

Họ và tên: Phạm Trọng Sơn - MSSV: 2052688
Môn: Xử lý Ảnh số và Thị giác Máy tính - MSMH: CO3057

REPORT Project1

1. Tách ảnh xám từ ảnh màu
Sử dụng thư viện opencv để đọc và xử lý ảnh.



Ảnh input

Ảnh có kích thước (612, 612, 3), dài rộng 612 với 3 channels biểu diễn 3 màu Đỏ Lục Lam

```
print(img.shape)
(612, 612, 3)
```

Sử dụng hàm cvtColor với *arg là COLOR_BGR2GRAY
Dựa trên công thức toán học sau:

RGB ↔ GRAY

Transformations within RGB space like adding/removing the alpha channel, reversing the channel order, conversion to/from 16-bit RGB color (R5:G6:B5 or R5:G5:B5), as well as conversion to/from grayscale using:

$$\text{RGB[A] to Gray: } Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

and

$$\text{Gray to RGB[A]: } R \leftarrow Y, G \leftarrow Y, B \leftarrow Y, A \leftarrow \max(\text{ChannelRange})$$

The conversion from a RGB image to gray is done with:

```
cvtColor(src, dst, cv::COLOR_BGR2GRAY);
```

```
rbg_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
```

- Image shape

```
[50] print(rbg_gray.shape)
```

```
(612, 612)
```

Ảnh xám có kích thước bằng với ảnh gốc như chỉ còn 1 channel



Ảnh xám của ảnh raw

Thử convert qua ảnh xám bằng cách chia đều sự phân bố của 3 màu, tức ta nhân nó với matrix $[\frac{1}{3}, \frac{1}{3}, \frac{1}{3}]$

```
2. Grayscale using RGB average
```

```
[52] img_float = img.astype(np.float32)
      grayscale_img = np.dot(img_float, [1/3, 1/3, 1/3])
      grayscale_img = np.clip(grayscale_img, 0, 255).astype(np.uint8)
```

```
[53] print(grayscale_img.shape)
```

```
(612, 612)
```

Output:



So sánh 3 ảnh:



Raw

COLOR_BGR2GRAY

Average RGB

Theo như OpenCV, công thức grayscale họ sử dụng liên quan đến độ nhạy cảm của mắt đối với các màu sắc. Ta có thể sự khác biệt rõ hơn nếu sử dụng cách chuyển đổi của opencv

2. Tách ảnh màu từ ảnh xám

Với ảnh xám khi ta load bằng định dạng ảnh xám của opencv.

```
[34] img_gray = cv2.imdecode(img_array, cv2.IMREAD_GRAYSCALE)
```

```
[35] print(img_gray.shape)
```

```
(612, 612)
```

```
cv2.imshow(img_gray)
```



Convert ngược lại thành ảnh 3 channels

```
gray_rgb = cv2.cvtColor(img_gray, cv2.COLOR_GRAY2BGR)
print(gray_rgb.shape)
cv2.imshow(gray_rgb)
```

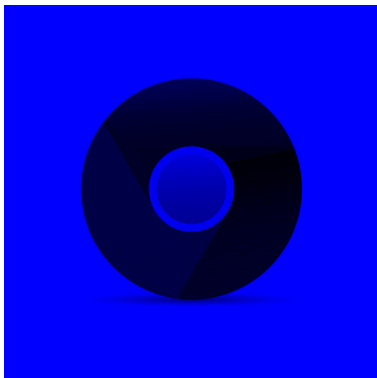
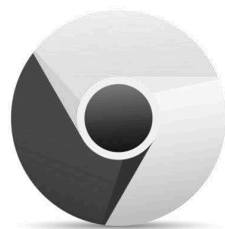
```
(612, 612, 3)
```



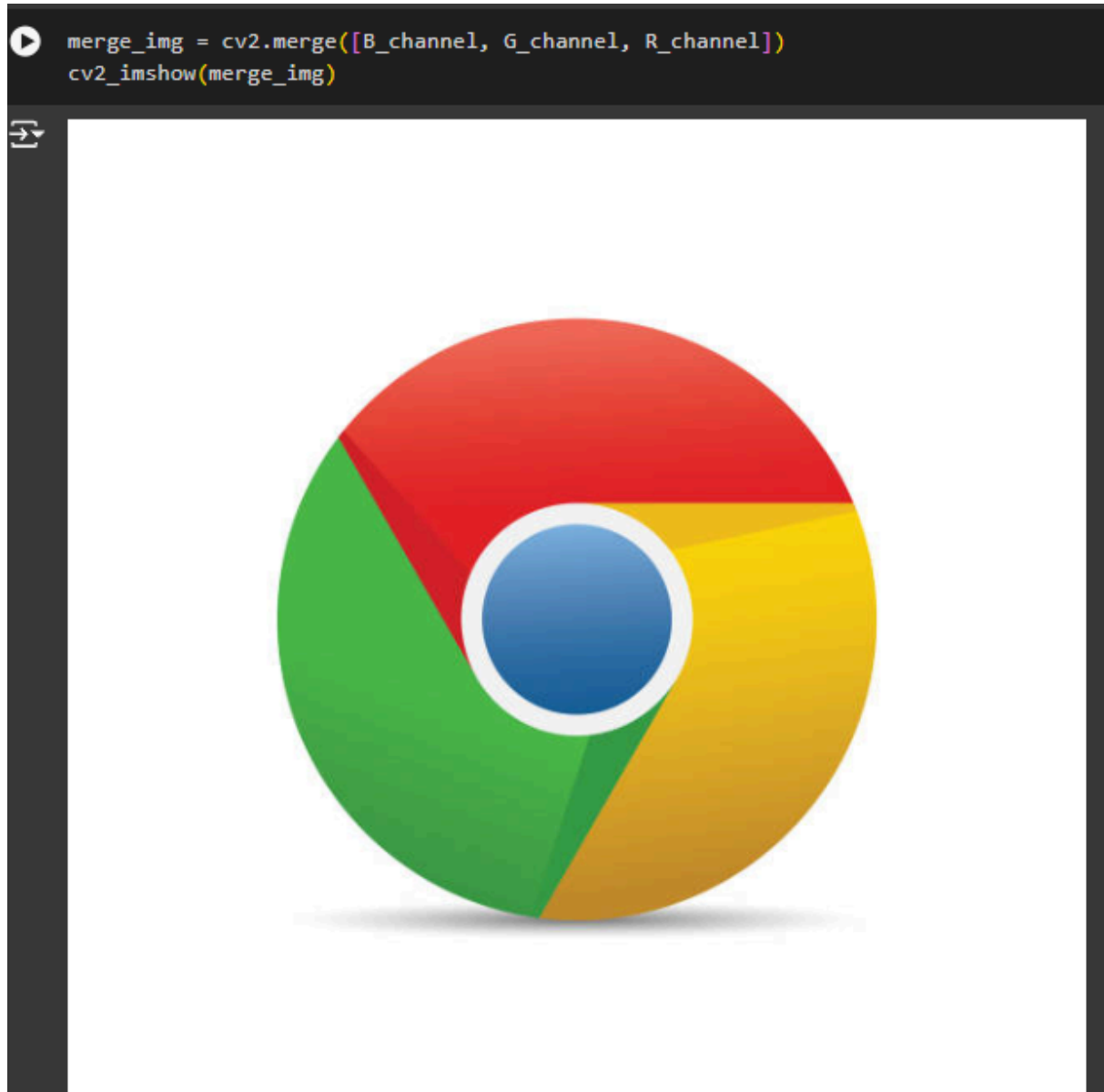
Ảnh trên là ảnh màu đã được khôi phục lại từ ảnh xám gốc

3. Kết hợp các channel màu để tạo ảnh màu từ các ảnh xám.

Khi split ảnh màu thành các channel. Ảnh sẽ là 1 channel aka ảnh xám. Hoặc có thể thêm những channel còn thiếu với giá trị bằng 0




Kết hợp 3 màu theo đúng thứ tự channel (BGR) ta có lại ảnh gốc:



Ngoài ra ta có thể khôi phục ảnh dựa theo ảnh xám của nó:

```
[25] red_channel = img_gray * 0.5
      green_channel = img_gray * 0.2
      blue_channel = img_gray * 0.3

      color_image = cv2.merge([blue_channel.astype('uint8'), green_channel.astype('uint8'), red_channel.astype('uint8')])
      cv2.imshow('color_image', color_image)
```



4. Kết luận

Việc chuyển đổi ảnh màu sang ảnh xám giúp ta giảm đi khối lượng thông tin rất lớn nhưng cũng làm mất đi lượng thông tin đó bởi ta không thể khôi phục chính xác ảnh lại ảnh màu từ ảnh xám.

Nguồn [Project1.ipynb](#)