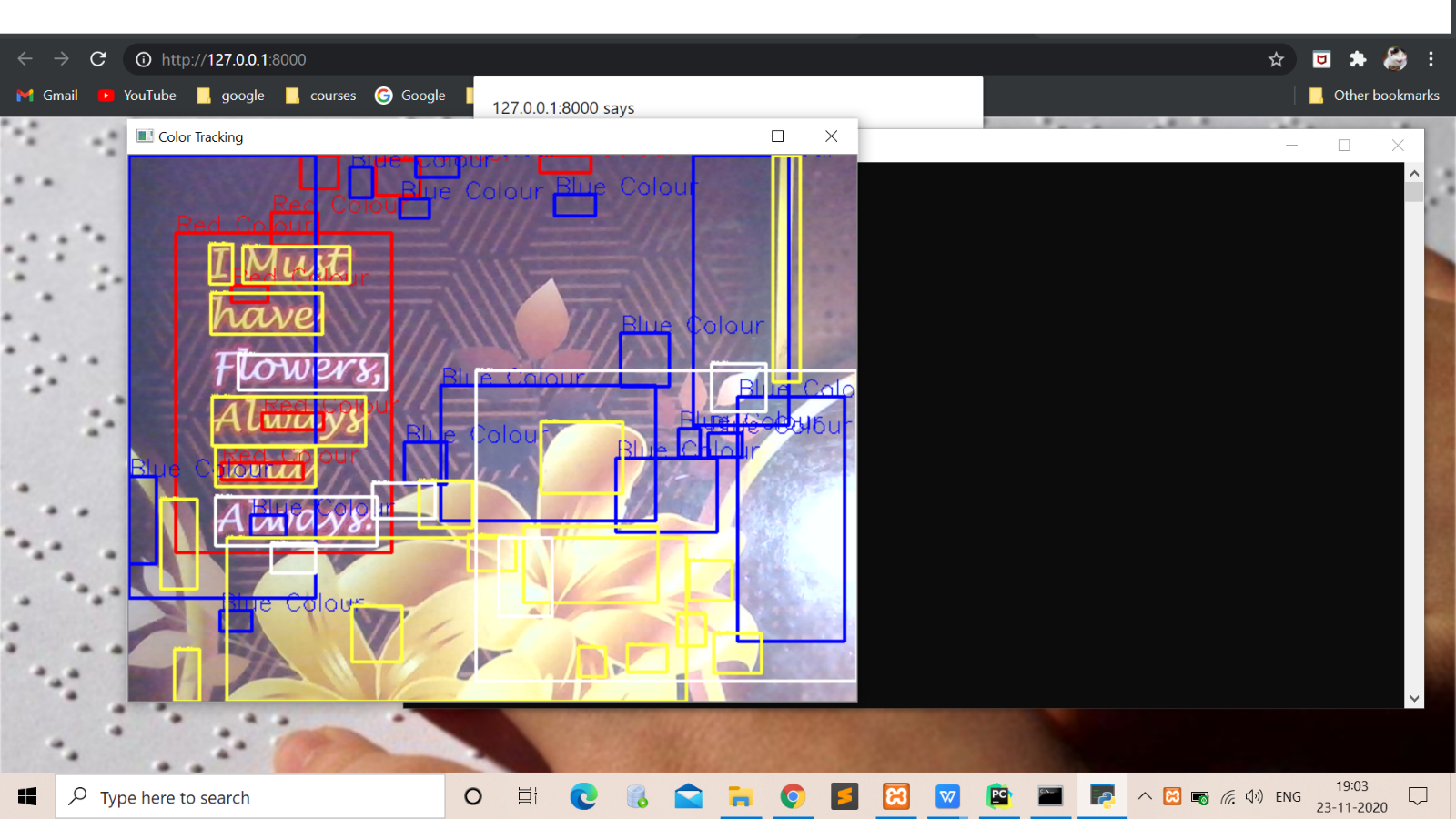
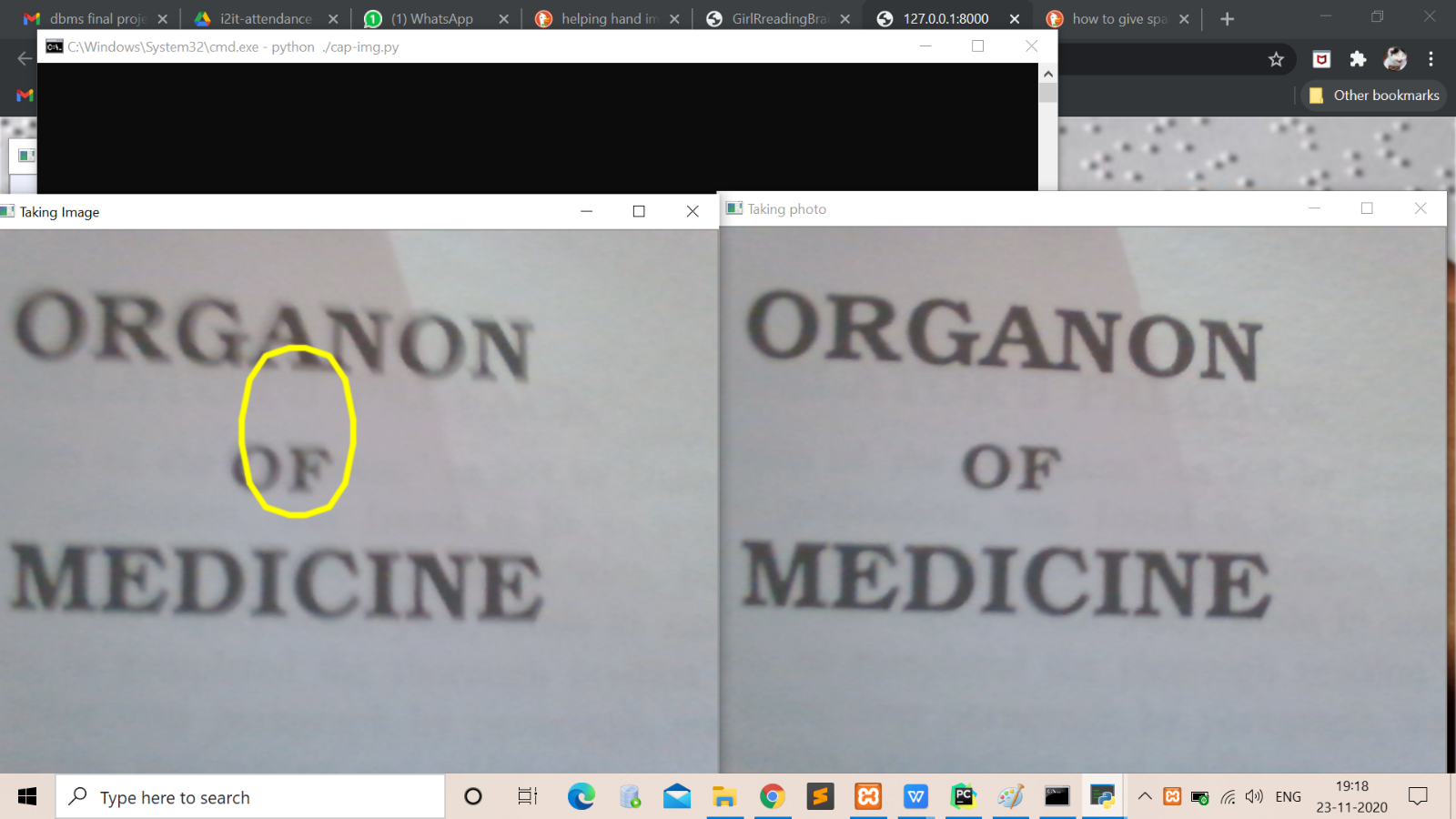
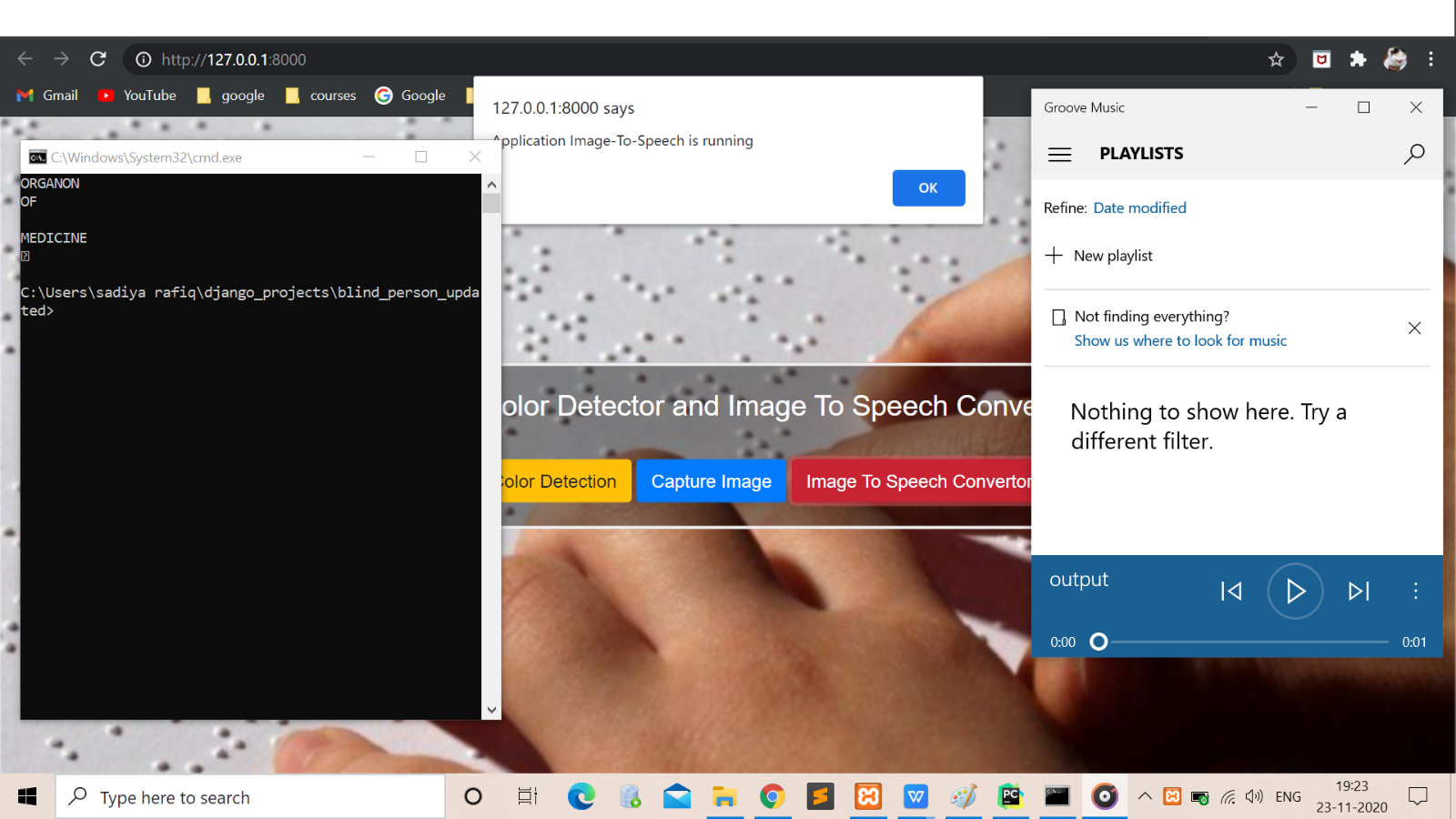


Image capturing :



**Image to speech :**



**6.CONCLUSION:**

**For colour detection:**  
 In particular, the focus was on the following important issues: (1) color invariance, (2) combining derivatives, (3) fusion of color models, (4) color saliency boosting, and (5) classifying image structures. To this end, the dichromatic reflection model has been outlined first. The dichromatic reflection model explains the RGB-values variations due to the image formation process. From the model, various color models are obtained showing a certain amount of invariance. Then, color (invariant) derivatives have been discussed. These derivatives include quasi-invariants which have proper noise and stability characteristics. To combine color derivatives into a single outcome, the color tensor has been used instead of taking themsum or Euclidean distance.

Tensors are convenient to describe color derivative vectors. Based on the color tensor, various image feature detection methods have been introduced to extract locale image structures such as edges, corners and circles. The experimental results of Canny color edge detection for several photometric quasi-invariants showed stable and accurate edge detection. Further, a proper model has been discussed to 24 *C*olor Image Processing: Emerging Applications select and weight color (invariant) models for discriminatory and robust image feature detection. The use of the fusion model is important as there are many color invariant models available. In addition, we used color to express saliency. It has been shown that after color saliency boosting, (less interesting) black and white structures in the image are ignored and more interesting color structures have been detected. Finally, a classification framework has been outlined to detect and classify local image structures based on photometrical and geometrical information. High classification accuracy is obtained by simple learning strategies.

In conclusion, this chapter provides a survey on methods solving important issues in the field of color feature detection. We hope that these solutions on low-level image feature detection will aid the continuing challenging task of handling higher level computer vision task such as object recognition and tracking.

**For Image To Speech:**

For the first stage project presentation, the required research work has been completed and the validation of project has been proved. The proposed system ensures to read text present in the image for assisting blind persons. Pre-processing part ensures efficient foreground extraction. But the system fails to extract the foreground when they possess a complex background and small text size. An improved algorithm for background subtraction can reduce the effects of complex backgrounds. The extracted text is then given to a spell corrector as OCR output is not perfect. After getting the corrected output we send it to the TTS engine which provides a speech output. . The future work will be concentrated on developing an efficient portable product that can extract text from any image enabling the blind people to read text present on the products, banners, books etc.

**7.        REFERENCES:**

[1] Portable Camera Based Assistive Text and Product Label Reading From Hand-Held Objects For Blind Persons. Chucai Yi, Student Member IEEE, Yingli Tian, Senior Member, IEEE, and Aries Arditi .

[2] Digital Image Processing, 3rd edition, Rafael C Gonzalez, Richard E Woods.

[3] “Text Pre-processing and Text Segmentation for OCR. International Journal of Computer Science Engineering and Technology”, ARCHANA A. SHINDE, D. pp. 810- 812, 2012.

[4] ”Optical Character Recognition" International Journal of Recent Technology and Engineering (IJRTE)”, R. Mithe, S. Indalkar and N. Divekar. ISSN: 2277- 3878,Volume- 2, Issue-1, March 2013.

[5] "High quality text-to-speech synthesis: a comparison of four candidate algorithms," Acoustics, Speech, and

1. Ajantha Devi and S Santhosh Baboo, "Optical Character Recognition on Tamil Text Image Using Raspberry Pi", International Journal of Computer Science Trends and Technology (IJCST), vol. 2, no. 4, Jul–Aug 2014.
2. Raja Venkatesan T, M. Karthigaa, P. Ranjith, C. Arunkumar and M. Gowtham, "Intelligent Transalate System for Visually Challenged People", International Journal for Scientific Research & Development (IJSRD), vol. 3, no. 12, 2016, ISSN 2321-0613.