Assignment – 2

Problem Statement: Given the attached image, write a program to implement a scheme that will extract only the elephant from the image. The output image will only have the elephant (in colour) and a white background.

Step 1: - Converting image from RGB model to HSI model

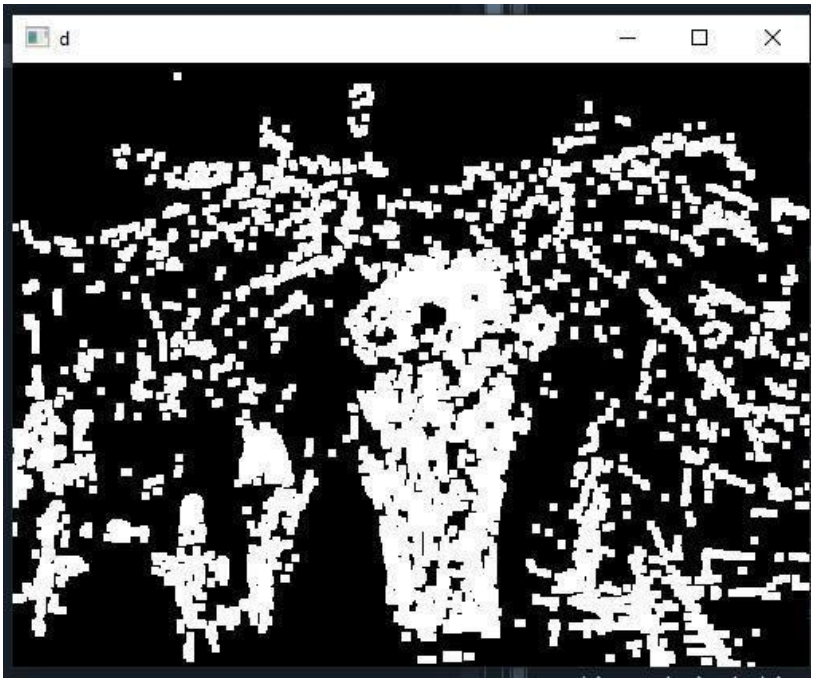

```
img = cv.imread("Assignment2.jpg")
showImage('rgb image',img)

hsv= cv.cvtColor(img,cv.COLOR_BGR2HSV)
showImage('hsvmage',hsv)

cv.imwrite('hsvImg.png', hsv)
```

Step 2: - Select pixels whose values match with the range given to us. Choosing a threshold value by analysing=

Low =[15,30,20] High=[25,70,180]

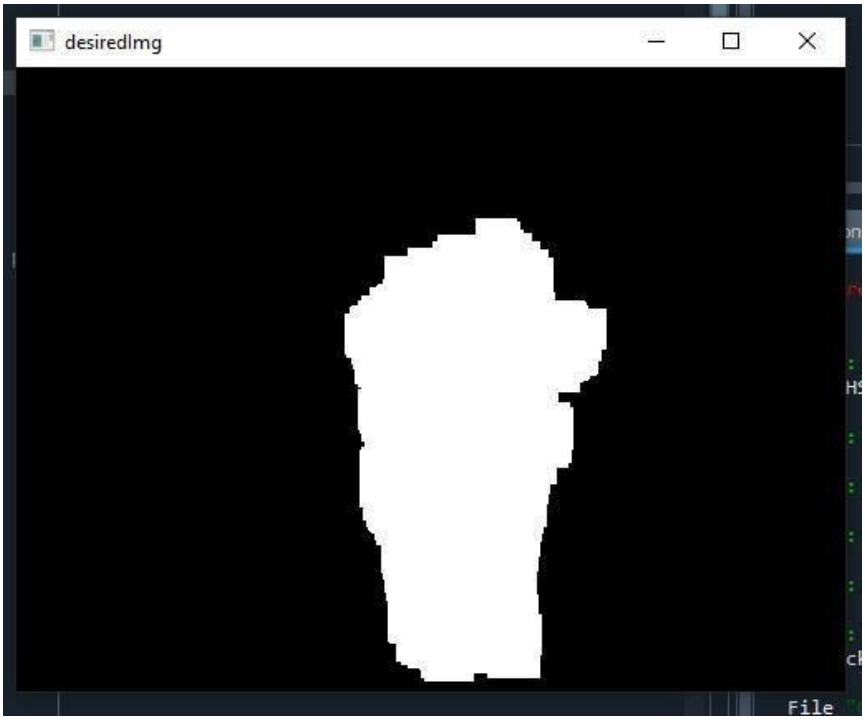
```
def threshold(hsv):
    low=np.array([15,30,20])
    high=np.array([25,70,180])

    x=cv.inRange(hsv,low,high)
    return x
```

Step 3: - Applying Dilation Morphological image processing



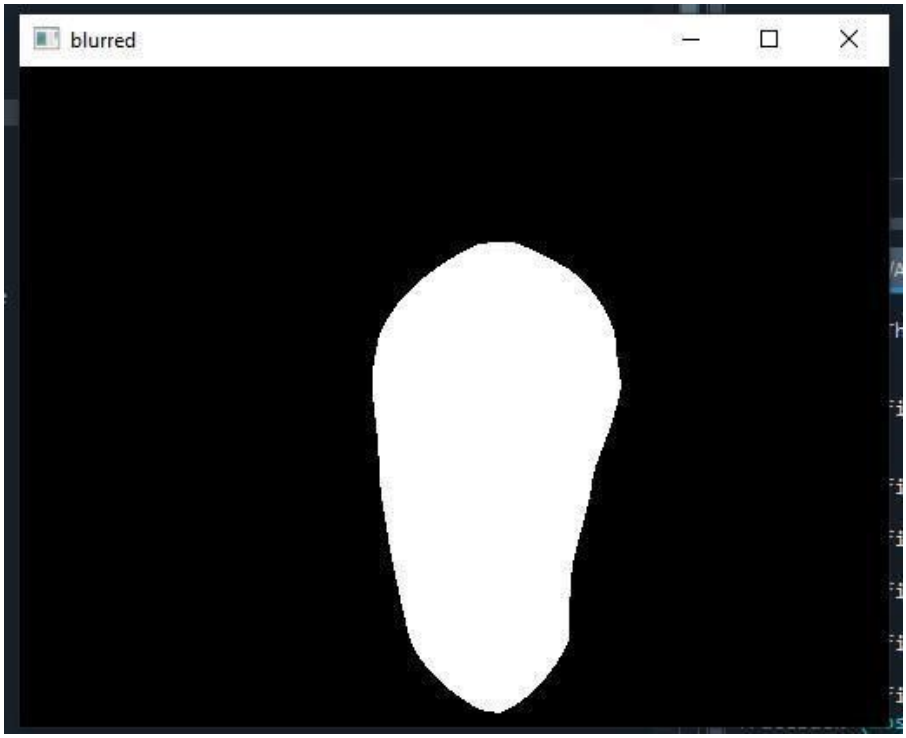
```
kernel5 = np.ones((5,5), np.uint8)
kernel4 = np.ones((4,4), np.uint8)
kernel3 = np.ones((3,3), np.uint8)
kernel2 = np.ones((2,2), np.uint8)
kernel1 = np.ones((1,1), np.uint8)

img_erosion = cv.erode(x, kernel2, iterations=1)
img_dilation = cv.dilate(x, kernel5, iterations=1)

showImage('d',img_dilation)

big =undesired_objects (img_dilation,img)

big = cv.dilate(big, kernel5, iterations=5)
showImage('desiredImg', big)
```

Step 4: - Find the largest connected components and extracting it give us the elephant only.



```
def overwriteOnimg(img,big):
    img_copy,temp,temp1=cv.split(img)
    height, width, depth = img.shape

    for i in range(0, height):
        for j in range(0, (width)):
            img_copy[i,j] = big[i,j]
    return img_copy

def undesired_objects (image,img):
    nbcomp, out, stats, centroids = cv.connectedComponentsWithStats(image, 4)
    sizes = stats[:, -1]

    max_label = 1
    max_size = sizes[1]
    for i in range(2, nbcomp):
        if sizes[i] > max_size:
            max_label = i
            max_size = sizes[i]
            img2 = np.zeros(out.shape)
    img2[out == max_label] = 255

    img2=overwriteOnimg(img, img2) #make img2 with help of img with only 1 plane
    return img2
```

Step 5: - Applying a median filter to the image.

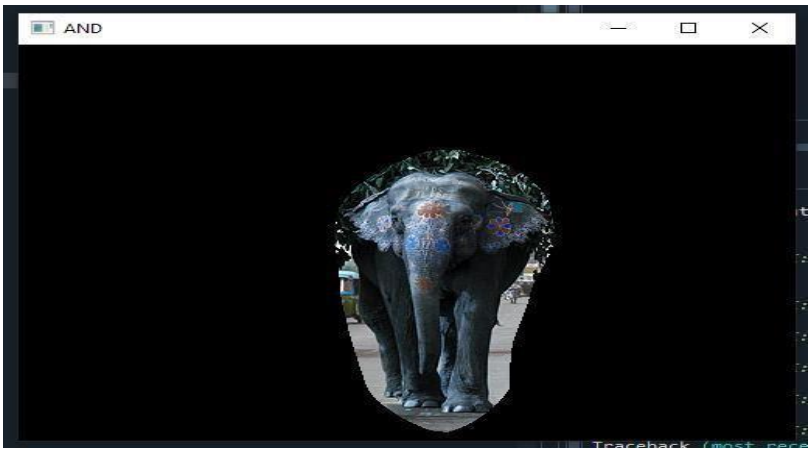


```
big = cv.dilate(big, kernel5, iterations=5)
showImage('desiredImg', big)

big = cv.medianBlur(big, 81)
showImage('blurred', big)
```

Made the median blur as maximum as possible and kernel of size 81 was the best fit.

Step 6: - Applying AND operation between image obtained above and the original image.

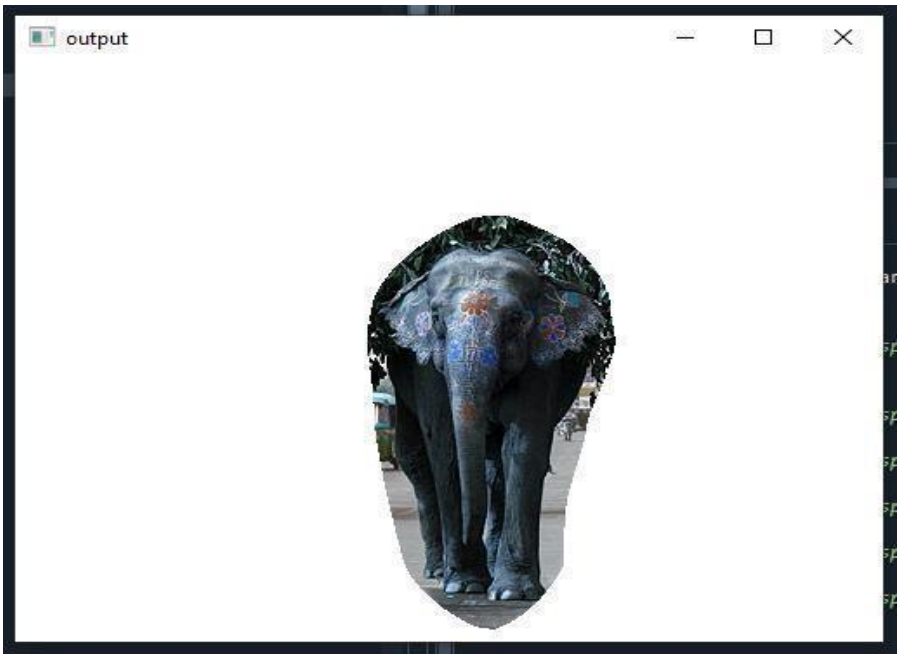
```
def bitwiseAndImages(img, big):

    b,g,r=cv.split(img)


    bitwiseAndB=b
    bitwiseAndG=g
    bitwiseAndR=r
    height, width, depth = img.shape
    for i in range(0, height):
        for j in range(0, (width)):
            bitwiseAndR[i,j] = big[i,j] and r[i,j]
            bitwiseAndG[i,j] = big[i,j] and g[i,j]
            bitwiseAndB[i,j] = big[i,j] and b[i,j]

    bitwiseAnd = cv.merge([bitwiseAndR,bitwiseAndG,bitwiseAndB])
    return bitwiseAnd
```



```
def whiteBlackgroung(bitwiseAnd,big):
    height, width, depth = bitwiseAnd.shape

    for i in range(0, height):
        for j in range(0, (width)):
            if(big[i,j]==0):
                bitwiseAnd[i,j,0]=255
                bitwiseAnd[i,j,1]=255
                bitwiseAnd[i,j,2]=255
    return bitwiseAnd
```

```
bitwiseAnd = bitwiseAndImages(img, big)
showImage("AND", bitwiseAnd)

output=whiteBlackgroung(bitwiseAnd,big)
showImage('output',output)
```

Final Output Image looks like this->



Functions used:

overwriteOnimg(img,big): makes big into 1 plane.
undesired_objects (image,img): extract the biggest cluster.
bitwiseAndImages(img, big): bitwise operation of 2 images.
threshold(hsv): extract the mask with a given threshold.

Prepared By -

21UCS027 Arin Khandelwal