COLOR DETECTION

To Perform Color Detection on Images

```
import numpy as np
import cv2
```

- We Have Imported Required Libraries like Numpy and cv2
- Numpy is used for Numerical Operations
- Cv2 is used for image processing

```
img = cv2.imread(r"red.jpg")
```

- This Reads an Image File using opencv imread function and stores in img

```
def empty(a):
    pass
```

- This is a placeholder function empty(a) that does nothing. It is used as a callback for trackbar events but has no actual functionality in this code.

```
cv2.namedWindow("TrackBars")
cv2.resizeWindow("TrackBars", 640, 240)
```

- Two lines create a named window called "TrackBars" and resize it to a width of 640 pixels and height of 240 pixels. This window will be used to display trackbars for adjusting color values.

```
cv2.createTrackbar("Hue Min", "TrackBars", 0, 179, empty)
cv2.createTrackbar("Hue Max", "TrackBars", 19, 179, empty)
cv2.createTrackbar("Saturation Min", "TrackBars", 110, 255, empty)
cv2.createTrackbar("Saturation Max", "TrackBars", 240, 255, empty)
cv2.createTrackbar("Value Min", "TrackBars", 153, 255, empty)
cv2.createTrackbar("Value Max", "TrackBars", 255, 255, empty)
```

- These lines create six trackbars in the "TrackBars" window. Each trackbar corresponds to a color channel (Hue, Saturation, Value) and its minimum and maximum values. The cv2.createTrackbar() function takes the following arguments:
- Trackbar name: The name of the trackbar displayed next to it.
- Window name: The name of the window to which the trackbar belongs.
- Default value: The initial position of the trackbar slider.
- Maximum value: The maximum value of the trackbar.
- Callback function: The function that will be called when the trackbar value changes. In this case, it uses the empty(a) function as a placeholder, which does nothing.

```
# Flag to check if the window is
created
window_created = False
```

- This line initializes the window_created variable to False. It is used to check if the window is created so that it only displays the original image and HSV image once.

```
while True:
    img = cv2.imread(r"red.jpg")
    imgHSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)

    # Check if the window is created
    if not window_created:
        cv2.imshow("Original Image", img)
        cv2.imshow('HSV IMAGE', imgHSV)
        window_created = True
```

- In the while loop, the code reads the "red.jpg" image and converts it to the HSV color space using cv2.cvtColor().
- It then checks if the window_created flag is False. If it is, it displays the original image and the HSV image once and sets the flag to True, so the images are not displayed again in subsequent iterations

```
    hmin = cv2.getTrackbarPos("Hue Min", "TrackBars")
    smin = cv2.getTrackbarPos("Saturation Min", "TrackBars")
    vmin = cv2.getTrackbarPos("Value Min", "TrackBars")
    hmax = cv2.getTrackbarPos("Hue Max", "TrackBars")
    smax = cv2.getTrackbarPos("Saturation Max", "TrackBars")
    vmax = cv2.getTrackbarPos("Value Max", "TrackBars")
    lower = np.array([hmin, smin, vmin])
    upper = np.array([hmax, smax, vmax])
```

- These lines get the current positions of the trackbars using cv2.getTrackbarPos(), and store them in variables (hmin, smin, vmin, hmax, smax, vmax).
- These trackbar values will define the lower and upper HSV range for color filtering.

```
    # Create mask
    mask = cv2.inRange(imgHSV, lower, upper)
    imgR = cv2.bitwise_and(img, img, mask=mask)

    cv2.imshow('Mask', mask)
    cv2.imshow("Result", imgR)
```

- The code then creates a mask using cv2.inRange() based on the selected HSV color range.
- The mask filters out the desired color, and the filtered result is obtained by performing a bitwise AND operation between the original image and the mask using cv2.bitwise_and().
- The filtered image is displayed in the "Mask" window, and the final result is displayed in the "Result" window.

```
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break
cv2.destroyAllWindows()
```

- Finally, all OpenCV windows are closed when the loop exits, ending the program.