Object Detection and Colour Detection GUI using Python

VATSAL PANDYA, SYBScIT, A032, 45207210028
DWITI PAREKH, SYBScIT, A033, 45207210020
HARSH PARMAR, SYBScIT, A034, 45207210056
AKHIL PATIL, SYBScIT, A036, 45207210054
JATIN MANDALIYA, SYBScIT, A053, 45207200054

A] Object Detection

ABSTRACT :
Object detection is a well-known computer technology connected with computer vision and image processing that focuses on detecting objects or its instances of a certain class (such as humans, flowers, animals) in digital images. There are various applications of object detection that have been well researched including face detection, character recognition, and vehicle calculator ,sentimental recognition. Object detection can be used for various purposes including retrieval and surveillance.helps to get a detail info of the grid data of where the object is on the picture and the % accuracy of the object being detected and the total number of objects that are accurately detected in the set picture .the various basic concepts used in object detection while making use of OpenCV library of python 3.10.0, imageai, improving the efficiency and accuracy of object detection that are presented.also a boundry line will be created on the object that is being detected plus the name of the detected object.

Objectives:

Object Recognition is a technology that lies under the broader domain of Computer Vision. This technology is capable of identifying objects that exist in images and videos and tracking them. Object Recognition also known as **Object Detection**.The two significant objectives of performing Object Recognition in the Python programming language using the **ImageAI** library.
that involve:

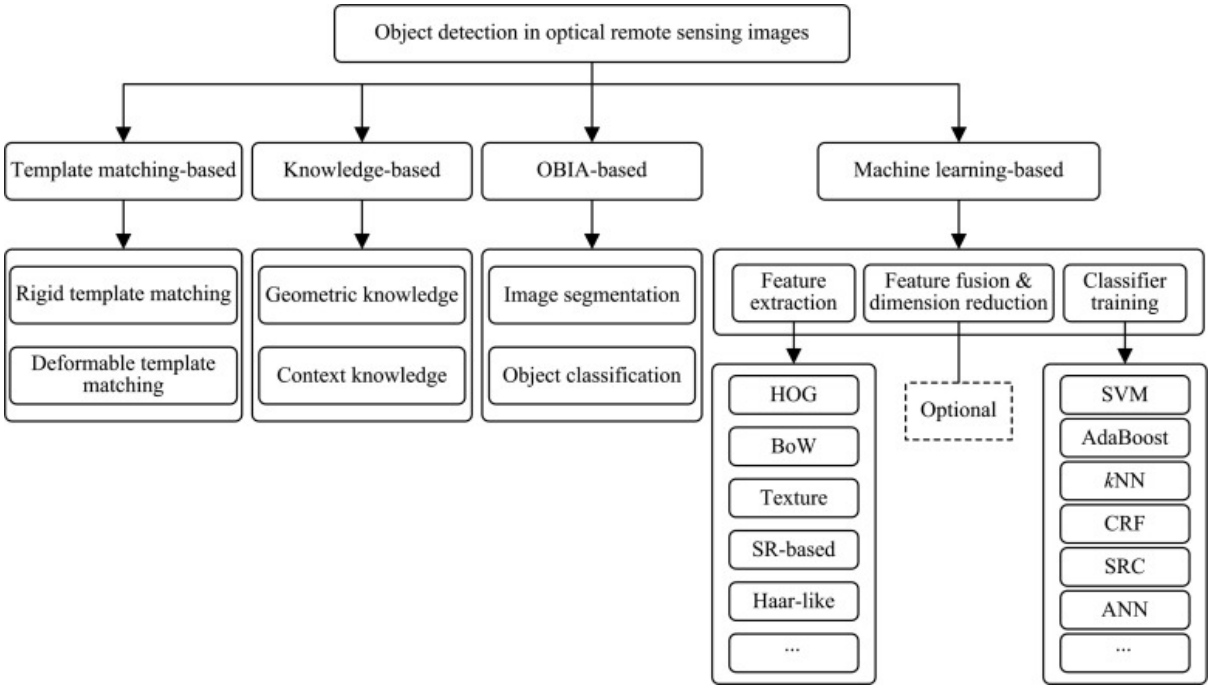
1. **Identification** of all objects that exist in an image

2. **Filtration** of the object that seeks attention

Keywords

Object Detection, IOU(Intersection over union), Python, tkinter,Imageai,face recognition,tensorflow,matplotlib ,threading

Detailed working of the Project :



Uses and purposes

- Medical feature detection in Healthcare.
  Can be used by the painters to get the accurate objects that are needed
- Can be used for survellience purposes
- Can be used by digital forensics to accuretly analyse footage and pictures of crime scences
- Can be used broadly for mutilple purposes by the military
- Can be used to create lists of objects detected with its coresponding accuracy and the grid location
- Can be used in VR games and AR (augmented reality)
- Can be used for detecting emotions on peoples faces to analyse their centiments.

Merits of Object detection to Real-world

Object detection is completely inter-linked with other similar computer vision techniques such as image segmentation and image recognition that assist us to understand and analyze the scenes in videos and images. Nowadays, several real-world use cases are implemented in the market of object detection which make a tremendous impact on different industries.

Autonomous driving :-

The primary reason behind the success of autonomous vehicles is real-time object detection artificial intelligence based models. These systems allow us to locate, identify and track the objects around them, for the purpose of safety and efficiency.

Crowd Counting :-

For heavily populated areas object detection application proves to be helpful to large enterprises and municipalities for tracking road traffic, violation of laws and number of vehicles passing in a particular time frame.

Anomaly detection :-

For instance, in agriculture, object detection models can accurately recognize and find the potential instances of plant disease. With the help of this, farmers will get notified and they will be able to prevent their crops from such threats.

Forest conservation – it can be used for forest,sanctury ,nursery conservation because program can accurately detect the no. of trees and their height corrsponding to the grid which can give you info abt the overall natural environmental condition and the growth or reduction in the environment.

Demerits of Object detection to Real-world :-

Dual Synchronization :-

To classify the image and position of the object, which is known as object localization. In order to address this problem, most developers often use a multi-tasking loss function to penalize both localization and misclassification errors.

Real-time detection speed

Fast speed of object detection algorithms has always been a major problem to classify and localize the crucial objects accurately at same

time to meet the real-time video processing. Over the years, several algorithms improved the test time from 0.02 frames per second to 155 fps.

Multiple aspects ratios and spatial scales

For several object detection applications, items of interest may appear in huge range of aspect ratios and sizes. Researchers proved numerous methods to ensure the detection algorithms which are able to recognize different objects at different views and scales.

Limited data

One of the undeniable facts to be considered is the limited amount of annotated data which becomes a hurdle to build an application. These datasets are specifically containing ground truth examples for dozens to hundreds of objects, while image classification datasets include approximately 100,000 different classes.

Scope for future enhancement :-

Geometric properties of the image can be included in the feature vector for recognition. Using unsupervised classifier instead of a supervised classifier for recognition of the object. The proposed object recognition system uses grey-scale image and discards the color information.

The colour information in the image can be used for recognition of the object. Colour based object recognition plays vital role in Robotics Although the visual tracking algorithm proposed here is robust in many of the conditions, it can be made more robust by eliminating some of the limitations as

In the Single Visual tracking, the size of the template remains fixed for tracking. If the sizeof the object reduces with the time, the background becomes more dominant than the object being tracked. In this case the object may not be tracked. Fully occluded object cannot be tracked and considered as a new object in the next frame. Foreground object extraction depends on the binary segmentation which is carried out byapplyingthreshold techniques. So blob extraction and tracking depends on the threshold value. Splitting and merging cannot be handled very well in all conditions using the single cameraduetothe loss

of information of a 3D object projection in 2D images.

For Night time visual tracking, night vision mode should be available as an inbuilt feature in CCTV camera. To make the system fully automatic and also to overcome the above limitations, infuture, multi-view tracking can be implemented using multiple cameras. Multi view tracking has the obvious advantage over single view tracking because of wide coverage range with different view in gangles for the objects to be tracked

the object Identification and Visual Tracking has been done through the use of ordinary camera. The concept is well extendable in applications like Intelligent Robots,Automatic Guided Vehicles, Enhancement of Security Systems to detect the suspicious behaviouralong with detection of weapons, identify the suspicious movements of enemies onboarderswiththe help of night vision cameras and many such applications.

In the proposed method, background subtraction technique has been used that is simpleandfast.This technique is applicable where there is no movement of camera. Object identification task with motion estimation needs to be fast enough to be implemented for the real time system. Still there is a scope for developing faster algorithms for object identification.
This program can be repurposed to be used in analysis of iris photograph data
Can be repurposed to be used to nalyse comments section data on various social media sites by using emojis as references to people emotions towards the presented content

CONCLUSION:

By using this thesis and based on experimental results we are able to detect obeject more precisely and identify the objects individually with exact location of an obeject in the picture in x,y axis.Thispaperalso provide experimental results on different methods for object detection and identification and compares each method for their efficiencies

Source Code:

```
from tkinter.messagebox import showinfo
```

```
from imageai.Detection import
ObjectDetection
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
from PIL import ImageTk, Image as
img
import pyttsx3
from tkinter import *
import time
from tkinter.filedialog import
askopenfile
import threading



detection = ""

root = Tk()
root.title("OBJECT DETECTION
USING PYTHON")
root.resizable(False,False)
HEIGHT = 600
WIDTH = 900
root.geometry(f"{WIDTH}x{HEIGH
T}+5+120")



font = ("Arial",30)

Label(root,background=
"#FFFF8F",text="OBJECT
DETECTION IN
PYTHON",font=font).pack()
image_1 =
ImageTk.PhotoImage(img.open(r"sa
mple1.jpg").resize((220, 150)))
image_2 =
ImageTk.PhotoImage(img.open(r"sa
mple2.jpeg ").resize((220, 150)))



detector = ObjectDetection()

e = pyttsx3.init()

model_path = r"YONO\yolo.h5"
input_path = ""


def set_img_1():
    global input_path
    input_path = r"sample1.jpg"
    d = detect()
    d.start()

def set_img_2():
    global input_path
    input_path = f"sample2.jpeg"
    # detect(input_path)
    d = detect()
    d.start()



def
perform_detection_on_img_frm_loca
l_disk():
    f_type = (
        ("image files","*.jpg"),
        ("image files","*.jpeg"),
        ("image files","*.png")
    )
    file =
askopenfile(filetypes=f_type,title="O
pen PNG,JPEG or JPG file")
    file_path = file.name

    global input_path
    input_path = file_path

    # detect(input_path)
    d = detect()
    d.start()



output_path =
r"OUTPUT/sample.jpg"

img_path = input_path
```

```
detector.setModelTypeAsYOLOv3()
detector.setModelPath(model_path)
detector.loadModel()



class
print_detect_details_in_screen(thread
ing.Thread):
   def run(self):
      def plot_input():
         img =
mpimg.imread(input_path)
         imgplot = plt.imshow(img)
         plt.title('Input Image')
         plt.show()


      def plot_output():
         img2 =
mpimg.imread(output_path)
         imgplot2 = plt.imshow(img2)
         plt.title('Output Image')
         plt.show()


      objects = []

      for eachItem in detection:
         if not eachItem['name'] in
objects:

objects.append(eachItem['name'])

      if(len(objects)):
         # plot_input()
         plot_output()
         sub_root = Tk()

sub_root.geometry("500x540")
         sub_root.title("Detected
Details")

sub_root.resizable(False,False)

         Label(sub_root,text =
"Objects with its percentage
probability").place(x = 2,y = 2)

         f1 = Frame(sub_root)
         f1.place(relx=0.01,rely=0.04)



list_of_objects_wth_percentage_prob
aility = Listbox(f1,width=27)

         s_b_1 = Scrollbar(f1)
         s_b_1.pack(side =
RIGHT,fill=Y)
         j = 1

         for eachItem in detection:
            print(eachItem["name"] , " :
",
eachItem["percentage_probability"])
            text =
str(eachItem["name"]) + " : " +
str(eachItem["percentage_probability
"])

list_of_objects_wth_percentage_prob
aility.insert(j,text)
            j+= 1


list_of_objects_wth_percentage_prob
aility.pack()

list_of_objects_wth_percentage_prob
aility.config(yscrollcommand=s_b_1.
set)
```

```
s_b_1.config(command=list_of_objects_wth_percentage_probaility.yview)
        Label(sub_root,text = "The Detected Objects are follows").place(relx=0.5,y = 2)


        f2 = Frame(sub_root)
        f2.place(relx=0.51,rely=0.04)


        s_b_2 = Scrollbar(f2)
        s_b_2.pack(side=RIGHT,fill = Y)

        list_of_objects = Listbox(f2,yscrollcommand=s_b_2.set,width=25)

        i = 1
        for object in objects:
            list_of_objects.insert(i , object)
            i += 1

        list_of_objects.pack()


s_b_2.config(command=list_of_objects.yview)


        Label(sub_root,text = f"Total Objects in an Image are {len(detection)}").place(rely=0.45,relx=0.01)

        total_numbers = [0 for i in objects]

        for item in detection:
            index = objects.index(item['name'])
            total_numbers[index]+=1

        f3 = Frame(sub_root)
        f3.place(rely=0.5,relx=0.01)

        s_b_3= Scrollbar(f3)
        s_b_3.pack(side = RIGHT,fill=Y)

        counted_objects = Listbox(f3,yscrollcommand=s_b_3.set,width=40)
        counted_objects.pack()


        l = 1
        for i in range(len(objects)):

counted_objects.insert(l,f"Total Number of \"{objects[i]}\" is {total_numbers[i]}")
            l += 1

        counted_objects.pack()

s_b_3.config(command=counted_objects.yview)

        def speak():
            e.say("The Detected Objects are")
            e.runAndWait()

            for object in objects:
                e.say(object)
                e.runAndWait()


            e.say(f"Total Objects in an Image are {len(detection)}")
            e.runAndWait()


            for i in range(len(objects)):
```

```
                e.say(f"Total Number of
{objects[i]} is {total_numbers[i]}")
                e.runAndWait()

            # e.say("Objects with its
percentage probability")
            # e.runAndWait()

            # for eachItem in detection:
            #     text =
str(eachItem["name"]) + " : " +
str(eachItem["percentage_probability
"])
            #     e.say(text)
            #     e.runAndWait()



        t =
threading.Thread(target=speak)
        t.start()




        sub_root.mainloop()

      else:
        plot_input()
        e.say("No Any Objects
Detected From the Image")
        e.runAndWait()
        showinfo("MASSAGE","No
Any Objects Detected From the
Image")

        plot_output()




class detect(threading.Thread):
   def run(self):
        global detection
        try:
            detection =
detector.detectObjectsFromImage(inp
ut_image=input_path,output_image_
path=output_path)
        except Exception:
            print("\nWrong Name/Path of
Input Image or Input Image Not
Found.Please Enter proper name with
full path\n")
            e.say("Wrong Name/Path of
Input Image or Input Image Not
Found.Please Enter proper name with
full path")
            e.runAndWait()
            exit(1)

        #
print_detect_details_in_screen(detecti
on)
        print_detail =
print_detect_details_in_screen()
        print_detail.start()


font = ("Helvetica",18)

Label(root,text="SAMPLE
1",font=font, background=
"#FFFF8F").place(x = 35,y = 140)
Button(root,image=image_1,comman
d=set_img_1).place(x = 165,y = 80)

Label(root,text="SAMPLE
2",font=font, background=
"#FFFF8F").place(x = 450,y = 140)
Button(root,image=image_2,comman
d=set_img_2).place(x =580,y = 80)


font = ("Helvetica",15)
Label(root,text="CLICK BUTTON
TO DETECT OBJECT IN IMAGE
```

```
FROM LOCAL IMAGE",font=font,
background= "#FFFF8F").place(x =
100,y = 260)

btn  = Button(root,text="DETECT
OBJECT FROM IMAGE FROM
THE LOCAL
DISK",wraplength=130,command=pe
rform_detection_on_img_frm_local_
disk, background= "#FFFF8F")
btn.place(y = 295,x = 353)

root.config(background= "#FFFF8F")

root.mainloop()
```

References:

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.greenteapress.com/thinkpython/thinkpython.pdf

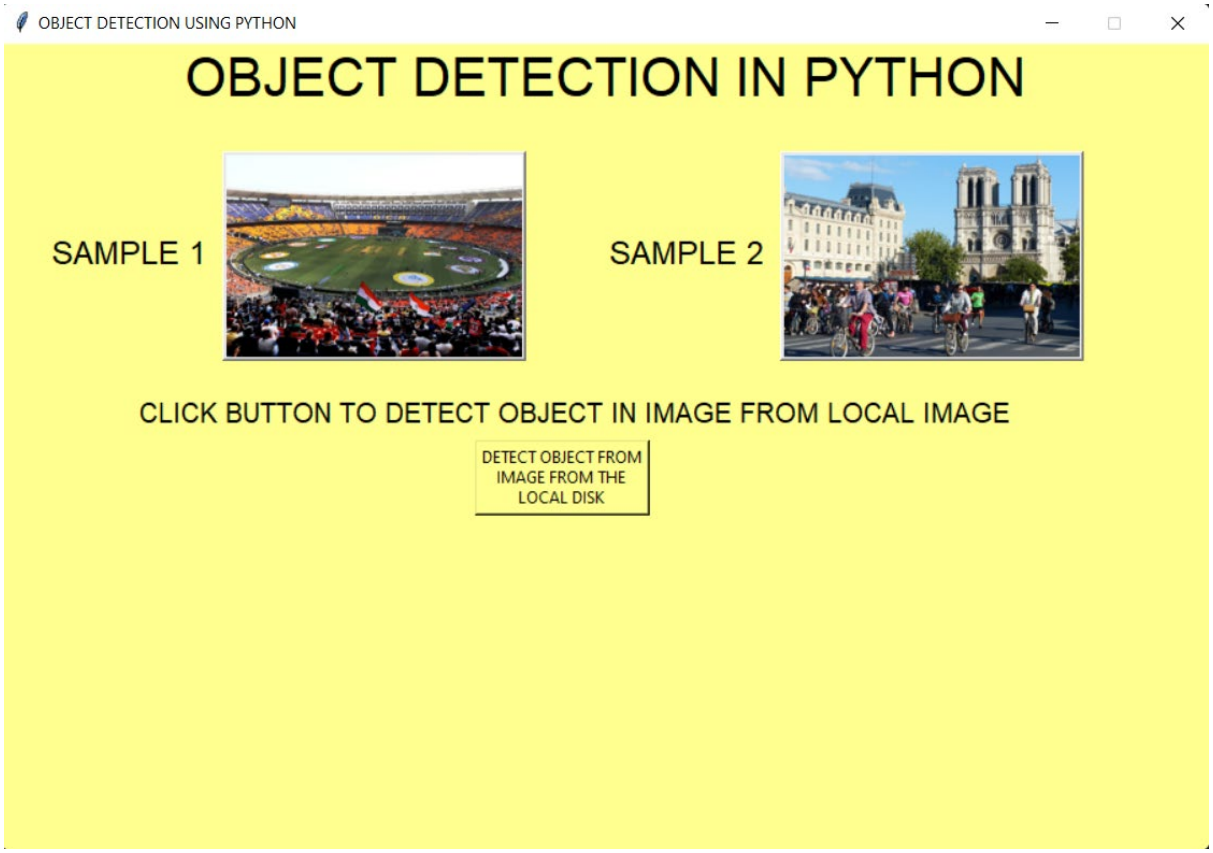
Web References:

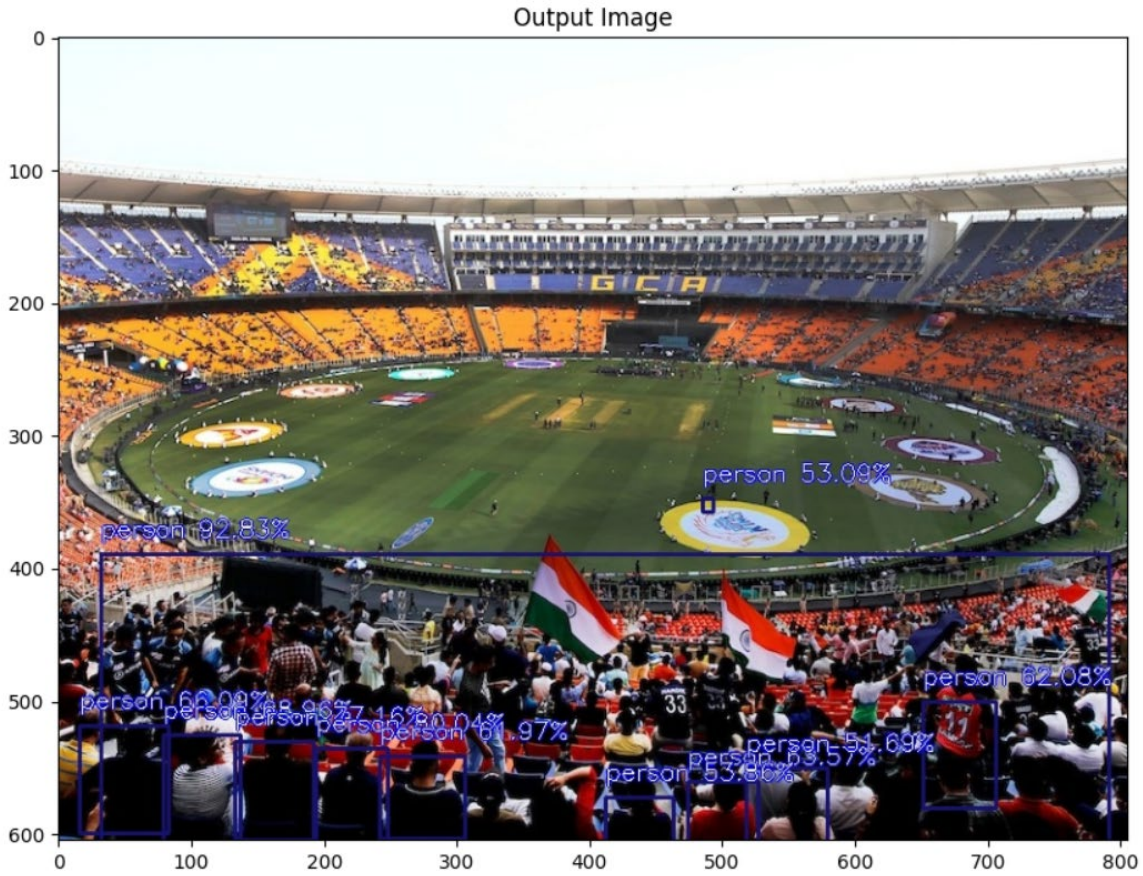
https://en.wikipedia.org/wiki/Object_detection

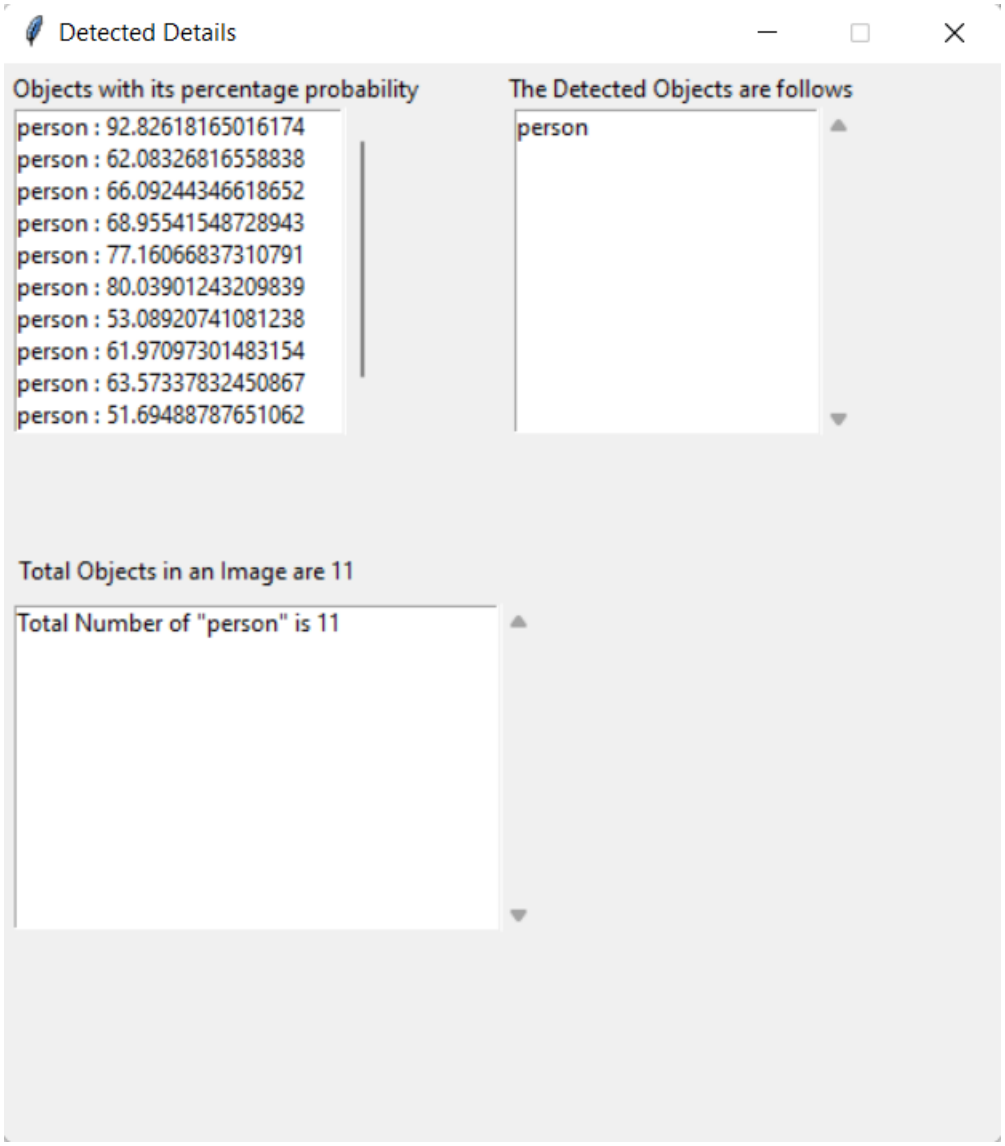
https://paperswithcode.com/task/object-detection

https://colab.research.google.com/github/fraziermatthew/ImageAI/blob/master/imageai.ipynb

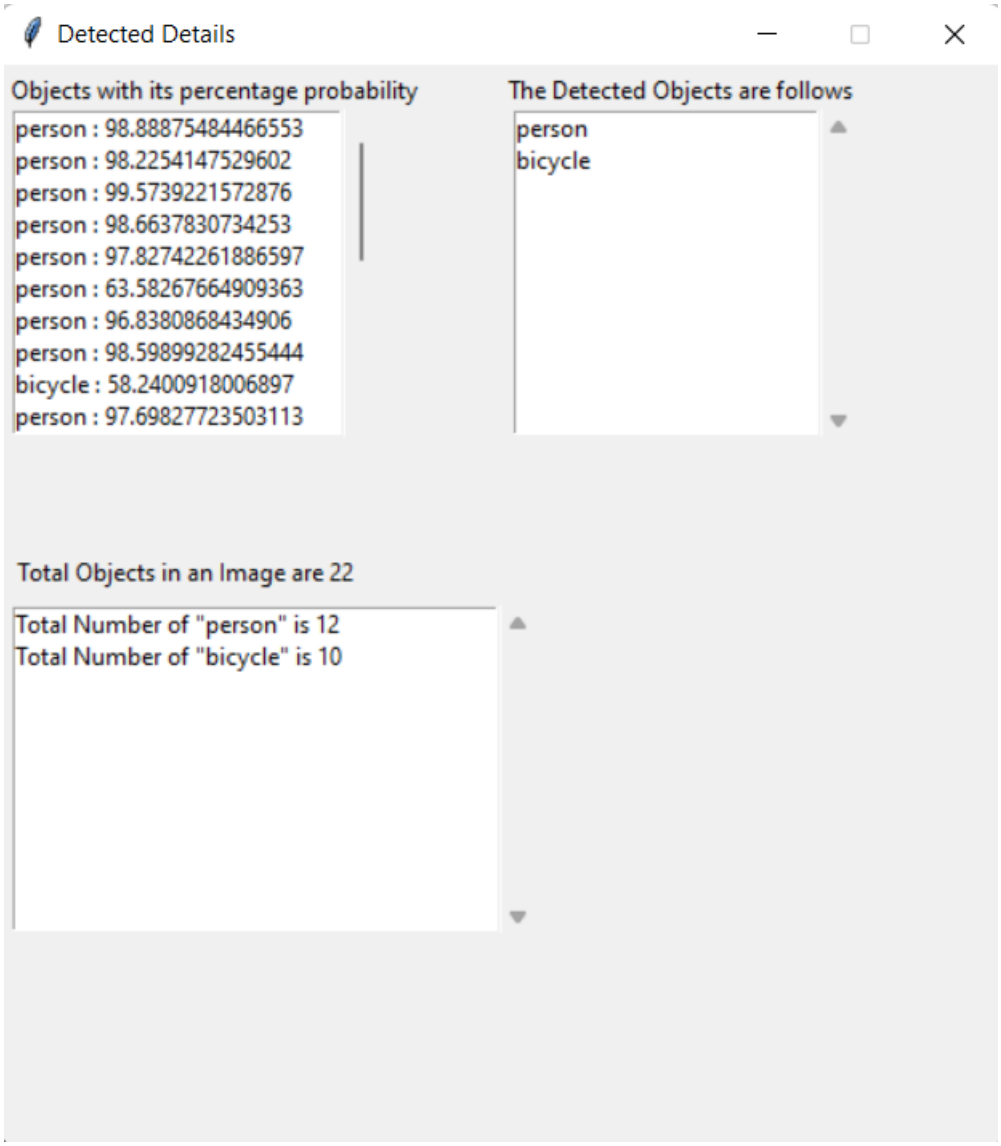
ScreenShots:






Detected Details
Objects with its percentage probability
person : 92.82618165016174
person : 62.08326816558838
person : 66.09244346618652
person : 68.95541548728943
person : 77.16066837310791
person : 80.03901243209839
person : 53.08920741081238
person : 61.97097301483154
person : 63.57337832450867
person : 51.69488787651062
The Detected Objects are follows
person
Total Objects in an Image are 11
Total Number of "person" is 11


Output Image
0
200
400
600
800
1000
0
200
400
600
800
1000
1200
1400
person 99.57%
bicycle 97.26%


Detected Details
Objects with its percentage probability
person : 98.88875484466553
person : 98.2254147529602
person : 99.5739221572876
person : 98.6637830734253
person : 97.82742261886597
person : 63.58267664909363
person : 96.8380868434906
person : 98.59899282455444
bicycle : 58.2400918006897
person : 97.69827723503113
The Detected Objects are follows
person
bicycle
Total Objects in an Image are 22
Total Number of "person" is 12
Total Number of "bicycle" is 10

B] Colour detection:

Synopsis:

A color detection algorithm identifies pixels in an image that match a specified color or color range. The color of detected pixels can then be changed to distinguish them from the rest of the image. For humans this is an extremely easy task but for computers, it is not straightforward. Human eyes and brains work together to translate light into color. Light receptors that are present in our eyes transmit the signal to the brain. Our brain then recognizes the color. Since childhood, we have mapped certain lights with their color names. We will be using the somewhat same strategy to detect color names.

Objectives:

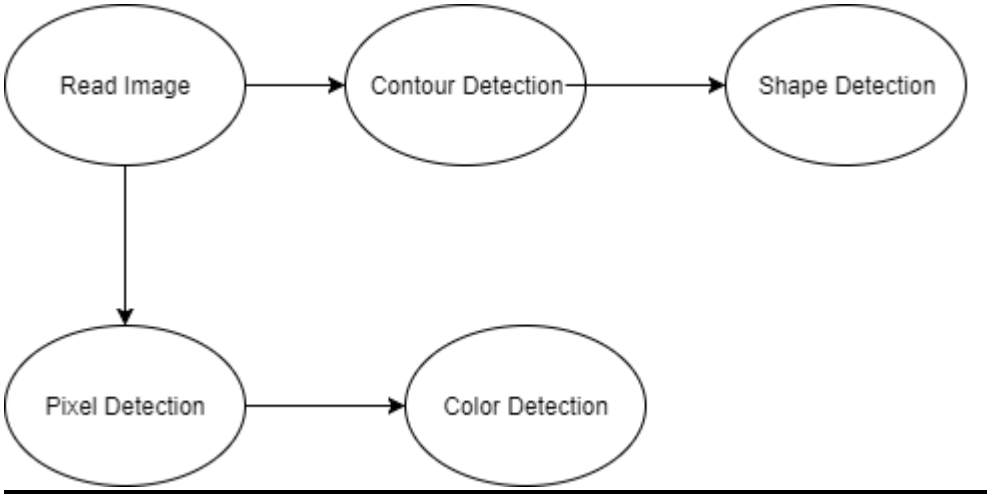
In this color detection Python project, we are going to build an application through which you can automatically get the name of the color by clicking on them. So, for this, we will have a data file that contains the color name and its values. Then we will calculate the distance from each color and find the shortest one.

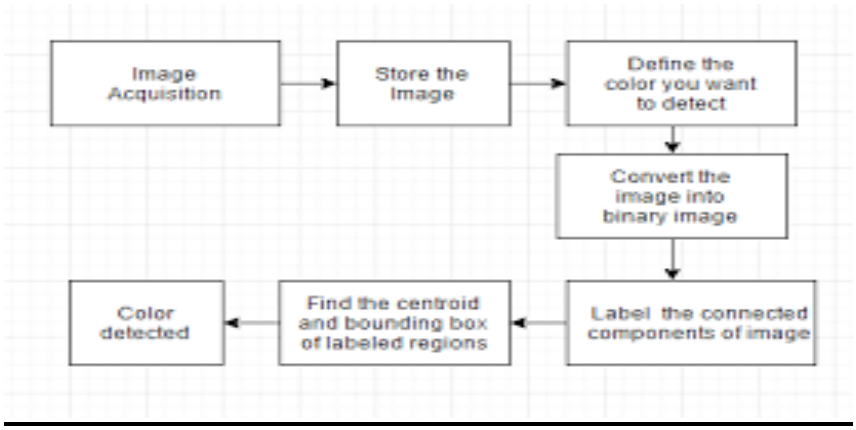
We have also made GUI to make it more user friendly using the tkinter module.

Keywords:

Color Detection, Pandas, Open CV, tkinter module, GUI

Detailed working of the Project:





Colors are made up of 3 primary colors; red, green, and blue. In computers, we define each color value within a range of 0 to 255. So, in how many ways we can define a color? The answer is 256*256*256 = 16,581,375. There are approximately 16.5 million different ways to represent a color. In our dataset, we need to map each color's values with their corresponding names. But don't worry, we don't need to map all the values. We will be using a dataset that contains RGB values with their corresponding names.

Before starting with this Python project with source code, you should be familiar with the computer vision library of Python that is Open CV and Pandas.

OpenCV, Pandas, and numpy are the Python packages that are necessary for this project in Python

Calculate distance to get color name:

We have the r,g and b values. Now, we need another function which will return us the color name from RGB values. To get the color name, we calculate a distance(d) which tells us how close we are to color and choose the one having minimum distance.

Our distance is calculated by this formula:

d = abs(Red – ithRedColor) + (Green – ithGreenColor) + (Blue – ithBlueColor)

Display image on the window:

Whenever a double click event occurs, it will update the color name and RGB values on the window.

Using the cv2.imshow() function, we draw the image on the window. When the user double clicks the window, we draw a rectangle and get the color name to draw text on the window using cv2.rectangle and cv2.putText() functions.

Python project is now complete, you can run the Python file from the command prompt. Make sure to give an image path using '-i' argument. If the image is in another directory, then you need to give full path of the image

Use and Purpose:

- In self-driving car, to detect the traffic signals.
- Multiple color detection is used in some industrial robots, to performing pick-and-place task in separating different colored objects.
- In medical field a prototype is designed in aid for color blind people in detecting color and edges of a given image which are of similar colors
- Color Detection is also used in many manufacturing factories to sort items, packages, Detecting register marks on a packaging sheet in food processing and pharmaceutical packaging units, Color mark detection of screw head in automobile factories, etc.

Merits / Demerits of Color detection:

Following are the **advantages of Color Detection**:

➡It helps in sorting of objects based on three color approach. It also helps in counting of objects.

➡Automated system can be built using color sensors which are made using color detection methods which help in completion of work in less time. Moreover, human intervention is not needed.

➡Powerful and large memory color sensor ICs are available at low cost. This has driven its use in many applications.

➡It is easy to change or modify manufacturing setups without even re-programing the sensor device. This is beneficial in low volume manufacturing applications having frequent color variations.

➡With the advancement of technology and memory loaded with color intensity data, color sensor controller can store and can make color matching decisions on unlimited number of colors virtually.

Following are the **disadvantages of Color sensor**:

➡The approach is costly for small scale industries.

➡It does color matching or identification in applications requiring only pass/fail output.

➡Operating distance range of the color sensors are matter of concern. This needs to be chosen appropriately with rigorous testing in the setup.

Future Enhancements:

- This program can be rewritten to detect all the colours in the imputed picture
- This program can be repurposed to not just detect but also suggest contrasting colour palette
- This program can be repurposed to better suit the visually impaired
- This program can be rewritten to detect colours of frames of a video with a video input.
- This program can be rewritten to detect two or more colours of the pixels in a picture and determine the distance between them .
- This program could be rewritten to read out the colour that is being detected in multiple different languages.
- This program can be rewritten to detect the colour of pixel from a camera feed directly .
- This program can be made into a mobile application

References:

chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.greenteapress.com/thinkpython/thinkpython.pdf

Web references:

https://www.geeksforgeeks.org/multiple-color-detection-in-real-time-using-python-opencv/

Project in Python - Colour Detection using Pandas & OpenCV - DataFlair (data-flair.training)

9748.pdf (journalijdr.com)

Source Code:

```
from tkinter import *
from tkinter.messagebox import *
from PIL import ImageTk, Image
from tkinter.filedialog import askopenfile
import cv2
import numpy as np
import pyttsx3 as pt
import pandas as pd
import argparse


# Creating argument parser to take image path from command line
ap = argparse.ArgumentParser()
ap.add_argument('-i', '--image', help="Image Path")
args = vars(ap.parse_args())
img_path = args['image']


e = pt.init()


root = Tk()
root.title("COLOR DETECTION USING PYTHON")
HEIGHT = 600
WIDTH = 600
root.resizable(False, False)
root.geometry(f"{WIDTH}x{HEIGHT}+2+2")
```

```
root.config(background="#FFFF8F"
)


# Reading the image with opencv

img = cv2.imread(r"sample1.jpg")

img1 = cv2.imread(r"sample2.jpeg")



image_1 =
ImageTk.PhotoImage(Image.open(r"
sample1.jpg").resize((220, 150)))

image_2 =
ImageTk.PhotoImage(Image.open(r"
sample2.jpeg").resize((220, 150)))



font = ("Arial", 20)


Label(root,
background="#FFFF8F",
    text="CLICK ON THE IMAGE
TO DETECT COLOR",
font=font).pack()


# declaring global variables (are
used later on)

clicked = False

r = g = b = xpos = ypos = 0


# Reading csv file with pandas and
giving names to each column

index = ["color", "color_name",
"hex", "R", "G", "B"]

csv = pd.read_csv(r'colors.csv',
names=index, header=None)


# function to calculate minimum
distance from all colors and get the
most matching color
```

```
def getColorName(R, G, B):
    minimum = 10000
    for i in range(len(csv)):
        d = abs(R - int(csv.loc[i, "R"])) + abs(G - int(csv.loc[i, "G"])) + abs(B - int(csv.loc[i, "B"]))
        if(d <= minimum):
            minimum = d
            cname = csv.loc[i, "color_name"]
    return cname


# function to get x,y coordinates of mouse double click


def draw_function(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        global b, g, r, xpos, ypos, clicked
        clicked = True
        xpos = x
        ypos = y
        b, g, r = img[y, x]
        b = int(b)
        g = int(g)
        r = int(r)


def draw_function1(event, x, y, flags, param):
    if event == cv2.EVENT_LBUTTONDBLCLK:
        global b, g, r, xpos, ypos, clicked
        clicked = True
        xpos = x
        ypos = y
        b, g, r = img1[y, x]
```

```
        b = int(b)
        g = int(g)
        r = int(r)



# image = int(input("Please Select
the Image!"))


def print_color_name_on_img(img):
    global clicked
    cv2.rectangle(img, (20, 20), (750,
60), (b, g, r), -1)


    text = getColorName(r, g, b) + ',
RGB(' + str(r) + \
        ',' + str(g) + ',' + str(b) + ')'


    cv2.putText(img, text, (50, 50), 2,
0.8,
                (255, 255, 255), 2,
cv2.LINE_AA)

    if(r+g+b >= 600):
        cv2.putText(img, text,  (50,
50), 2, 0.8,
                    (0, 0, 0), 2,
cv2.LINE_AA)


    e.say(getColorName(r, g, b))
    e.runAndWait()


    clicked = False


# if(image == 1):


def show_img_1():
    cv2.namedWindow('image')
    cv2.setMouseCallback('image',
draw_function)
```

```
    while(1):
        cv2.imshow("image", img)
        if (clicked):

print_color_name_on_img(img)


        # Break the loop when user hits
'esc' key
        if cv2.waitKey(20) & 0xFF ==
ord('q'):
            break

    cv2.destroyAllWindows()



def show_img_2():
    cv2.namedWindow('image1')
    cv2.setMouseCallback('image1',
draw_function1)

    while(1):
        cv2.imshow("image1", img1)
        if (clicked):

print_color_name_on_img(img1)


        # Break the loop when user hits
'esc' key
        if cv2.waitKey(20) & 0xFF ==
ord('q'):
            break

    cv2.destroyAllWindows()



def show_other_image():
    f_types = [
        ('image files','*.jpg'),
        ('image files','*.png'),
        ('image files','*.jpeg')
    ]
```

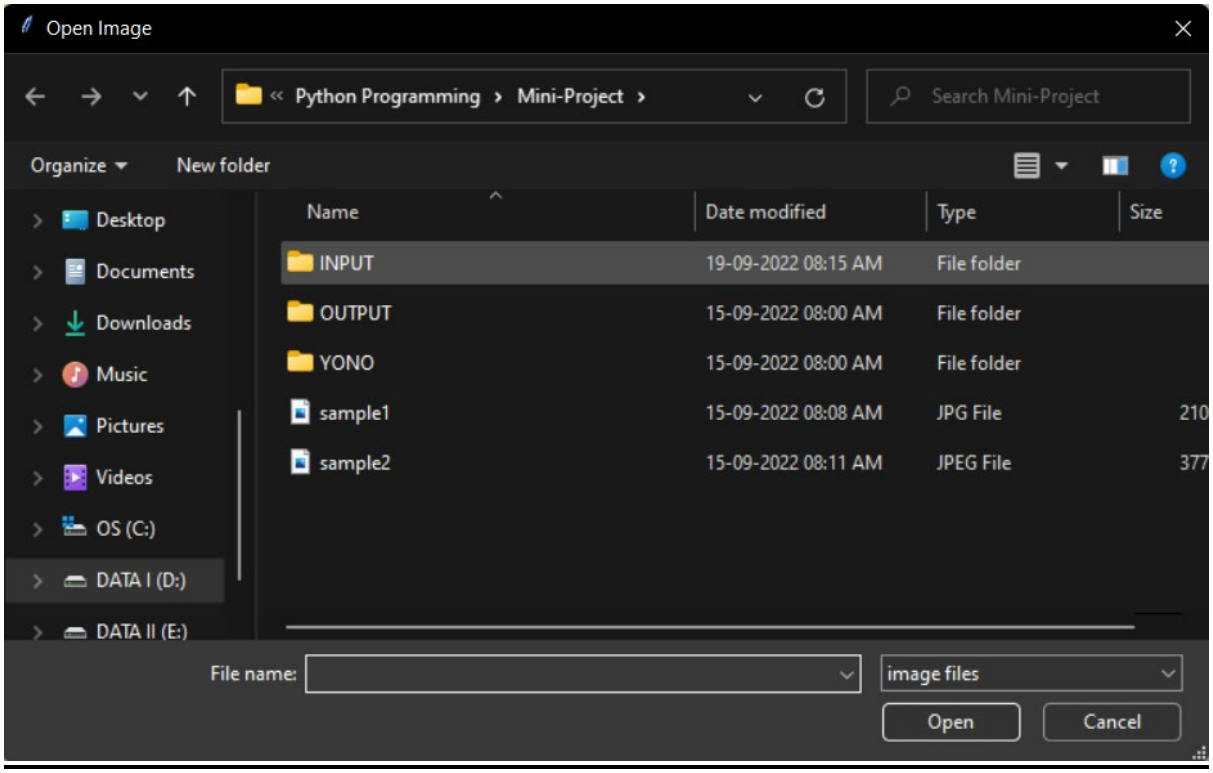
```
    file =
askopenfile(filetypes=f_types,
title="Open Image")



    def draw_function2(event, x, y,
flags, param):
        if event ==
cv2.EVENT_LBUTTONDBLCLK:
            global b, g, r, xpos, ypos,
clicked
            clicked = True
            xpos = x
            ypos = y
            b, g, r = _image[y, x]
            b = int(b)
            g = int(g)
            r = int(r)


    try:
        file_path = file.name
        _image = cv2.imread(file_path)


        cv2.namedWindow('Image3')


cv2.setMouseCallback('Image3',
draw_function2)



        while(1):
            cv2.imshow("Image3",
_image)
            if (clicked):

print_color_name_on_img(_image)


            # Break the loop when user
hits 'esc' key
            if cv2.waitKey(20) & 0xFF
== ord('q'):
                break
```

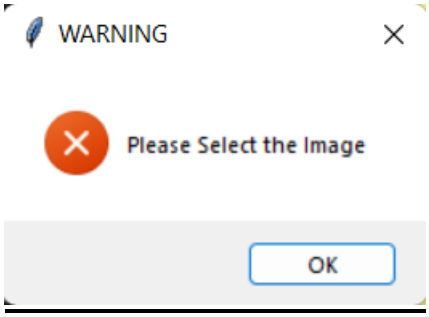
```
        cv2.destroyAllWindows()


    except Exception:


showerror("WARNING","Please
Select the Image")



def onHover(colorOnHover,
colorWhenLeave, button):


    button.bind('<Enter>',
func=lambda f: button.config(

        background=colorOnHover))

    button.bind('<Leave>',
func=lambda g: button.config(

        background=colorWhenLeave))



Label(root,
background="#FFFF8F",
font=("Arial", 18),

        text="SAMPLE 1 ").place(x=10,
y=130)

button1 = Button(root,
background="#FFFF8F",
image=image_1,
command=show_img_1)

button1.place(x=140, y=70)



Label(root,
background="#FFFF8F",
font=("Arial", 18),

        text="SAMPLE 2 ").place(x=10,
y=308)

button2 = Button(root,
background="#FFFF8F",
image=image_2,
command=show_img_2)

button2.place(x=140, y=250)



Label(root,
background="#FFFF8F",
font=("Arial", 18),
```

```
    text="Want to Detect Color in Other Image ? ").place(x=10, y=440)


btn = Button(root, activebackground="#DFFF00", background="#FFFF8F",

          text="CLICK HERE TO OPEN IMAGE FROM LOCAL DISK", wraplength=100, command=show_other_image)

btn.place(x=440, y=435)


onHover("#FFF8DC", "#FFFF8F", btn)

root.mainloop()
```

Screen Shots:


image
Indian Yellow, RGB(219,167,81)


Open Image
Python Programming › Mini-Project ›
Search Mini-Project
Organize
New folder
Desktop
Documents
Downloads
Music
Pictures
Videos
OS (C:)
DATA I (D:)
DATA II (E:)
Name
Date modified
Type
Size
INPUT
19-09-2022 08:15 AM
File folder
OUTPUT
15-09-2022 08:00 AM
File folder
YONO
15-09-2022 08:00 AM
File folder
sample1
15-09-2022 08:08 AM
JPG File
210
sample2
15-09-2022 08:11 AM
JPEG File
377
File name:
image files
Open
Cancel


WARNING
Please Select the Image
OK