OpenCV – Overview :

Computer Vision

Computer vision is a process by which we can understand the images and videos how they are stored and how we can manipulate and retrieve data from them. Computer Vision is the base or mostly used for Artificial Intelligence. Computer-Vision is playing a major role in self-driving cars, robotics as well as in photo correction apps.

OpenCV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing and now it plays a major role in real-time operation which is very important in today's systems. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When it integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To Identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both academic and commercial use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing.

Look at the following images

from the above original image, lots of pieces of information that are present in the original image can be obtained. Like in the above image there are two faces available and the person(I) in the images wearing a bracelet, watch, etc so by the help of OpenCV we can get all these types of information from the original image.

It's the basic introduction to OpenCV we can continue the Applications and all the things in our upcoming articles.

Applications of OpenCV :

There are lots of applications which are solved using OpenCV, some of them are listed below

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anamoly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

OpenCV Functionality :

- Image/video I/O, processing, display (core, imgproc, highgui)
- Object/feature detection (objdetect, features2d, nonfree)
- Geometry-based monocular or stereo computer vision (calib3d, stitching, videostab)
- Computational photography (photo, video, superres)
- Machine learning & clustering (ml, flann)

  CUDA acceleration (gpu)

Image-Processing

Image processing is a method to perform some operations on an image, in order to get an enhanced image and or to extract some useful information from it.

If we talk about the basic definition of image processing then “Image processing is the analysis and manipulation

of a digitized image, especially in order to improve its quality".

Digital-Image :

An image may be defined as a two-dimensional function f(x, y), where x and y are spatial(plane) coordinates, and the amplitude of fat any pair of coordinates (x, y) is called the intensity or grey level of the image at that point.

In another word An image is nothing more than a two-dimensional matrix (3-D in case of coloured images) which is defined by the mathematical function f(x, y) at any point is giving the pixel value at that point of an image, the pixel value describes how bright that pixel is, and what colour it should be.

Image processing is basically signal processing in which input is an image and output is image or characteristics according to requirement associated with that image.

Image processing basically includes the following three steps:

- Importing the image
- Analysing and manipulating the image
- Output in which result can be altered image or report that is based on image analysis

How Does A Computer Read An Image?

We are humans we can easily make it out that is the image of a person who is me. But if we ask computer "is it my photo?". The computer can't say anything because the computer is not figuring out it all on its own.

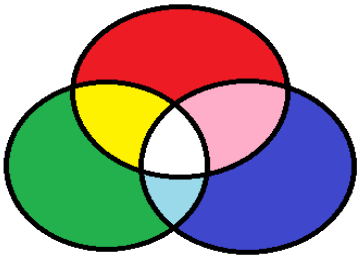
The computer reads any image as a range of values between 0 and 255. For any color image, there are 3 primary channels -red, green and blue.

Color Spaces in OpenCV | Python :

Color spaces are a way to represent the color channels present in the image that gives the image that particular hue. There are several different color spaces and each has its own significance.
Some of the popular color spaces are *RGB* (Red, Green, Blue), *CMYK* (Cyan, Magenta, Yellow, Black), *HSV* (Hue, Saturation, Value), etc.

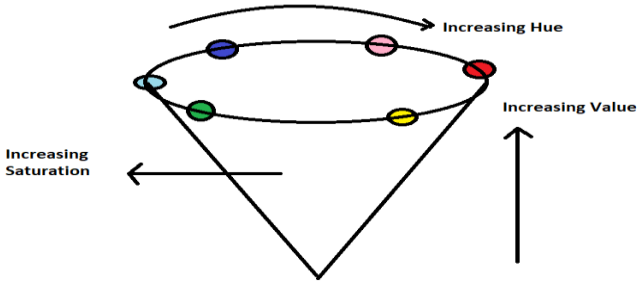
BGR color space:

OpenCV's default color space is RGB. However, it actually stores color in the BGR format. It is an additive color model where the different intensities of Blue, Green and Red give different shades of color.



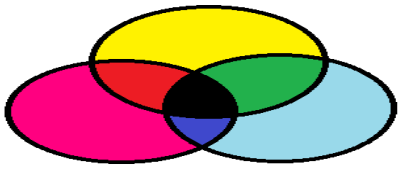
HSV color space:

It stores color information in a cylindrical representation of RGB color points. It attempts to depict the colors as perceived by the human eye. Hue value varies from 0-179, Saturation value varies from 0-255 and Value value varies from 0-255. It is mostly used for color segmentation purpose.



CMYK color space:

Unlike, RGB it is a subtractive color space. The CMYK model works by partially or entirely masking colors on a lighter, usually white, background. The ink reduces the light that would otherwise be reflected. Such a model is called subtractive because inks “subtract” the colors red, green and blue from white light. White light minus red leaves cyan, white light minus green leaves magenta, and white light minus blue leaves yellow.



OpenCV HSV range :

The HSV or Hue, Saturation and Value of a given object is the color space associated with the object in OpenCV where Hue represents the color, Saturation represents the greyness and Value represents the brightness and it is used to solve the problems related to computer vision because of its better performance when compared to RGB or Red, Blue and Green color space and the Hue range in HSV is [0,179], the Saturation range in HSV is [0,255] and the Value range in HSV is [0,255] and to perform object detection, finding the range of HSV is necessary.

The syntax to define HSV range in OpenCV is as follows:

```
hsvcolorspace = cv.cvtColor(image, cv.COLOR_BGR2HSV)

lower_hsvcolorspace = np.array([Hue range, Saturation range, Value range])

upper_hsvcolorspace = np.array([Hue range, Saturation range, Value range])
```

where hsvcolorspace is the conversion of the given image in RGB format to HSV format,

lower_hsvcolorspace is the lower threshold for a range of some color,

upper_hsvcolorspace is the upper threshold for a range of some color,

Hue range is the Hue range in HSV which is [0,179],

Saturation range is the Saturation range in HSV which is [0,255] and

The value range is the Value range in HSV which is [0,255].

Working of HSV range in OpenCV :

- The HSV or Hue, Saturation, and value of a given object is the color space associated with the object in OpenCV.

- The Hue in HSV represents the color, Saturation in HSV represents the greyness, and Value in HSV represents the brightness.

- Whenever we want to solve problems related to object detection, it is necessary to use HSV and find the range of HSV.

- The Hue, Saturation, and Value in HSV have their own range of values.

- The Hue range in HSV is [0,179], the Saturation range in HSV is [0,255] and the Value range in HSV is [0,255].

- There is also an upper bound and lower bound range for a range of each color in HSV.

- The HSV or Hue, Saturation, and value of a given object provide better performance when compared to RGB or Red, Blue, and Green color space and hence it is used widely in the area of computer vision.

Basic color detection using hsv ranges:

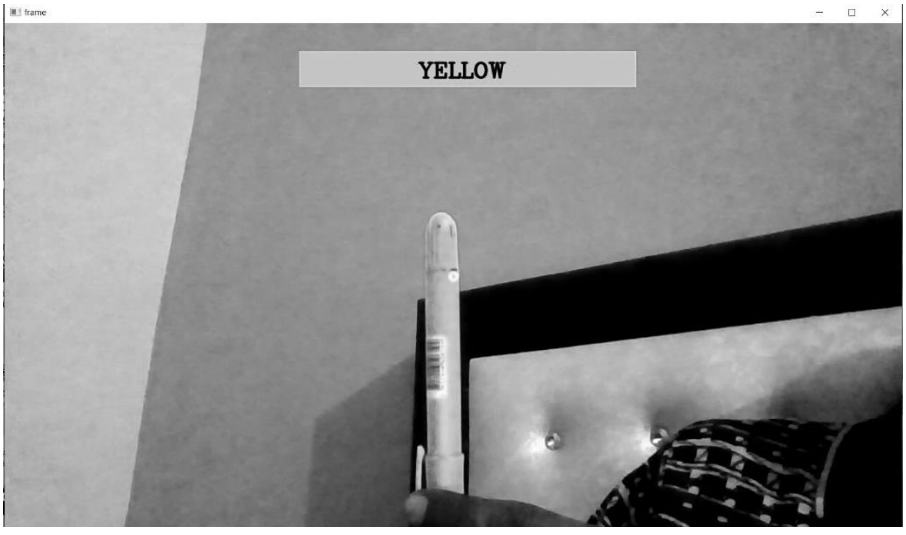
Source code:

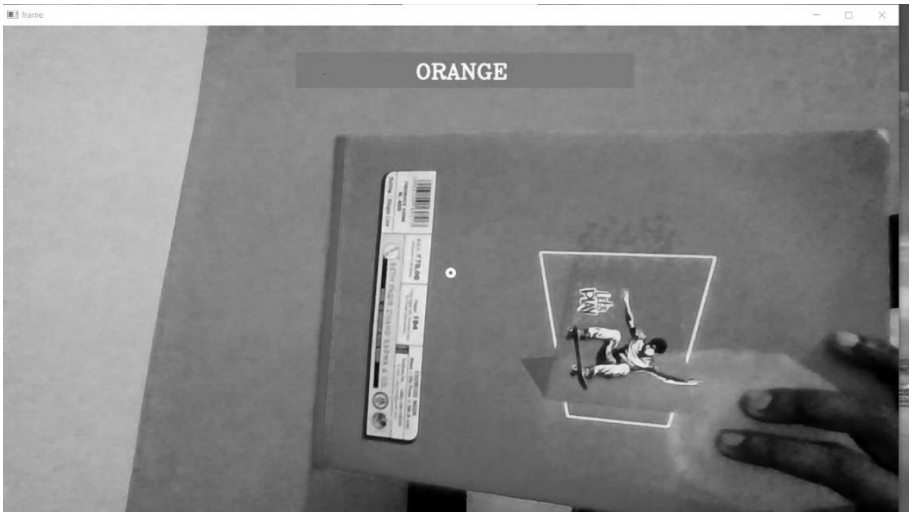
```
import cv2

cap = cv2.VideoCapture(0)
cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

while True:
    _,frame = cap.read()
    height,width,_ = frame.shape
    hsv_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)

    cx = int(width/2)
    cy = int(height/2)

    #Picking pixel center
    pixel_center= hsv_frame[cy,cx]

    hue_value = pixel_center[0]
    s_value = pixel_center[1]
    v_value = pixel_center[2]

    color ="Undefined"

    #RED
    if hue_value <4 :
        if v_value <15:
            color ="BLACK"
        elif v_value > 244 & s_value<7:
            color ="WHITE"
        else:
            color = "RED"

    #ORANGE
    elif hue_value < 18:
        if v_value <15:
            color ="BLACK"
        elif v_value > 244 & s_value < 7:
            color = "WHITE"
        else:
            color = "ORANGE"

    #YELLOW
    elif hue_value < 34:
        if v_value <15:
            color ="BLACK"
        elif v_value > 244 & s_value<7:
```

```
        color ="WHITE"
    else:
        color = "YELLOW"

#GREEN
elif hue_value < 68:
    if v_value <15:
        color ="BLACK"
    elif v_value > 244 & s_value<7:
        color ="WHITE"
    else:
        color = "GREEN"

#CYAN
elif hue_value < 92:
    if v_value <15:
        color ="BLACK"
    elif v_value > 244 & s_value<7:
        color ="WHITE"
    else:
        color = "CYAN"

#BLUE
elif hue_value < 126:
    if v_value <15:
        color ="BLACK"
    elif v_value > 244 & s_value<7:
        color ="WHITE"
    else:
        color = "BLUE"

#VIOLET
elif hue_value < 145:
    if v_value <15:
        color ="BLACK"
    elif v_value > 244 & s_value<7:
        color ="WHITE"
    else:
        color = "VIOLET"

#PINK
elif hue_value < 170:
    if v_value <15:
        color ="BLACK"
    elif v_value > 244 & s_value<7:
        color ="WHITE"
    else:
        color = "PINK"

else:
    if v_value < 15:
        color = "BLACK"
    elif v_value > 244 & s_value < 7:
        color = "WHITE"
```

```
    else:
        color = "RED"

    pixel_center_bgr = frame[cy,cx]
    b,g,r = int(pixel_center_bgr[0]), int(pixel_center_bgr[1]), int(pixel_center_bgr[2])

    cv2.circle(frame, (cx,cy),5, (255,255,255), 3)
    cv2.rectangle(frame, (cx-220, 40), (900, 90), (b,g,r), -1)
    if v_value > 244 & s_value<7:
        cv2.putText(frame, color, (cx-50, 77), 3, 1, (0, 0, 0), 2,cv2.LINE_AA)
    else:
        cv2.putText(frame, color, (cx-50,77),3,1, (255,255,255), 2,cv2.LINE_AA)

    cv2.imshow("frame", frame)
    key = cv2.waitKey(1)
    if key == 27:
        break

cap.release()
cv2.destroyAllWindows()
```

Output :



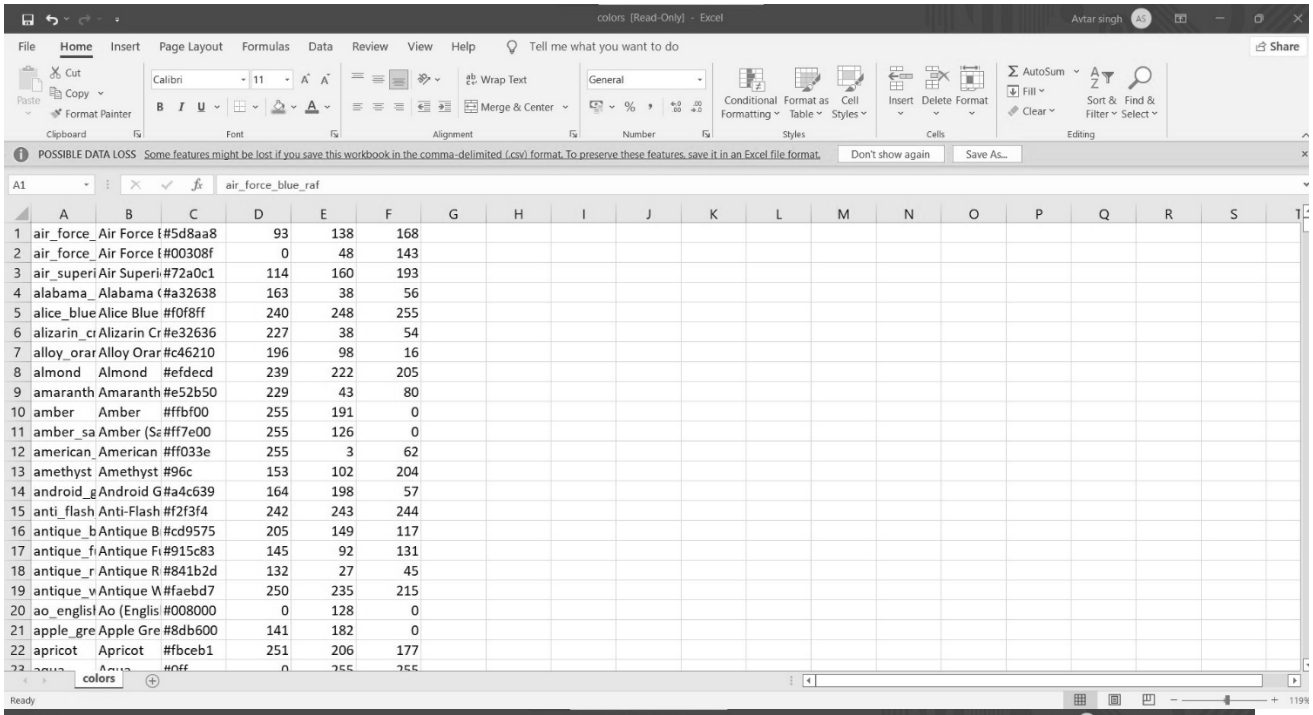


Color detection using bgr :

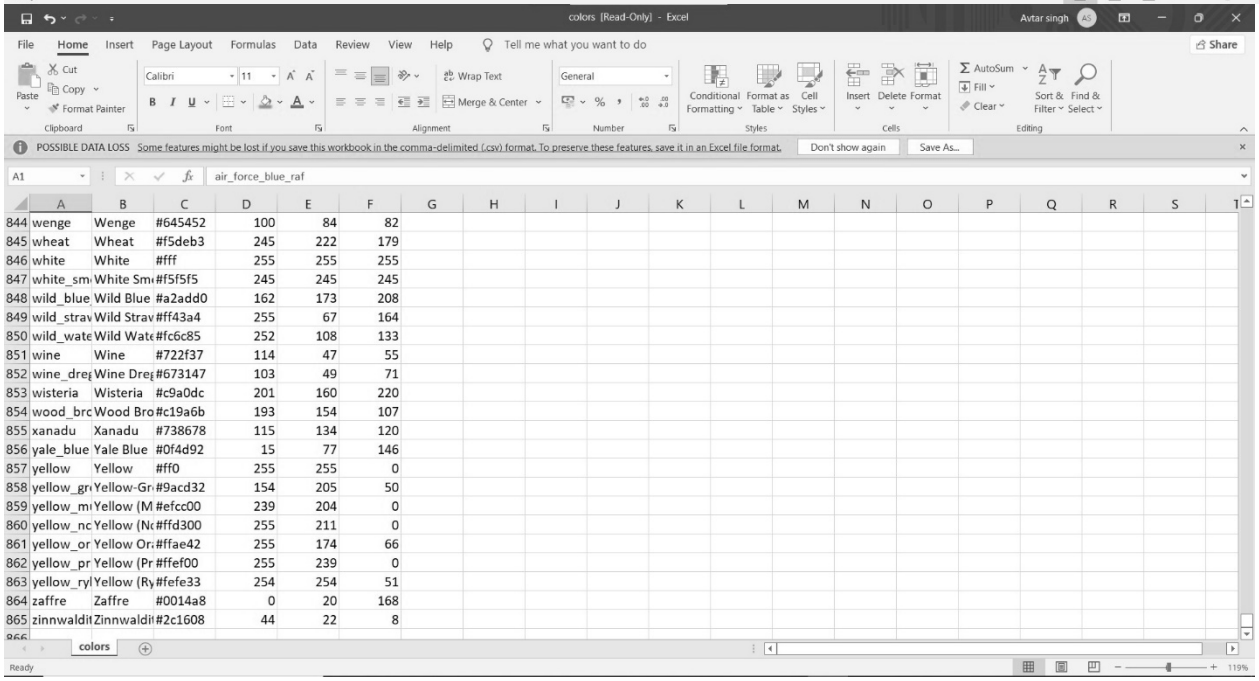
In this, we will use the color stored in the csv file to get the nearest color to the object shown in webcam. We will find distance using :

```
d = abs(R - int(csv.loc[i, "R"])) + abs(G - int(csv.loc[i, "G"])) + abs(B - int(csv.loc[i, "B"]))
```

CSV File:



	A	B	C	D	E	F
1	air_force_	Air Force B	#5d8aa8	93	138	168
2	air_force_	Air Force B	#00308f	0	48	143
3	air_superi	Air Superi	#72a0c1	114	160	193
4	alabama_	Alabama (	#a32638	163	38	56
5	alice_blue	Alice Blue	#f0f8ff	240	248	255
6	alizarin_cr	Alizarin Cr	#e32636	227	38	54
7	alloy_orar	Alloy Orar	#c46210	196	98	16
8	almond	Almond	#efdecd	239	222	205
9	amaranth	Amaranth	#e52b50	229	43	80
10	amber	Amber	#ffbf00	255	191	0
11	amber_sa	Amber (Sa	#ff7e00	255	126	0
12	american_	American	#ff033e	255	3	62
13	amethyst	Amethyst	#96c	153	102	204
14	android_g	Android G	#a4c639	164	198	57
15	anti_flash	Anti-Flash	#f2f3f4	242	243	244
16	antique_b	Antique B	#cd9575	205	149	117
17	antique_fi	Antique Fı	#915c83	145	92	131
18	antique_r	Antique R	#841b2d	132	27	45
19	antique_v	Antique V	#faebd7	250	235	215
20	ao_englis	Ao (Englis	#008000	0	128	0
21	apple_gre	Apple Gre	#8db600	141	182	0
22	apricot	Apricot	#fbceb1	251	206	177



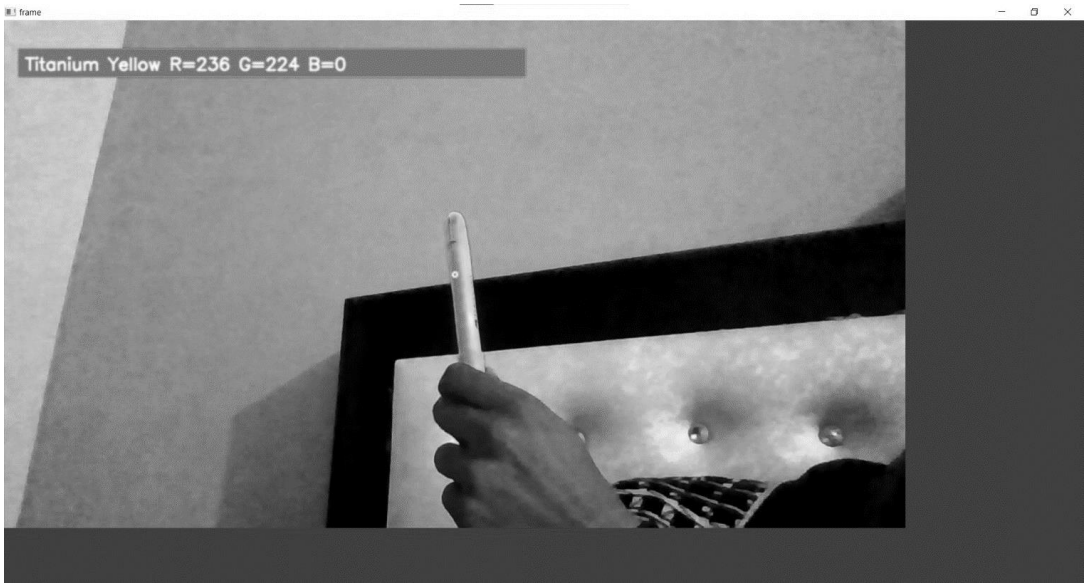
	A	B	C	D	E	F
844	wenge	Wenge	#645452	100	84	82
845	wheat	Wheat	#f5deb3	245	222	179
846	white	White	#fff	255	255	255
847	white_sm	White Sm	#f5f5f5	245	245	245
848	wild_blue	Wild Blue	#a2add0	162	173	208
849	wild_strav	Wild Strav	#ff43a4	255	67	164
850	wild_wate	Wild Wate	#fc6c85	252	108	133
851	wine	Wine	#722f37	114	47	55
852	wine_dreg	Wine Dreg	#673147	103	49	71
853	wisteria	Wisteria	#c9a0dc	201	160	220
854	wood_brc	Wood Bro	#c19a6b	193	154	107
855	xanadu	Xanadu	#738678	115	134	120
856	yale_blue	Yale Blue	#0f4d92	15	77	146
857	yellow	Yellow	#ff0	255	255	0
858	yellow_gr	Yellow-Gr	#9acd32	154	205	50
859	yellow_mi	Yellow (M	#efcc00	239	204	0
860	yellow_nc	Yellow (Nc	#ffd300	255	211	0
861	yellow_or	Yellow Or	#ffae42	255	174	66
862	yellow_pr	Yellow (Pr	#ffef00	255	239	0
863	yellow_ryl	Yellow (Ry	#fefe33	254	254	51
864	zaffre	Zaffre	#0014a8	0	20	168
865	zinnwaldit	Zinnwaldit	#2c1608	44	22	8

Source code:

```
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)

cap = cv2.VideoCapture(0)

cap.set(cv2.CAP_PROP_FRAME_WIDTH, 1280)
cap.set(cv2.CAP_PROP_FRAME_HEIGHT, 720)

while True:
    _,frame = cap.read()
    height,width, _ = frame.shape

    cx = int(width/2)
    cy = int(height/2)

    pixel_center = frame[cy,cx]
    b,g,r = int(pixel_center[0]), int(pixel_center[1]), int(pixel_center[2])
    print(pixel_center)
    cv2.circle(frame,(cx,cy),3, (255,255,255), 2)

    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


    text = getColorName(r, g, b) + ' R=' + str(r) + ' G=' + str(g) + ' B=' + str(b)

    cv2.rectangle(frame,(20,40), (cx+100, 80),(b,g,r),-1)
    cv2.putText(frame,text,(30,70),0,0.8,(255,255,255),2,cv2.LINE_AA)

    if (r + g + b >= 600):
        cv2.putText(frame, text, (30, 70), 2, 0.8, (0, 0, 0), 2, cv2.LINE_AA)

    cv2.imshow("frame", frame)

    key = cv2.waitKey(1)
    if key == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

Output:


frame
Blue (Ncs) R=0 G=134 B=190

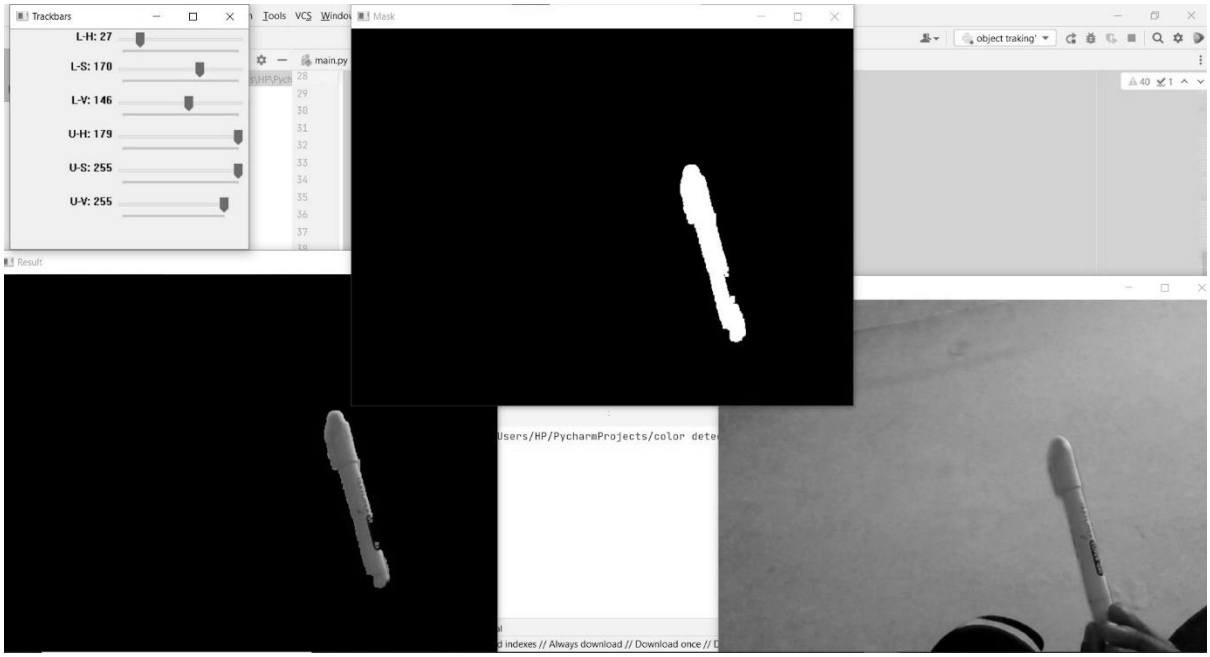

frame
Titanium Yellow R=236 G=224 B=0

Object tracking using hsv masks:

Source Code:

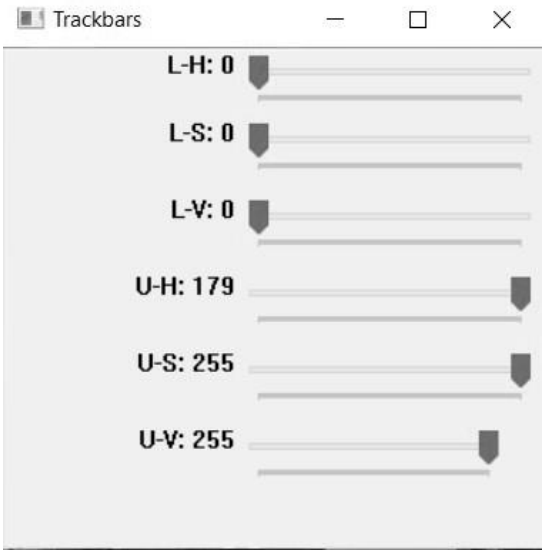
```
import cv2
import numpy as np

def nothing(x):
    pass

cv2.namedWindow("Trackbars")

cv2.createTrackbar("L-H","Trackbars",0,179,nothing)
cv2.createTrackbar("L-S","Trackbars",0,255,nothing)
cv2.createTrackbar("L-V","Trackbars",0,255,nothing)
cv2.createTrackbar("U-H","Trackbars",179,179,nothing)
cv2.createTrackbar("U-S","Trackbars",255,255,nothing)
cv2.createTrackbar("U-V","Trackbars",255,255,nothing)

cap = cv2.VideoCapture(0)

while True:
    _,frame = cap.read()
    hsv_frame = cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
    l_h = cv2.getTrackbarPos("L-H","Trackbars")
    l_s = cv2.getTrackbarPos("L-V", "Trackbars")
    l_v = cv2.getTrackbarPos("L-S", "Trackbars")
    u_h = cv2.getTrackbarPos("U-H", "Trackbars")
    u_s = cv2.getTrackbarPos("U-S", "Trackbars")
    u_v = cv2.getTrackbarPos("U-V", "Trackbars")

    lower = np.array([l_h,l_s,l_v])
    upper = np.array([u_h,u_s,u_v])

    mask = cv2.inRange(hsv_frame,lower,upper)

    kernal = np.ones((5, 5), "uint8")
    mask = cv2.dilate(mask,kernal)
    result = cv2.bitwise_and(frame,frame,mask=mask)

    cv2.imshow("Frame", frame)
    cv2.imshow("Mask", mask)
    cv2.imshow("Result", result)

    key = cv2.waitKey(1)
    if key == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

Output:

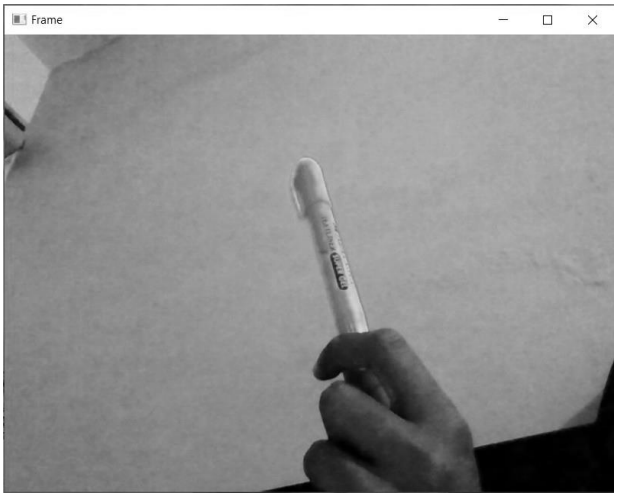


There are 4 windows as the output:

1. Trackbar : To adjust the lower values of the h, s, v individually so that the object shown in webcam comes in the range of lower and upper hsv.
2. Frame : The actual video stream of webcam.
3. Mask : The black and white mask of the frame after setting the h,s,v values. The white portion is the object that comes in the range of selected lower and upper hsv values
4. Result : The object that is masked is shown in original color in this window.



TRACKBAR



FRAME

MASK

RESULT

REFERENCES:

- ✓ https://www.geeksforgeeks.org/
- ✓ https://opencv.org/
- ✓ https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html
- ✓ https://stackoverflow.com/