

**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**  
**ĐẠI HỌC KHOA HỌC TỰ NHIÊN**  
**NGÀNH CÔNG NGHỆ THÔNG TIN**



## **PROJECT 02**

Hoàng Nguyên Trúc

18127055

**Môn học: Toán ứng dụng và thống kê**  
**Thành phố Hồ Chí Minh – 2020**

---

## MỤC LỤC

---

MỤC LỤC.....	2
NHỮNG CHỨC NĂNG HOÀN THÀNH.....	3
Ý TƯỞNG THỰC HIỆN CÁC HÀM CHỨC NĂNG .....	4
HÌNH ẢNH KẾT QUẢ CỦA TỪNG CHỨC NĂNG.....	10
TÀI LIỆU THAM KHẢO.....	13

## NHỮNG CHỨC NĂNG HOÀN THÀNH

CHỨC NĂNG	MỨC ĐỘ HOÀN THÀNH
Thay đổi độ sáng cho ảnh	100
Thay đổi độ tương phản	100
Chuyển đổi ảnh RGB thành ảnh xám	100
Chồng 2 ảnh cùng kích thước	100
Xoay dọc ảnh	100
Xoay ngang ảnh	100
Làm mờ ảnh	100

# Ý TƯỞNG THỰC HIỆN CÁC HÀM CHỨC NĂNG

## 1) Hàm chuyển đổi hình ảnh → ma trận :

```
def data(path):  
    img = Image.open(path).convert('RGB')  
    img_np = np.asarray(img) # convert to numpy array  
    return img_np
```

- Input: nhận đường dẫn tới hình ảnh.
- Đọc ảnh và chuyển mode từ 'RGBA' sang 'RGB'.
- Chuyển đổi hình ảnh sang numpy array.

## 2) Hàm thay đổi độ sáng hình ảnh:

```
def change_light(img_npa, scalar):  
    temp = img_npa.copy()  
    for ele in temp:  
        for ele_child in ele:  
            for pos in range(3):  
                if (ele_child[pos] + scalar) > 255:  
                    ele_child[pos] = 255  
                else:  
                    ele_child[pos] += scalar  
    return temp
```

- Cộng thêm 1 số tự nhiên (dương: tăng độ sáng, âm: giảm độ sáng) cho từng pixel(R,G,B) trong mảng ảnh 3 chiều.

### 3) Hàm thay đổi độ tương phản hình ảnh:

```
def change_contrast(img_npa, scalar):  
    temp = img_npa.copy()  
    for ele in temp:  
        for ele_child in ele:  
            for pos in range(3):  
                if (ele_child[pos] * scalar) > 255:  
                    ele_child[pos] = 255  
                else:  
                    ele_child[pos] *= scalar  
    return temp
```

- Nhân thêm 1 số tự nhiên cho từng pixel(R,G,B) trong mảng ảnh 3 chiều.

### 4) Hàm chuyển một ảnh sang ảnh màu xám:

```
def change_grey(img_npa):  
    temp = img_npa.copy()  
    for ele in temp:  
        for ele_child in ele:  
            value = ele_child[0]*0.3 + ele_child[1]*0.59 + ele_child[2]*0.11  
            for pos in range(3):  
                ele_child[pos] = value  
    return temp
```

- Tính greyscale theo tỉ lệ  $R*0.3 + G*0.59 + B*0.11$
- Gán greyscale của mỗi pixel vào vị trí R G B của pixel đó:  
 $\text{pixel}(R,G,B) \rightarrow \text{pixel}(\text{greyscale}, \text{greyscale}, \text{greyscale})$

## 5) Hàm chồng 2 ảnh cùng kính thước:

```
def stack_photo(img1, img2):  
    shape = np.array(img1).shape  
    width = shape[0]  
    height = shape[1]  
    img1 = np.reshape(img1, (width*height, 3))  
    img2 = np.reshape(img2, (width*height, 3))  
    ans = []  
    for i in range(width*height):  
        ans.append((img1[i]+img2[i])//2)  
    return ans, width, height
```

- Truyền 2 ảnh cùng kính thước vào hàm.
- Gọi hàm chuyển thành ảnh xám lần lượt cho từng ảnh.
- Cộng hai ảnh chia 2 để giá trị R,G,B không vượt quá 255 và ảnh không bị cháy sáng.

## 6) Hàm xoay dọc ảnh:

```
def vertical(img_npa):  
    ans = []  
    shape = np.array(img_npa).shape  
    width = shape[0]  
    height = shape[1]  
    img = np.reshape(img_npa, (width*height, 3))  
    for i in range((width*height)):  
        ans.append(img[width*height - i - 1])  
    return ans, width, height
```

- Đảo thứ tự các dòng pixel trong ảnh. Ví dụ:  
123      789  
456 → 456  
789      123

## 7) Hàm xoay ngang ảnh:

```
def horizontal(img_npa):  
    re = []  
    for ele in img_npa:  
        ans = []  
        for e in range(len(ele)-1, -1, -1):  
            ans.append(ele[e])  
        re.append(ans)  
    return re
```

➤ Đổi thứ tự các cột pixel trong ảnh. Ví dụ:

123	321
456	→ 654
789	987

## 8) Hàm làm mờ ảnh:

```
def cal_avg(img_npa, y, x): #y is i, x is j
    shape = np.array(img_npa).shape
    w = shape[0]
    h = shape[1]
    c, r = [], []

    for i in np.arange(-1,1):
        if x + i < 0:
            c.append(0)
        elif x+i > w - 1:
            c.append(w - 1)
        else:
            c.append(x + i)

    for i in np.arange(-1,1):
        if y + i < 0:
            r.append(0)
        elif y+i > h-1:
            r.append(h - 1)
        else:
            r.append(y + i)

    sum = np.zeros(3)
    for i in r:
        for j in c:
            sum += img_npa[i][j]/9

    sum = [int(i) for i in sum]
    return sum
```

- Truyền vị trí của ô pixel đang xét và mảng hình ảnh 3 chiều đọc từ hàm data(path).
- Dùng box blur kernel để tính giá trị mới cho ô pixel bằng cách nhân  $1/9$  cho 8 ô xung quanh pixel đang xét và tính tổng những ô đó.

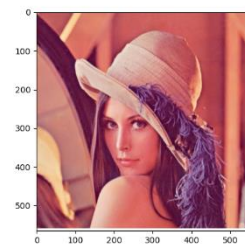
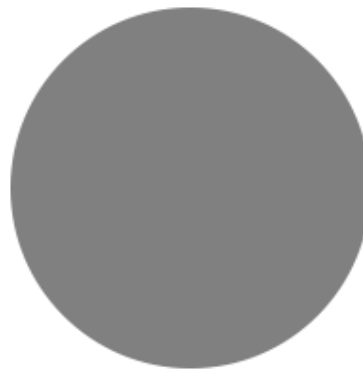
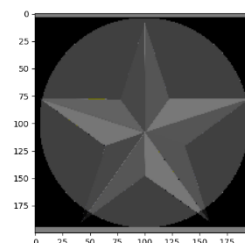


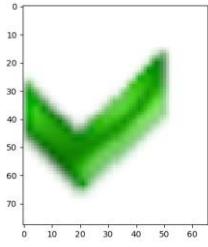
```
def blur(img_npa):  
    img_temp = img_npa.copy()  
    for pos, ele in enumerate(img_temp):  
        for e in range(len(ele)):  
            temp = cal_avg(img_npa, pos, e)  
            ele[e] = temp  
    return img_npa
```

- Duyệt từng pixel của mảng hình ảnh 3 chiều, gán vào R,G,B của ô pixel được xét bằng giá trị nhận được từ hàm `cal_avg`.

# HÌNH ẢNH KẾT QUẢ CỦA TỪNG CHỨC NĂNG

<b>ẢNH GỐC</b>	
<b>ẢNH SAU KHI THAY ĐỔI ĐỘ SÁNG</b>	
<b>ẢNH SAU KHI THAY ĐỔI ĐỘ TƯƠNG PHẢN</b>	
<b>ẢNH XÁM</b>	
<b>ẢNH XOAY DỌC</b>	

**ẢNH XOAY NGANG****ẢNH 1****ẢNH 2****CHỖNG HAI ẢNH CÙNG KÍNH THƯỚC**

<b>ẢNH GỐC</b>	✓
<b>ẢNH SAU KHI LÀM MỜ</b>	

---

## TÀI LIỆU THAM KHẢO

---

- 1) [https://www.tutorialspoint.com/dip/concept\\_of\\_blurring.htm](https://www.tutorialspoint.com/dip/concept_of_blurring.htm)
- 2) [https://en.wikipedia.org/wiki/Kernel\\_\(image\\_processing\)](https://en.wikipedia.org/wiki/Kernel_(image_processing))