Training Report

On

### “Color detection and k-means clustering using Python and OpenCV”

#### at

Defence Laboratory, Jodhpur

Defence Research and Development Organization (DRDO)

Date: 24 June 2021 – 09 August 2021



SUBMITTED BY: SUBMITTED TO:

Aviral Goyal Mr. R K Khatri

B.Tech Scientist – ‘F’

(Computer Science – 4th semester)

(Vellore Institute of Technology,

Vellore, Tamil Nadu)

**ACKNOWLEDGEMENT**

I take immense pleasure in thanking **Sh. Ravindra Kumar,** Director, Defence Lab Jodhpur for having permitted me to carry out this project work. I would like to thank **Dr. R Vijay Raghavan**, HOD, Camouflage Division for his support and facilities made available.

I wish to express my deep sense of gratitude to **Sh. R K Khatri, Scientist ‘F',** for his able guidance and useful suggestions, which helped me in completing the project work, in time.

I would like to express thanks to the HRD Section for providing necessary administrative support.

**-Aviral Goyal**

**CERTIFICATE**

This is to certify that Aviral Goyal, student of B.Tech (Computer Science, 4th Semester), Vellore Institute of Technology, Vellore, Tamil Nadu, has undertaken Practical Training on “**Color detection and k-means clustering using Python and OpenCV**”.The training was conducted at Defence Laboratory, Jodhpur under my supervision and guidance during 24 June 2021 – 09 August 2021.

**Date: R K Khatri**

**Place: Defence Laboratory, Jodhpur Scientist – ‘F’**

**ABSTRACT**

In this project we are making a system that will help with the identification of various colors present in any image. For this, the system will use python programming language along with libraries like OpenCV, PIL etc. The system that we are making uses the k-means clustering method to segregate various clusters of colors and show them as output. This system can use an image in any format as input and perform various functions on it to identify the colors present in the image. The system also forms contours and help us identify the shapes of various objects present in the image. This system can be made part of various drones which can then help determine the geographical features of various landscapes. This system can run on any hardware system with a python compiler. This system is also accompanied with a user-friendly GUI to make its processing understandable and easier.

**Contents**

### Page No. PART-I: DEFENCE LABORATORY, JODHPUR: 6-8

### INTRODUCTION 7

### Vision

### Mission 7

### Camouflage Division

### NRMA Division 8

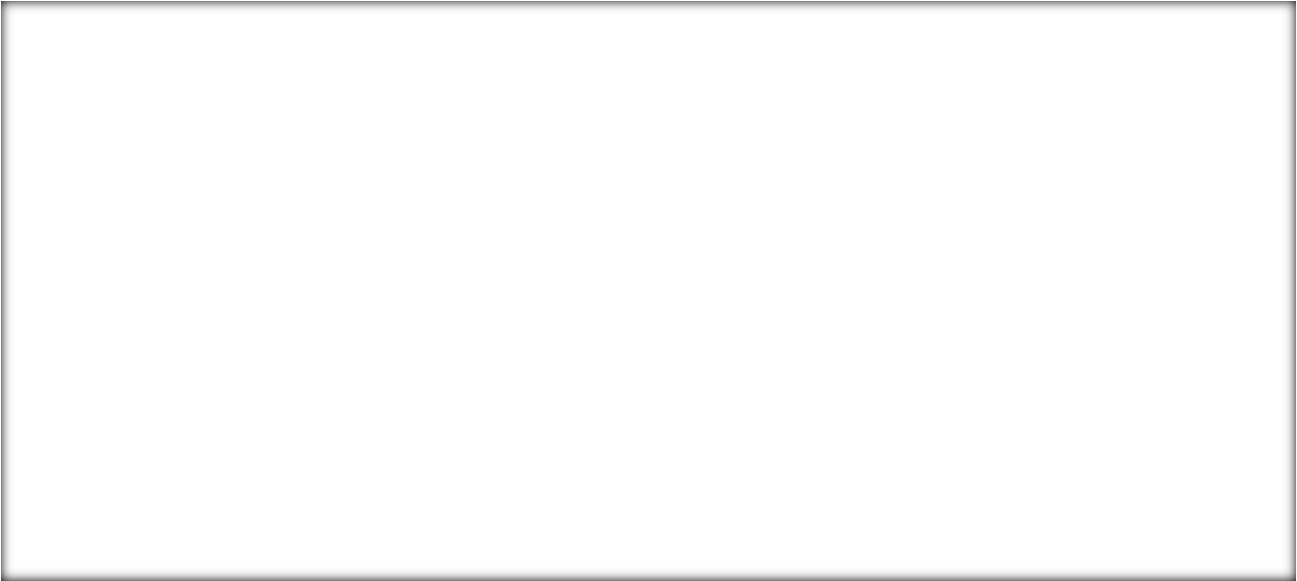
### CDST Division 8

**PART-II: COLOR DETECTION SYSTEM 9-32**

1. Introduction 9
2. Tools used 11
3. Procedure 15
4. Code 17
5. Output 26
6. Conclusion 32
7. References 32

**PART – I**

### DEFENCE LABORATORY, JODHPUR: INTRODUCTION



Defence Laboratory, Jodhpur (DLJ) was established in May 1959 to deal with the problems related to environmental conditions in desert and their impact on desert warfare. The initial charter allotted to the laboratory was:

"Undertaking field trials on weapons and equipment which were either newly designed or developed in the country or were being manufactured indigenously with imported know-how, besides conducting basic research as applicable in the arid zone, physiological studies, Radio-wave propagation studies and solar energy."

Subsequently, with expansion of the laboratory, the charter of duties was enriched by adding Operational Research, Camouflage in Desert, Electronics & Communications in Desert, Water Problems in Desert, Transportation and Navigation Systems in Desert, and Weapons, Ammunitions, and Stores areas of activities.

The laboratory is pursuing R&D in the strategic areas of Camouflage, Deception, Detection, Reconnaissance, Performance of weapons & ammunition, clothing, equipment and stores under desert condition, Integrated water management, Soil stabilization, and Applications of Radioisotopes in defence. Thrust areas of the laboratory are: -

* Camouflage & Low Observable Technologies
* Nuclear Radiation Management & Applications
* DTDG

The laboratory has a unique feature wherein it pursues basic research in all the three thrust areas, converts it into technology and finally applies it to systems that are introduced into services.

### Vision:

To become a centre of excellence in Camouflage, Desert Sciences & Nuclear Radiation Management Technologies.

### Mission:

To achieve excellence and self-reliance in the areas of Multispectral Camouflage and Low Observable Technologies, provide solutions to Desert related problems, and develop Nuclear Radiation Sensor Technologies and Applications of Radioisotopes.

### Camouflage Division:

The Camouflage Division of Defence Laboratory, Jodhpur, is actively engaged in the R&D on the Development of Materials and Technologies for Camouflage and Low-Observability. The focus is towards Signature Management of Ground-based and Airborne Platforms.

The Materials Development activity of the Division has a strong research and development focus on synthesis and characterization of advanced materials, and for the development and testing of Material Products for Camouflage and Stealth.

### Nuclear Radiation Management & Application (NRMA):

NRMA Division is pursuing R & D in the field of radiation detectors, dosimeters, dose rate meters, and area monitors. Facilities have been established for the testing and calibration of radiation monitors including thermo luminescent dosimeters-based personnel monitoring services. The facilities have been made traceable with the accreditation by national bodies (NABL / BARC). Assorted training courses for armed forces personnel in Nuclear Defence and Radiological Safety are being conducted. Scientists of DLJ represent a number of National Committees related to nuclear defence /safety.

### DTDG Division:

DTDG Division is engaged in the R&D in the fields of water, soil & heat to enhance the efficiency of the troops operating under harsh desert conditions of Desert. Basic as well as applied research is being pursued leading to development of technologies related to water, heat and terrain management for military applications.

**PART – II**

**COLOR DETECTION SYSTEM**

1. **Introduction:**

**What is Color Detection?**

Color detection is the process of detecting the name of any color. 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.

**About Project**

In this project, we are constructing a system that will use Python and OpenCV to detect various colors present in the image. The detection of various colors is mainly done using the k-means clustering algorithm. The system built can take images of any format like .jpeg, .jpg etc. as input and perform various functions in it. The image can be uploaded in the system using the already saved images on your computer or using a webcam. Along with the image the user will also give the number of clusters as input. The number of clusters thus obtained will represent the number of colors into which each and every color present in the image will be segregated using k-means clustering algorithm. The system will give us a graphical representation of various colors and their amount present in the image. It will also tell the percentage of each color present in the image and the maximum and minimum colors present. The system will also enable us to draw contours in order to detect closely bounded shapes in the image. Along with contours, the system uses them to determine the shapes of the object and label the shapes accordingly in the image. The system constructed is also accompanied with interactive GUI to make the functions easier to work with.

1. **Tools used:**

**Software:**

Software used for implementing this system is **Microsoft Visual Code.** Visual Studio Code is a [source-code editor](https://en.wikipedia.org/wiki/Source-code_editor) made by [Microsoft](https://en.wikipedia.org/wiki/Microsoft) for [Windows](https://en.wikipedia.org/wiki/Windows), [Linux](https://en.wikipedia.org/wiki/Linux) and [macOS](https://en.wikipedia.org/wiki/MacOS). Features include support for [debugging](https://en.wikipedia.org/wiki/Debugging), [syntax highlighting](https://en.wikipedia.org/wiki/Syntax_highlighting), [intelligent code completion](https://en.wikipedia.org/wiki/Intelligent_code_completion), [snippets](https://en.wikipedia.org/wiki/Snippet_(programming)), [code refactoring](https://en.wikipedia.org/wiki/Code_refactoring), and embedded [Git](https://en.wikipedia.org/wiki/Git). Users can change the [theme](https://en.wikipedia.org/wiki/Theme_(computing)), [keyboard shortcuts](https://en.wikipedia.org/wiki/Keyboard_shortcut), preferences, and install [extensions](https://en.wikipedia.org/wiki/Plug-in_(computing)) that add additional functionality.

*Download Link:* [*https://code.visualstudio.com/download*](https://code.visualstudio.com/download)

**Features:**

1. Microsoft visual studio code is available for macOS, Linux and Windows.
2. We can edit, build and debug our code easily with Microsoft visual studio code. It supports hundreds of languages, VS Code helps you be instantly productive with syntax highlighting, bracket-matching, auto-indentation, box-selection, snippets, and more.
3. We can customize every feature of vs code to our liking and install any number of third-party extensions.
4. VS Code includes enriched built-in support for Node.js development with JavaScript and TypeScript, powered by the same underlying technologies that drive Visual Studio. VS Code also includes great tooling for web technologies such as JSX/React, HTML, CSS, SCSS, Less, and JSON.
5. VS code has robust and extensible architecture. Architecturally, Visual Studio Code combines the best of web, native, and language-specific technologies. Visual Studio Code includes a public extensibility model that lets developers build and use extensions, and richly customize their edit-build-debug experience.

**Algorithm:**

The algorithm used to perform color detection is the k-means clustering algorithm. K-means clustering algorithm is an iterative algorithm that tries to partition the dataset into *K* pre-defined distinct non-overlapping subgroups(clusters) where each data point belongs to only one group. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different (far) as possible. It assigns data points to a cluster such that the sum of the squared distance between the data points and the cluster’s centroid (arithmetic mean of all the data points that belong to that cluster) is at the minimum. The less variation we have within clusters, the more homogeneous (similar) the data points are within the same cluster.

**Python Libraries:**

* **tkinter:** Tkinter is a Python binding to the Tk GUI toolkit. It is the standard Python interface to the Tk GUI toolkit, and is Python's de facto standard GUI. Tkinter is included with standard Linux, Microsoft Windows and Mac OS X installs of Python.
* **PIL:** Python Imaging Library is a free and open-source additional library for the Python programming language that adds support for opening, manipulating, and saving many different images file formats. It is available for Windows, Mac OS X and Linux.
* **OpenCV:** OpenCV is a library of programming functions mainly aimed at real-time computer vision. Originally developed by Intel, it was later supported by Willow Garage then Itseez. The library is cross-platform and free for use under the open-source Apache 2 License
* **Sklearn.cluster:** [Clustering](https://en.wikipedia.org/wiki/Cluster_analysis) of unlabeled data can be performed with the module [sklearn.cluster](https://scikit-learn.org/stable/modules/classes.html#module-sklearn.cluster). Each clustering algorithm comes in two variants: a class, that implements the fit method to learn the clusters on train data, and a function, that, given train data, returns an array of integer labels corresponding to the different clusters.
* **Numpy:** NumPy is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays.
* **Pandas:** Pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. It is free software released under the three-clause BSD license.

**Dataset:**

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. There are approximately 16.5 million different ways to represent a color. In our dataset, we map each color’s values with their corresponding names. We will be using a dataset that contains RGB values with their corresponding names.

*Link to dataset:* [*https://github.com/codebrainz/color-names/blob/master/output/colors.csv*](https://github.com/codebrainz/color-names/blob/master/output/colors.csv)

The colors.csv file includes 865 color names along with their RGB and hex values.

1. **Procedure:**
2. The GUI is created using the tkinter library. Through GUI we can either upload an image already saved on your system or we can capture an image using the webcam of your system.
3. If the image is captured using webcam. The image is saved on the computer also to be referred to later.
4. After uploading an image, the number of desirable clusters is given as input.
5. The options to perform various functions are then visible in a new window. The options include: color detected, percentage of color detected, colors present maximum and minimum and draw contours.
6. Color detected: This option will use k-means clustering to obtain the rgb values of the centroid of clusters of colors of the image. Each color data point of the image will then be included in only one cluster. The cluster will be decided according to the distance between the rgb values of the centroid and the data point. After the formation of clusters and segregation of data points into various clusters, a horizontal bar graph is formed and the proportion of each cluster is shown graphically.
7. Percentage of color detected: This option will link the dataset to our program. After that it will implement the k-means clustering and extract the rgb values of the centroid of various clusters formed. These rgb values are compared with the rgb values present in our dataset and then the name of the color of the cluster is extracted from the dataset. The name of each cluster along with the percentage of that cluster present in the image is obtained as output.
8. Colors present maximum and minimum: This option compares the percentage obtained of each cluster and finds the color clusters present in maximum percentage and minimum percentage in the image.
9. Draw contours: In this option the program uses the contour feature of OpenCV library to draw contours on the image. These contours are drawn and are used to detect the bounded objects/features present in the image.
10. Shape Detection: In this function the program uses the contours and makes an approx. polygon form the contours of that shape. After that based on the number of sides of the contours polygon the program determines its shape and label it on the image.
11. Each of these options function independently, i.e., we can use any option at any time and no interdependency between the functions are there.
12. The code for each option is written in separate python files. This makes the independent use of each feature easier and makes the functions more accessible.
13. **Code:**

**kmean.py**

from sklearn.cluster import KMeans

import matplotlib.pyplot as plt

import cv2

import numpy as np

def centroid\_histogram(clt):

    numLabels = np.arange(0, len(np.unique(clt.labels\_)) + 1)

    (hist, \_) = np.histogram(clt.labels\_, bins = numLabels)

    hist = hist.astype("float")

    hist /= hist.sum()

    return hist

def plot\_colors(hist, centroids):

    bar = np.zeros((50, 300, 3), dtype = "uint8")

    startX = 0

    #i = 0

    for (percent, color) in zip(hist, centroids):

        endX = startX + (percent \* 300)

        #print(i)

        #i = i+1

        cv2.rectangle(bar, (int(startX), 0), (int(endX), 50), color.astype("uint8").tolist(), -1)

        startX = endX

    return bar

def nameit(name, num):

    #path = r'D:\Documents\Desktop\DRDO\_Project\colorpicpng.png'

    image = cv2.imread(name)

    image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

    plt.figure()

    plt.axis("off")

    plt.imshow(image)

    image = image.reshape((image.shape[0] \* image.shape[1], 3))

    clt = KMeans(n\_clusters = num)

    clt.fit(image)

    hist = centroid\_histogram(clt)

    bar = plot\_colors(hist, clt.cluster\_centers\_)

    plt.figure()

    plt.axis("off")

    plt.imshow(bar)

    plt.show()

**percent.py**

from sklearn.cluster import KMeans

import cv2

import numpy as np

import pandas as pd

index = ["color","color\_name","hex","r","g","b"]

csv = pd.read\_csv('colors.csv',names = index, header = None)

def getcolorname(r,g,b):

    minimum = 100000000000

    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

def centroid\_histogram(clt):

    numLabels = np.arange(0, len(np.unique(clt.labels\_)) + 1)

    (hist, \_) = np.histogram(clt.labels\_, bins = numLabels)

    hist = hist.astype("float")

    hist /= hist.sum()

    return hist

def plot\_colors(hist, centroids):

    bar = np.zeros((50, 300, 3), dtype = "uint8")

    startX = 0

    z = []

    for (percent, color) in zip(hist, centroids):

        endX = startX + (percent\*100)

        #cv2.rectangle(bar, (int(startX), 0), (int(endX), 50), color.astype("uint8").tolist(), -1)

        z.append(endX - startX)

        startX = endX

    return z

def percentage(name, num):

    #path = r'D:\Documents\Desktop\DRDO\_Project\colorpicpng.png'

    image = cv2.imread(name)

    image = cv2.cvtColor(image, cv2.COLOR\_BGR2RGB)

    image = image.reshape((image.shape[0] \* image.shape[1], 3))

    clt = KMeans(n\_clusters = num)

    clt.fit(image)

    cent = clt.cluster\_centers\_

    #print(cent.shape)

    hist = centroid\_histogram(clt)

    bar = plot\_colors(hist, cent)

    #bar1 = bar

    colname = []

    for i in range(num):

        R = cent[i][0]

        G = cent[i][1]

        B = cent[i][2]

        colname.append(getcolorname(R,G,B))

        n = i

        #print(str(n)+"\n\n")

    fin = {}

    #print(len(colname))

    for i in range(len(colname)):

        #print(i)

        fin[colname[i]] = bar[i]

        #print(len(fin))

        #bar.remove(j)

        #break

    #print(len(fin.shape))

    return fin

**contour.py**

import cv2 as cv

def con(im):

    img = cv.imread(im)

    gray = cv.cvtColor(img, cv.COLOR\_BGR2GRAY)

    blur = cv.blur(gray, (10,10))

    ret, thresh = cv.threshold(blur, 1, 255, cv.THRESH\_OTSU)

    contours, heirarchy = cv.findContours(thresh, cv.RETR\_TREE, cv.CHAIN\_APPROX\_SIMPLE)

    cv.drawContours(img, contours, -1, (0,0,0), 3)

    cv.namedWindow('Contours',cv.WINDOW\_NORMAL)

    cv.namedWindow('Thresh',cv.WINDOW\_NORMAL)

    cv.imshow('Contours', img)

    cv.imshow('Thresh', thresh)

    if cv.waitKey(0):

        cv.destroyAllWindows()

**shape.py**

import numpy as np

import cv2

def shapedet(im):

    img = cv2.imread(im)

    imgGry = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

    ret, thrash = cv2.threshold(imgGry, 240 , 255, cv2.CHAIN\_APPROX\_NONE)

    contours, hierarchy = cv2.findContours(thrash, cv2.RETR\_TREE, cv2.CHAIN\_APPROX\_NONE)

    for contour in contours:

        approx = cv2.approxPolyDP(contour, 0.01\* cv2.arcLength(contour, True), True)

        cv2.drawContours(img, [approx], 0, (0, 0, 0), 5)

        x = approx.ravel()[0]-50

        y = approx.ravel()[1]-5

        if len(approx) == 3:

            cv2.putText( img, "Triangle", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0) )

        elif len(approx) == 4:

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

            aspectRatio = float(w)/h

            #print(aspectRatio)

            if aspectRatio >= 0.95 and aspectRatio < 1.05:

                cv2.putText(img, "square", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

            else:

                cv2.putText(img, "rectangle", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

        elif len(approx) == 5:

            cv2.putText(img, "pentagon", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

        elif len(approx) == 6:

            cv2.putText(img, "hexagon", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

        elif len(approx) == 10:

            cv2.putText(img, "star", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

        else:

            cv2.putText(img, "circle", (x, y), cv2.FONT\_HERSHEY\_COMPLEX, 0.5, (0, 0, 0))

    #cv2.imshow('contours', cv2.drawContours(img, [approx], 0, (0, 0, 0), 5)

    small = cv2.resize(img, (700,700))

    cv2.imshow('shapes', img)

    cv2.waitKey(0)

    cv2.destroyAllWindows()

**color detection.py**

from tkinter import \*

from PIL import ImageTk, Image

from tkinter import filedialog

from kmean import nameit

from percent import percentage

import cv2

from contour import con

from shape import shapedet

import os

root = Tk()

root.title("Colour Detection")

root.geometry("550x300")

root.resizable(width = True, height = True)

def colours():

    n = int(num.get())

    nameit(x,n)

def percent():

    root2 = Toplevel(root1)

    root2.title("Percentage of each colour")

    root2.geometry("550x300")

    m = int(num.get())

    per = percentage(x,m)

    #print(per)

    #print(len(per))

    key = list(per.keys())

    value = list(per.values())

    t = Text(root2)

    #print(m)

    #print(len(value))

    for i in range(0,len(value)):

        #print(i)

        s = "{:.3f}".format(value[i])

        s1 = str(s)

        s2 = str(key[i])

        t.insert(END, s2 + "\t\t\t\t\t\t\t" + s1 + "%\n")

    t.pack()

def sort():

    root3 = Toplevel(root1)

    root3.title("Maximum and Minimum Colors")

    root3.geometry("550x300")

    mn = int(num.get())

    sor = percentage(x,mn)

    sor1 = sorted(sor.items(),key = lambda x:x[1])

    min1 = sor1[0][0]

    min2 = "{:.3f}".format(sor1[0][1])

    s = str(min2)

    max1 = sor1[mn-1][0]

    max2 = "{:.3f}".format(sor1[mn-1][1])

    s1 = str(max2)

    t1 = Text(root3)

    t1.insert(END, "Minimum Color: "+ min1 + "\t\t\t\t\t\t\t" + s +"%\nMaximum Color: " + max1 + "\t\t\t\t\t\t\t" + s1 + "%")

    t1.pack()

def shapedet():

    shapedet(x)

def drawc():

    con(x)

def km():

    global root1

    root1 = Toplevel(root)

    root1.title("Colours Detected")

    root1.geometry("500x500")

    btn1 = Button(root1, text ='Color Detected', command = colours, width = 50, height = 5)

    btn1.pack()

    btn2 = Button(root1, text ='Percentage of Color Detected', command = percent, width = 50, height = 5)

    btn2.pack()

    btn3 = Button(root1, text ='Colours Present maximum and minimum', command = sort, width = 50, height = 5)

    btn3.pack()

    btn8 = Button(root1, text ='Draw Contours', command = drawc, width = 50, height = 5)

    btn8.pack()

    btn9 = Button(root1, text ='Shape Detection', command = shapedet, width = 50, height = 5)

    btn9.pack()

def cap(img\_name):

    ca = cv2.imread(img\_name)

    ca1 = cv2.cvtColor(ca, cv2.COLOR\_BGR2RGB)

    ca3 = Image.fromarray(ca1)

    im1 = ca3.resize(( 950, 750), Image.ANTIALIAS)

    im1 = ImageTk.PhotoImage(ca3)

    panel = Label(root, image = im1)

    panel.image = im1

    panel.grid(row = 2)

    global num

    num = Entry(root)

    num.place(x = 1000, y = 150)

    btn7 = Button(root,text="Set Clusters",command=km)

    btn7.place(x = 1000, y = 200)

def captured(ima):

    global x

    x = "D:/Documents/Desktop/DRDO\_Project/opencv\_frame\_.png"

    ca2 = cv2.cvtColor(ima, cv2.COLOR\_BGR2RGB)

    cv2.imwrite(x, ca2)

    cap(x)

def web():

    #im = None

    win = Toplevel(root)

    win.geometry("700x700")

    label =Label(win)

    label.grid(row=0, column=0)

    cap = cv2.VideoCapture(0, cv2.CAP\_DSHOW)

    def show\_frames():

        # im = None

        im = cap.read()[1]

        #print(im)

        global cv2image

        cv2image= cv2.cvtColor(im, cv2.COLOR\_BGR2RGB)

        #print(cv2image)

        #global img

        img = Image.fromarray(cv2image)

        imgtk = ImageTk.PhotoImage(image = img)

        label.imgtk = imgtk

        label.configure(image=imgtk)

        label.after(20, show\_frames)

    show\_frames()

    btn6 = Button(win, text = 'Capture', command = lambda:captured(cv2image), width = 40, height = 5)

    btn6.place(x = 200, y = 500)

    win.mainloop()

def open\_img():

    global x

    x = filedialog.askopenfilename(title ='Open File')

    img = Image.open(x)

    img1 = img.resize(( 950, 750), Image.ANTIALIAS)

    img1 = ImageTk.PhotoImage(img1)

    panel = Label(root, image = img1)

    panel.image = img1

    panel.grid(row = 2)

    global num

    num = Entry(root)

    num.place(x = 1000, y = 150)

    btn4 = Button(root,text="Set Clusters",command=km)

    btn4.place(x = 1000, y = 200)

btn = Button(root, text = 'Open Image', command = open\_img)

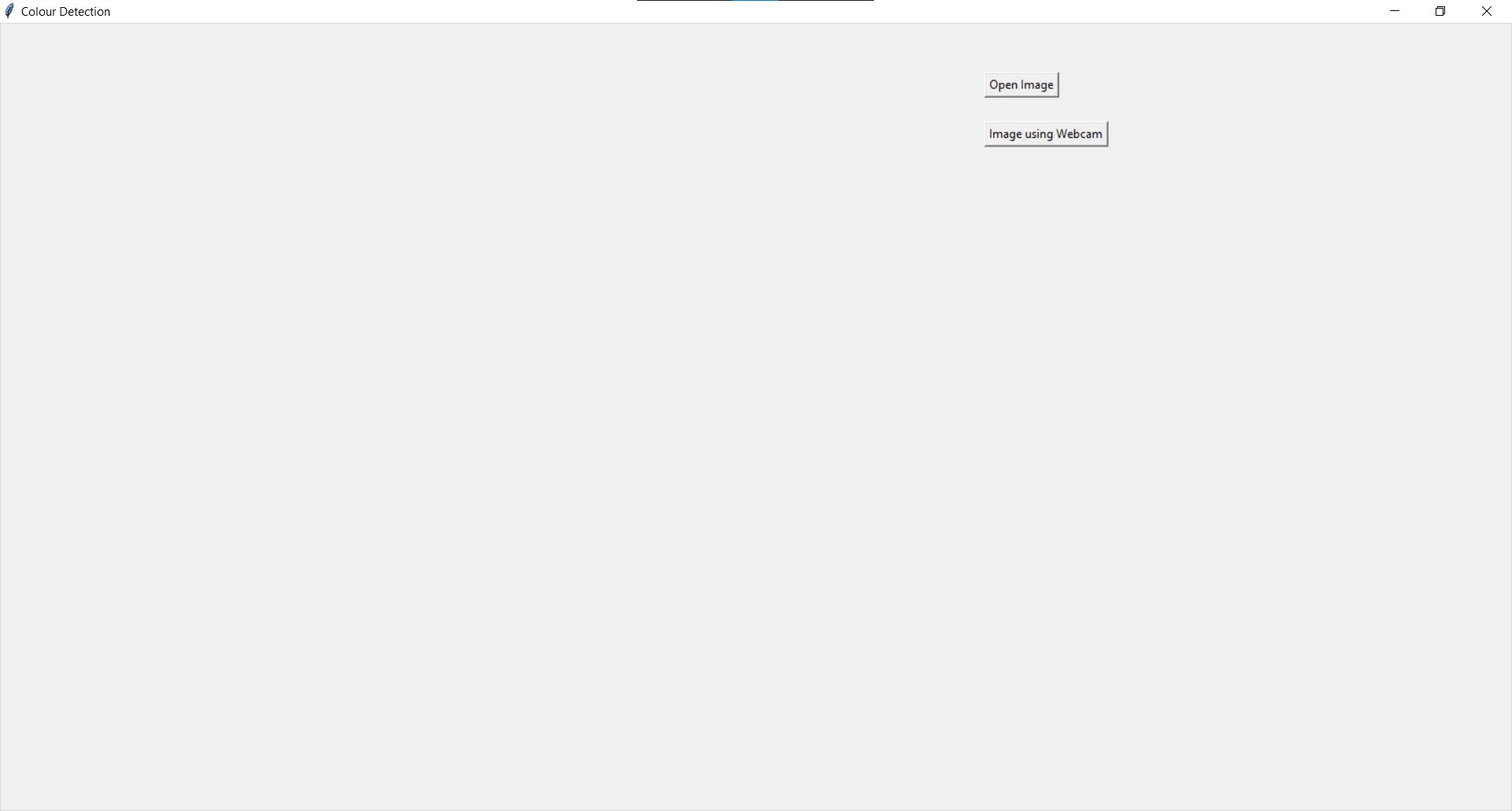
btn.place(x = 1000, y = 50)

btn5 = Button(root, text = 'Image using Webcam', command = web)

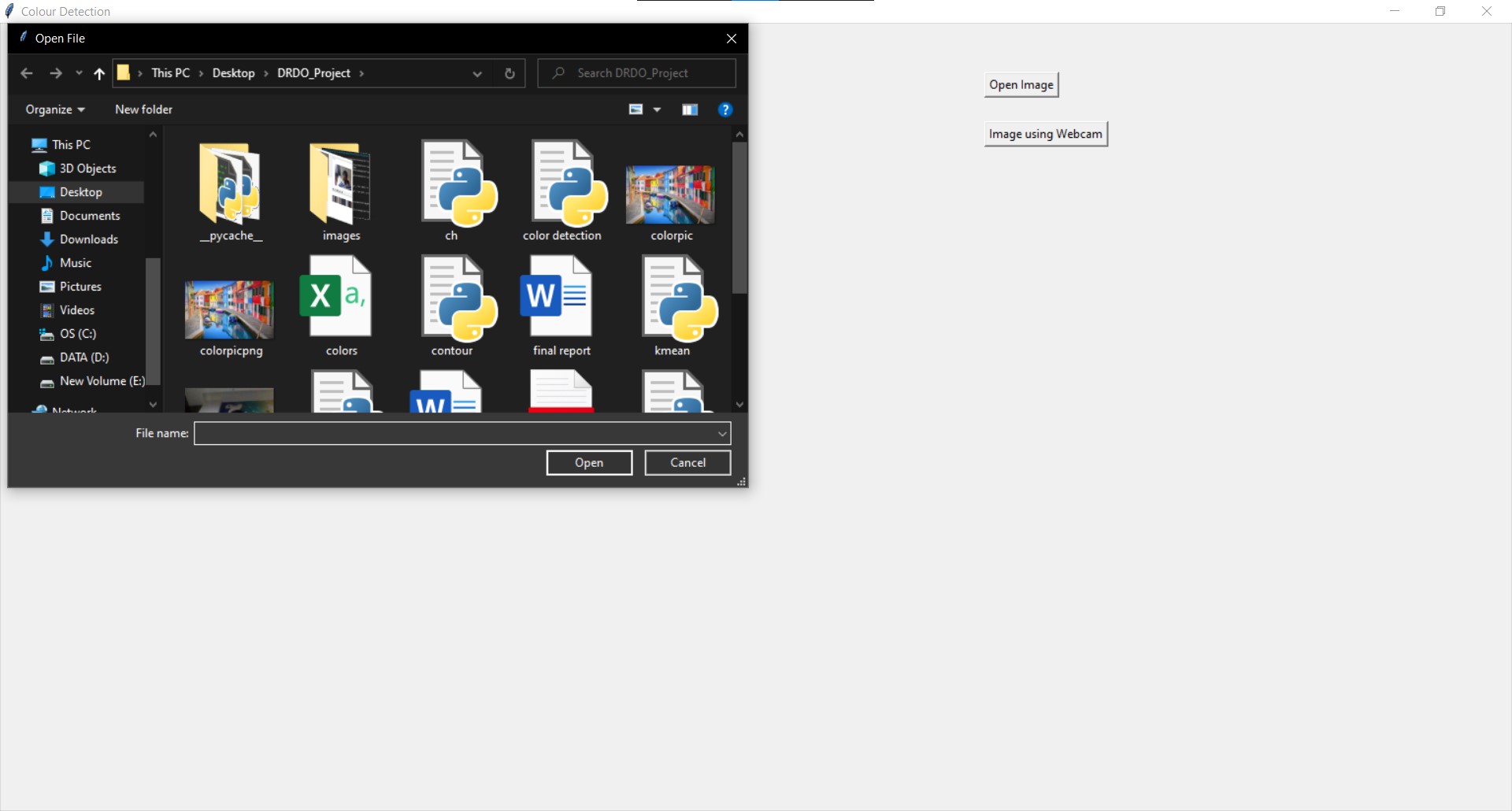
btn5.place(x = 1000, y = 100)

root.mainloop()

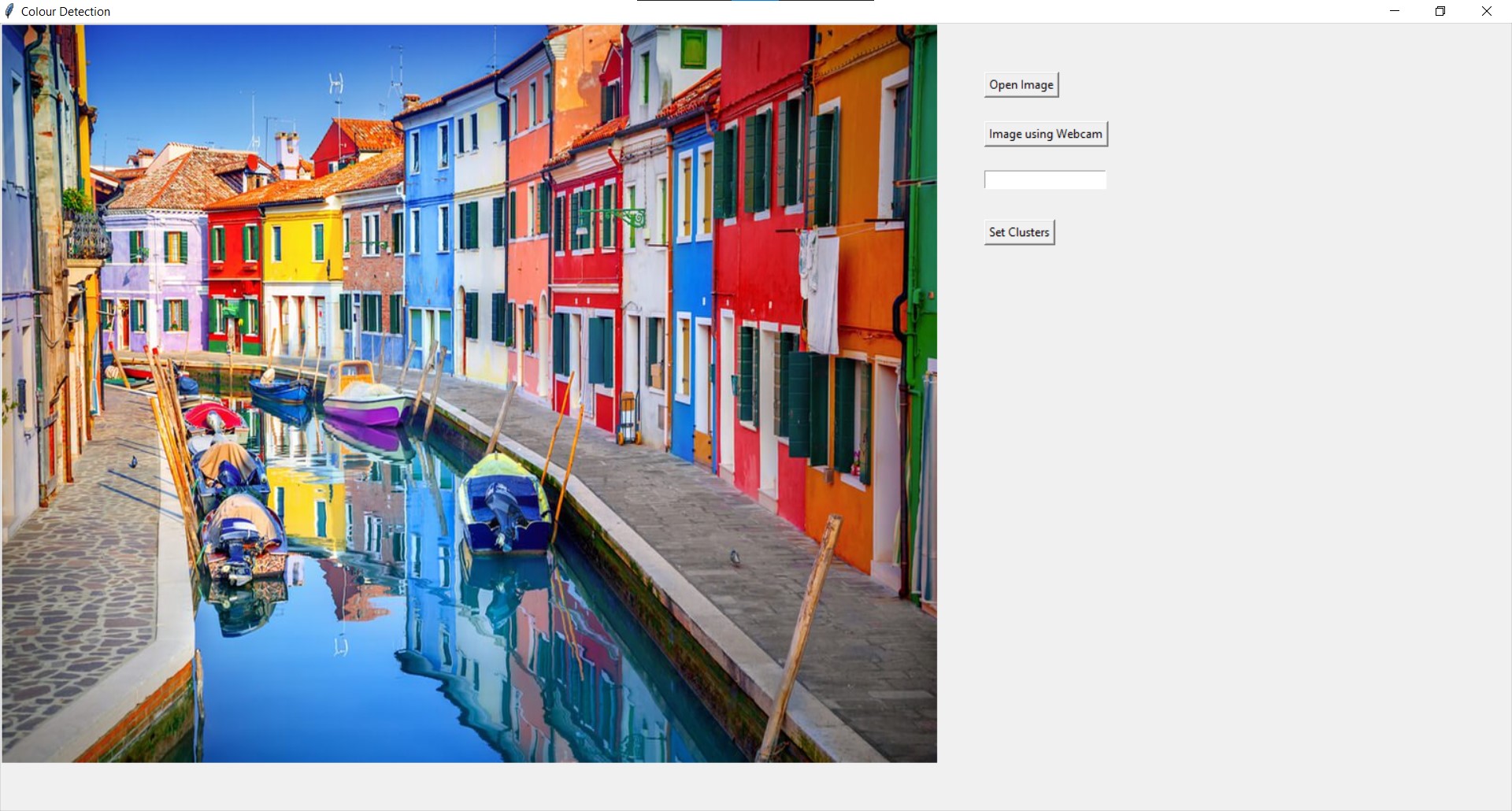
1. **Output:**

****

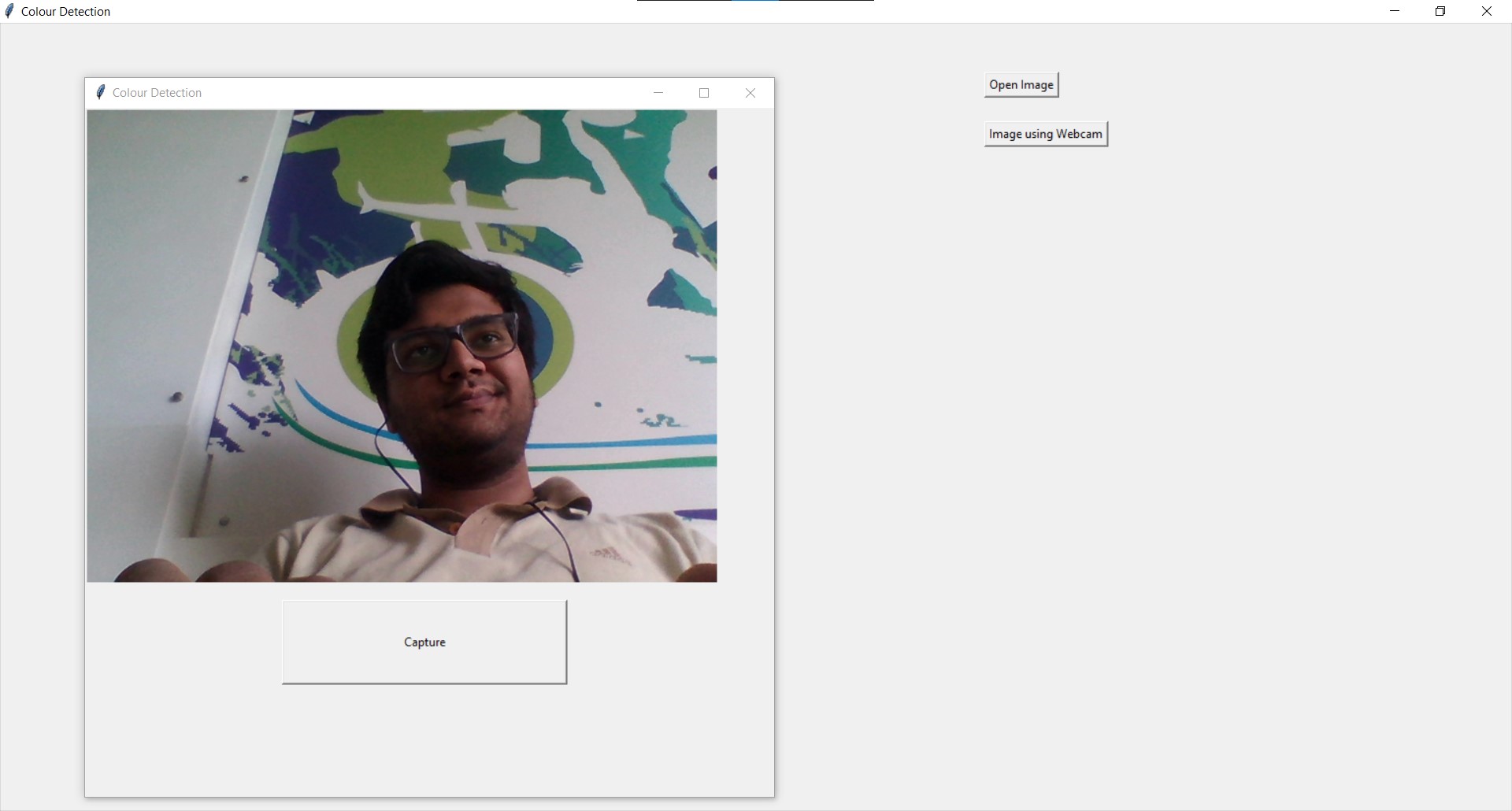
**(Basic GUI to upload image or capture image)**

****

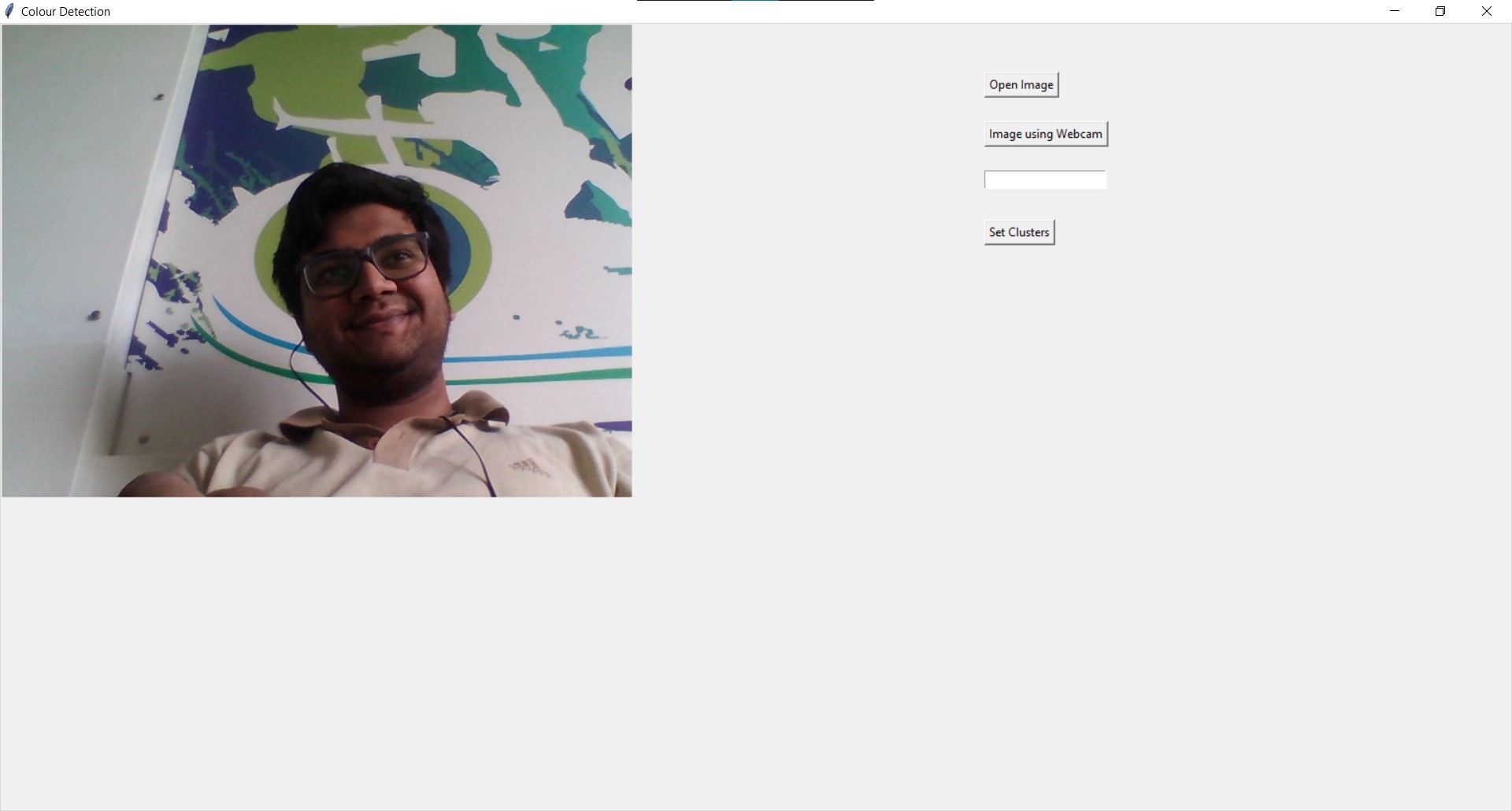
**(Image uploaded that is already present in system)**

****

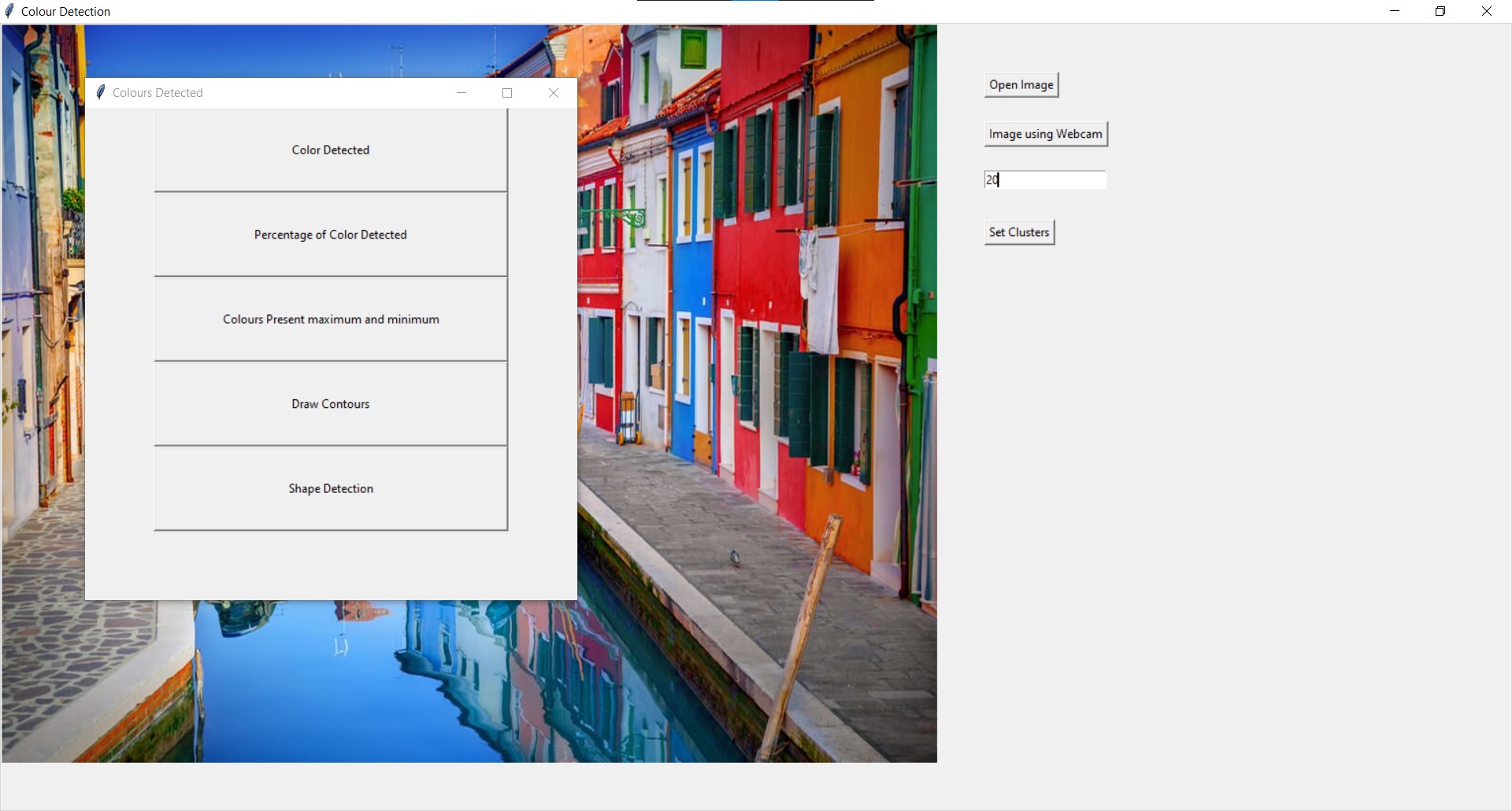
**(Image uploaded from computer)**

****

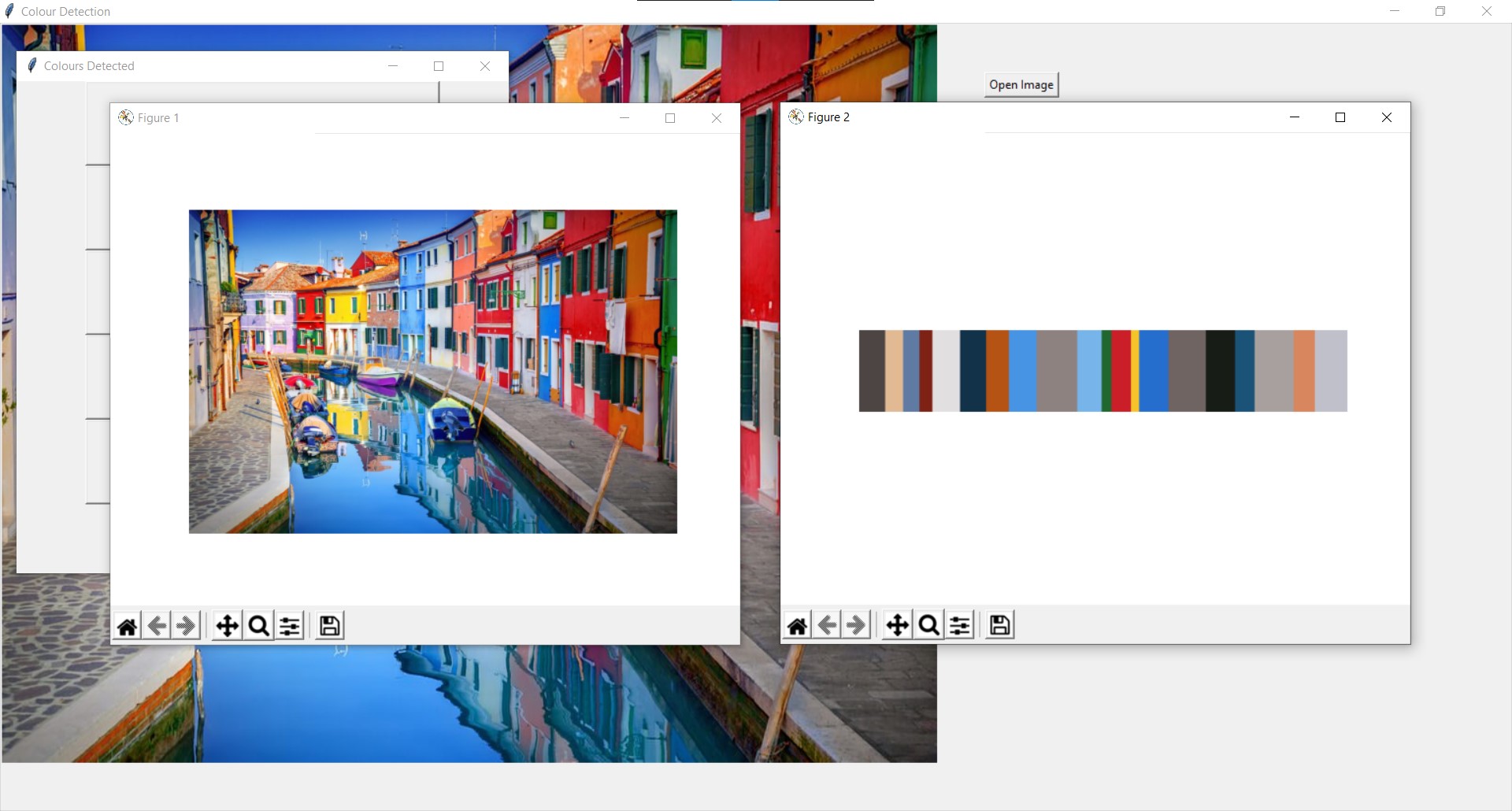
**(Capture and upload image using webcam)**

****

**(Image captured and uploaded using webcam)**

****

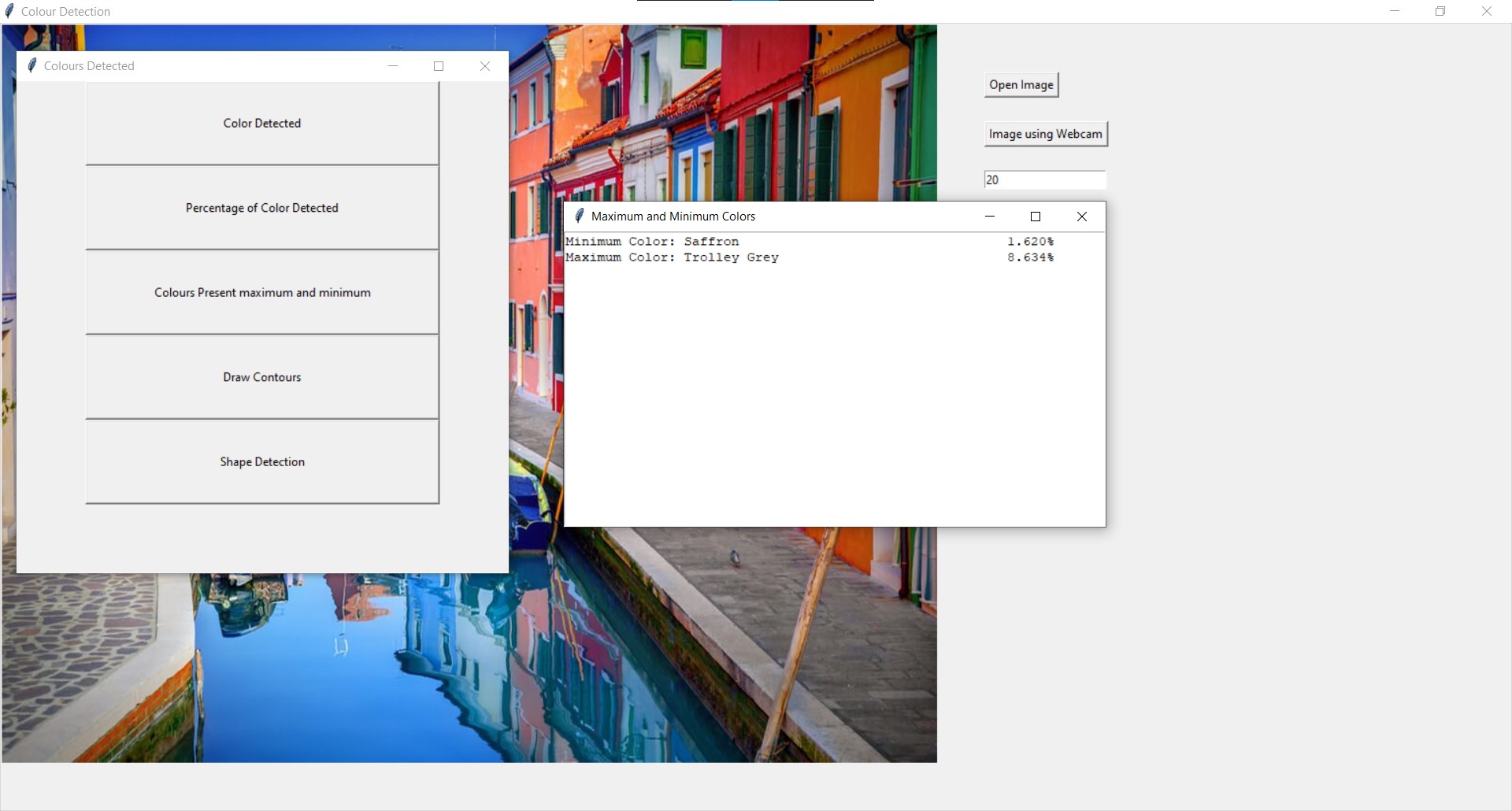
**(Number of clusters set, options of various functions enabled)**

****

**(Output of Color Detected)**

****

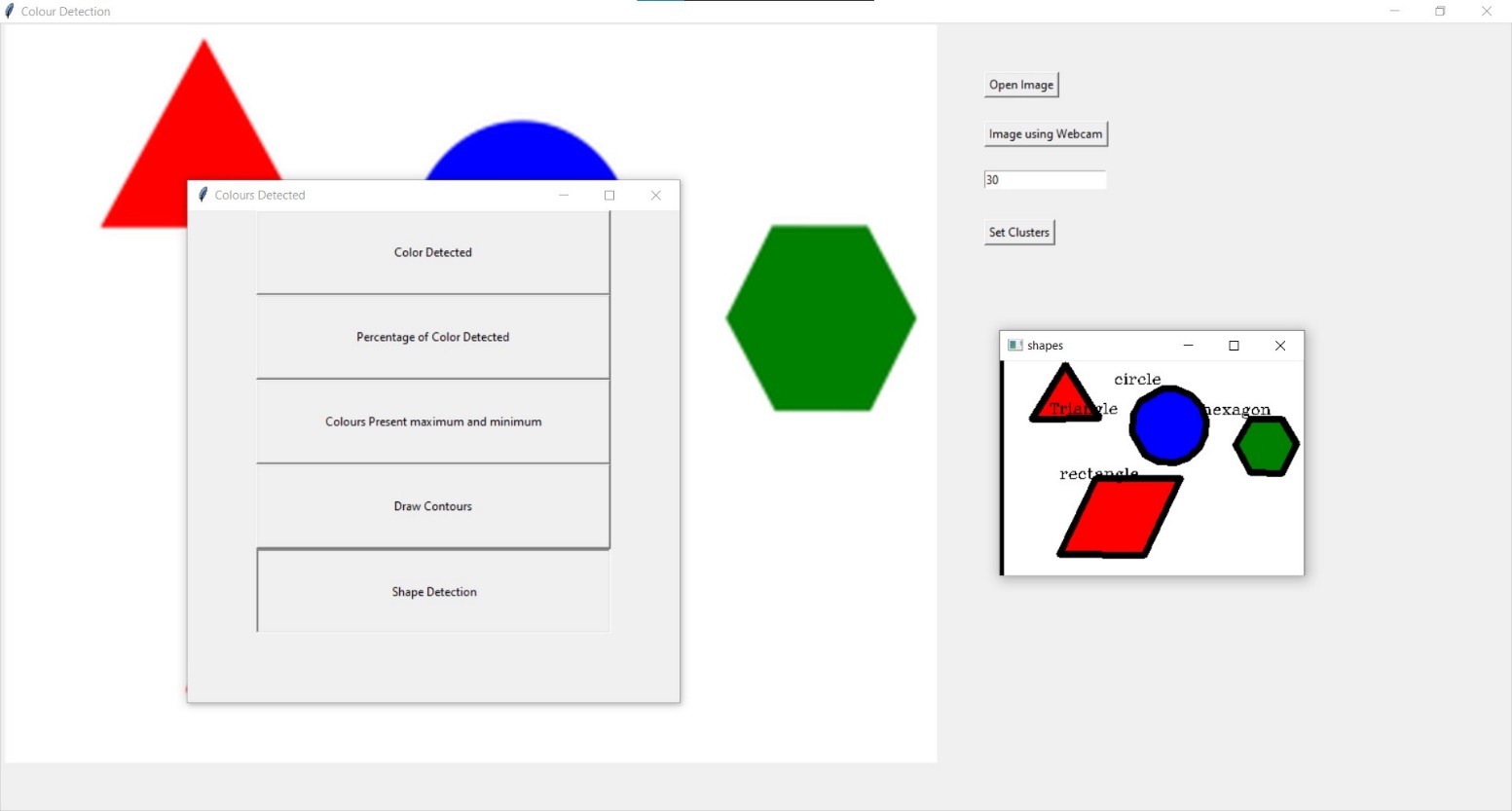
**(Output of percentage of color detected)**

****

**(Output of color present maximum and minimum)**

****

**(Output of draw contour, contours drawn on uploaded image)**



**(Output of Shape Detection, shapes of objects detected and labelled)**

1. **Conclusion:**

The given system successfully uses k-means clustering to identify various colors present in the uploaded image. The image can be uploaded using the files on the computer and using the webcam of your computer. It also gives us the percentage of each cluster present up to the accuracy of 3 decimal points. The contours of the image are clearly visible showing us the various bounded objects present in the image uploaded.

1. **References:**
2. <https://www.tutorialspoint.com/python/python_gui_programming.htm>
3. <https://www.javatpoint.com/opencv>
4. <https://www.geeksforgeeks.org/find-and-draw-contours-using-opencv-python/>
5. <https://www.analyticsvidhya.com/blog/2019/08/comprehensive-guide-k-means-clustering/>

# Digital Image Processing, 4th Edition, Rafael C. Gonzalez, University of Tennessee, Richard E. Woods, MedData Interactive