COLOR DETECTION WITH OPENCV AND PYTHON

Before of start with the explanation of the program color detection something that is important and should be clear is the type of HSV model (Hue, Saturation, Value) or also known as HSB (Hue, Saturation, Brightness) that is used in this type of detections, in this model each channel has a value with which obtains get each color.

- Hue has a value between 0 and 179.
- Saturation has a value from 0 to 255.
- Value or brightness has a value from 0 to 255.

Hue is responsible for modeling the color type, saturation we could understand as the strength with which the color is displayed and finally value is the brightness or intensity of the color, the figure 1 shows the relationship of this.

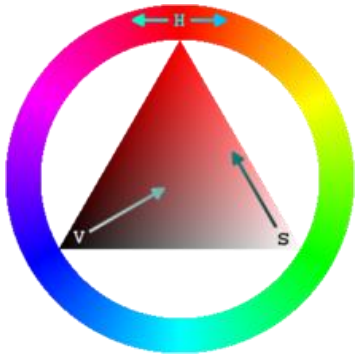


Figure 1: HSV color space. Wikipedia

Once understand this we will pass to the programming, we will use the Open cv and NumPy library to this project.

We will start by importing these libraries into the archive and initialize our video capture, in this project was used the extension .ipynb but you can use .py without problems.

```
import cv2
import numpy as np
import imutils

#START THE VIDEOCAPTURE
Cap=cv2.VideoCapture(0)
```

Now we will put in practice what we have seen in the beginning about the HSV model, first we have to take into account that to detect correctly and of better way the colors we need of HSV values of the color both dark and the brightest level this to have a margin of between these values is the color to detect, in this case we can observe in the code shown later that the value for the variables "Dark_red" and "Light_red" appear twice this is because in HSV the color red appears twice, in the next graphic HSV shown in the figure 2 we can see how the color red is found both in the right side and the left side of the image.

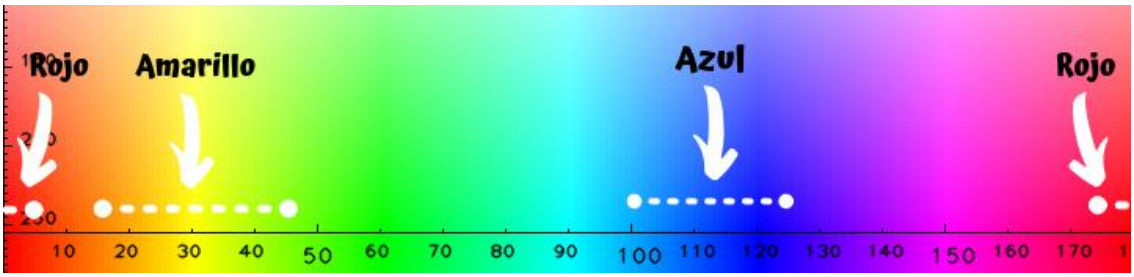


Figure 2: Graphic of Hue values to colors in HSV model. OMES

Here we can see that the values shown below in the graphic are the Hue values, these are values that we will see in the next code of programming.

In this part of the code what is done is to assign the dark and light values of each color to detect in a NumPy array, you will see that add "uint8", this is the usual datatype to images.

```
#VALUES TO DETECT COLORS IN HSV SCALE
Dark_red1= np.array([0,50,120], np.uint8)
Light_red1= np.array([10,255,255], np.uint8)

Dark_red2= np.array([170,100,100], np.uint8)
Light_red2= np.array([179,255,255], np.uint8)

Dark_green= np.array([40,70,80], np.uint8)
Light_green= np.array([70,255,255], np.uint8)

Dark_blue= np.array([90,60,0], np.uint8)
Light_blue= np.array([121,255,255], np.uint8)

Dark_yellow= np.array([25,70,100], np.uint8)
Light_yellow= np.array([30,255,255], np.uint8)
```

So that these values will be optimal to its detection should be tried with different values until get the appropriate one, you can use as guide the HSV graphic shown in the figure 2.

Following with the code we will create our while cycle in which we will verify if the "Ret" variable is True and our variable "Frame" that will read the contain of the video capture, followed by this we will have that convert our frames from BGR (format that Open cv use by default) to HSV that is the format that we will use to the detection.

```
while True:
    #READ THE FRAMES AND CHECK THE RET
    Ret, Frame= Cap.read()
    if Ret==True:
    #TRANSORM THE ORIGINALS BGR FRAMES TO HSV TO DETECT COLORS
        Transform_hsv= cv2.cvtColor(Frame,cv2.COLOR_BGR2HSV)
```

Now we will create the mask for each color with the function .inRange() that use three parameters that are the principal matrix in this case it will be "Transform_hsv" followed by the matrix with the value dark of the color and finally the brightness matrix, the function of these mask is specific that values we want to keep.

For the case of the mask of color red that had two values low and high, they will have to add both masks to create only one as it will be appreciated in the variable "Mask_red_end".

```
#CREATE A MASK TO EVERY COLOR WITH THE DARK AND LIGHT VALUE
        Mask_red1=cv2.inRange(Transform_hsv, Dark_red1, Light_red1)
        Mask_red2=cv2.inRange(Transform_hsv, Dark_red2, Light_red2)
        Mask_red_end=Mask_red1 + Mask_red2

        Mask_green=cv2.inRange(Transform_hsv, Dark_green, Light_green)
        Mask_blue=cv2.inRange(Transform_hsv, Dark_blue, Light_blue)
        Mask_yellow=cv2.inRange(Transform_hsv, Dark_yellow, Light_yellow)
```

Continuing with the code, we will proceed to create a threshold for each mask since this will help us to the hour of drawing the contours of the objects of better way later.

The value for the "Thresh" variable was of 100 but is recommended to try with other values to find the one that best suit to the needs.

In this function .threshold() we will give four parameters which are:

- The variable to apply the thresholding to.
- The value of the threshold (in this case declared in the variable "Thresh").
- The maximum value (usually 255)
- The method of thresholding.

```
#THRESHOLD
Thresh=100
Ret, Thresh_red= cv2.threshold(Mask_red_end,Thresh,255,cv2.THRESH_BINARY)
Ret, Thresh_green= cv2.threshold(Mask_green,Thresh,255,cv2.THRESH_BINARY)
Ret, Thresh_blue= cv2.threshold(Mask_blue,Thresh,255,cv2.THRESH_BINARY)
Ret, Thresh_yellow= cv2.threshold(Mask_yellow,Thresh,255,cv2.THRESH_BINARY)
```

To use this thresholding that we just finish of make we will need of two functions, the first .findContours() to which we will give three parameters, the first will be the variable binary of Thresh assigned to each color, as second parameter we will use the detection mood and finally the detection method, to know about of the different mood and methods you can read the next document https://goo.su/QBb8Ms2.

```
#FIND THE CONTOUR OF THE OBJECTS TO DETECT
      Countour1= cv2.findContours(Thresh_red, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
      Countour1= imutils.grab_contours(Countour1)

      Countour2= cv2.findContours(Thresh_green, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
      Countour2= imutils.grab_contours(Countour2)

      Countour3= cv2.findContours(Thresh_blue, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
      Countour3= imutils.grab_contours(Countour3)

      Countour4= cv2.findContours(Thresh_yellow, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
      Countour4= imutils.grab_contours(Countour4)
```

The second function will be .grab_contours() of the library imutils to which we will give the parameter of the variable crated to find the contours previously in .findContours().

Already having found the contours we will pass to create a for loop to each contour in which we going to define the area and from this we will draw the contour found join with a circle and a text.

First, we going to create a variable call "c", next we will find the area of the contour with the function .contourArea(), we will pass to a conditional if in which if the value of the area is greater than 4000 it will draw the contours with the function .drawContours() this function will have as parameters the image in which will be drawn the contour, next of the elements of the tuple of the variable "c" [c]. Then contourIdx, then the color of the contour and finally the width of the line of the contour line.

To draw the circle and put the text we need the coordinates, for this we use the function .moments() which gives us the values in the axes x,y. To learn more about this function moments visit the page https://goo.su/JYiT.

Having the values of x,y we will pass to draw the circle with the function .circle() to which we will give as parameters:

- The image on which the circle will be drawn.
- The x,y coordinates.
- The radius of the circle.
- The color.
- The thickness of the circle.

And finally, we will show the text in the image with the parameters of:

- Image in which the text will be displayed.
- Text to show.
- Coordinates in x,y.
- Type of font.
- Size of the font.
- Color of the text.
- Thickness of the text.

```
#DRAW THE CONTOURS, A CIRCLE AND PUT TEXT FOR EVERY CONTOUR IN THE FRAMES
        for c in Countour1:
            Area1=cv2.contourArea(c)
            if Area1 > 4000:
                cv2.drawContours(Frame,[c],-1,(0,255,0),2)
                M=cv2.moments(c)
                x=int(M["m10"]/M["m00"])
                y=int(M["m01"]/M["m00"])
                cv2.circle(Frame,(x,y),5,(255,255,255),-1)
                cv2.putText(Frame,"Red",(x-20,y),cv2.FONT_HERSHEY_SIMPLEX,2,(255,255,255),2)
```

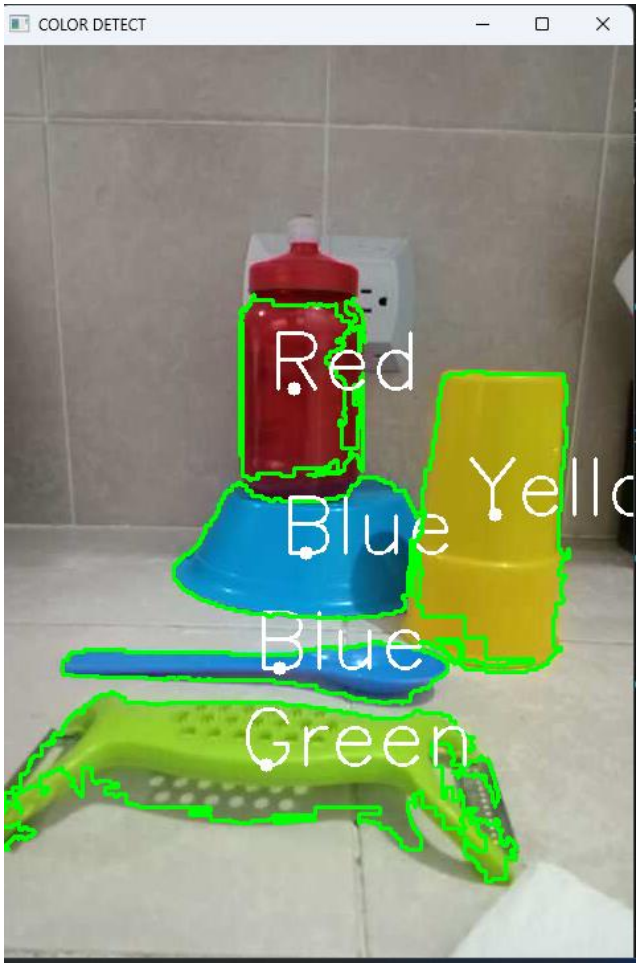
All the for-loops to each contour have the same syntax, except that some variables should be changed as shown in the following code.

```
#DRAW THE CONTOURS, A CIRCLE AND PUT TEXT FOR EVERY CONTOUR IN THE FRAMES
        for c in Countour1:
            Area1=cv2.contourArea(c)
            if Area1 > 4000:
                cv2.drawContours(Frame,[c],-1,(0,255,0),2)
                M=cv2.moments(c)
                x=int(M["m10"]/M["m00"])
                y=int(M["m01"]/M["m00"])
                cv2.circle(Frame,(x,y),5,(255,255,255),-1)
                cv2.putText(Frame,"Red",(x-20,y),cv2.FONT_HERSHEY_SIMPLEX,2,(255,255,255),2)

        for c in Countour2:
            Area2=cv2.contourArea(c)
            if Area2 > 4000:
                cv2.drawContours(Frame,[c],-1,(0,255,0),2)
                M=cv2.moments(c)
                x=int(M["m10"]/M["m00"])
                y=int(M["m01"]/M["m00"])
                cv2.circle(Frame,(x,y),5,(255,255,255),-1)
                cv2.putText(Frame,"Green",(x-20,y),cv2.FONT_HERSHEY_SIMPLEX,2,(255,255,255),2)

        for c in Countour3:
            Area3=cv2.contourArea(c)
            if Area3 > 4000:
                cv2.drawContours(Frame,[c],-1,(0,255,0),2)
                M=cv2.moments(c)
                x=int(M["m10"]/M["m00"])
                y=int(M["m01"]/M["m00"])
                cv2.circle(Frame,(x,y),5,(255,255,255),-1)
                cv2.putText(Frame,"Blue",(x-20,y),cv2.FONT_HERSHEY_SIMPLEX,2,(255,255,255),2)

        for c in Countour4:
            Area4=cv2.contourArea(c)
            if Area4 > 4000:
                cv2.drawContours(Frame,[c],-1,(0,255,0),2)
                M=cv2.moments(c)
                x=int(M["m10"]/M["m00"])
                y=int(M["m01"]/M["m00"])
                cv2.circle(Frame,(x,y),5,(255,255,255),-1)
                cv2.putText(Frame,"Yellow",(x-20,y),cv2.FONT_HERSHEY_SIMPLEX,2,(255,255,255),2)
```

Finally, we will show the video in screen with the function .imshow() and as parameters the name that will have the window in screen and the image that will be shown in this case the "Frame" variable that will contain the video capture and we will create a condition to close the screen of video with the key ESC of the keyboard, the key ESC in code ASCII correspond to number 27.

```
        cv2.imshow("COLOR DETECT", Frame)
        t=cv2.waitKey(1)
        if t==27:break

Cap.release()
cv2.destroyAllWindows()
```

Now will be seen the results obtained with this code, it can be seen as detect the colors correctly and the contour of the images is also detected, something to keep in mind when we make the detection it is about the lighting in some occasions some colors are not detected due to the lack of lighting like the contours can be drawn in shadows as the case of the color green that takes the reflection of the shadow as a contour of the object.



If you got to this part, I thank you for the time that you gave me and I hope that I have helped to you with you doubts and concerns of learning more, this is just a small fragment of explain if you want to continue learning and improve you can visit these pages that will help you in the topic of artificial vision and Python.

1.- https://omes-va.com/

2.- https://www.pythonpool.com/

3.- https://hetpro-store.com/TUTORIALES/

4.- https://learnopencv.com/

Wisdom is the principal thing; therefore get wisdom: yea, with all thou hast gotten get understanding. Proverbs 4:7