

## DIGITAL IMAGE PROCESSING

This project is done under the supervision of “[Dr. Anukriti Bansal](#)”

### Project Aim :

Given the attached images, 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.



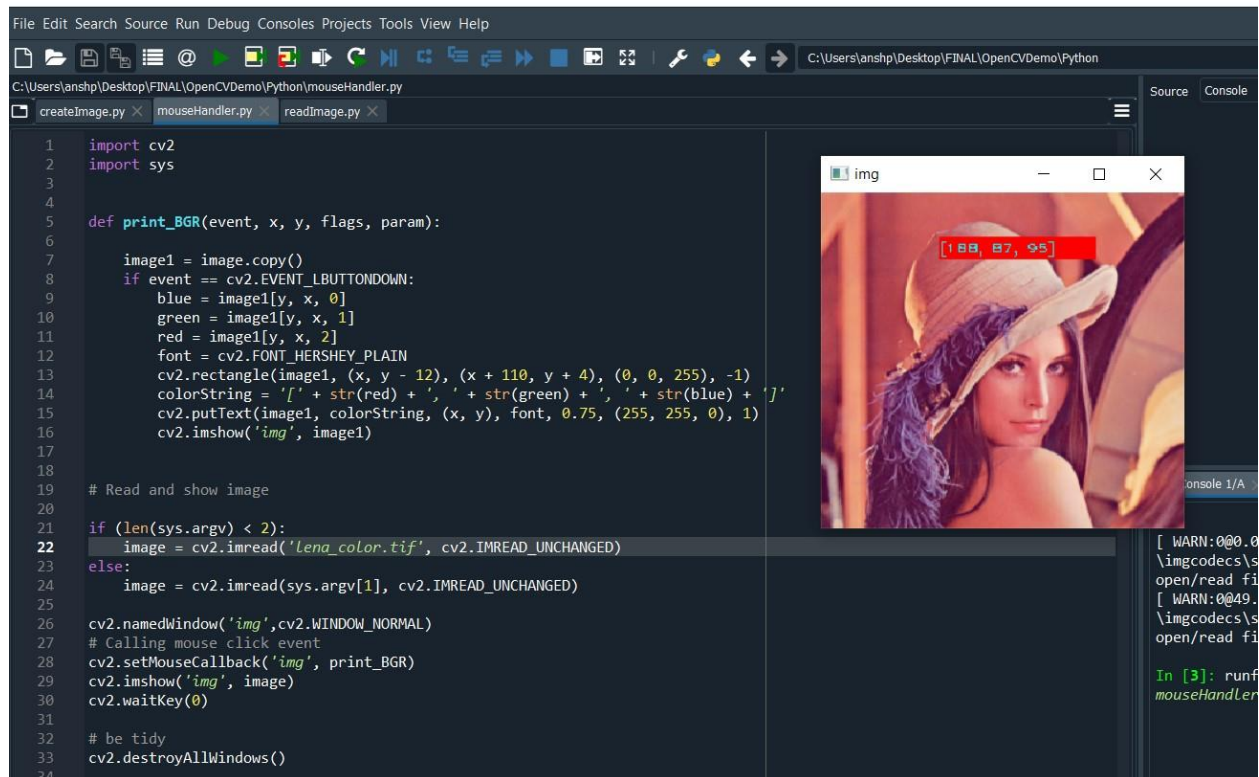
OUTPUT (after execution of code):



# Report:-

We have found used the code of our previous lab demonstration from classroom to find an approximate pixel value for the threshold.

Below is the snippet of the code and running project (openCV\_demo from the classroom)



```
1 import cv2
2 import sys
3
4
5 def print_BGR(event, x, y, flags, param):
6
7     image1 = image.copy()
8     if event == cv2.EVENT_LBUTTONDOWN:
9         blue = image1[y, x, 0]
10        green = image1[y, x, 1]
11        red = image1[y, x, 2]
12        font = cv2.FONT_HERSHEY_PLAIN
13        cv2.rectangle(image1, (x, y - 12), (x + 110, y + 4), (0, 0, 255), -1)
14        colorString = '[' + str(red) + ', ' + str(green) + ', ' + str(blue) + ']'
15        cv2.putText(image1, colorString, (x, y), font, 0.75, (255, 255, 0), 1)
16        cv2.imshow('img', image1)
17
18
19 # Read and show image
20
21 if (len(sys.argv) < 2):
22     image = cv2.imread('lena_color.tif', cv2.IMREAD_UNCHANGED)
23 else:
24     image = cv2.imread(sys.argv[1], cv2.IMREAD_UNCHANGED)
25
26 cv2.namedWindow('img', cv2.WINDOW_NORMAL)
27 # Calling mouse click event
28 cv2.setMouseCallback('img', print_BGR)
29 cv2.imshow('img', image)
30 cv2.waitKey(0)
31
32 # be tidy
33 cv2.destroyAllWindows()
34
```

The screenshot shows a Python IDE with the above code. A window titled 'img' displays a color image of a woman. A red rectangle is drawn on the image, and a text box displays the BGR pixel values: [188, 87, 95].

We then compare HSI values with respect to the threshold and only extract those values which lie within the threshold range.

Snippet:

```
### Function for thresholding
def create_mask(image):
    (height, width) = image.shape[:2]
    masked_image = np.zeros((height, width), np.uint8)
    for i in range(height):
        for j in range(width):
            if((0.45< image[i][j][0] and image[i][j][0] < 1) and (0.1 < image[i][j][1] and image[i][j][1] < 0.6) and (0.05 < image[i][j][2] and image[i][j][2] < 0.8)):
                masked_image[i][j] = 255
    # cv2.imshow("IMG_Thresh_", masked_image)
    cv2.imwrite("eleB.jpeg", masked_image)
```

This will give us our mask.



This mask is then dilated and eroded as per the desire, to reduce unwanted components which got set to one along with those of the ELEPHANT's area and then filter the mask.

## SNIPPET:

Below are the mask that are used for erosion and dilation

```
# smallMask of Dilation and Erosion
smallMask = np.ones((2, 2), np.uint8)
smallMask2= np.ones((3, 3), np.uint8)
smallMask3 = np.ones((1, 3), np.uint8)
```

After erosion :



**After Dilation:**



Erosion will help us to dilate all those values which are smaller than the size of the mask and then dilation will help us to obtain the image of the elephant back.

Then we find the largest connected area.

There are two ways to find the largest connected area. One by findingContour() function and the other by the function that is shown in the code.

## SNIPPET

```
def undesired_objects(image):
    image = image.astype('uint8')
    nb_components, output, stats, centroids = cv2.connectedComponentsWithStats(
        image, connectivity=8)
    sizes = stats[:, -1]
    max_label = 1
    max_size = sizes[1]
    for i in range(2, nb_components):
        if sizes[i] > max_size:
            max_label = i
            max_size = sizes[i]

    img2 = np.zeros(output.shape)
    img2[output == max_label] = 255
    # cv.imshow("Biggest component", img2)
    # cv.waitKey()
    return img2
```

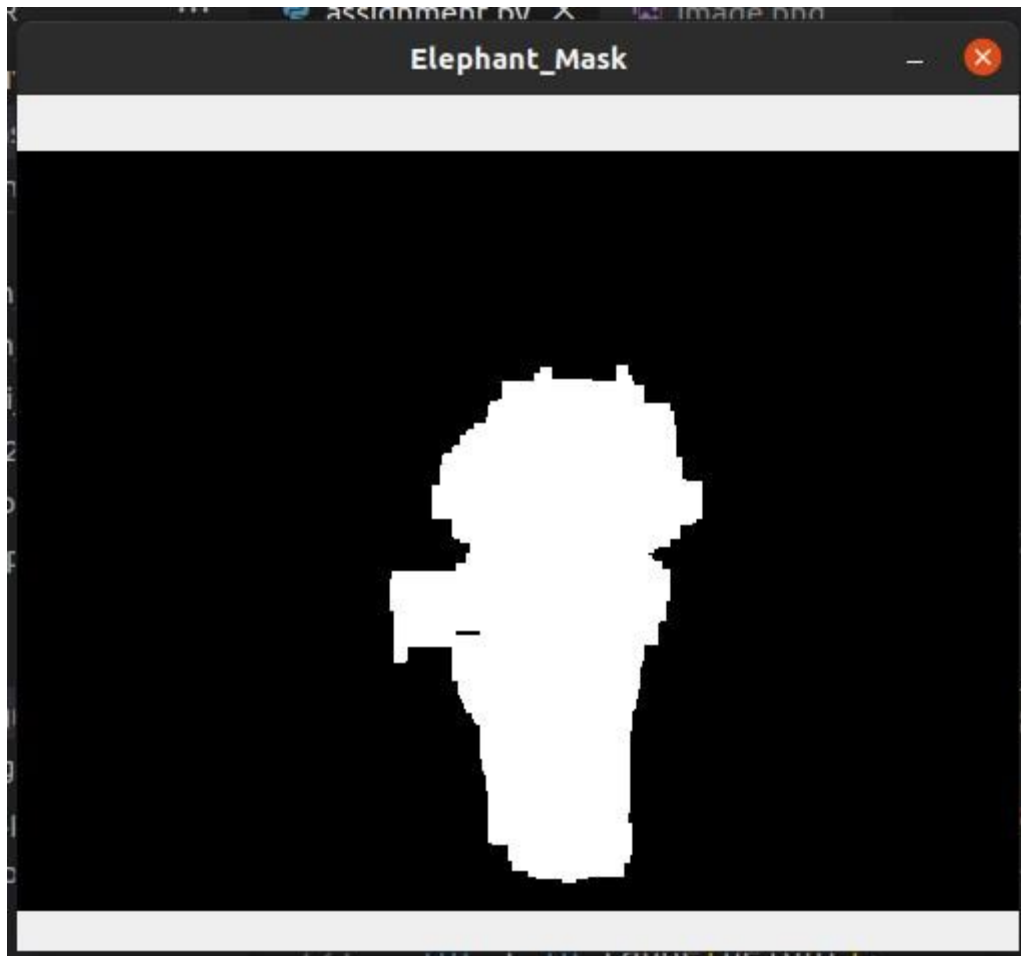
## Contoured Image (largest connected area):



We have used the second method: In this function, we first find the largest connected area which then returns the desired area.

Above Elephant\_mask is then eroded and dilated multiple times to get the desired image.

## OUR FINAL MASK



This updated mask is then used to extract the colored image using the method shown below. Bitwise\_and() method can also be used to extract the image.

Hence, we obtain the extracted image.



