Homework 1 : Color Transform

Name: 周睦鈞

Student ID: 311553060

Spec:

- Please represent “lena.png” in terms of RGB, YUV, and YCbCr.

1. RGB -> YUV:
   $$Y = 0.299 * R + 0.587 * G + 0.114 * B$$
   $$U = -0.169 * R - 0.331 * G + 0.5 * B + 128$$
   $$V = 0.5 * R - 0.419 * G - 0.081 * B + 128$$
2. RGB -> YCbCr: in the slides

- In any programming language you are comfortable with (C/C++/Python/MATLAB).
- Output 8 grayscale images representing R, G, B, Y, U, V, Cb, and Cr, respectively.
- **Do not** use any ready-made functions to transform the color.
- You are allowed to use image reading/writing APIs.
- Deadline: 2024/09/30 13:19.
- Compressed as a single ZIP file.
- Required files :

1. **VC_HW1_[student_id].pdf**: Report PDF
2. **VC_HW1_[student_id].zip**: Source code (C/C++/Python/MATLAB) with a **README** file instructing the TAs on how to run your code.

Programming Description:

1. Main block

Obtain the input image and pass it to the function `convert_rgb_to_yuv_ycbcr`.

Then, extract the 8 grayscale output images corresponding to R, G, B, Y, U, V, Cb, and Cr, respectively, and save them to the output folder.

```python
if __name__ == '__main__':
    # Path to the uploaded image
    image_path = 'lena.png'  # Replace this with your local image path

    # Call the function to convert and generate images
    generated_images = convert_rgb_to_yuv_ycbcr(image_path)

    # Output paths of generated images
    for channel, path in generated_images.items():
        print(f"{channel} channel image saved at: {path}")
```

2. `convert_rgb_to_yuv_ycbcr` function

Load the image using the PIL API and convert it to RGB format.

Next, generate the grayscale images for the R, G, and B channels.

Then, using the YUV and YCbCr formulas, derive the grayscale images for Y, U, V, Cb, and Cr.

```python
    # Load the image
    lena_image = Image.open(image_path)

    # Convert the image to RGB format
    lena_rgb = np.array(lena_image.convert('RGB'))

    # Separate R, G, B channels
    R = lena_rgb[:, :, 0].astype(float)
    G = lena_rgb[:, :, 1].astype(float)
    B = lena_rgb[:, :, 2].astype(float)

    # Calculate Y, U, and V using the provided formula
    Y = 0.299 * R + 0.587 * G + 0.114 * B
    U = -0.169 * R - 0.331 * G + 0.5 * B + 128
    V = 0.5 * R - 0.419 * G - 0.081 * B + 128

    # Calculate Cb and Cr using the provided formula
    Cb = 128 - 0.168736 * R - 0.331264 * G + 0.5 * B
    Cr = 128 + 0.5 * R - 0.418688 * G - 0.081312 * B
```

Use a dictionary to store the 8 grayscale output images.

Then, save these images to the output folder and return their corresponding paths

```
    images = {
        'R': Image.fromarray(R.astype(np.uint8), 'L'),
        'G': Image.fromarray(G.astype(np.uint8), 'L'),
        'B': Image.fromarray(B.astype(np.uint8), 'L'),
        'Y': Image.fromarray(Y.astype(np.uint8), 'L'),
        'U': Image.fromarray(U.astype(np.uint8), 'L'),
        'V': Image.fromarray(V.astype(np.uint8), 'L'),
        'Cb': Image.fromarray(Cb.astype(np.uint8), 'L'),
        'Cr': Image.fromarray(Cr.astype(np.uint8), 'L')
    }

    # Create output directory if it doesn't exist
    output_dir = 'output'
    if not os.path.exists(output_dir):
        os.makedirs(output_dir)

    # Save the images and return their paths
    image_paths = {}
    for key, img in images.items():
        output_path = os.path.join(output_dir, f'lena_{key}.png')
        img.save(output_path)
        image_paths[key] = output_path

    return image_paths
```

3. Output

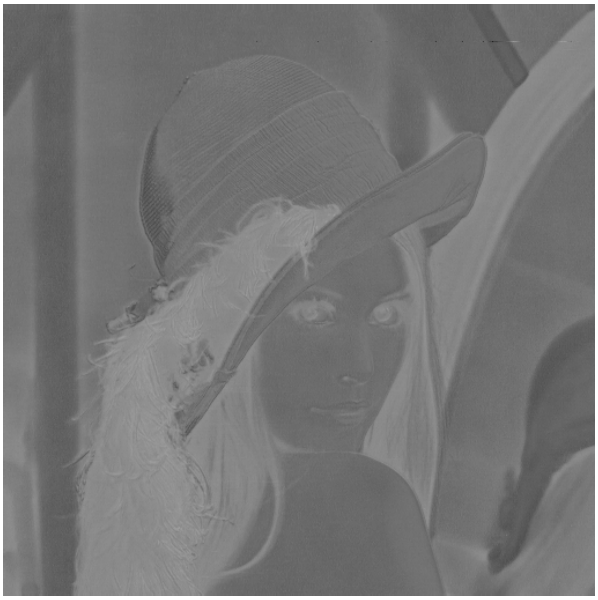
R:

G:

B:

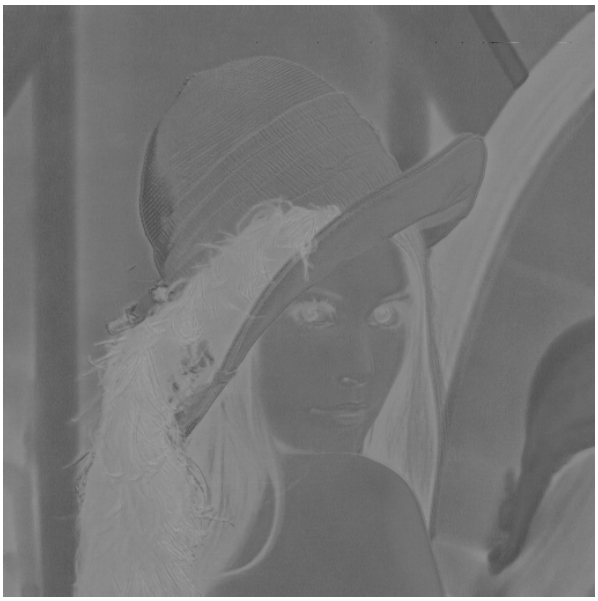
Y:

U:



V:

Cb:



Cr: