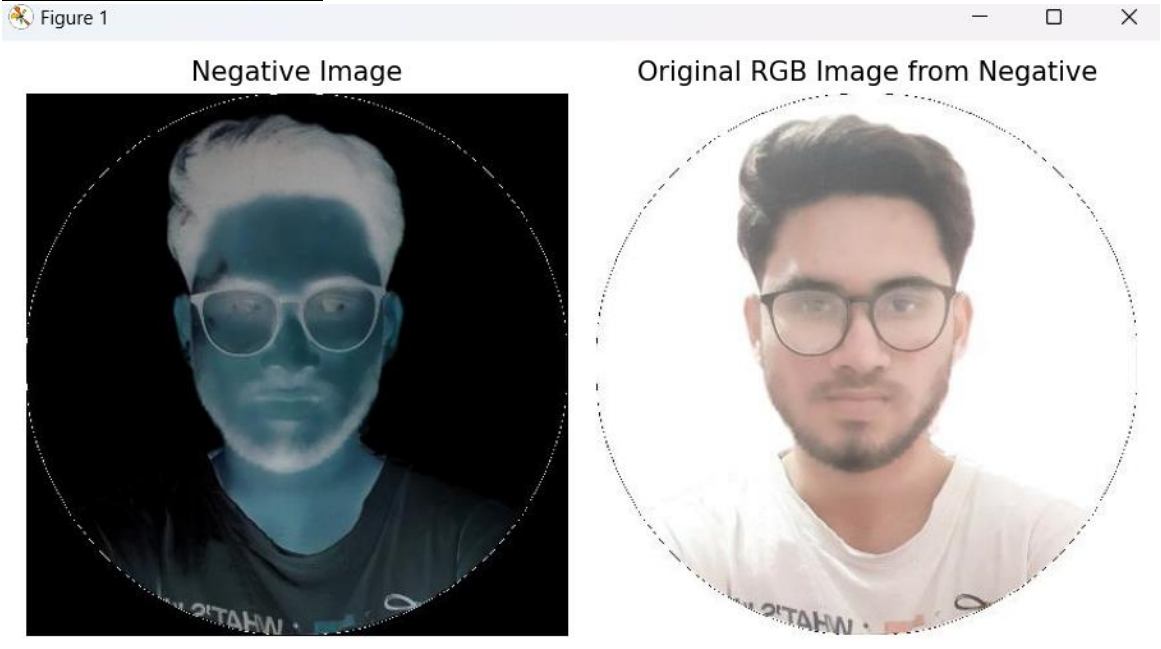
Name: Harshit Kumar
Registration number: 21ETMC412011
Subject: Computer Vision (Implementation)
Topic: Color models= RGB, CMYK, HSV, CIE

[1.] Program: RGB

RGB.py > ...

```python
import cv2                                          # Import the OpenCV library for image processing
import numpy as np                                  # Import NumPy for numerical operations
import matplotlib.pyplot as plt                     # Import Matplotlib for plotting images

negative_image = cv2.imread('negative_image.jpg')       # Load the negative image from a file
negative_image_rgb = cv2.cvtColor(negative_image, cv2.COLOR_BGR2RGB)  # Convert the image from BGR to RGB color space
original_image = 255 - negative_image_rgb               # Invert the colors to get the original image

plt.figure(figsize=(6, 5))                          # Create a figure to display images

plt.subplot(1, 2, 1)                                # Create a subplot for the negative image
plt.imshow(negative_image_rgb)                      # Show the negative image in the subplot
plt.title('Negative Image')                         # Add a title to the negative image
plt.axis('off')                                     # Turn off the axis for the negative image

plt.subplot(1, 2, 2)                                # Create a subplot for the original image
plt.imshow(original_image)                          # Show the original image in the subplot
plt.title('Original RGB Image from Negative')       # Add a title to the original image
plt.axis('off')                                     # Turn off the axis for the original image

plt.tight_layout()                                  # Adjust the layout so images do not overlap
plt.show()                                          # Display the figure with the images
```

Input and Output:

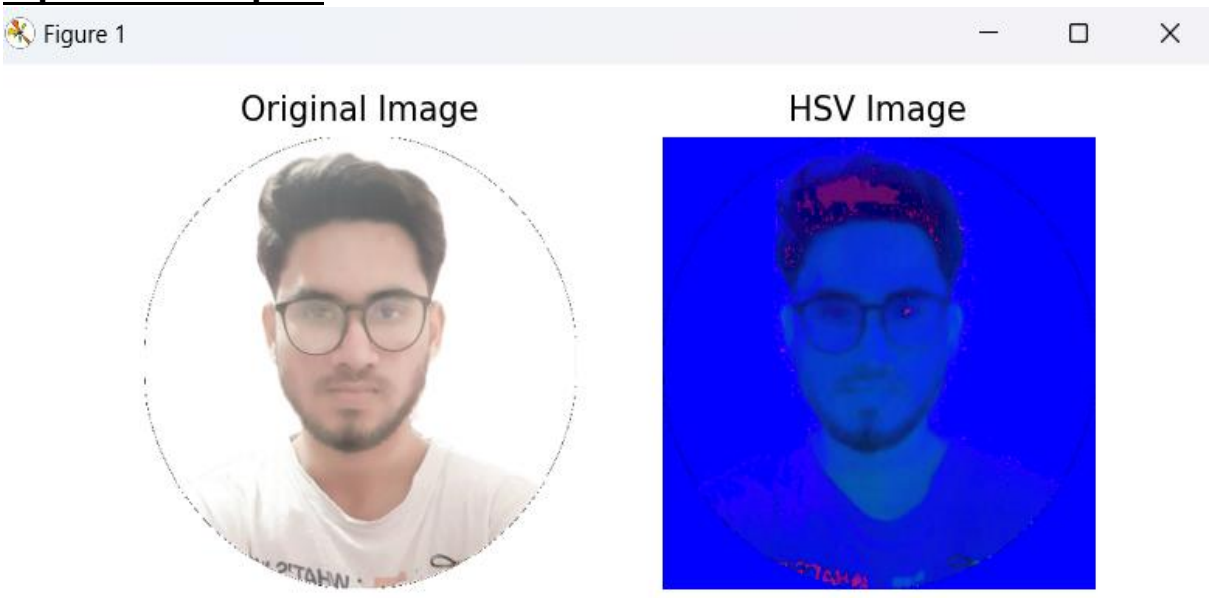


[2.] Program: HSV

HSV.py > ...

```python
import cv2                                          # Import OpenCV for image processing
import matplotlib.pyplot as plt                     # Import Matplotlib for plotting images

# Load the image
image = cv2.imread('20231019_183114.jpg')           # Read the image from a file

# Convert the image from BGR (OpenCV default) to HSV
image_hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)  # Change the image color from BGR to HSV

# Display the original and HSV images side by side
plt.figure(figsize=(10, 5))                         # Create a figure to hold the images

# Display the original image (converted to RGB for correct display)
plt.subplot(1, 2, 1)                                # Create a subplot for the original image
image_rgb = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)  # Change the image color from BGR to RGB
plt.imshow(image_rgb)                               # Show the original image
plt.title('Original Image')                         # Add a title to the original image
plt.axis('off')                                     # Turn off the axis for the original image

# Display the HSV image
plt.subplot(1, 2, 2)                                # Create a subplot for the HSV image
plt.imshow(image_hsv)                               # Show the HSV image
plt.title('HSV Image')                              # Add a title to the HSV image
plt.axis('off')                                     # Turn off the axis for the HSV image

plt.show()                                          # Show the figure with both images

```

Input and Output:



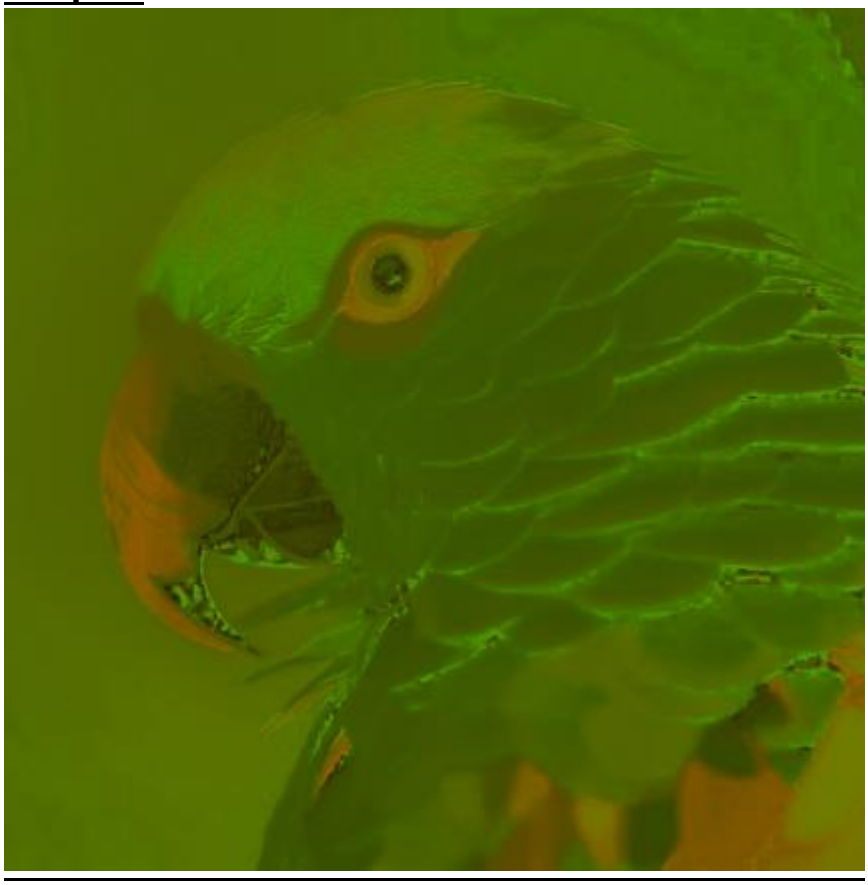
[3.] Program: CIE

CIE-Chromaticity.py > main

```python
import cv2                                          # Import OpenCV for image processing
import numpy as np                                  # Import NumPy for numerical operations

def rgb_to_xyz(image):
    # Conversion matrix from RGB to XYZ
    M = np.array([[0.412453, 0.357580, 0.180423],   # Define the conversion matrix from RGB to XYZ
                  [0.212671, 0.715160, 0.072169],
                  [0.019334, 0.119193, 0.950227]])

    # Normalize the RGB values to [0, 1]
    image = image / 255.0                           # Scale the RGB values to be between 0 and 1

    # Convert the image to XYZ
    XYZ_image = cv2.transform(image, M)             # Apply the conversion matrix to the image to get XYZ values

    return XYZ_image                                # Return the converted XYZ image

def xyz_to_chromaticity(XYZ_image):
    X = XYZ_image[:, :, 0]                          # Extract the X channel from the XYZ image
    Y = XYZ_image[:, :, 1]                          # Extract the Y channel from the XYZ image
    Z = XYZ_image[:, :, 2]                          # Extract the Z channel from the XYZ image

    sum_XYZ = X + Y + Z                             # Calculate the sum of X, Y, and Z values
    sum_XYZ[sum_XYZ == 0] = 1                       # Avoid division by zero by setting zero sums to one

    x = X / sum_XYZ                                 # Calculate the chromaticity x coordinate
    y = Y / sum_XYZ                                 # Calculate the chromaticity y coordinate

    chromaticity_image = np.zeros_like(XYZ_image)   # Create an empty image to hold the chromaticity values
    chromaticity_image[:, :, 0] = x                 # Set the x values in the new image
    chromaticity_image[:, :, 1] = y                 # Set the y values in the new image
    chromaticity_image[:, :, 2] = 0                 # Set the third channel to zero (not used)

    return chromaticity_image                       # Return the chromaticity image

def main():
    # Load the original image
    image = cv2.imread('WhatsApp Image 2024-07-15 at 16.46.34.jpeg')  # Read the image from a file

    # Convert the image from BGR to RGB
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)      # Change the image color from BGR to RGB

    # Convert RGB image to XYZ
    XYZ_image = rgb_to_xyz(image)                       # Convert the RGB image to XYZ

    # Convert XYZ image to CIE Chromaticity
    chromaticity_image = xyz_to_chromaticity(XYZ_image) # Convert the XYZ image to chromaticity coordinates

    # Scale the chromaticity coordinates to [0, 255] for display
    chromaticity_image = (chromaticity_image * 255).astype(np.uint8)  # Scale the chromaticity image values to 0-255

    # Save the chromaticity image
    chromaticity_image_bgr = cv2.cvtColor(chromaticity_image, cv2.COLOR_RGB2BGR)  # Convert back to BGR for saving
    cv2.imwrite('chromaticity_image.jpg', chromaticity_image_bgr)  # Save the chromaticity image to a file

if __name__ == "__main__":                              # If this script is run directly (not imported)
    main()                                              # Run the main function
```

Input:

Output:



[4.] Program: CMYK

CMYK.py > ...

```python
from PIL import Image                    # Import the Python Imaging Library (PIL) for image handling

# Open an image file
img = Image.open("20231019_183114.jpg") # Load the image from a file

# Convert the image to CMYK
cmyk_img = img.convert("CMYK")          # Change the image color mode to CMYK

# Save the CMYK image
cmyk_img.save("cmyk_image.jpg")         # Save the new CMYK image to a file

# Show the CMYK image
cmyk_img.show()                         # Display the CMYK image

```

Input:

Output: