BÁO CÁO PROJECT 2

1. Thông tin cá nhân

- Họ và tên: Phan Trí Tài
- MSSV: 20127318

2. Ý tưởng thực hiện, mô tả các hàm

a) Ý tưởng

- Tìm hiểu các thư viện `NumPy` (tính toán ma trận), `PIL` (đọc, ghi ảnh), `matplotlib` (hiển thị ảnh).
- Tìm nguồn trên mạng hoặc trong sách.
- Nếu còn thắc mắc thì nên hỏi GVBM để hiểu rõ hơn về đồ án.

b) Mô tả hàm

- Import thư viện

```
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
```

- Hàm đọc ảnh

```
input_file = 'women.png'
image = Image.open(input_file)
image = np.array(image)

input_file_2 = 'women.png'
image_2 = Image.open(input_file_2)
image_2 = np.array(image_2)
```

1. Hàm thay đổi độ sáng

```
def adjust_brightness(img, brightness):
    return np.uint8(np.clip(img + np.array([brightness], dtype=np.int16), 0, 255))
```

- Hàm test (thay đổi độ sáng)

```
# Brightness: -255 tới 255
brightness = 123
result = adjust_brightness(image, brightness)
```

```
# Show ảnh trên notebook
plt.imshow(result)
```

```
# Xuất ra file
output_file = input_file.split('.')[0] + '_brightness' + '.png'
Image.fromarray(result).save(output_file)
```

2. Hàm thay đổi độ tương phản

```
def adjust_contrast(img, contrast):
    contrast = np.clip(float(contrast), -255, 255)
    factor = (259 * (contrast + 255)) / (255 * (259 - contrast))
    return np.uint8(np.clip(factor * (img.astype(float) - 128) + 128, 0, 255))
```

- Hàm test (thay đổi độ tương phản)

```
# Contrast: -255 tới 255
contrast = 123
result = adjust_contrast(image, contrast)
```

```
# Show ảnh trên notebook
plt.imshow(result)
```

```
# Xuất ra file
output_file = input_file.split('.')[0] + '_contrast' + '.png'
Image.fromarray(result).save(output_file)
```

3. Hàm lật ảnh ngang dọc

```
def flip(img, direction):
    return np.flipud(img) if direction == 'horizontal' else np.fliplr(img)
```

- Hàm test (lật ảnh ngang dọc)

```
# Direction: 'vertical' hoặc 'horizontal'
direction = 'horizontal'
result = flip(image, direction)
```

```
# Show ảnh trên notebook
plt.imshow(result)
```

```
# Xuất ra file
output_file = input_file.split('.')[0] + '_flip' + '.png'
Image.fromarray(result).save(output_file)
```

4. Hàm chuyển đổi ảnh RGB thành ảnh xám

```
def to_grayscale(img, weight):
    return np.uint8(np.dot(img[..., :3], weight))
```

- Hàm test (chuyển đổi ảnh RGB thành ảnh xám)

```
# Weight: Dưới đây là công thức luma của các ảnh thuộc format CCIR 601
weight = [0.299 , 0.587, 0.114]
result = to_grayscale(image, weight)
```

```
# Show ảnh trên notebook
    # cmap='gray': để show ảnh với 3 kênh màu khi ảnh xám chỉ có duy nhất 1 kênh
plt.imshow(result, cmap='gray')
```

```
# Xuất ra file
output_file = input_file.split('.')[0] + '_grayscale' + '.png'
Image.fromarray(result).save(output_file, cmap='gray')
```

5. Hàm chồng hai ảnh cùng kích thước

```
def blend(img_1, img_2, alpha):
    return np.uint8(alpha * img_1.astype(float) + (1 - alpha) * img_2.astype(float))
```

```
# Chuyển input thành ảnh xám, tuy nhiên vẫn có thể thực hiện được trên ảnh màu
gray_image_1 = to_grayscale(image, weight)
gray_image_2 = flip(to_grayscale(image_2, weight), 'horizontal')
```

- Hàm test (chồng hai ảnh cùng kích thước)

```
# Alpha: 0.0 tới 1.0
alpha = 0.7
result = blend(gray_image_1, gray_image_2, alpha)
```

```
# Show ảnh trên notebook
plt.imshow(result, cmap='gray')
```

```
# Xuất ra file
output_file = input_file.split('.')[0] + '_blend' + '.png'
Image.fromarray(result).save(output_file, cmap='gray')
```

6. Hàm mờ ảnh

```python
def Gaussian_func(x, sigma):
    return np.array(1 / (np.sqrt(2 * np.pi) * sigma) * (np.exp(-np.power(x / sigma, 2) / 2)))
```

```python
def calc_Gaussian_kernel(kernel_size, sigma):
    kernel_1d = np.linspace(-(kernel_size // 2), kernel_size // 2, num=kernel_size)
    kernel_1d = Gaussian_func(kernel_1d, sigma)
    kernel_2d = np.outer(kernel_1d.T, kernel_1d.T)
    kernel_2d *= 1.0 / np.sum(kernel_2d)
    return kernel_2d
```

```python
def convolve_layer(layer, kernel):
    view = kernel.shape + tuple(np.subtract(layer.shape, kernel.shape) + 1)
    submatrices = np.lib.stride_tricks.as_strided(layer, shape = view, strides = layer.strides * 2)
    return np.einsum('ij,ijkl->kl', kernel, submatrices)
```

```python
def convolution(img, kernel):
    return np.dstack((convolve_layer(img[:,:,0], kernel), convolve_layer(img[:,:,1], kernel), convolve_layer(img[:,:,2], kernel)))
```

```python
def Gaussian_blur(img, kernel_size):
    kernel = calc_Gaussian_kernel(kernel_size, sigma=(kernel_size-1)/6)
    return np.uint8(convolution(img, kernel))
```

- Hàm test (mờ ảnh)

```python
# Kernel size: từ 1 đến dương vô cùng, phải là số lẻ
kernel_size = 50
result = Gaussian_blur(image, kernel_size)
```

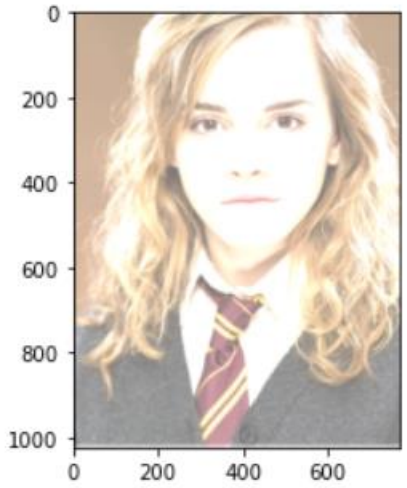
```python
# Show ảnh trên notebook
plt.imshow(result)
```

```python
# Xuất ra file
output_file = input_file.split('.')[0] + '_blur' + '.png'
Image.fromarray(result).save(output_file)
```

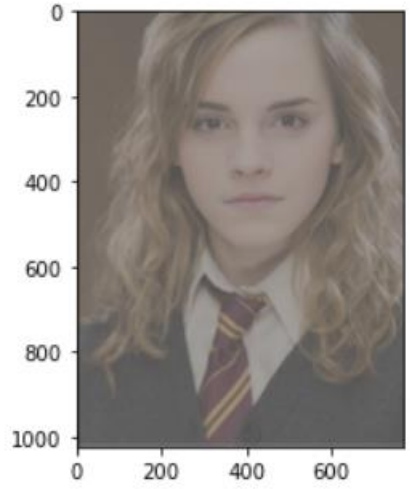
7. Hàm main (chưa làm được)

3. Hình ảnh kết quả

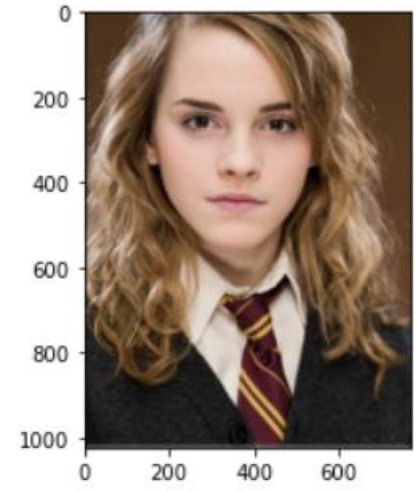
- Thay đổi độ sáng



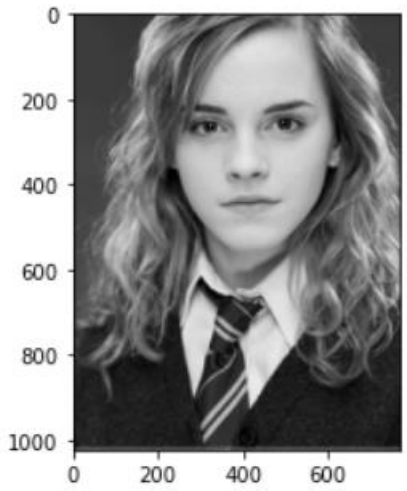
- Thay đổi độ tương phản



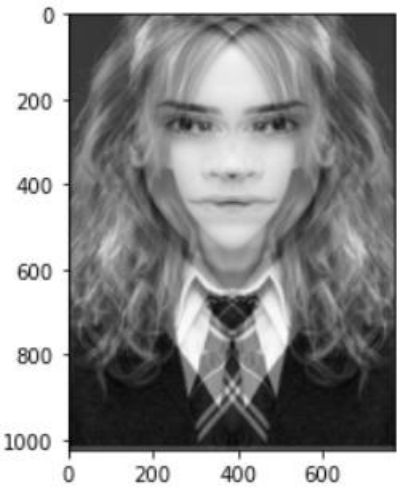
- Lật ảnh



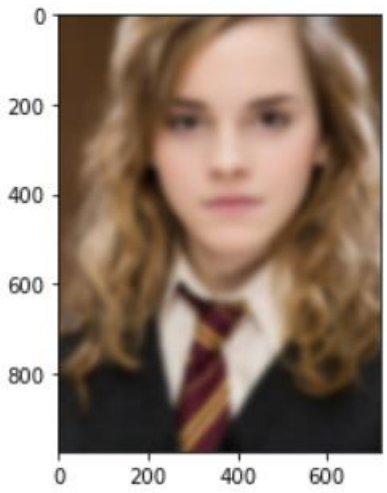
- Chuyển đổi ảnh RGB sang ảnh xám



- Chồng 2 ảnh cùng kích thước



- Làm mờ ảnh



4. Nhận xét

- Bài làm chỉ hoàn thành được 50-60% vì còn nhiều thiếu sót và vẫn chưa hoàn toàn hiểu được hết project.

5. Tài liệu tham khảo

- https://en.wikipedia.org/wiki/Kernel_(image_processing)
- https://www.tutorialspoint.com/dip/grayscale_to_rgb_conversion.htm
- https://tek4.vn/khoa-hoc/python-machine-learning/su-dung-jupyter-notebook-cho-python
- https://vi.wikipedia.org/wiki/YCbCr