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

TGMT01

BÁO CÁO

Xử Lý Ảnh Số và Video Số

LAB 01

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

TGMT01

BÁO CÁO
ĐỒ ÁN
XỬ LÍ ẢNH VÀ VIDEO

Giảng Viên Hướng Dẫn

Phạm Minh Hoàng

Nguyễn Mạnh Hùng

Lý Quốc Ngọc

Sinh viên: Phan Hữu Đoàn Anh - 20127110

MỤC LỤC

1. THÔNG TIN CÁ NHÂN

MSSV	Họ và Tên
20127110	Phan Hữu Đoàn Anh

2. CÁC CHỨC NĂNG ĐÃ HOÀN THÀNH

Color Transformations Geometric Transformations	**Mức độ hoàn thành**
Contrast modification	100%
Brightness modification	100%
Brightness+ Contrast modification	100%
Logarithmic mapping	100%
Exponential mapping	100%
Histogram Equalization	100%
Bilinear transform	0%
Affine transform	0%
Brightness interpolation	0%

Image smoothing	**Mức độ hoàn thành**
Averaging filter	100%
Median filter	100%
Gaussian filter	100%

Edge Detection	**Mức độ hoàn thành**
Gradient operator	100%
Laplace operator	100%
Laplace of Gaussian	0%
Canny method	0%

3. CÁC HÀM CÓ TRONG CHƯƠNG TRÌNH

1 Contrast modification

- Input:
    - Image
- Công thức:

Contrast modification

$$g(x,y) = a.f(x,y)$$

- Mô tả:

Vì đây là ảnh RGB nên từng pixel có 3 giá trị. Nên ta biến ảnh thành mảng 2 chiều và nhân mảng với số a và biến ảnh mảng 3 chiều lại.

Hàm numpy.clip() của thư viện numpy để giới hạn giá trị sau khi cộng/trừ vào pixel. Vì RGB hiển thị màu trong khoảng 0 đến 255

2 Brightness modification

- Input:
    - Image
- Công thức:

Brightness modification

$$g(x,y) = f(x,y) + b$$

- Mô tả:

Vì đây là ảnh RGB nên từng pixel có 3 giá trị. Nên ta biến ảnh thành mảng 2 chiều và cộng mảng với số b và biến ảnh mảng 3 chiều lại.

Hàm numpy.clip() của thư viện numpy để giới hạn giá trị sau khi cộng/trừ vào pixel. Vì RGB hiển thị màu trong khoảng 0 đến 255

3 Brightness+ Contrast modification

- Input:
    - Image
- Công thức:

Brightness+ Contrast modification

$$g(x,y) = a.f(x,y) + b$$

- Mô tả:

Vì đây là ảnh RGB nên từng pixel có 3 giá trị. Nên ta biến ảnh thành mảng 2 chiều.

Sau đó cộng cho b để tăng độ sáng và nhận cho a để tăng độ tương phản

Hàm numpy.clip() của thư viện numpy để giới hạn giá trị sau khi cộng/trừ vào pixel. Vì RGB hiển thị màu trong khoảng 0 đến 255

Sau đó biến mảng lại thành 3 giá trị

4 Logarithmic mapping

- Input:
    - Image
- Công thức:

Logarithmic mapping function

$$g(x,y) = c\log f(x,y)$$

- Mô tả:

Vì đây là ảnh RGB nên từng pixel có 3 giá trị. Nên ta biến ảnh thành mảng 2 chiều.

Sau đó tạo vòng lập và nhân tường log(i,j) với c

Hàm numpy.clip() của thư viện numpy để giới hạn giá trị sau khi cộng/trừ vào pixel. Vì RGB hiển thị màu trong khoảng 0 đến 255

Sau đó biến mảng lại thành 3 giá trị

5 Exponential mapping :

- Input:
    - Image
- Công thức:

Exponential mapping function

$$g(x,y) = e^{f(x,y)}$$

- Mô tả:

Vì đây là ảnh RGB nên từng pixel có 3 giá trị. Nên ta biến ảnh thành mảng 2 chiều.

Chạy vòng lặp, với mỗi f(I,j) t sẽ cho bằng với e mũ f(i,j)

Dùng hàm numpy.clip() của thư viện numpy để giới hạn giá trị sau khi cộng/trừ vào pixel. Vì RGB hiển thị màu trong khoảng 0 đến 255

Sau đó gán lại bằng ảnh có 3 giá trị

6 Histogram Equalization:

- Input:
    - Image
- Mô tả:
    1. Thống kê số lượng pixel cho từng mức sáng
    2. Tính hàm tích luỹ bằng "calc_hist" cho từng mức sáng theo công thức dưới và Z(i) là tổng số các pixel có giá trị <= i

$$Z(i) = \sum_{j=0}^{i} H(i)$$

    3. Sau khí tính hàm tích luỹ thì sẽ tiến hành cân bằng tại một điểm i với hàm equal_hist và công thức dưới đây. Công thức ngày có tác dụng làm giảm khoảng cách phân bố dày đặc pixel và co khoảng phân bố thưa pixel.

$$K(i) = \frac{Z(i) - min(Z)}{max(Z) - min(Z)} * 255$$

    4. Cân bằng histogram là cân bằng lại mức cường độ sáng, tức chỉ là 1 trong 3 chanel của hệ màu HSV. Vậy nên sẽ sử dụng ảnh xám.

6 Gaussian filter:

- Input: Image.
- Mô tả:

  - Vì hàm tính tích chập (convolution) chỉ hoạt động trên ảnh có một channel nên chúng ta phải chuyển đổi ảnh sang xám. (Hàm convolution được lấy ở github)
  - Đầu tiên chúng ta tính h(I,j) theo công thức dưới đây, nhưng vì ảnh chỉ có một kênh màu nên i và j sẽ được gộp lại.

$$h(i,j) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{i^2+j^2}{2\sigma^2}}$$

  - Sau đó chúng ta tạo ra Gaussain kernel bằng hàm gaussian_kernel

  - Tiếp theo chúng ta sẽ nhân tích chập Gaussian Kernel này với ảnh Img thông qua convolution.

7 Averaging filter

- Input:
  - image
- Mô tả:
  - Chúng ta sẽ tạo Kernel riêng cho Averaging filter bằng công thức

$$h = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

  - Sau đó chúng ta sử dụng hàm convolution để nhân tích chập với ảnh. Vì hàm convolution chỉ có 1 kênh nên chúng ta phải đổi sang ảnh xám.

8 Median filter

- Input:

- image

- Mô tả:
  Hàm Median_filter chạy vòng lặp duyệt qua từng pixel. Với mỗi pixel và vị trí của nó, ta truyền vào hàm the_box để lấy các pixel lân cận rồi tiến hành sắp xếp bằng hàm nummpy.sort() sau đó trả về trung vị.

9 Edge Detection:

Giải thuật chung:

- Chúng ta đều lấy đạo hàm theo x và y của f(x,y) bằng cách nhân tích chập với Kernel riêng của thuật toán là Wx và Wy. Sau đó gộp chúng lại thông qua hàm gradient_magnitude.

9.1 Pixel Difference:

Công thức Kernel:

$$W_x = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & -1 \\ 0 & 0 & 0 \end{bmatrix}, W_y = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

9.2 Separeted Pixel difference:

Công thức kernel:

$$W_x = \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & -1 \\ 0 & 0 & 0 \end{bmatrix}, W_y = \begin{bmatrix} 0 & -1 & 0 \\ 0 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix}$$

9.3 Robert Operator:

Công thức Kernel:

$$W_x = \begin{bmatrix} 0 & 0 & -1 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}, W_y = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

9.4 Prewitt Operator

Công thức Kernel:

$$W_x = \frac{1}{3}\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}, W_y = \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

9.5 Sobel Operator:

Công thức Kernel:

$$W_x = \frac{1}{4}\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}, W_y = \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

9.6 Frei-chen Operator:

Công thức Kernel:

$$W_x = \frac{1}{2+\sqrt{2}} \begin{bmatrix} 1 & 0 & -1 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 1 & 0 & -1 \end{bmatrix},$$

$$W_y = \frac{1}{2+\sqrt{2}} \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix}$$

Associate Prof. Lý Quốc Ngọc

9.7 Laplace Operator:

- Công thức laplace xác định đạo hàm bậc 2 của f thì sẽ sấp xỉ bằng f tích chập với Laplace Kernel

Công thức Laplace 1:

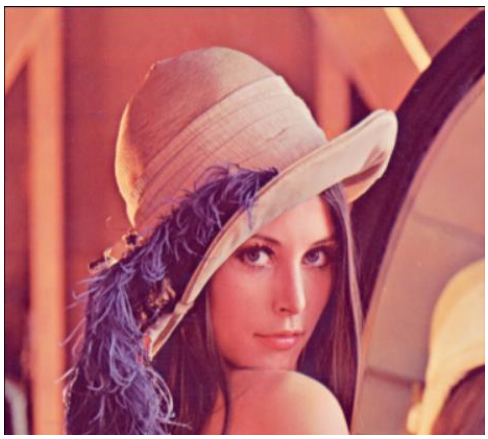
$$Laplace = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Công thức Laplace 2:

$$Laplace = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

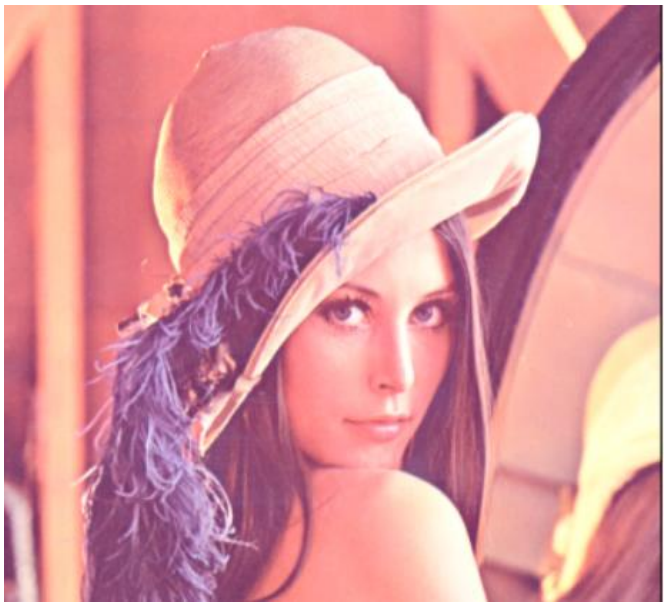
4 HÌNH ẢNH KẾT QUẢ

Ảnh gốc:

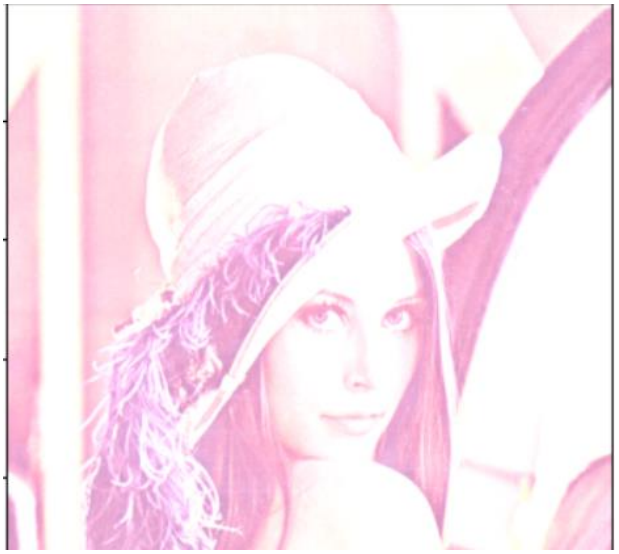


Ảnh gray.png:

1. Brightness Modification:
   Với bias = 50



Với bias = 150



2. Contrast Modification:

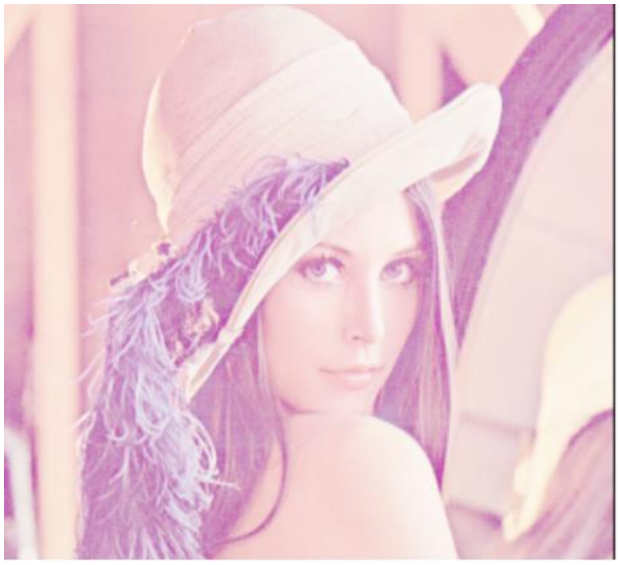
   Với alpha = 3

Với alpha = 5

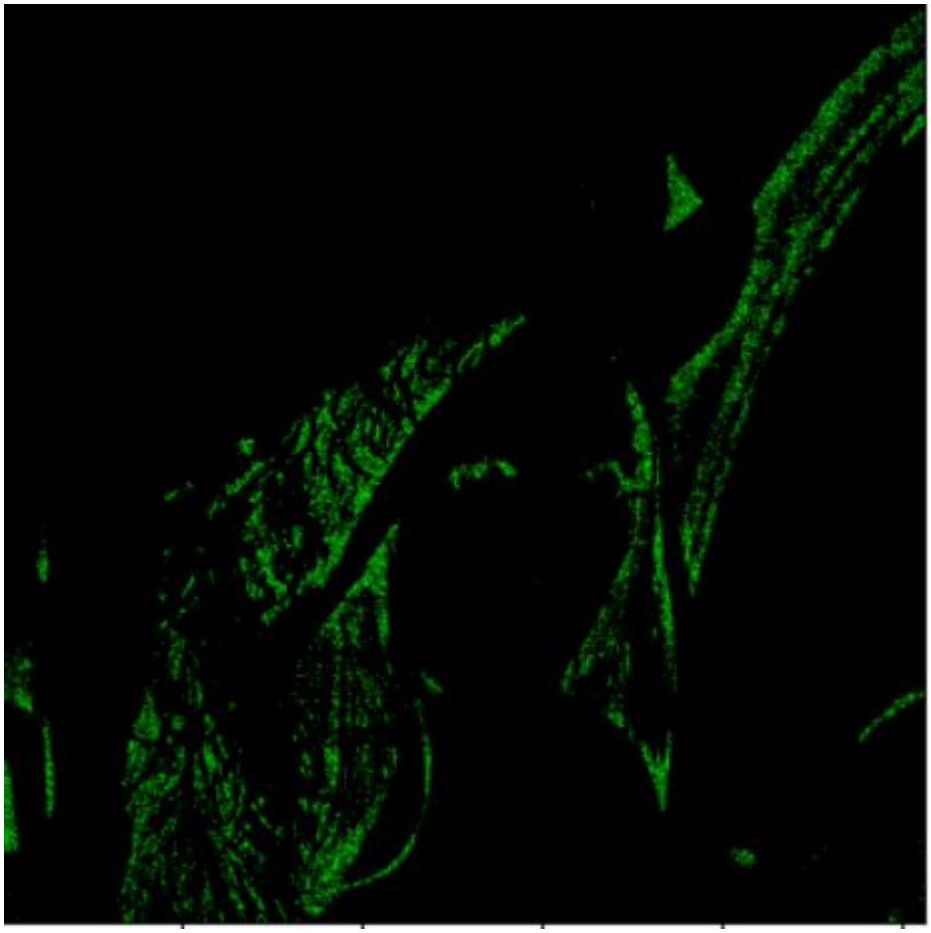
3. Brightness + Contrast Modification:

Với apha = 3 và bias = 50

4. Logarithm Modification:

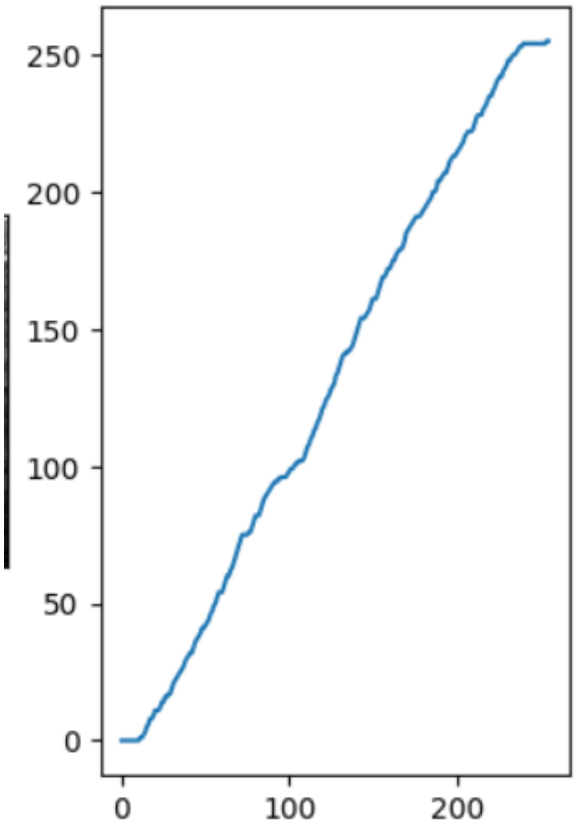


5. Exponential_mapping:



6. Histogram Equalization:

Bảng cân bằng Histogram các pixel trên ảnh.



7. Gaussian Filter:

8. Average Filter:

9. Median Filter:

10. Pixel Difference:

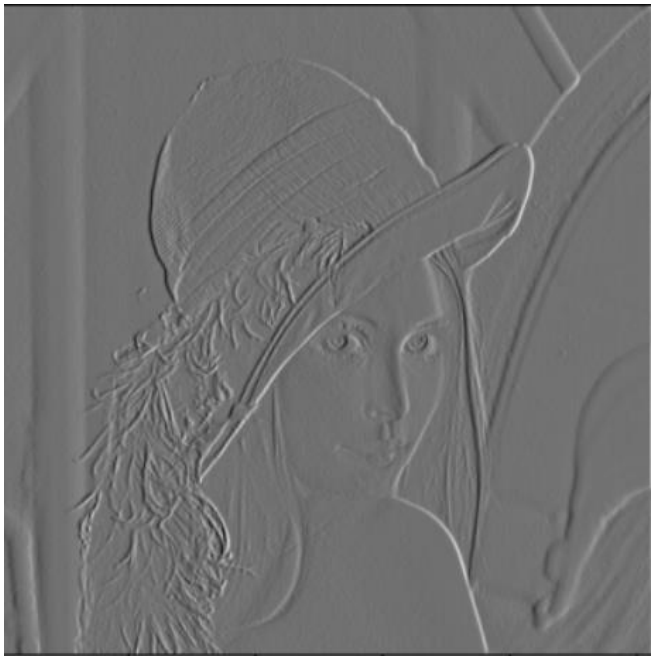
Fx:

Fy:

Kết quả:

11. Separate Pixel Difference:

Fx:

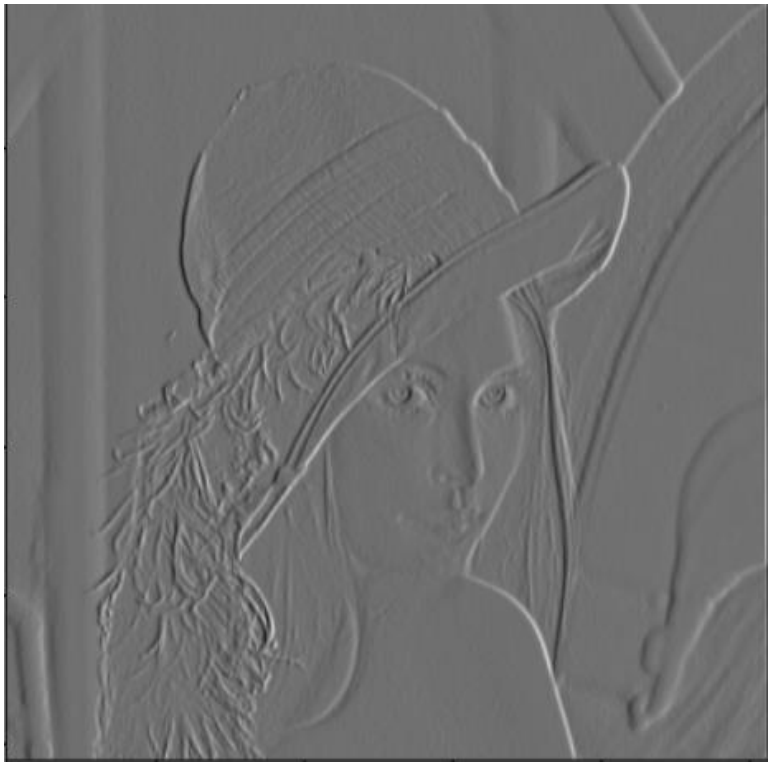


Fy:

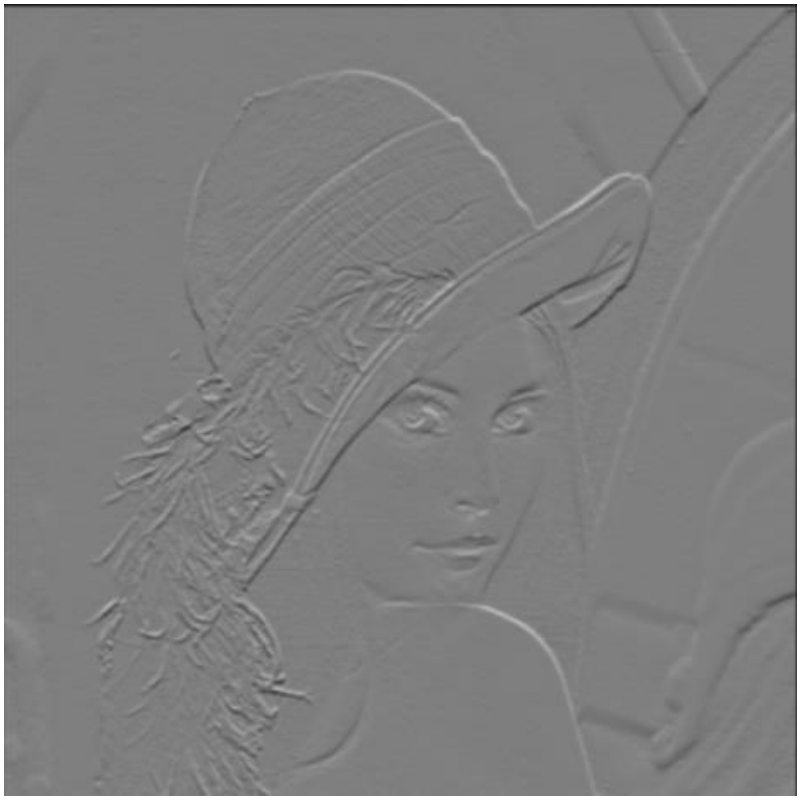
Result:

12. Prewitt operator:

Fx:



Fy:



Result:

13. Robert operator:

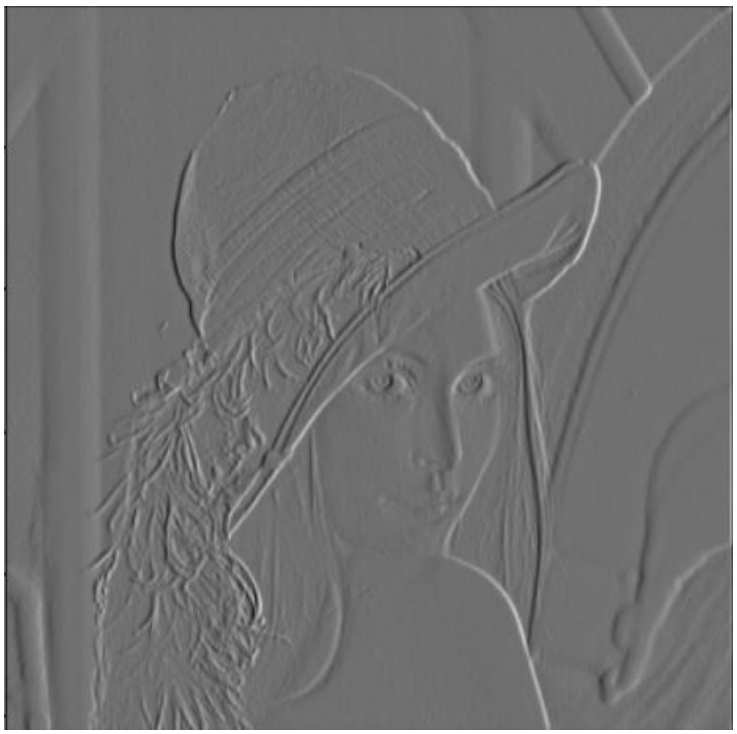
Fx:

Fy:

Result:

14. Sobel Operator:

Fx:



Fy:

Result:

15. Frei-chen Operator:

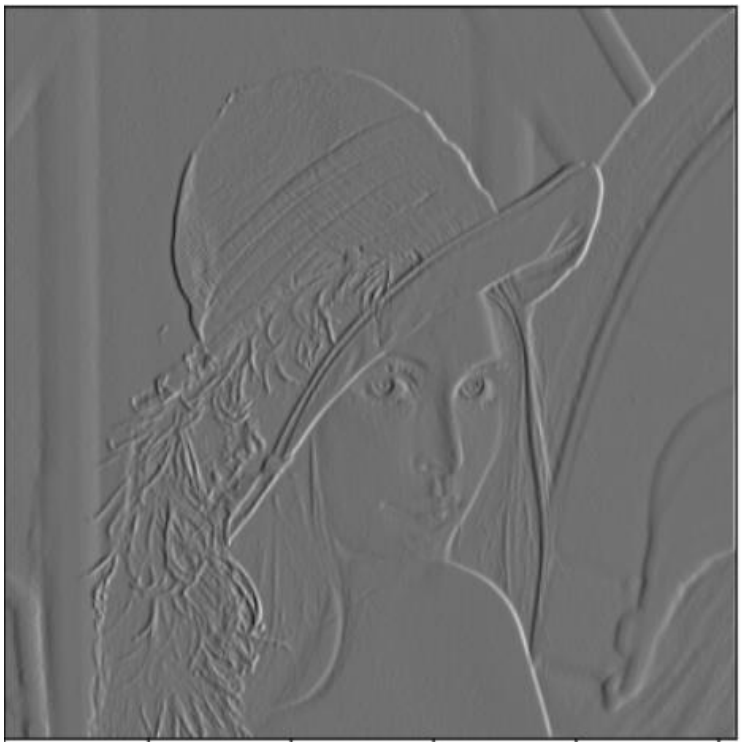
Fx:

Fy:

Result:

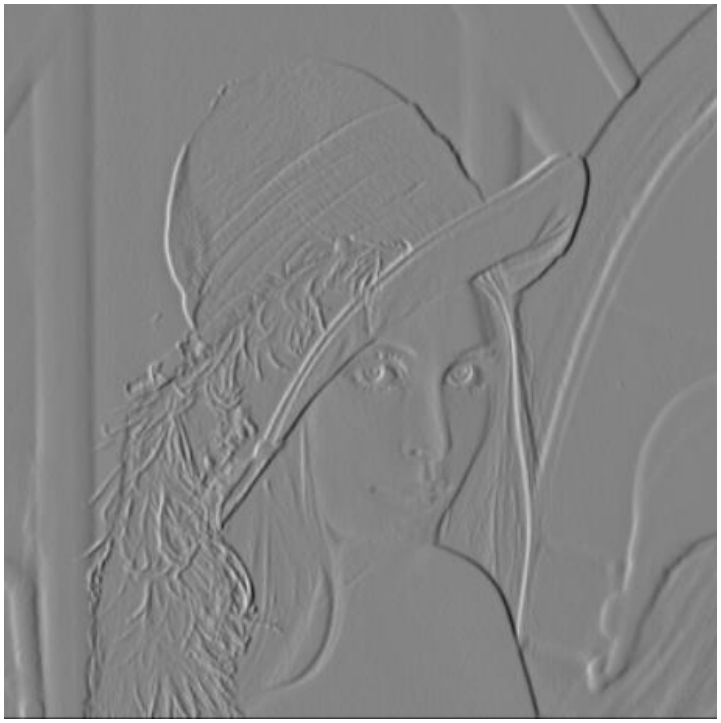
16. Laplace:

Laplace1:



Laplace 2:

OpenCV Edge Detection:
Đạo hàm theo x



5.Đánh giá

Các thuật toán Edge Detection đã được khai báo và chạy đúng với khai báo và giải thuật nhưng hàm Edge Detection của OpenCV có runtimes nhanh hơn các thuật toán Edge Detection đã khai báo.

6.TÀI LIỆU THAM KHẢO

https://github.com/adeveloperdiary/blog/tree/master/Computer_Vision

https://viblo.asia/p/xu-li-anh-thuat-toan-can-bang-histogram-anh-GrLZDOogKk0

Slide bài giảng của thầy Lý Quốc Ngọc (Các công thức)

Toán Ứng Dụng Thống Kê của cô Phan Thị Phương Uyên (Median Smoothing)