

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN TP.HCM
KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO TIẾN ĐỘ ĐỒ ÁN
Toán ứng dụng và thống kê cho công nghệ thông tin
21CLC04

Đồ án

IMAGE PROCESSING



Giáo viên hướng dẫn

Nguyễn Văn Quang Huy
Ngô Đình Hy
Nguyễn Đình Thúc

Thành viên

Lý Nhật Hào - 21127041

NIÊN KHOÁ 2022 - 2023



LỜI CẢM ƠN

Để hoàn thành được bài báo cáo này, em đã nhận được sự giúp đỡ rất nhiều từ phía thầy cô giảng viên, trợ giảng và bạn bè. Nay em xin gửi lời cảm ơn chân thành và sâu sắc nhất đến giảng viên môn Toán ứng dụng và thống kê cho Công Nghệ Thông Tin lớp 21CLC4, Khoa Công nghệ thông tin :

- Giảng viên **Nguyễn Đình Thúc**
- Giảng viên **Ngô Đình Huy**
- Giảng viên **Nguyễn Văn Quang Huy**

Các thầy đã đồng hành, đã luôn quan tâm, hướng dẫn và truyền đạt, cung cấp kiến thức, tài liệu và các thủ thuật cần thiết để em có thể hoàn thành đồ án !

Trong quá trình thực hiện đồ án không thể tránh khỏi những thiếu sót. Em rất mong nhận được nhiều ý kiến đóng góp từ các giảng viên và bạn bè để đồ án ngày càng hoàn thiện hơn!



THÀNH VIÊN NHÓM BÁO CÁO

X

Lý Nhật Hào

Xin chân thành cảm ơn!



MỤC LỤC

LỜI CẢM ƠN	1
Chương I: GIỚI THIỆU ĐỒ ÁN	4
1.1. Thông tin cá nhân sinh viên thực hiện đồ án:	4
1.2. Tổng quát yêu cầu đồ án và mức độ hoàn thiện:	4
CHƯƠNG II: CÀI ĐẶT THUẬT TOÁN VÀ HƯỚNG DẪN SỬ DỤNG CHƯƠNG TRÌNH	5
2.1. Pseudocode:	5
• Xử lý tăng độ sáng của ảnh	5
• Xử lý tăng độ tương phản của ảnh:	6
• Xử lý lật ảnh theo chiều ngang hoặc dọc	7
• Chuyển đổi ảnh màu sang ảnh xám	8
• Áp dụng bộ lọc sepia vào một hình ảnh	9
• Xử lý làm mờ hình ảnh	10
• Xử lý làm sắc nét hình ảnh	11
• Cắt một phần của hình ảnh	12
• Vẽ một hình tròn để làm khung hình ảnh	13
• Vẽ 2 hình elip chéo nhau để làm khung cho hình ảnh	14
2.2. Hướng Dẫn Sử Dụng và Kết Quả Của Chương Trình Image Processing:	15
CHƯƠNG III: TÀI LIỆU THAM KHẢO	21

Chương I: GIỚI THIỆU ĐỒ ÁN

1.1. Thông tin cá nhân sinh viên thực hiện đồ án:

Họ & Tên	Lý Nhật Hào
MSSV	21127041
EMAIL	lnhao21@clc.fitus.edu.vn

1.2. Tổng quát yêu cầu đồ án và mức độ hoàn thiện:

Bài nộp đã hoàn thành 100% yêu cầu đồ án 2 đưa ra cùng với 5% nâng cao

LINK VIDEO DEMO: <https://youtu.be/qDIapY3J00g>

Các chức năng xử lý ảnh cơ bản đã được hoàn thành cụ thể như sau:

1. Thay đổi độ sáng cho ảnh (1 điểm)
2. Thay đổi độ tương phản (1 điểm)
3. Lật ảnh (ngang - dọc) (1 điểm)
4. Chuyển đổi ảnh RGB thành ảnh xám/sepia (2 điểm)
5. Làm mờ/sắc nét ảnh (2 điểm)
6. Cắt ảnh theo kích thước (cắt ở trung tâm) (1 điểm)
7. Cắt ảnh theo khung hình tròn (1 điểm)
8. Viết hàm main xử lý (1 điểm) với các yêu cầu sau:

- Cho phép người dùng nhập vào tên tập tin ảnh mỗi khi hàm main được thực thi.

- Cho phép người dùng lựa chọn chức năng xử lý ảnh (từ 1 đến 7, đối với chức năng 4 cho phép lựa chọn giữa lật ngang hoặc lật dọc). Lựa chọn 0 cho phép thực hiện tất cả chức năng với tên file đầu ra tương ứng với từng chức năng. Ví dụ:

- Đầu vào: `cat.png`
- Chức năng: Làm mờ
- Đầu ra: `cat_blur.png`

9. **Nâng cao - Không bắt buộc (Cộng 0.5 điểm vào điểm đồ án 2)**

- Thực hiện cắt nội dung ảnh theo khung được áp lên, với khung là 2 hình elip chéo nhau.

CHƯƠNG II: CÀI ĐẶT THUẬT TOÁN VÀ HƯỚNG DẪN SỬ DỤNG CHƯƠNG TRÌNH

2.1. Pseudocode:

- Xử lý **tăng độ sáng** của ảnh bởi hàm Bright(picture2, picture1, temp).
Cụ thể:
 - Đầu vào của hàm bao gồm **picture2** và **picture1** là hai ảnh đầu vào, và **temp** là hằng số tăng độ sáng (mặc định là 0,4).
 - Chuyển đổi kiểu dữ liệu của **picture2** thành kiểu dữ liệu số nguyên để thực hiện phép tính toán.
 - Thực hiện phép tăng độ sáng bằng cách cộng **picture2** với giá trị tương ứng của **255*temp**.
 - Giới hạn giá trị của **picture2** trong khoảng từ 0 đến 255 để đảm bảo rằng ảnh không bị tràn số.
 - Chuyển đổi lại kiểu dữ liệu của **picture2** thành kiểu uint8 để có thể hiển thị được trên các trình xem ảnh
 - Tạo một đối tượng **fig** và **ax** để hiển thị ảnh gốc và ảnh đã được tăng độ sáng sử dụng thư viện matplotlib. Cuối cùng, hàm trả về ma trận ảnh **picture2** đã được tăng độ sáng.

```
def Bright(picture2,picture1, temp = 0.4):
    picture2 = picture2.astype(int)
    picture2 = picture2 + int(255*temp)
    picture2[255<picture2] = 255
    picture2 = picture2.astype(np.uint8)
    fig, ax = plt.subplots(1, 2, figsize = (10, 6))
    ax[0].imshow(picture1)
    ax[0].set_title('PICTURE WITHOUT BRIGHTEN')
    ax[1].imshow(picture2)
    ax[1].set_title('BRIGHTEN PICTURE')
    for ax in fig.axes:
        ax.axis('on')

    return picture2
```

✓ 0.0s

- Xử lý **tăng độ tương phản** của ảnh trong hàm Contrast(picture2, picture1, temp). Cụ thể:

- Đầu vào của hàm bao gồm **picture2** và **picture1** là hai ảnh đầu vào, và **temp** là hằng số tăng độ tương phản (mặc định là 0,5).
- Chuyển đổi kiểu dữ liệu của **picture2** thành kiểu dữ liệu số thực để thực hiện phép tính toán.
- Chuẩn hóa giá trị của **picture2** về khoảng từ -0.5 đến 0.5.
- Tăng độ tương phản của **picture2** bằng cách nhân với giá trị của **(temp+1)**2**.
- Chuyển đổi lại giá trị của **picture2** về khoảng từ 0 đến 255.
- Giới hạn giá trị của **picture2** trong khoảng từ 0 đến 255 để đảm bảo rằng ảnh không bị tràn số.
- Chuyển đổi lại kiểu dữ liệu của **picture2** thành kiểu uint8 để có thể hiển thị được trên các trình xem ảnh.
- Tạo một đối tượng **fig** và **ax** để hiển thị ảnh gốc và ảnh đã được tăng độ tương phản sử dụng thư viện matplotlib. Cuối cùng, hàm trả về ma trận ảnh **picture2** đã được tăng độ tương phản.

```
def Contrast(picture2, picture1, temp = 0.5):
    picture2 = picture2.astype(float)

    picture2 = picture2/255 - 0.5
    picture2 = picture2* (float(temp+1)**2)

    picture2 = 255*(0.5+picture2)
    picture2[picture2 > 255] = 255
    picture2[picture2 < 0] = 0
    picture2 = picture2.astype(np.uint8)
    fig, ax = plt.subplots(1, 2, figsize = (10, 6))
    ax[0].imshow(picture1)
    ax[0].set_title('PICTURE WITHOUT CONTRAST')
    ax[1].imshow(picture2)
    ax[1].set_title('CONTRAST PICTURE')
    for ax in fig.axes:
        ax.axis('on')

    return picture2
```

✓ 0.0s

- Xử lý ***lật ảnh theo chiều ngang hoặc dọc*** trong hàm Flippic(**image**, **picture1**, **tam**, **mode**). Cụ thể:

- Đầu vào của hàm bao gồm **image** là ma trận ảnh đầu vào, **picture1** là ảnh gốc, **tam** là kích thước của ảnh và **mode** là chế độ lật ảnh (0 - lật theo chiều dọc, 1 - lật theo chiều ngang).
- Chuyển đổi kích thước của **image** thành kích thước tương ứng với **tam**.
- Thực hiện phép lật ảnh dựa trên **mode** bằng cách sử dụng hàm **np.flip()**.
 - Nếu **mode** là 0, ảnh sẽ được lật theo chiều dọc.
 - Nếu **mode** là 1, ảnh sẽ được lật theo chiều ngang.
- Chuyển đổi lại kiểu dữ liệu của **image** thành kiểu uint8 để có thể hiển thị được trên các trình xem ảnh.
- Tạo một đối tượng **fig** và **ax** để hiển thị ảnh gốc và ảnh đã được lật sử dụng thư viện **matplotlib**.
- Cuối cùng, hàm trả về ma trận ảnh **image** đã được lật và được chuyển đổi thành một mảng một chiều bằng cách sử dụng hàm **flatten()**.

```
def Flippic(image,picture1, tam, mode):  
    image = image.reshape(tam)  
    image = np.flip(image, mode)  
  
    image = image.astype(np.uint8)  
    fig, ax = plt.subplots(1, 2, figsize = (10, 6))  
    ax[0].imshow(picture1)  
    ax[0].set_title('PICTURE WITHOUT FLIP')  
    ax[1].imshow(image)  
    ax[1].set_title('FLIP PICTURE')  
    for ax in fig.axes:  
        ax.axis('on')  
    image = image.flatten()  
    return image
```

✓ 0.0s

- **Chuyển đổi ảnh màu sang ảnh xám** trong hàm Blackwhite(**picture2, picture1, x, y, z**). Cụ thể:

- Đầu vào của hàm bao gồm **picture2** và **picture1** là hai ảnh đầu vào, và **x,y,z** là các hằng số được sử dụng để tính toán giá trị xám (mặc định là **0,399, 0,687 và 0,214**).
- Chuyển đổi kiểu dữ liệu của **picture2** thành kiểu dữ liệu số thực để thực hiện phép tính toán.
- Tính tổng của **x+y+z**.
- Thực hiện phép nhân ma trận giữa **picture2** và một ma trận 1×3 chứa các giá trị của **x, y, z / sum**.
- Chuyển đổi vị trí của các phần tử trong ma trận **picture2** để chuyển đổi thành ảnh xám.
- Chuyển đổi lại kiểu dữ liệu của **picture2** thành kiểu **uint8** để có thể hiển thị được trên các trình xem ảnh.
- Tạo một đối tượng **fig** và **ax** để hiển thị ảnh gốc và ảnh đã được chuyển đổi thành ảnh xám sử dụng thư viện **matplotlib**.
- Cuối cùng, hàm trả về ma trận ảnh **picture2** đã được chuyển đổi thành ảnh xám.

```
def Blackwhite(picture2, picture1, x = 0.399, y = 0.687, z = 0.214):  
    picture2 = picture2.astype(float)  
    sum=x+y+z  
    picture2 = np.dot(picture2, np.array([x, y, z])/(sum))  
    picture2 = (picture2*np.ones((3,1))).transpose()  
  
    picture2 = picture2.astype(np.uint8)  
    fig, ax = plt.subplots(1, 2, figsize = (10, 6))  
    ax[0].imshow(picture1)  
    ax[0].set_title('PICTURE WITHOUT GREY')  
    ax[1].imshow(picture2)  
    ax[1].set_title('GREY PICTURE')  
    for ax in fig.axes:  
        ax.axis(['on'])  
    return picture2
```

✓ 0.0s

- **Áp dụng bộ lọc sepia vào một hình ảnh bằng hàm `sepia(picture2,picture1, depth=10)`. Cụ thể:**

- Chuyển đổi hình ảnh đầu vào sang kiểu dữ liệu float để thực hiện các phép tính toán.
- Tạo ma trận bộ lọc sepia với các giá trị đã được xác định trước.
- Áp dụng ma trận bộ lọc sepia vào hình ảnh đầu vào bằng cách sử dụng phép nhân ma trận.
- Tăng độ sâu của màu sepia bằng cách thêm giá trị `depth` vào hình ảnh.
- Giới hạn giá trị tối đa của hình ảnh tại 255 và chuyển đổi lại sang kiểu dữ liệu uint8.
- Hiển thị hình ảnh gốc và hình ảnh đã được xử lý với bộ lọc sepia trên một **figure** với hai trực
- Trả về hình ảnh đã được xử lý với bộ lọc sepia
- Hàm này cung cấp cho người dùng tùy chọn để tăng độ sâu của màu sepia bằng cách chỉ định giá trị `depth`.

```
def sepia(picture2,picture1, depth=10):
    picture2 = picture2.astype(float)

    # Apply sepia filter
    sepia_filter = np.array([[0.393, 0.769, 0.189],
                           [0.349, 0.686, 0.168],
                           [0.272, 0.534, 0.131]])
    picture2 = np.dot(picture2, sepia_filter.T)

    # Scale the depth of sepia color
    picture2 = picture2 + depth

    # Cap the maximum value at 255 and convert back to uint8
    picture2[picture2 > 255] = 255
    picture2 = picture2.astype(np.uint8)
    fig, ax = plt.subplots(1, 2, figsize = (10, 6))
    ax[0].imshow(picture1)
    ax[0].set_title('PICTURE WITHOUT SEPIA')
    ax[1].imshow(picture2)
    ax[1].set_title('SEPIA PICTURE')
    for ax in fig.axes:
        ax.axis('on')

    return picture2
```

- Xử lý **làm mờ hình ảnh** từ hàm Blur(picture2, picture1, tam, kernel).
Cụ thể:

- Tính toán kích thước của ma trận bộ lọc.
- Thêm lè cho hình ảnh đầu vào để có thể áp dụng ma trận bộ lọc cho các điểm ở rìa của hình ảnh.
- Khởi tạo một mảng kết quả với kích thước bằng với hình ảnh đầu vào.
- Lặp qua từng điểm ảnh của hình ảnh đầu vào và tính toán giá trị mới của điểm ảnh đó bằng cách áp dụng ma trận bộ lọc. Giá trị mới được tính bằng cách lấy tổng các giá trị pixel trong vùng lân cận được quy định bởi ma trận bộ lọc và chia cho tổng giá trị trong ma trận bộ lọc.
- Hiển thị hình ảnh gốc và hình ảnh đã được làm mờ trên một **figure** với hai trục.
- Trả về hình ảnh đã được làm mờ.

```
def blur(picture2, picture1, tam, kernel):  
    kernel_tam = kernel.shape[0]  
    pad_size = kernel_tam // 2  
    picture2_pad = np.pad(picture2, ((pad_size, pad_size), (pad_size, pad_size), (0, 0)), mode='edge')  
    picture2_result = np.zeros(picture2.shape)  
  
    for i in range(picture2.shape[0]):  
        for j in range(picture2.shape[1]):  
            kernel_sum = 0  
            for k in range(kernel_tam):  
                for l in range(kernel_tam):  
                    pixel_value = picture2_pad[i+k, j+l]  
                    kernel_value = kernel[k, l]  
                    picture2_result[i, j] += pixel_value * kernel_value  
                    kernel_sum += kernel_value  
            picture2_result[i, j] /= kernel_sum  
    #picture2 = picture2.astype(np.uint8)  
    fig, ax = plt.subplots(1, 2, figsize = (10, 6))  
    ax[0].imshow(picture1)  
    ax[0].set_title('PICTURE WITHOUT BLUR')  
    ax[1].imshow(picture2)  
    ax[1].set_title('BLUR PICTURE')  
    for ax in fig.axes:  
        ax.axis('on')  
    return picture2_result
```

- Xử lý **làm sắc nét hình ảnh bằng hàm Sharpen(picture2, picture1, tam, kernel())**. Cụ thể:
 - Tạo ma trận bộ lọc cạnh để tăng cường độ tương phản của các cạnh trong hình ảnh.
 - Áp dụng bộ lọc Gaussian làm mờ để loại bỏ nhiễu trong hình ảnh đầu vào.
 - Tính toán đạo hàm của hình ảnh để tìm các cạnh trong hình ảnh.
 - Trừ hình ảnh đã được làm mờ từ hình ảnh gốc để tăng độ sắc nét của hình ảnh.
 - Giới hạn giá trị pixel trong khoảng từ 0 đến 255 và chuyển đổi lại sang kiểu dữ liệu uint8.
 - Hiển thị hình ảnh gốc và hình ảnh đã được làm sắc nét trên một figure với hai trục.
 - Trả về hình ảnh đã được làm sắc nét.

```
def Sharpen(picture2, picture1, tam, kernel):  
  
    # Apply high-pass filter to enhance edges  
    kernelDim = kernel.shape[0]  
  
    edge_kernel = np.array([[-1,-1,-1], [-1,9,-1], [-1,-1,-1]])  
    picture2 = blurforsharp(picture2, picture1, tam, edge_kernel)  
  
    # Subtract the blurred picture2 from the original picture2 to increase sharpness  
    picture2 = picture2.reshape(tam)  
    picture2_result = np.zeros(picture2.shape)  
    for i in range(kernelDim):  
        for j in range(kernelDim):  
            rowShiftValue = int(i - kernelDim/2)  
            colShiftValue = int(j - kernelDim/2)  
            shiftedArray = np.roll(picture2, (rowShiftValue, colShiftValue), axis=(0, 1))  
            picture2_result += shiftedArray * kernel[i,j]  
  
    picture2_result = picture2_result.flatten()  
  
    picture2_result[picture2_result > 255] = 255  
    picture2_result[picture2_result < 0] = 0  
  
    picture2_result = picture2_result.astype(np.uint8)  
    fig, ax = plt.subplots(1, 2, figsize = (10, 6))  
    ax[0].imshow(picture1)  
    ax[0].set_title('PICTURE WITHOUT SHARPEN')  
    ax[1].imshow(picture2)  
    ax[1].set_title('SHARPEN PICTURE')  
    for ax in fig.axes:  
        ax.axis('on')  
    return picture2_result
```

✓ 0.0s

- **Cắt một phần của hình ảnh** theo hàm **Cut(picture2, picture1, size)**.
Hàm này thực hiện các bước sau:

- Lấy kích thước của hình ảnh gốc.
- Tính toán giá trị **left, top, right và bottom** để xác định phần cần cắt của hình ảnh.
- Sử dụng phương thức **crop()** của đối tượng hình ảnh để cắt phần cần cắt.
- Chuyển đổi hình ảnh đã cắt sang kiểu dữ liệu **numpy array** và kiểu dữ liệu **uint8**.
- Hiển thị hình ảnh gốc và hình ảnh đã được cắt trên một **figure** với hai trục.
- Trả về hình ảnh đã được cắt.
- Hàm này yêu cầu ba đối số đầu vào: hình ảnh đầu vào, hình ảnh gốc và kích thước của phần cần cắt.

```

def Cut(picture2, picture1, size):
    width, height = picture1.size
    left = (width - size) / 2
    top = (height - size) / 2
    right = (width + size) / 2
    bottom = (height + size) / 2
    cropped_picture2 = picture1.crop(left, top, right, bottom)
    cropped_picture2 = np.array(cropped_picture2)
    cropped_picture2 = cropped_picture2.astype(np.uint8)
    fig, ax = plt.subplots(1, 2, figsize = (10, 6))
    ax[0].imshow(picture1)
    ax[0].set_title('PICTURE WITHOUT CUT')
    ax[1].imshow(cropped_picture2)
    ax[1].set_title('CUT PICTURE')
    for ax in fig.axes:
        ax.axis('on')
    return cropped_picture2

```

✓ 0.0s

- **Vẽ một hình tròn để làm khung hình ảnh** theo hàm circleK(picture2, tam, mode = 0). Cụ thể:

- Chuyển đổi hình ảnh đầu vào thành một mảng **numpy** 2D.
- Tính toán tọa độ của tâm của hình ảnh.
- Tính toán bán kính của hình tròn.
- Tạo một mảng **boolean mask_circle** với kích thước bằng với hình ảnh đầu vào, với giá trị True tại các điểm trong hình tròn và False tại các điểm bên ngoài hình tròn.
- Đặt giá trị các điểm bên ngoài hình tròn trong mảng hình ảnh đầu vào thành 0.
- Chuyển đổi mảng hình ảnh trở lại thành một mảng 1D và trả về.
- Hàm này yêu cầu ba đối số đầu vào:
 - Hình ảnh đầu vào
 - Kích thước của hình ảnh.
 - Chế độ hiển thị hình tròn (0 để vẽ một hình tròn có bán kính bằng với chiều nhỏ nhất của hình ảnh, 1 để vẽ một hình tròn có bán kính bằng với chiều lớn nhất của hình ảnh).

```

✓ def circleK(picture2, tam, mode = 0):
    picture2 = picture2.reshape(tam)

    height = tam[0]
    width = tam[1]
    |
    center = np.array([height/2, width/2])
    radius = mode*max(height/2, width/2) + (1 - mode)*min(height/2, width/2)

    x, y = np.ogrid[:height, :width]

    mask_circle = (x - center[0])**2 + (y - center[1])**2 > radius**2
    picture2[mask_circle] = np.zeros(tam[2])

    picture2 = picture2.flatten()
    return picture2
  ✓ 0.0s

```

- **Vẽ 2 hình elip chéo nhau để làm khung cho hình ảnh đầu vào. Hàm này thực hiện các bước sau:**

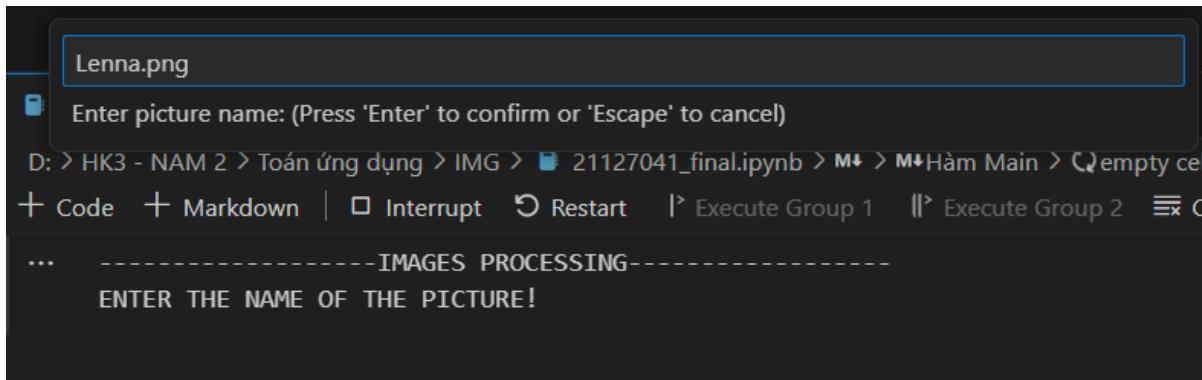
- Chuyển đổi hình ảnh đầu vào thành một mảng **numpy** 2D.
- Tính toán kích thước của hình ảnh.
- Tính toán góc quay của hình bầu dục.
- Tính toán kích thước của khung hình chữ nhật.
- Tạo một mảng boolean **ellips_final** đại diện cho hình bầu dục.
- Xác định tọa độ của các điểm trong hình bầu dục sau khi quay.
- Tạo một mảng boolean **rotatedellips_final** với kích thước bằng với khung hình chữ nhật, với giá trị **True** tại các điểm nằm trong hình bầu dục và **False** tại các điểm bên ngoài hình bầu dục.
- Đặt giá trị các điểm trong mảng hình ảnh đầu vào tại các điểm bên ngoài hình bầu dục trong **rotatedellips_final** thành 0.
- Chuyển đổi mảng hình ảnh trở lại thành một mảng 1D và trả về.

```
def interlaceEllipseFrame(picture2, tam, temp = 1.25, angle = 45):
    picture2 = picture2.reshape(tam)
    height = tam[0]
    width = tam[1]
    angle = math.radians(angle)
    sine = math.sin(angle)
    cosine = math.cos(angle)
    edge = int(min(width, height)*temp)
    new_edge = np.ceil(abs(edge * cosine) + abs(edge * sine)).astype(int)
    x_ellips, y_ellips = np.ogrid[:edge, :edge]
    smallRadius = edge/4
    largeRadius = edge/2
    ellips1 = (x_ellips-edge/2)**2/largeRadius**2 + (y_ellips-edge/2)**2/smallRadius**2 > 1
    ellips2 = (x_ellips-edge/2)**2/smallRadius**2 + (y_ellips-edge/2)**2/largeRadius**2 > 1
    ellips_final = ellips1 & ellips2
    y = edge/2 - y_ellips
    x = edge/2 - x_ellips
    x_new = (new_edge/2 - np.ceil(x*cosine + y*sine)).astype(int)
    y_new = (new_edge/2 - np.ceil(-x*sine + y*cosine)).astype(int)
    rotatedellips_final = np.ones((new_edge, new_edge), dtype = bool)
    rotatedellips_final[x_new, y_new] = ellips_final[x_ellips, y_ellips]
    c = int(np.ceil((rotatedellips_final.shape[0] - width)/2))
    rotatedellips_final = rotatedellips_final[c:-c, c:-c]

    temp = np.ones((height, width), dtype=bool)
    startIndexH = int((height - rotatedellips_final.shape[0])/2)
    startIndexW = int((width - rotatedellips_final.shape[1])/2)
    temp[startIndexH:rotatedellips_final.shape[0]+startIndexH,startIndexW:rotatedellips_final.shape[1]+startIndexW] = rotatedellips_final
    rotatedellips_final = temp
    picture2[rotatedellips_final] = np.zeros(tam[2])
    picture2 = picture2.flatten()
    return picture2
```

2.2. Hướng Dẫn Sử Dụng và Kết Quả Của Chương Trình Image Processing:

- **Bước 1:** Chạy chương trình
- **Bước 2:** Nhập vào tên bức ảnh cần compress

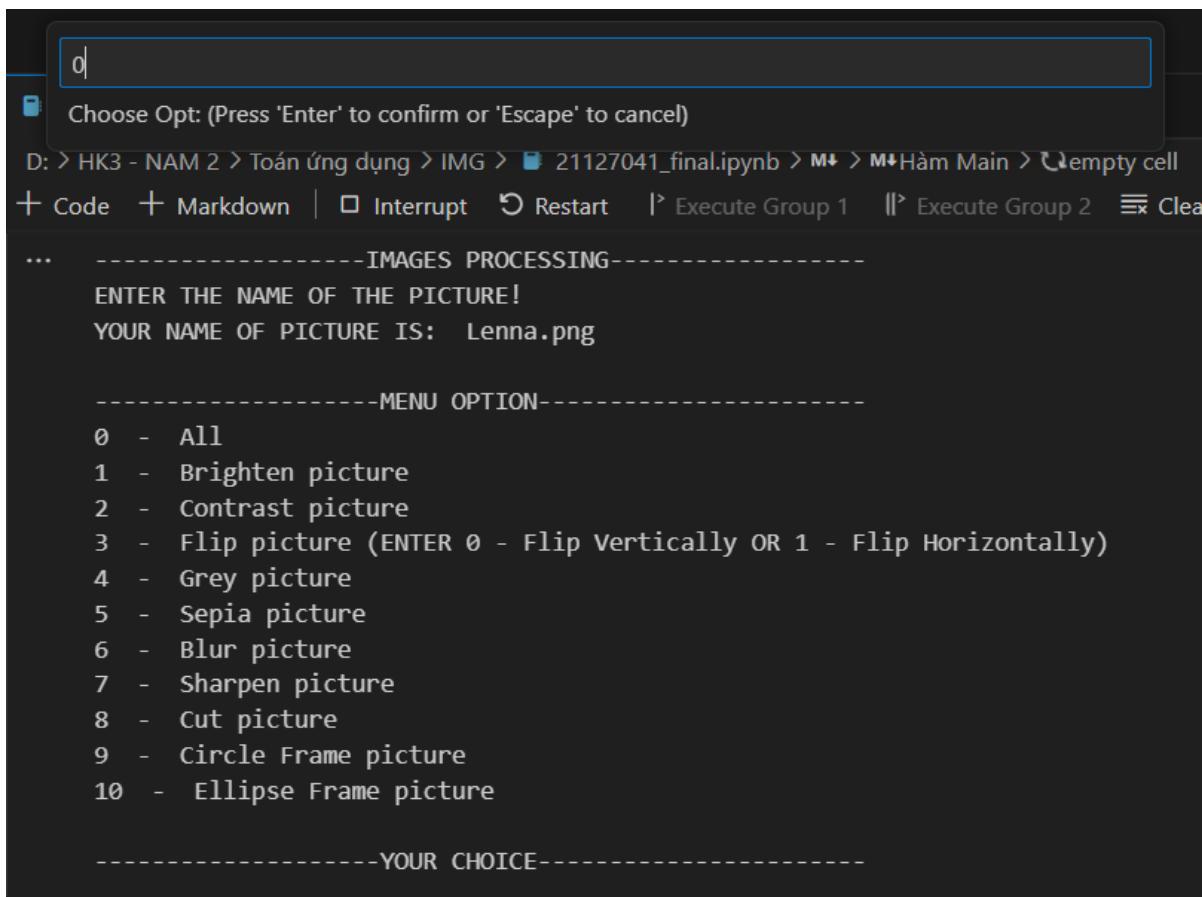


```

Lenna.png
 Enter picture name: (Press 'Enter' to confirm or 'Escape' to cancel)

D: > HK3 - NAM 2 > Toán ứng dụng > IMG > 21127041_final.ipynb > M+ > M+Hàm Main > Qempty cell
+ Code + Markdown | ⚡ Interrupt ⚡ Restart | ⌂ Execute Group 1 ⌂ Execute Group 2 ⌂ Clear Cell
... -----IMAGES PROCESSING-----
ENTER THE NAME OF THE PICTURE!
  
```

- **Bước 3:** Chọn lựa việc chỉnh sửa ảnh mong muốn trong menu lựa chọn



```

 Choose Opt: (Press 'Enter' to confirm or 'Escape' to cancel)

D: > HK3 - NAM 2 > Toán ứng dụng > IMG > 21127041_final.ipynb > M+ > M+Hàm Main > Qempty cell
+ Code + Markdown | ⚡ Interrupt ⚡ Restart | ⌂ Execute Group 1 ⌂ Execute Group 2 ⌂ Clear Cell
... -----IMAGES PROCESSING-----
ENTER THE NAME OF THE PICTURE!
YOUR NAME OF PICTURE IS: Lenna.png

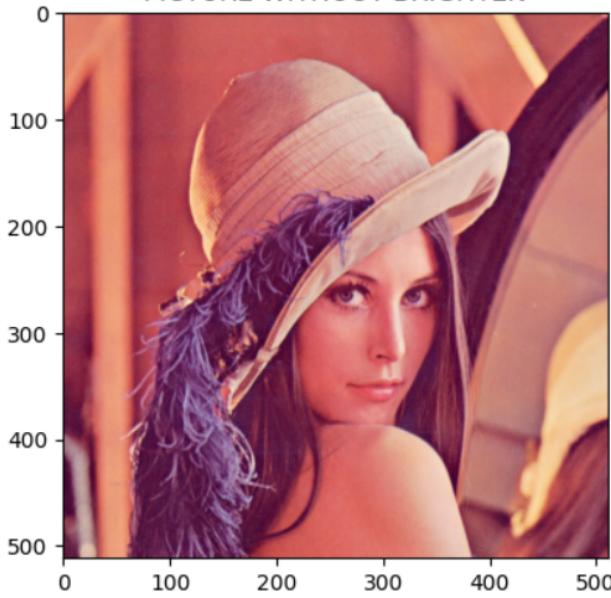
-----MENU OPTION-----
0 - All
1 - Brighten picture
2 - Contrast picture
3 - Flip picture (ENTER 0 - Flip Vertically OR 1 - Flip Horizontally)
4 - Grey picture
5 - Sepia picture
6 - Blur picture
7 - Sharpen picture
8 - Cut picture
9 - Circle Frame picture
10 - Ellipse Frame picture

-----YOUR CHOICE-----
  
```

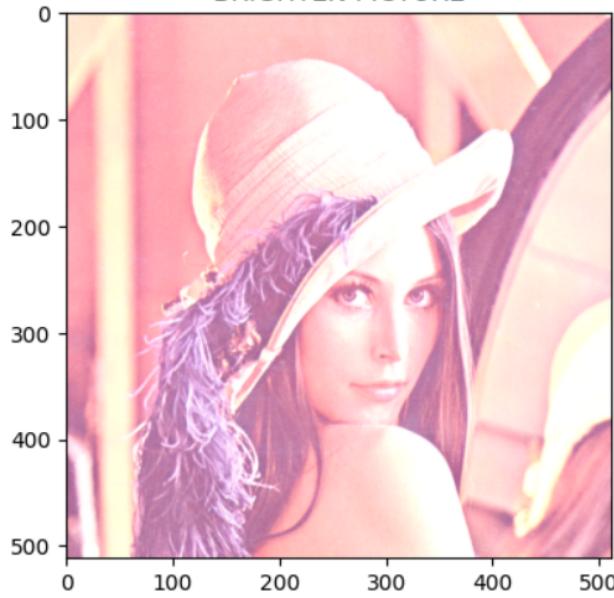
- **Bước 4:** xem kết quả được hiển thị trên màn hình console, thời gian thực hiện thuật toán và các ảnh được lưu trong folder theo định dạng đã yêu cầu.

Option: 1 - Brighten picture

PICTURE WITHOUT BRIGHTEN



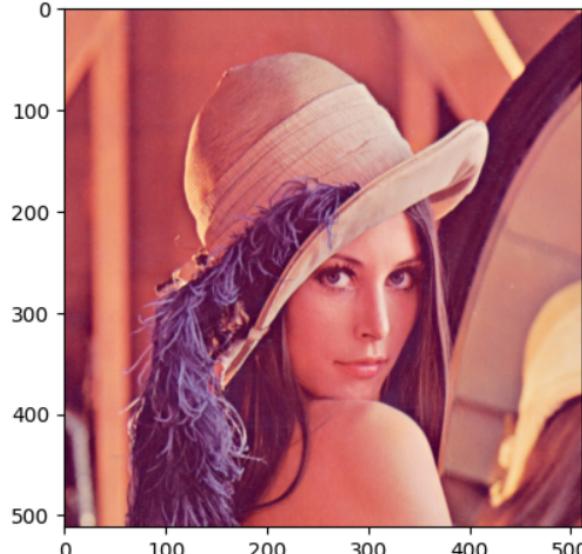
BRIGHTEN PICTURE



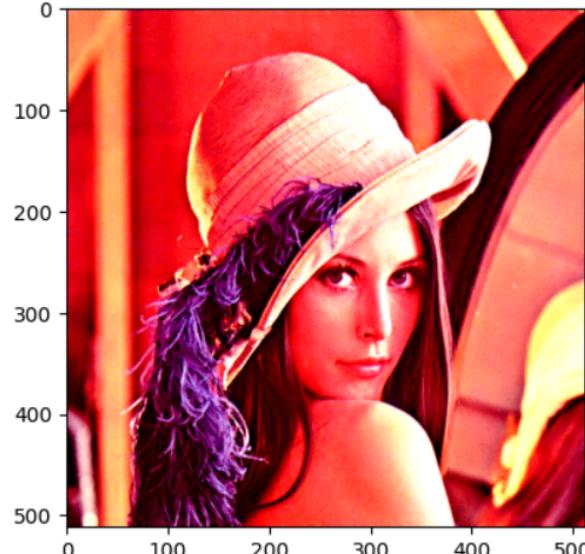
Your picture Lenna.png has been processed as Brighten picture in 0.7405579090118408 s

Option: 2 - Contrast picture

PICTURE WITHOUT CONTRAST



CONTRAST PICTURE



Your picture Lenna.png has been processed as Contrast picture in 0.8853929042816162 s

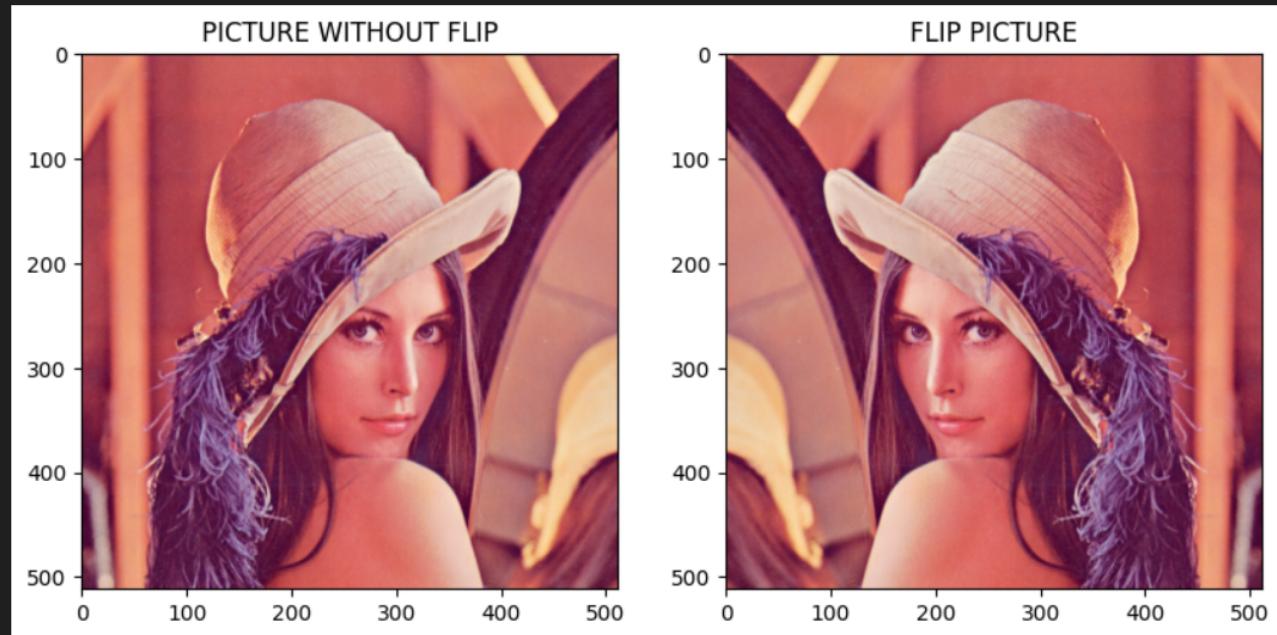
Opt 3 : FLIP IMAGE

ENTER 0 OR 1 TO FLIP BY DIRECTION

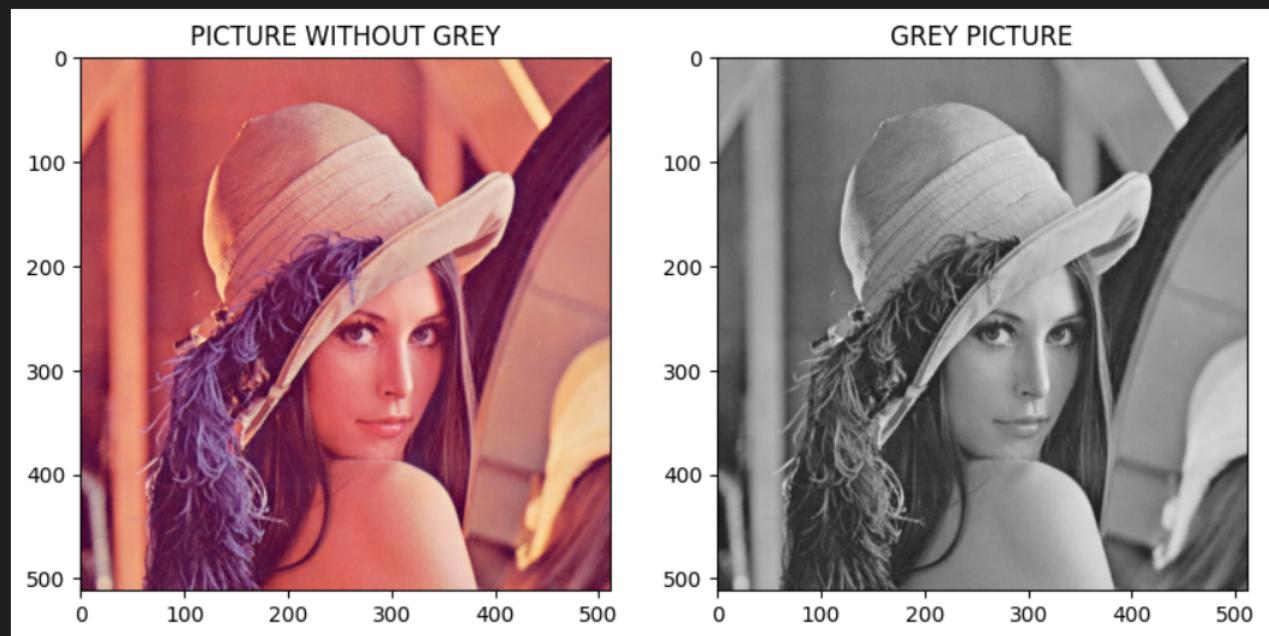
0 - Flip Vertically

1 - Flip Horizontally

Your picture has been processed as Flip Picture in 0.03838944435119629 s



Option: 4 - Grey picture



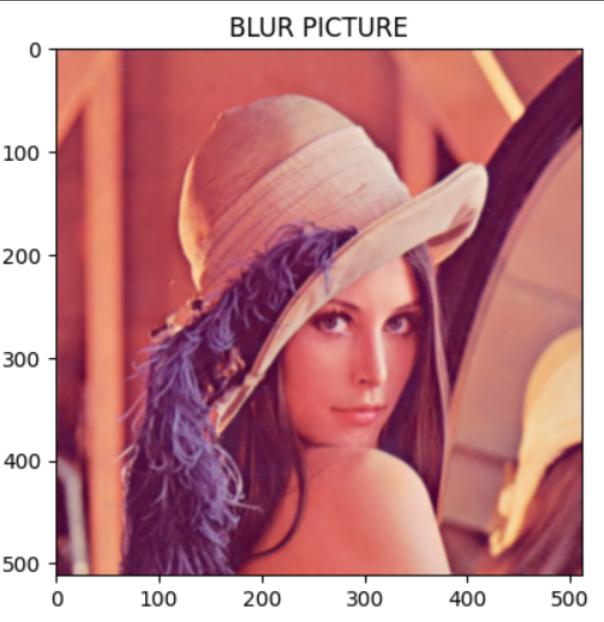
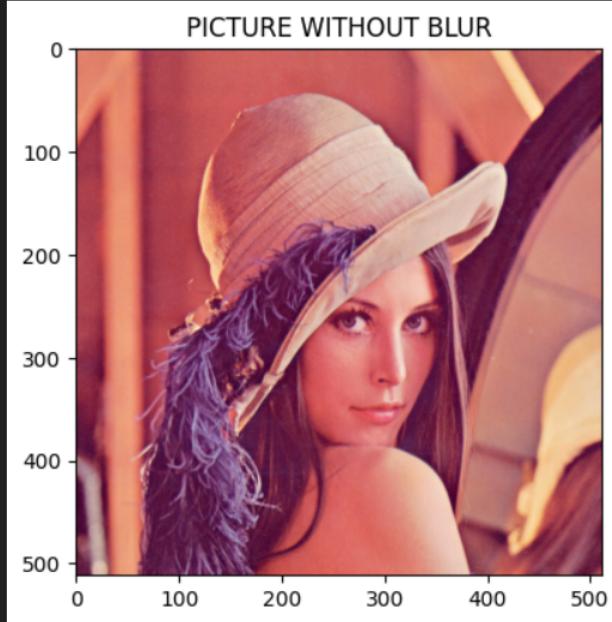
Your picture Lenna.png has been processed as Grey picture in 0.7017576694488525 s

Option: 5 - Sepia picture



Your picture Lenna.png has been processed as Sepia picture in 0.7885093688964844 s

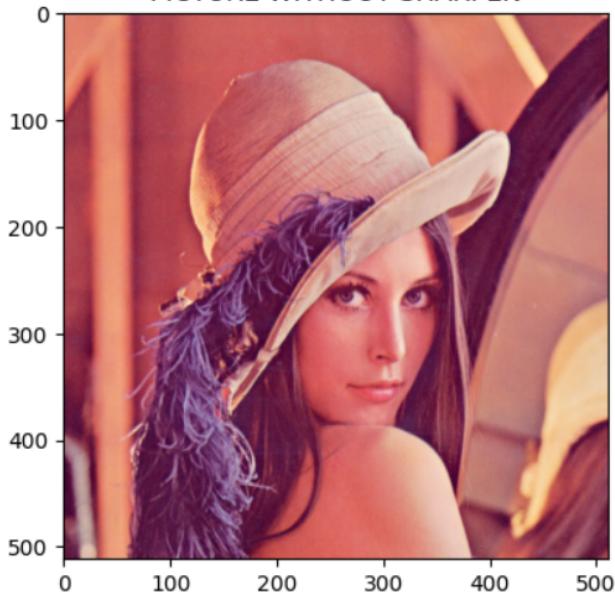
Option: 6 - Blur picture



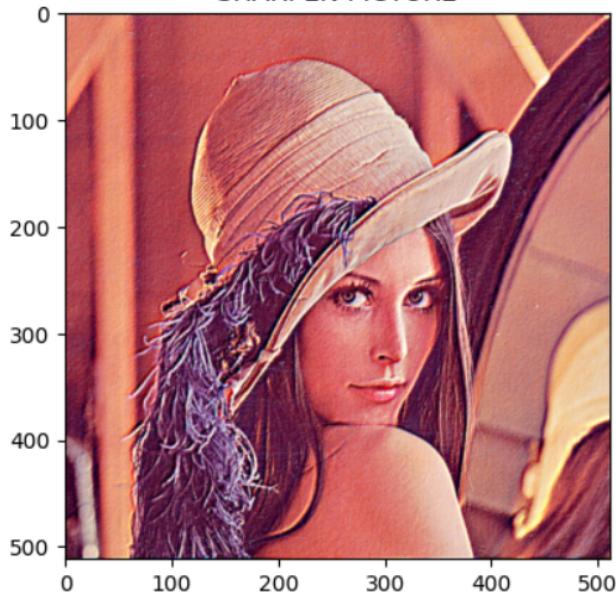
Clipping input data to the valid range for imshow with RGB data ([0..1] for floats or [0..255] for integers).
Your picture Lenna.png has been processed as Blur picture in 1.006216049194336 s

Option: 7 - Sharpen picture

PICTURE WITHOUT SHARPEN



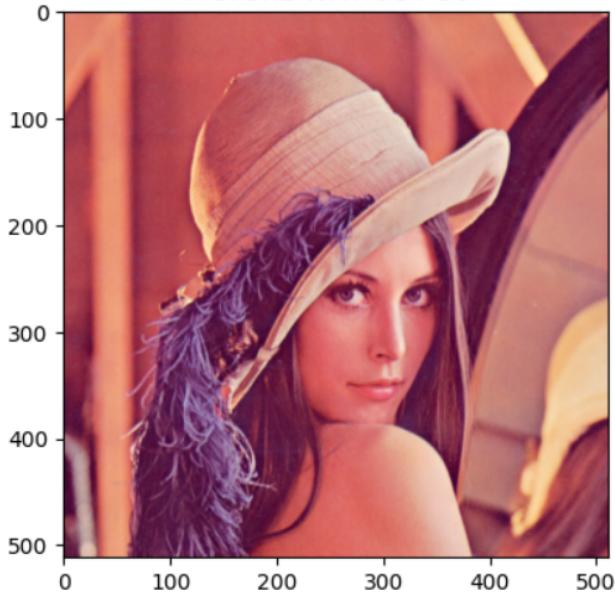
SHARPEN PICTURE



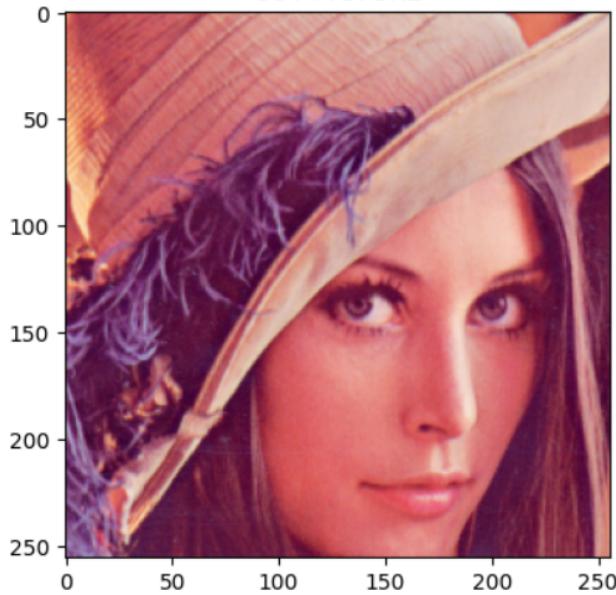
Your picture Lenna.png has been processed as Sharpen picture in 0.8104169368743896 s

Option: 8 - Cut picture

PICTURE WITHOUT CUT

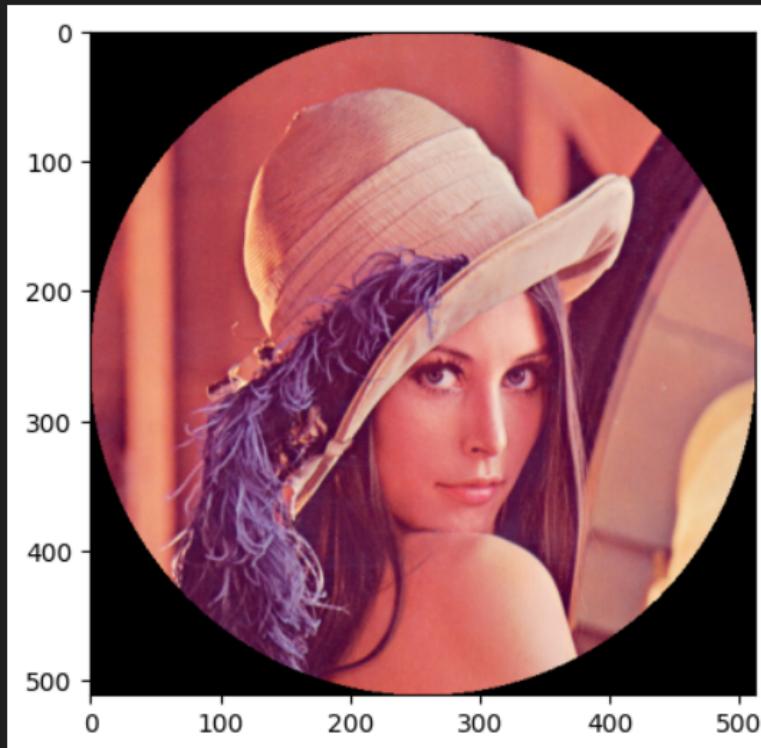


CUT PICTURE



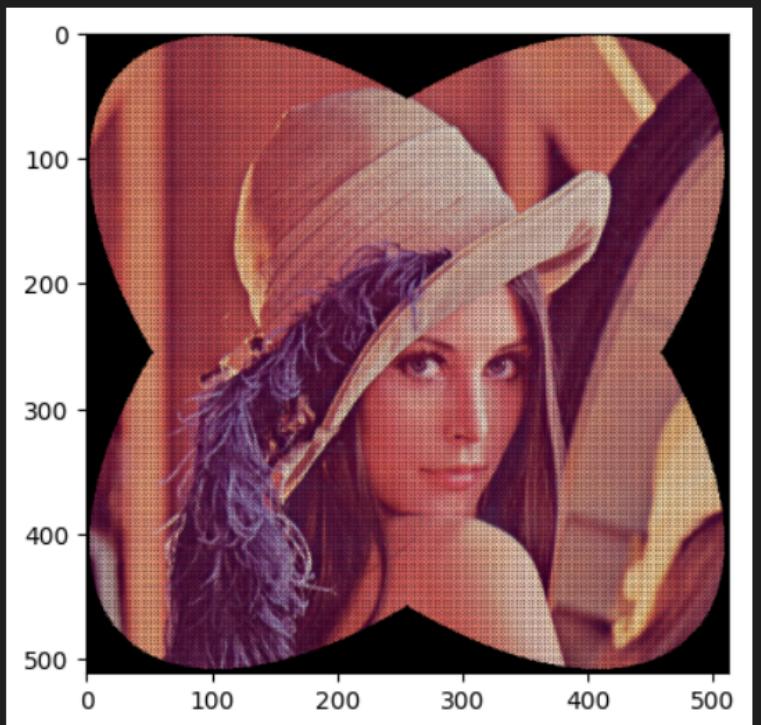
Your picture Lenna.png has been processed as Cut picture in 0.6058714389801025 s

Option: 9 - Circle Frame picture



Your picture Lenna.png has been processed as Circle Frame picture in 0.36357784271240234 s

Option: 10 - Ellipse Frame picture



Your picture Lenna.png has been processed as Ellipse Frame picture in 0.33464908599853516 s

CHƯƠNG III: TÀI LIỆU THAM KHẢO

1. [Image Processing without OpenCV | Python - GeeksforGeeks](#)
2. [Kernel \(image processing\) - Wikipedia](#)
3. [\[Example code\]-Processing an image to sepia tone in Python](#)
4. [Display an image with Python - matplotlib](#)
5. [How to show a PIL Image in Jupyter Notebook | bobbyhadz](#)