# Computer Vision

My Personal Notes

## Color Filtering/Segmentation/Detection – HSV

One of the first hands-on application that fledgling Computer Vision enthusiasts start with is **color filtering/detection**.

**Why using HSV color space for this?**

** In real world images there is always **variations** in the image color values due to various *lightening conditions, shadows* and, even due to *noise* added by the camera while clicking and subsequently processing the image.

- To over the above color variations, we will perform color detection in the HSV color space.

The whole idea is, we will be creating a binary mask where the white region will represent our target color (red in this case) and black will represent rest of the colors.

So we will be using the `inRange()` function to filter out the required color.

```
mask = cv2.inRange(src, lowerb, upperb)
```

- `src:` This will be src array, in our case it will be the target Hsv image
- `lowerb:` This is the lower boundary array, in our case it will be an array of 3 values , these values will be the lower range of hsv values.
- `lowerb:` This is the upper boundary array, in our case it will be an array of 3 values , these values will be the upper range of hsv values.

Now using inRange() function, we will filter all the colors that are in between the range of lower and upper boundaries. For example to capture red color we can filter out **Hue** in range 160-180 and the values and saturation can be in range 50-255. This will give us different variations of red both dark-bright and dull-pure red color variations are captured.

**How to get HSV values?**

OpenCV halves the H values to fit the range [0,255], so H value instead of being in range [0, 360], is in range [0, 180]. S and V are still in range [0, 255].

Look at this site which gives you HSV values for any RGB value.

```
import cv2
import numpy as np
greenBGR = np.uint8([[[0,255,0 ]]])

hsv_green = cv2.cvtColor(greenBGR,cv2.COLOR_BGR2HSV)
print (hsv_green)
```

Output :

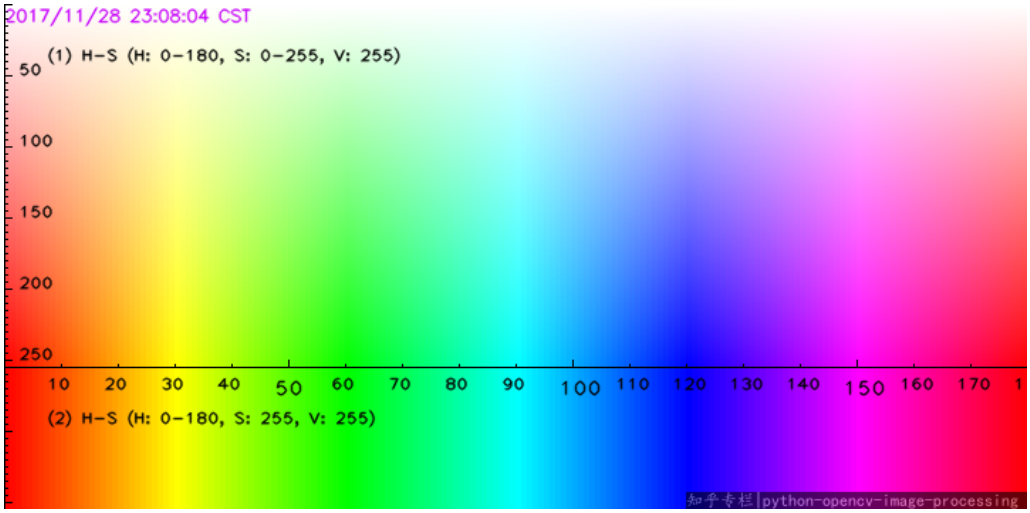
```
[[[ 60 255 255]]]
```

Now you take [H-10, 100,100] and [H+10, 255, 255] as lower bound and upper bound respectively.

---

In this blog post we will try to detect the rose flower in the image below using color detection:

In *OpenCV*, **Hue** has values from 0 to 180, **Saturation** and **Value** from 0 to 255. Thus, OpenCV uses HSV ranges between (0-180, 0-255, 0-255). In OpenCV, the H values 179, 178, 177 and so on are as close to the true RED as




From the above HSV color map, to find a color, usually just look up for the range of H and S, and set v in range (20, 255).

```
To find the orange color, we look up for the map, and find the best range:
H :[10, 25], S: [100, 255], and V: [20, 255]


So the mask is
cv2.inRange(hsv,(10, 100, 20), (25, 255, 255) )
```

---

Our rose flower is predominantly red, so we will set the **cv2.inrange** function with the range of HSV values of red color.The HSV values for **true RED** are (0, 255, 255) and to accommodate for color variations, we will consider a **range** of **HSV values for the red color**.

- The *red* color, in *OpenCV*, has the hue values approximately in the range of 0 to 10 and 160 to 180.

We will use the `cv2.inRange` to generate the mask that has a value of 255 for pixels where the HSV values fall within the specified **color range** and a value of 0 for pixels whose values don't lie in this interval. The mask values are either 255 or 0. 255 represents color pixels and 0 represents non color pixels.

Full code :

```
1 import cv2
2 import numpy as np
```



```
9 image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
```

```
10
11  # lower boundary RED color range values; Hue (0 - 10)
12  lower1 = np.array([0, 100, 20])
13  upper1 = np.array([10, 255, 255])
14
15  # upper boundary RED color range values; Hue (160 - 180)
16  lower2 = np.array([160,100,20])
17  upper2 = np.array([179,255,255])
18
19  lower_mask = cv2.inRange(image, lower1, upper1)
20  upper_mask = cv2.inRange(image, lower2, upper2)
21
22  full_mask = lower_mask + upper_mask;
23
24  result = cv2.bitwise_and(result, result, mask=full_mask)
25
26  cv2.imshow('mask', full_mask)
27  cv2.imshow('result', result)
28
29  cv2.waitKey(0)
30  cv2.destroyAllWindows()
```

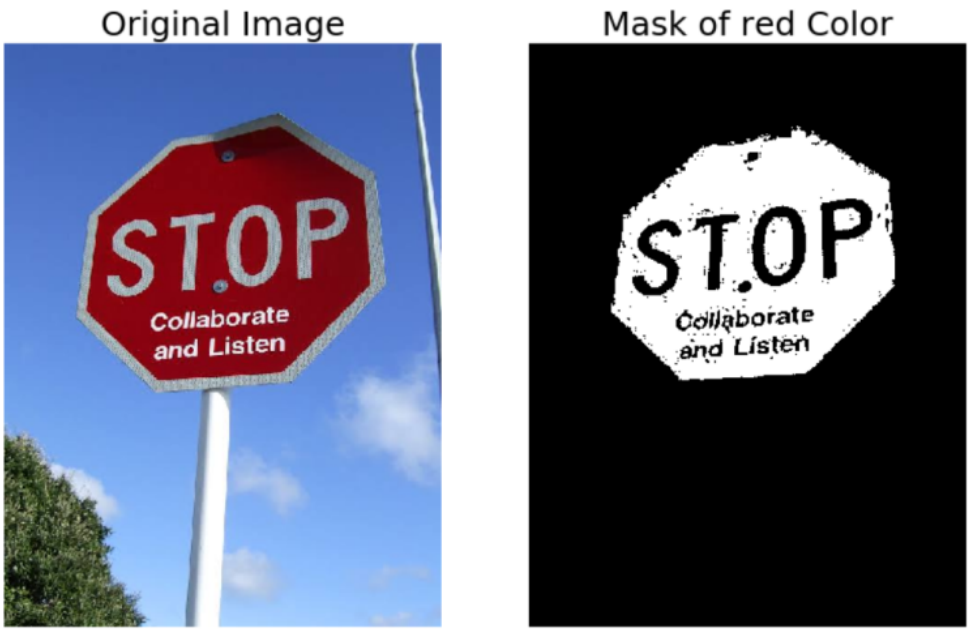
Output :

*MASK*

Another example :

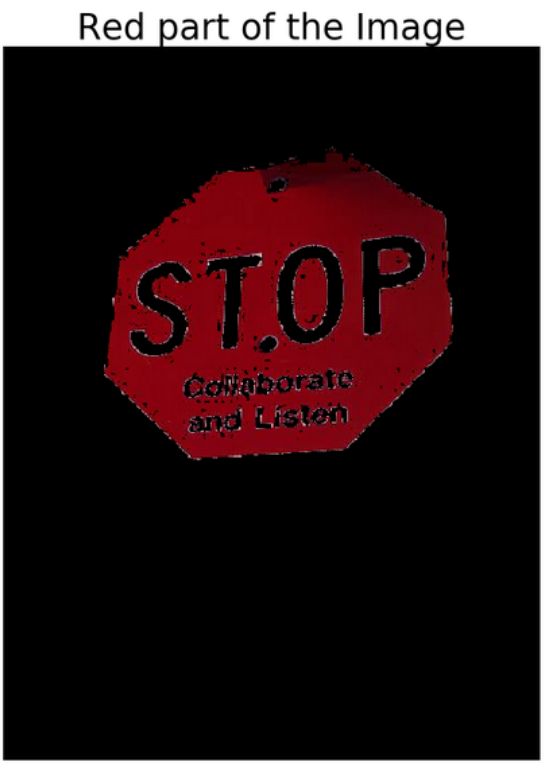
```
import cv2
import matplotlib.pyplot as plt
import numpy as np

image = cv2.imread('media/M3/stop.jpg')

# Converting the image to hsv
hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)

# define range of red color in HSV
lower_red = np.array([160,50,50])
upper_red = np.array([180,255,255])

# Threshold the HSV image using inRange function to get only red colors
mask = cv2.inRange(hsv, lower_red, upper_red)

plt.figure(figsize=[13,13])
plt.subplot(121);plt.imshow(image[:,:,::-1]);plt.title("Original Image",fontdict={'fonts
plt.subplot(122);plt.imshow(mask, cmap='gray');plt.title("Mask of red Color",fontdict={'
```

Output :




Now we can use this Mask and pass into the bitwise_and() function along with the original image and it will give the part of the image which has red color.

```
res = cv2.bitwise_and(image,image, mask= mask)
plt.figure(figsize=[13,13])
```




**Capture a Colored object in Real time** / Color Detection / Find Color object

```
import cv2
import matplotlib.pyplot as plt
import numpy as np

cap = cv2.VideoCapture(0)

while(1):

    # Take each frame
    _, frame = cap.read()

    # Convert BGR to HSV
    hsv = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

    # define range of red color in HSV
    lower_red = np.array([160,50,50])
    upper_red = np.array([180,255,255])

    #Threshold the HSV image to get only red colors
    mask = cv2.inRange(hsv, lower_red, upper_red)

    # Bitwise-AND mask and original image
    res = cv2.bitwise_and(frame,frame, mask= mask)

    # we are just adding 2 more channels on the mask so we can stack it along other imag
    mask_3 = cv2.cvtColor(mask, cv2.COLOR_GRAY2BGR)
```

```
34            break
35
36    cv2.destroyAllWindows()
37    cap.release()
```

This entry was posted in Uncategorized and tagged Color Detection, Color Filtering on April 28, 2020 [https://cvexplained.wordpress.com/2020/04/28/color-detection-hsv/] .