ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH

TRƯỜNG ĐẠI HỌC KHOA HỌC TỰ NHIÊN

KHOA CÔNG NGHỆ THÔNG TIN

BÁO CÁO

Lab 01

Xử lý ảnh số và video số 20_23

Giảng viên hướng dẫn – Nguyễn Mạnh Hùng

Thành phố Hồ Chí Minh - 2022

MỤC LỤC

THÔNG TIN SINH VIÊN

MSSV	Họ Tên	Email	Ghi chú
20120201	Phạm Gia Thông	20120201@student.hcmus.edu.vn	

1. Các thao tác xử lý cơ bản trên hình ảnh với thư viện OpenCV

a) Setup

Cài đặt thư viện opencv2

#pip install opencv-python

Cài đặt thư viện numpy

#pip install numpy

Cài đặt thư viện scipy

#pip install scipy

Cài đặt thư viện matplotlib

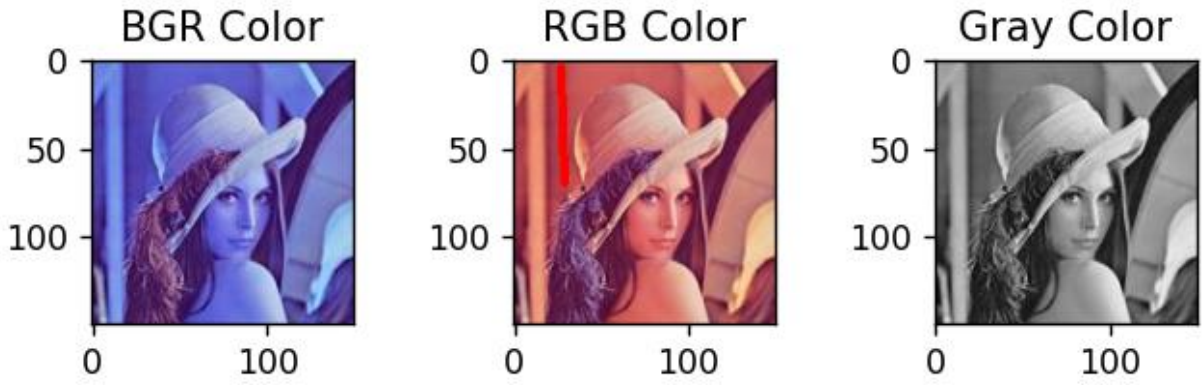
#pip install matplotlib

b) Read and display image

Dùng ảnh Lenna.jpg

Đầu ra của image lúc này sẽ là 1 ma trận ba chiều nhưng bởi vì cv2 mặc định đọc ảnh màu theo thứ là BGR (là blue, green và red) nên khi đọc ra sẽ ra theo thứ tự màu BGR.

Muốn ra đúng các kênh màu theo thứ tự, ta phải convert lại màu của ảnh thành RGB với hàm thư viện opencv.



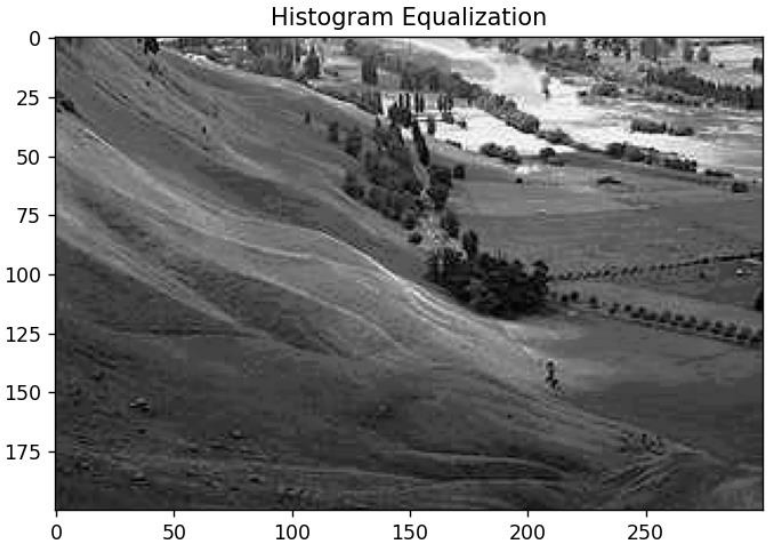
Ngoài ra còn có các chức năng như là đọc thông tin, thông số của bức ảnh, vẽ một đường thẳng, đường tròn lên bức ảnh, cắt một phần của bức ảnh ra và đổi màu phần bị cắt đó.

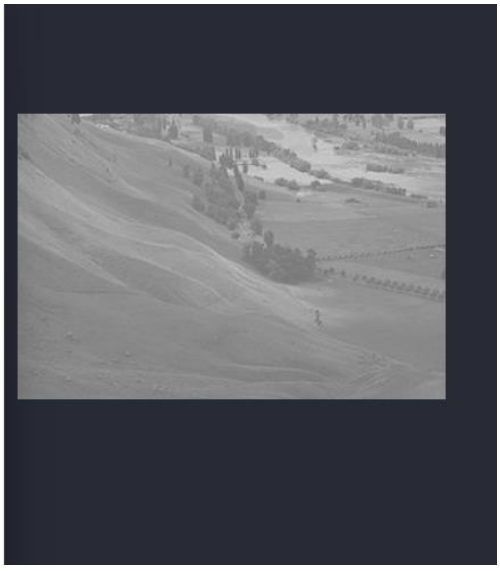
2. Giải thuật

a) Biến đổi tuyến tính và phi tuyến

Thực hiện được hàm cân bằng lược đồ xám (Histogram Equalization)

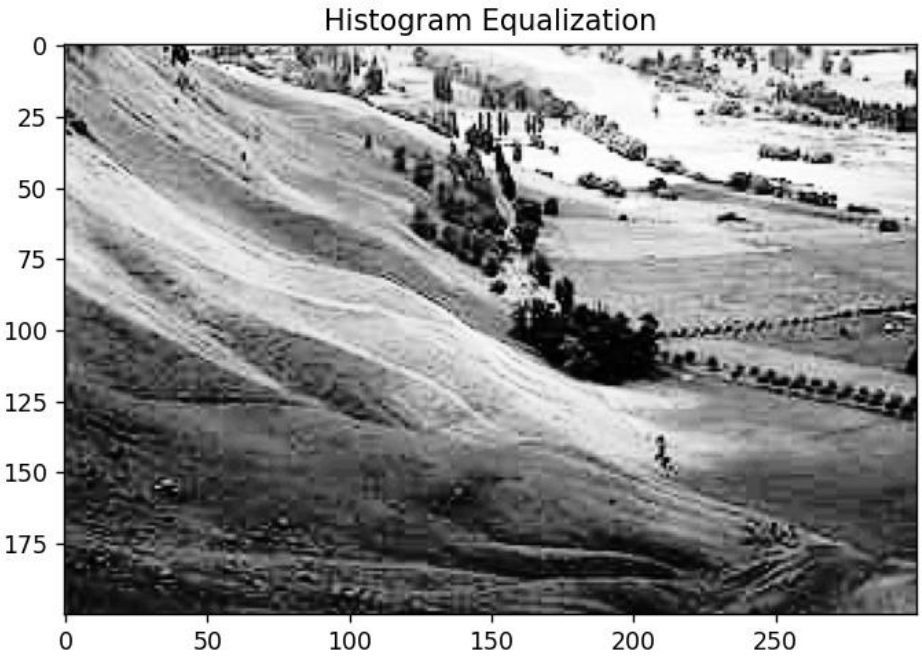
Dùng ảnh Hill.jpg





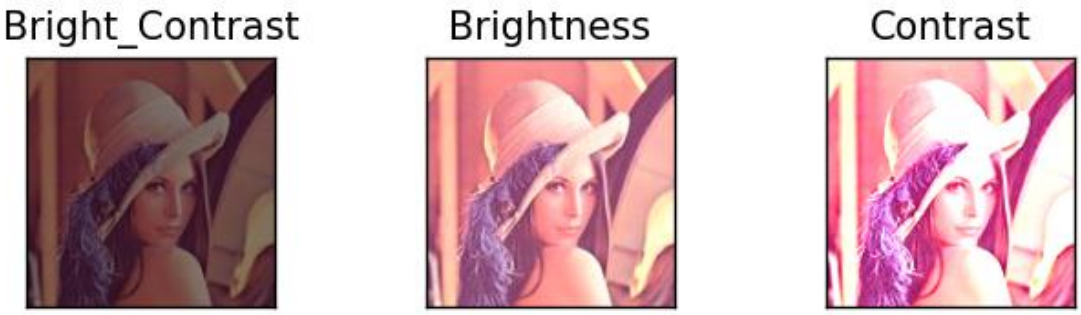
Thời gian thực hiện lâu hơn khi dùng hàm có sẵn, xấp xỉ: ~0.225[sec]

Khi thực hiện bằng thư viện có sẵn của OpenCV (cv2.equalizeHist()), ta cho được một ảnh đầu ra có thời gian thực hiện nhanh hơn, tối ưu và không phải tốn nhiều bước để cài đặt các thuật toán phức tạp. Ảnh đầu ra cũng có độ phủ màu đậm và cường độ hình ảnh lớn hơn.



Thời gian thực hiện xấp xỉ: ~0.126 – 0.164[sec]

Thực hiện được phép biến đổi tuyến tính



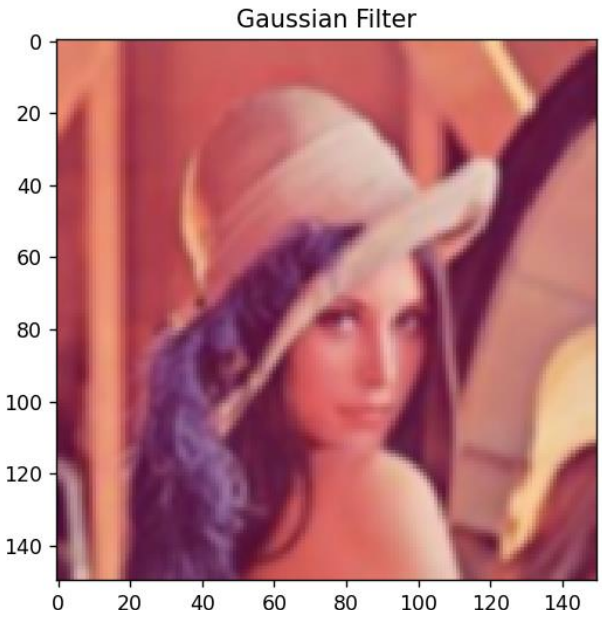
Thời gian thực hiện xấp xỉ: ~0.1307[sec]

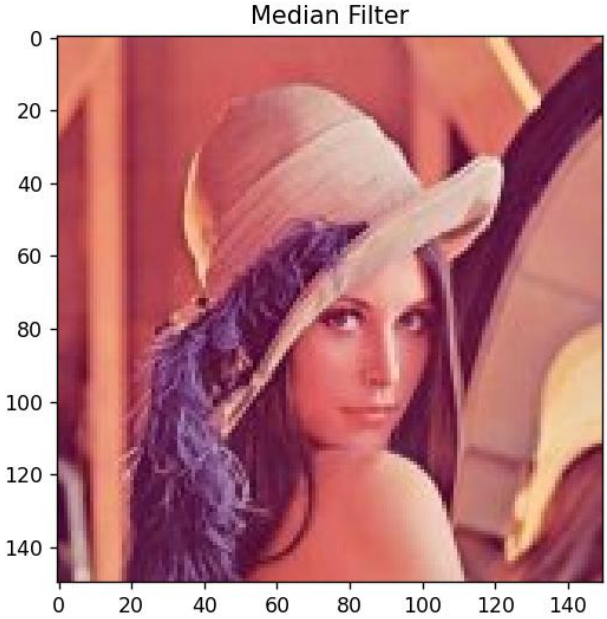
Chưa thực hiện được phép biến đổi phi tuyến và đặc tả lược đồ xám.

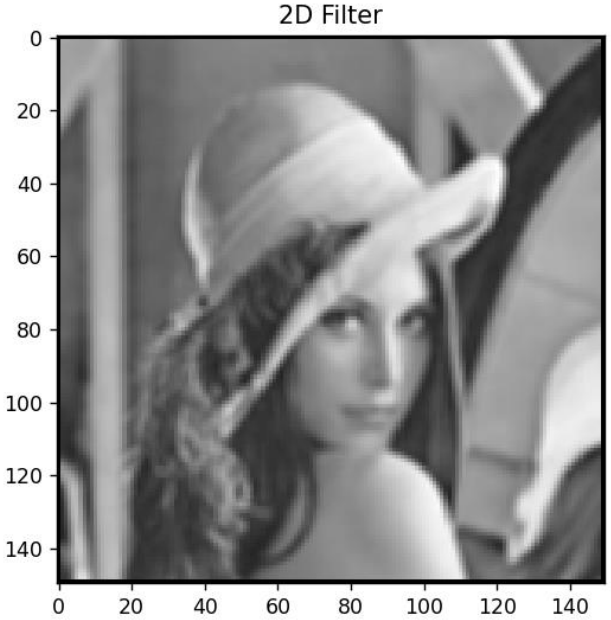
b) Biến đổi vị trí điểm ảnh và phép nội suy giá trị màu của điểm ảnh (chưa hoàn thành được)

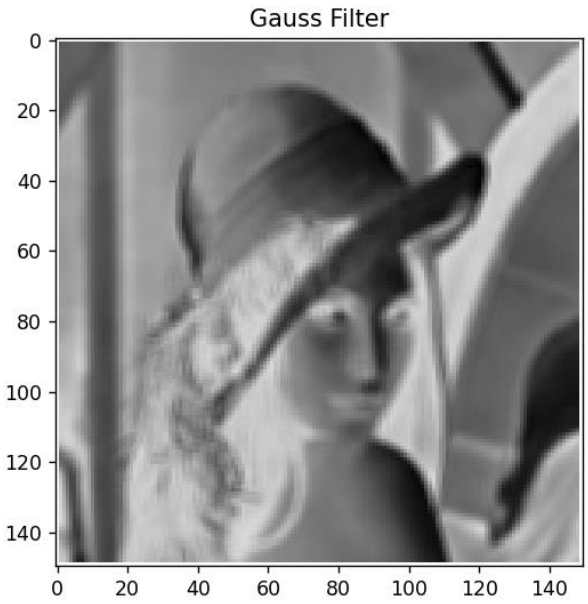
c) Làm trơn ảnh

Thực hiện được các phép làm trơn ảnh với toán tử Gaussian, toán tử Trung vị. Với toán tử Trung vị có thể bị lỗi



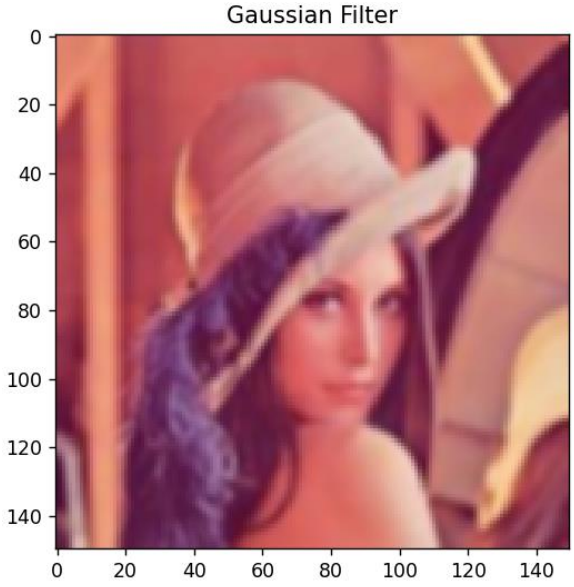


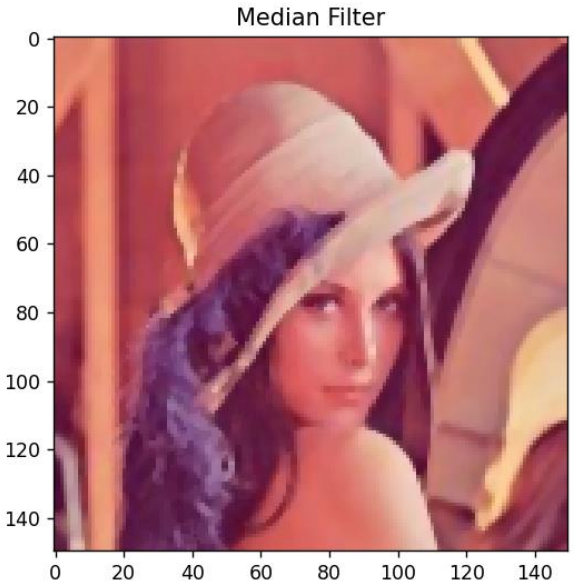




- Thời gian toán tử Gaussian thực hiện xấp xỉ: ~0.7511[sec]
- Thời gian toán tử Trung vị thực hiện xấp xỉ: ~0.6935[sec]
- Thời gian toán tử Trung bình thực hiện xấp xỉ: ~0.1971[sec]
- Thời gian toán tử Gauss thực hiện xấp xỉ: ~0.3137[sec]

Khi thực hiện bằng các hàm có sẵn trong thư viện openCV, ta có thể thấy thời gian thực hiện nhanh hơn, tối ưu, và độ chính xác cao hơn.

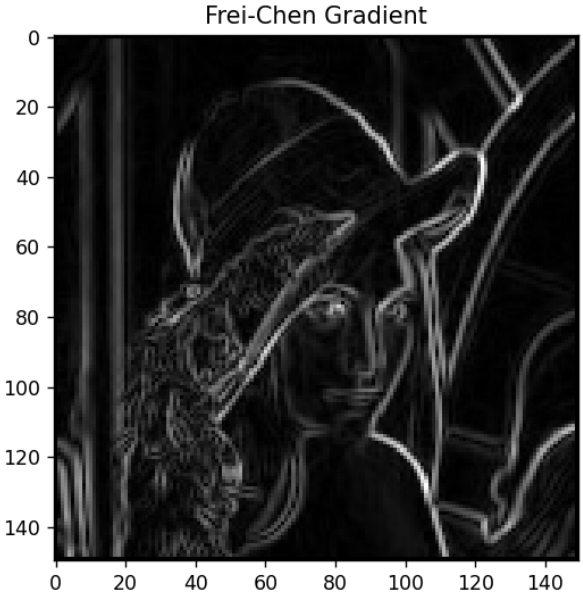


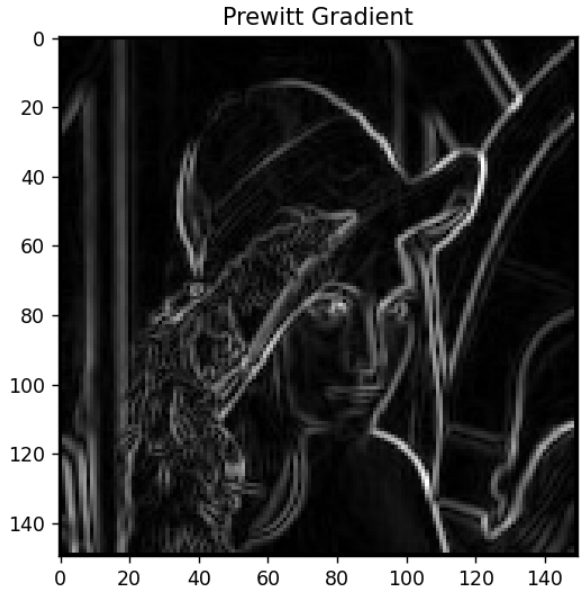


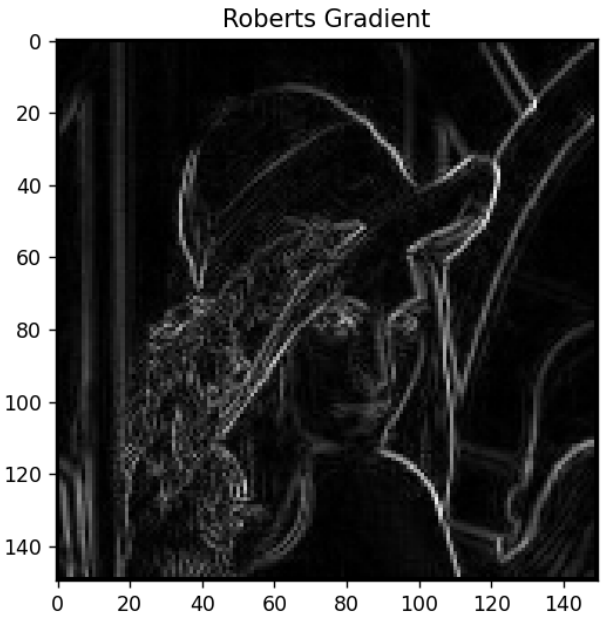
- Thời gian toán tử Gaussian thực hiện xấp xỉ: ~0.1263[sec]
- Thời gian toán tử Trung vị thực hiện xấp xỉ: ~0.1140[sec]

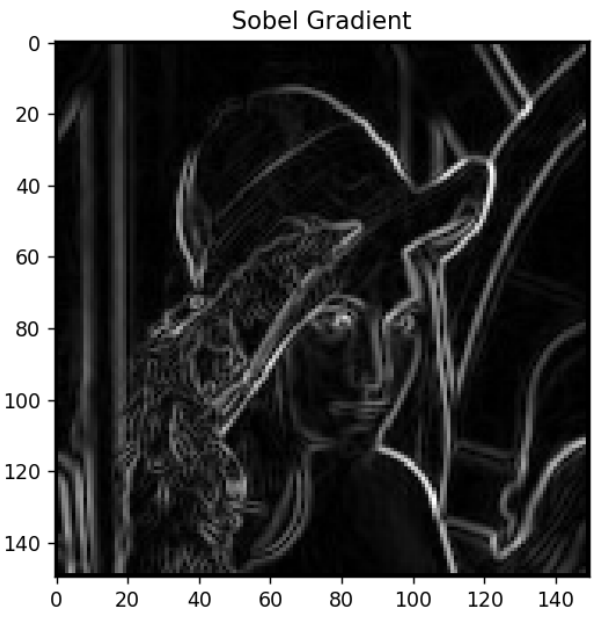
d) Phát hiện biên cạnh

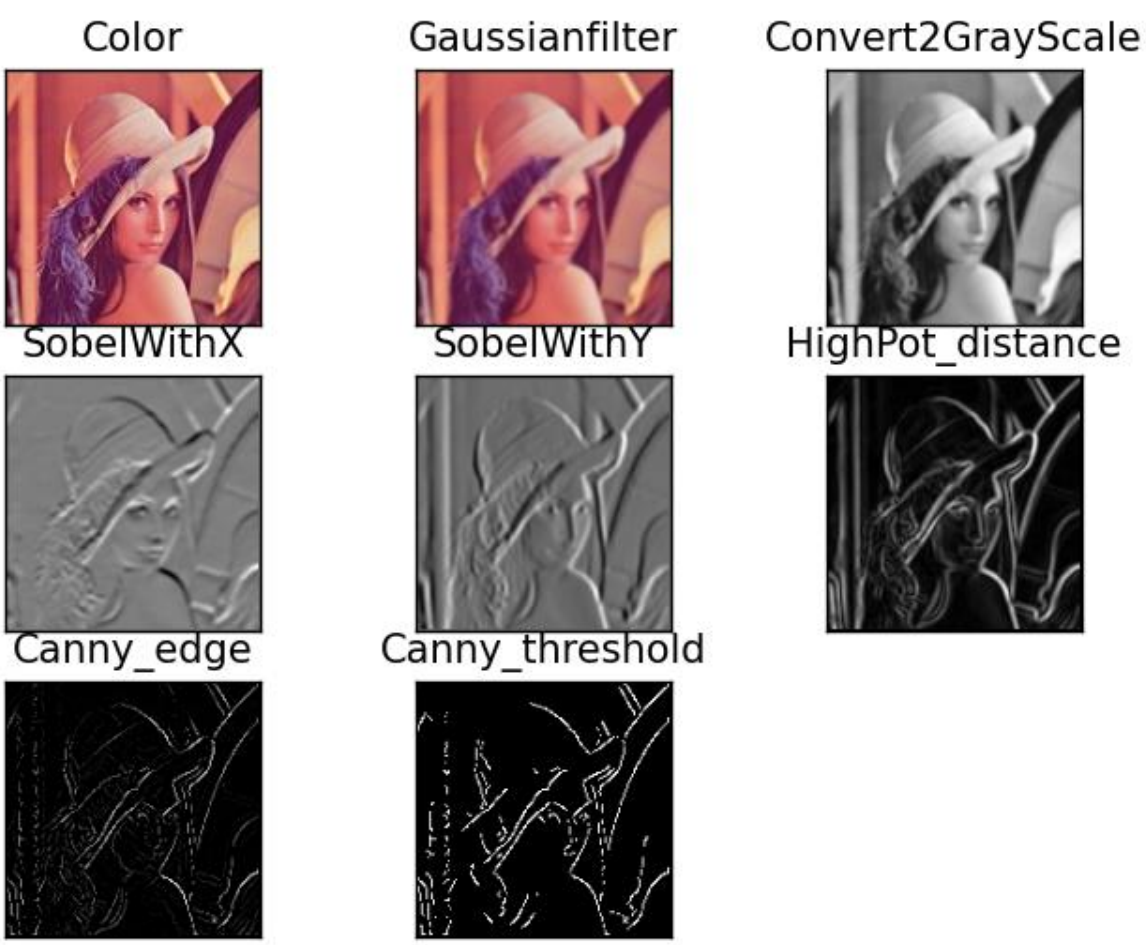
Thực hiện được các phép phát hiện biên cạnh với toán tử Gradient(Roberts, Prewitt, Sobel, Frei-Chen), toán tử Laplace, phương pháp Canny. Với toán tử Laplace of Gaussian, Hough phát hiện đường thẳng, đường tròn, đường cong chưa hoàn thành được.





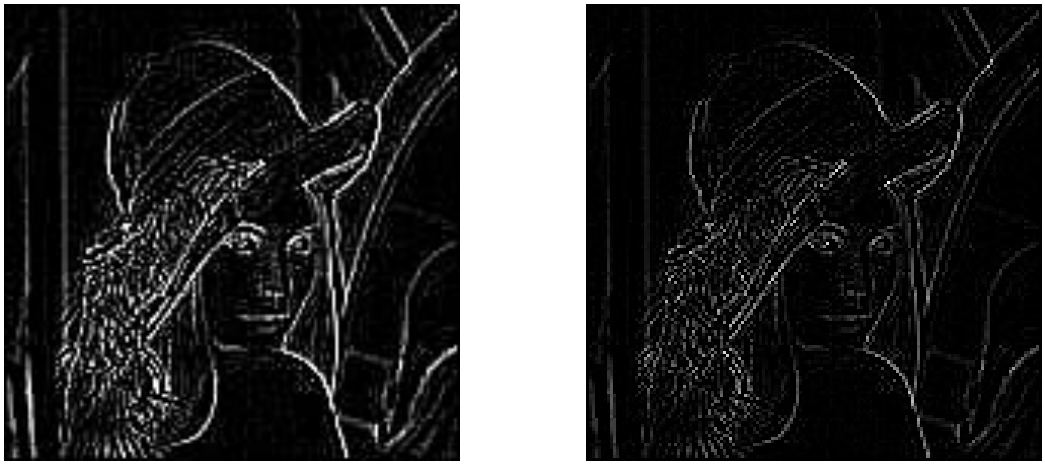

Roberts Gradient
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


Sobel Gradient
0
20
40
60
80
100
120
140
0
20
40
60
80
100
120
140


Color
Gaussianfilter
Convert2GrayScale
SobelWithX
SobelWithY
HighPot_distance
Canny_edge
Canny_threshold

Phương pháp Canny

Toán tử Laplace



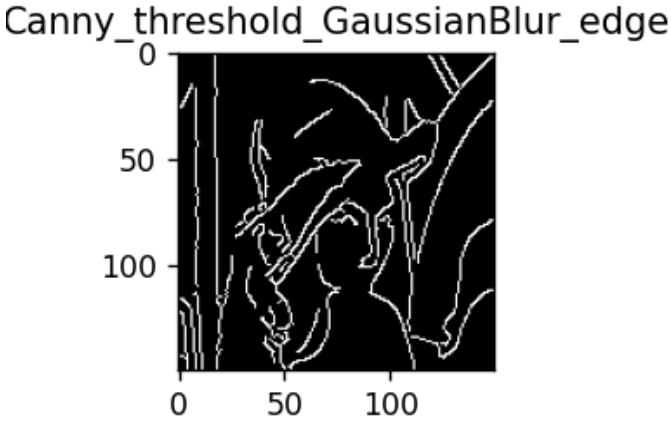
- Thời gian toán tử Gradient(Sobel) thực hiện xấp xỉ: ~0.4416[sec]
- Thời gian toán tử Gradient(Roberts) thực hiện xấp xỉ: ~0.2979[sec]
- Thời gian toán tử Gradient(Prewitt) thực hiện xấp xỉ: ~0.2979[sec]
- Thời gian toán tử Gradient(Frei-Chen) thực hiện xấp xỉ: ~0.2959[sec]
- Thời gian toán tử Laplace thực hiện xấp xỉ: ~0.3062[sec]
- Thời gian phương pháp Canny thực hiện xấp xỉ: ~1.0731[sec]

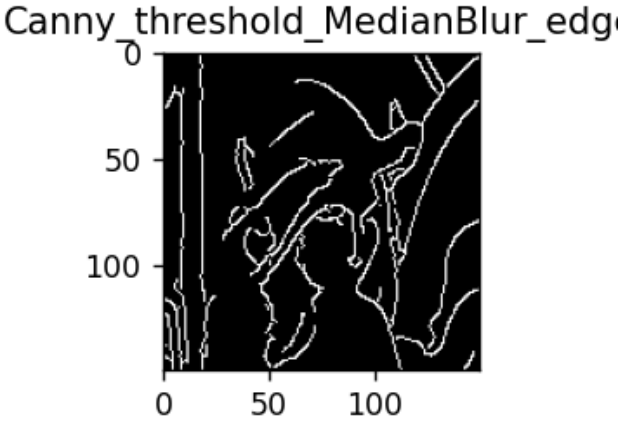
Khi thực hiện bằng các hàm có sẵn trong thư viện OpenCV:

Toán tử Gradient(Sobel)

Thời gian thực hiện xấp xỉ: ~0.00336 – 0.00825[sec]

Phương pháp Canny





Thời gian thực hiện xấp xỉ: ~0.1266 – 0.1808[sec]

Nhìn chung, khi sử dụng các hàm thư viện có sẵn, chúng ta sẽ có được một ảnh đầu ra với thời gian thực hiện nhanh hơn, chính xác và tối ưu hơn, giảm được thời gian tính toán ở các bước giải thuật phức tạp, vì các hàm thư viện này đã được nghiên cứu và thử nghiệm nhiều lần trước khi công bố. Nhưng hạn chế ở việc dùng thư viện là có một số hàm không có, nên nếu bạn muốn dùng chức năng của hàm đó, bạn bắt buộc phải giải thuật ra từng bước.

TÀI LIỆU THAM KHẢO

Danh mục tài liệu tham khảo:

[1] *https://docs.opencv.org/master/d6/d10/tutorial_py_houghlines.html*
[2]*https://docs.opencv.org/3.4/d8/d6a/group__imgcodecs__flags.html#ga61d9b0126a3e57d9277ac48327799c80*
[3]*https://docs.opencv.org/3.0beta/doc/py_tutorials/py_gui/py_drawing_functions/py_drawing_functions.html*
[4] *https://opencvexamples.blogspot.com/2013/10/sobel-edge-detection.html*
[5]*http://docs.opencv.org/3.0beta/doc/py_tutorials/py_imgproc/py_gradients/py_gradients.html*