BỘ GIAO THÔNG VẬN TẢI

**TRƯỜNG ĐẠI HỌC HÀNG HẢI VIỆT NAM**

****

**KHOA CÔNG NGHỆ THÔNG TIN**

**SVTH : Đỗ Văn Quang - 87085**

**BÁO CÁO BÀI TẬP LỚN MÔN HỌC**

**XỬ LÝ ẢNH**

**Xây dựng chương trình xử lý ảnh cho phép đọc ảnh từ file**

**Giảng viên hướng dẫn:**

**TS. Nguyễn Hữu Tuân**

**Mục Lục**

[DANH MỤC HÌNH ẢNH 3](#_Toc131680539)

[LỜI NÓI ĐẦU 4](#_Toc131680540)

[CHƯƠNG 1 : GIỚI THIỆU ĐỀ TÀI 5](#_Toc131680541)

[1 Đặt vấn đề 5](#_Toc131680542)

[2.Mục tiêu nghiên cứu 5](#_Toc131680543)

[3.Ứng dụng 6](#_Toc131680544)

[CHƯƠNG 2 : Cơ sở lý thuyết 6](#_Toc131680545)

[1.Tổng quan về xử lý ảnh 6](#_Toc131680546)

[2.Giới thiệu về Python 7](#_Toc131680547)

[3.Các thư viện của Python được dùng trong chương trình 8](#_Toc131680548)

[CHƯƠNG 3 : Chương trình 9](#_Toc131680549)

[1.Các thao tác xử lý dựa trên điểm ảnh: 9](#_Toc131680550)

* [Cân bằng histogram. 9](#_Toc131680551)
* [Tách ngưỡng (THRESHOLDING) 11](#_Toc131680552)
* [Lấy ảnh âm bản (NEGATIVE IMAGE) 12](#_Toc131680553)
* [Biến đổi logarit. 13](#_Toc131680554)
* [Tăng độ tương phản. 13](#_Toc131680555)

[2.Các phép lọc không gian: 14](#_Toc131680556)

* [Mean filter 14](#_Toc131680557)
* [Median filter. 15](#_Toc131680558)
* [Gaussian filter. 16](#_Toc131680559)
* [Linear sharpen. 17](#_Toc131680560)
* [Lấy ảnh gradient. 18](#_Toc131680561)
* [Edge detection. 19](#_Toc131680562)

[3. Các phép lọc ảnh trong miền tần số: 20](#_Toc131680563)

* [Lọc thông thấp. 20](#_Toc131680564)
* [Lọc thông cao. 21](#_Toc131680565)

[4.Thực hiện lọc ảnh bằng bộ lọc Bilateral và NonLocalMeans. 21](#_Toc131680566)

* [Bilateral 21](#_Toc131680567)
* [NonLocalMeans. 22](#_Toc131680568)

[CHƯƠNG 4 : Kết luận 23](#_Toc131680569)

[4.1 Giao diện khi chạy chương trình 23](#_Toc131680570)

[4.2.Ưu và nhược điểm 23](#_Toc131680571)

# DANH MỤC HÌNH ẢNH

[Hình 1.Quá trình xử lý ảnh 7](#_Toc131679510)

[Hình 2. Các bước cơ bản trong một hệ thống xử lý ảnh 7](#_Toc131679511)

[Hình 3.Ảnh minh họa 10](#_Toc131679512)

[Hình 4. Công thức tách ngưỡng 11](#_Toc131679513)

[Hình 5.Công thức lấy ảnh âm bản 12](#_Toc131679514)

[Hình 6. Công thức tăng độ tương phản của ảnh 14](#_Toc131679515)

[Hình7. Lọc trung vị (Nhiễu muối tiêu) 16](#_Toc131679516)

[Hình8. Lọc Gaussian fliter 17](#_Toc131679517)

[Hình9. Công thức lọc Linear sharpe 18](#_Toc131679518)

[Hình10. Công thức lọc thông thấp Butterworth 21](#_Toc131679519)

[Hình11. Công thức lọc thông cao Butterworth 22](#_Toc131679520)

[Hình 12.Giao diện chương trình 24](#_Toc131679521)

# 

# LỜI NÓI ĐẦU

Trong thời đại số hóa ngày nay, ảnh kỹ thuật số trở thành một phương tiện không thể thiếu trong giao tiếp và truyền tải thông tin. Chính vì thế, việc xử lý ảnh số trở nên ngày càng quan trọng trong nhiều lĩnh vực khác nhau như y tế, công nghệ, khoa học, nghệ thuật, v.v.

Để giúp đỡ việc xử lý ảnh số của người dùng, chúng tôi xin giới thiệu một chương trình xử lý ảnh đơn giản. Chương trình cho phép đọc ảnh từ file và thực hiện một số thao tác xử lý ảnh phổ biến như cân bằng histogram, tách ngưỡng, lấy ảnh gradient, v.v. Ngoài ra, chương trình cũng hỗ trợ các phép lọc ảnh trong miền không gian và miền tần số.

Chương trình được xây dựng trên nền tảng Python và sử dụng các thư viện mã nguồn mở như NumPy, OpenCV, Matplotlib. Tôi hi vọng rằng chương trình này sẽ hữu ích cho các bạn trong quá trình xử lý ảnh số.

# 

# CHƯƠNG 1 : GIỚI THIỆU ĐỀ TÀI

## 1 . Đặt vấn đề

Trong thời đại công nghệ hiện nay, ảnh đã trở thành một phần không thể thiếu trong cuộc sống của chúng ta. Từ những bức ảnh kỷ niệm đến những bức ảnh sản phẩm, tất cả đều có một vai trò quan trọng trong việc ghi lại và truyền tải thông tin.

Tuy nhiên, khi chụp ảnh hoặc lưu trữ ảnh, chúng ta thường gặp phải những vấn đề như ảnh bị mờ, không đủ sáng tối, tương phản kém, nhiễu và độ phân giải thấp. Điều này làm cho việc xem và sử dụng ảnh trở nên khó khăn và không đạt được chất lượng tốt nhất.

Để giải quyết những vấn đề này, chúng ta cần một chương trình xử lý ảnh có khả năng thực hiện các thao tác để cải thiện chất lượng ảnh. Chương trình sẽ cho phép đọc ảnh từ file và thực hiện một số thao tác xử lý như cân bằng histogram, tách ngưỡng, lấy ảnh gradient, lọc thông thấp, lọc thông cao và thực hiện lọc ảnh bằng bộ lọc Bilateral và NonLocalMeans.

## 2.Mục tiêu nghiên cứu

Mục tiêu của chương trình xử lý ảnh từ file là cung cấp cho người dùng một công cụ đơn giản và hiệu quả để xử lý ảnh. Chương trình cho phép người dùng đọc ảnh từ file và thực hiện một số thao tác xử lý như cân bằng histogram, tách ngưỡng, lấy ảnh nghịch đảo, biến đổi logarit, tăng độ tương phản, phép lọc không gian, phép lọc ảnh trong miền tần số và lọc ảnh bằng bộ lọc Bilateral và NonLocalMeans. Mục tiêu của chương trình là giúp người dùng tiết kiệm thời gian và công sức trong việc xử lý ảnh, đồng thời cung cấp kết quả xử lý ảnh chính xác và đáng tin cậy.

## 3.Ứng dụng

- Xử lý ảnh y khoa: Chương trình có thể được sử dụng để phân tích các hình ảnh y khoa, bao gồm các loại chẩn đoán hình ảnh như chụp X-quang, siêu âm và MRI. Các thao tác xử lý ảnh như tách ngưỡng và lọc ảnh giúp cải thiện chất lượng hình ảnh và giúp các bác sĩ đưa ra chẩn đoán chính xác hơn.

- Xử lý ảnh kỹ thuật số: Chương trình có thể được sử dụng để xử lý các hình ảnh kỹ thuật số, bao gồm các bản vẽ kỹ thuật, bản vẽ kiến ​​trúc, hình ảnh máy tính và nhiều loại hình ảnh khác. Các thao tác xử lý ảnh như cân bằng histogram và tăng độ tương phản giúp cải thiện chất lượng hình ảnh và giúp người sử dụng phân tích chi tiết hơn.

- Xử lý ảnh trong sản xuất: Chương trình có thể được sử dụng để xử lý các hình ảnh trong quá trình sản xuất, bao gồm kiểm tra chất lượng sản phẩm và phát hiện các lỗi sản xuất. Các phép lọc không gian và lọc ảnh trong miền tần số giúp cải thiện chất lượng hình ảnh và giúp người sử dụng phát hiện các lỗi sản xuất.

- Xử lý ảnh trong đời sống: Chương trình có thể được sử dụng để xử lý các hình ảnh trong đời sống hàng ngày, bao gồm các bức ảnh chụp từ điện thoại, máy ảnh, hay các hình ảnh trên mạng. Các thao tác xử lý ảnh như cân bằng histogram và biến đổi logarit giúp cải thiện chất lượng hình ảnh và giúp người sử dụng có thể chỉnh sửa và cải thiện các bức ảnh của mình.

# CHƯƠNG 2 : Cơ sở lý thuyết

## 1.Tổng quan về xử lý ảnh

Con người thu nhận thông tin qua các giác quan, trong đó thị giác đóng vai trò quan trọng nhất. Những năm trở lại đây với sự phát triển của phần cứng máy tính, xử lý ảnh và đồ hoạ đó phát triển một cách mạnh mẽ và có nhiều ứng dụng trong cuộc sống. Xử lý ảnh và đồ hoạ đóng một vai trò quan trọng trong tương tác người máy.

Quá trình xử lý ảnh được xem như là quá trình thao tác ảnh đầu vào nhằm cho ra kết quả mong muốn. Kết quả đầu ra của một quá trình xử lý ảnh có thể là một ảnh “tốt hơn” hoặc một kết luận.

Sơ đồ tổng quát của hệ thống xử lý ảnh

**A picture containing diagram

Description automatically generated**

Hình 1.Quá trình xử lý ảnh

Ảnh có thể xem là tập hợp các điểm ảnh và mỗi điểm ảnh được xem như là đặc trưng cường độ sáng hay một dấu hiệu nào đó tại một vị trí nào đó của đối tượng trong không gian và nó có thể xem như một hàm n biến P(c1, c2,..., cn). Do đó, ảnh trong xử lý ảnh có thể xem như ảnh n chiều.

Sơ đồ tổng quát của một hệ thống xử lý ảnh

Hệ quyết định

Thu nhận ảnh (Scanner, Camera,Sensor)

Trích chọn

đặc điểm

Hậu xử lý

Đối sánh rút ra kết luận

Tiền xử

lý

Lưu trữ

Hình 2. Các bước cơ bản trong một hệ thống xử lý ảnh

## 2.Giới thiệu về Python

- Python là một ngôn ngữ lập trình thông dịch được phát triển bởi Guido van Rossum vào những năm 1990. Với tính năng đa mục đích, dễ học và dễ sử dụng, Python đã trở thành một trong những ngôn ngữ lập trình phổ biến nhất trên thế giới. Nó được sử dụng trong nhiều lĩnh vực khác nhau, từ phân tích dữ liệu, xử lý ảnh, web development, machine learning và nhiều hơn nữa.

- Python được phát triển ban đầu vào năm 1989 và phát hành vào năm 1991 bởi Guido van Rossum tại Hà Lan. Ban đầu, Python được thiết kế để là một ngôn ngữ script, nhưng nó đã phát triển thành một ngôn ngữ lập trình đa mục đích với nhiều tính năng tiên tiến.

Các phiên bản chính của Python bao gồm Python 2 và Python 3. Python 2 được phát hành vào năm 2000 và đã được sử dụng rộng rãi trong nhiều năm. Tuy nhiên, Python 2 đã chính thức dừng được hỗ trợ vào năm 2020 và nên chuyển sang sử dụng Python 3. Hiện tại, phiên bản mới nhất của Python là Python 3.10, được phát hành vào tháng 10 năm 2021.

## 3.Các thư viện của Python được dùng trong chương trình

1. NumPy: Thư viện NumPy cung cấp các chức năng để thực hiện tính toán số học và hình học trên mảng số, giúp cho việc xử lý ảnh trở nên nhanh chóng và hiệu quả.
2. OpenCV: Thư viện OpenCV là một trong những thư viện xử lý ảnh phổ biến nhất và được sử dụng rộng rãi trong các ứng dụng xử lý ảnh. OpenCV cung cấp các chức năng để đọc, ghi và xử lý ảnh.
3. Pillow: Thư viện Pillow cung cấp các chức năng để đọc và ghi các định dạng ảnh phổ biến như JPEG, PNG, GIF và BMP. Nó cũng cung cấp các chức năng để thực hiện các phép tính xử lý ảnh như cắt, xoay, phóng to, thu nhỏ và lọc ảnh.
4. Matplotlib: Thư viện Matplotlib là một thư viện đồ họa mạnh mẽ được sử dụng để tạo ra các biểu đồ, đồ thị và hình ảnh.
5. Scikit-image: Thư viện Scikit-image cung cấp các chức năng để thực hiện các phép tính xử lý ảnh như phân đoạn ảnh, xử lý ảnh y tế và nhận dạng đối tượng.
6. TensorFlow và Keras: TensorFlow và Keras là các thư viện được sử dụng để thực hiện các nhiệm vụ deep learning và máy học trong xử lý ảnh
7. Tkinter : là một thư viện giao diện người dùng (GUI) tiêu chuẩn được tích hợp sẵn trong Python. Nó cung cấp các công cụ để tạo ra các ứng dụng có giao diện đồ họa với các phần tử như cửa sổ, button, textbox, label, menu, scrollbar và nhiều hơn nữa.

# CHƯƠNG 3 : Chương trình

## 1.Các thao tác xử lý dựa trên điểm ảnh:

### • Cân bằng histogram.

Cân bằng Histogram (Histogram Equalization) là một kỹ thuật xử lý ảnh được sử dụng để cải thiện độ tương phản và độ sáng của ảnh. Kỹ thuật này là một phương pháp điều chỉnh độ sáng tự động của ảnh bằng cách phân bố lại các giá trị mức xám trong ảnh.

Histogram là biểu đồ thể hiện tần số xuất hiện của mỗi mức xám trong ảnh. Khi thực hiện cân bằng Histogram, chúng ta sẽ phân bố lại các giá trị mức xám sao cho histogram của ảnh đạt được phân bố đều trên toàn bộ khoảng giá trị.

Khi cân bằng Histogram, các vùng tối và sáng của ảnh sẽ được cân bằng, giúp tăng độ tương phản và độ sáng của ảnh, từ đó giúp tăng cải thiện chất lượng ảnh. Cân bằng Histogram thường được sử dụng trong các ứng dụng như xử lý ảnh y tế, nhận dạng khuôn mặt, nhận dạng vân tay, các ứng dụng trong công nghiệp,...

Các đặc điểm của ảnh được thể hiện qua histogram của ảnh:

+ Đối với ảnh có độ tương phản thấp thì đỉnh và phần histogram chính tập trung vào một vùng hẹp trên histogram.

+ Đối với bức ảnh tối thì histogram tập trung vào phía trái của lược đồ.

+ Đối với bức ảnh sáng thì histogram tập trung vào phía phải của lược đồ.

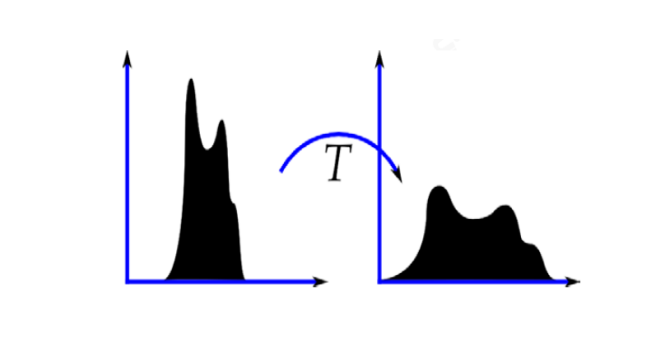
Histogram của ảnh đóng vai trò quan trọng trong các ứng dụng xử lý ảnh và nhận dạng:

+ Đối sánh ảnh: ảnh được biểu diễn bởi các dãy histogram và đối sánh với nhau.

+ Nhận dạng: biểu diễn các đặc trưng (features, intrinsic characteristics, discriminative features) qua các histogram cục bộ và đối sánh với các ảnh đã biết.

+ Phân đoạn ảnh

Hệ thống thị giác của con người (Human vision system) có thể nhìn được các chi tiết ảnh tốt hơn nếu histogram của ảnh nằm trong một dải rộng thay vì một dải hẹp=>có thể thực hiện điều chỉnh histogram của ảnh để đạt được hiệu quả nhận thức tốt hơn



Hình 3.Ảnh minh họa

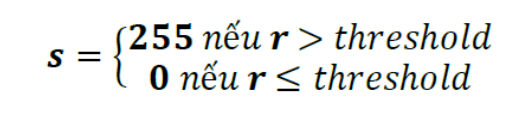
Trong Python để cân bằng Histogram của ảnh bằng phương pháp sử dụng hàm **equalizeHist()** và thư viện Open-Cv:

**def** Histogram():  
 **while**(**True**):  
 *# Load ảnh gốc* img = cv2.imread(**'anh.jpg'**)  
 *# Chuyển ảnh sang ảnh xám* gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 *# Cân bằng histogram của ảnh* ketqua = cv2.equalizeHist(gray)  
 *# Hiển thị ảnh gốc và ảnh đã cân bằng histogram* cv2.imshow(**'Anh Xam'**, gray)  
 cv2.imshow(**'Anh sau khi can bang Histogram'**, ketqua)  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:**if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

### • Tách ngưỡng (THRESHOLDING)

Là thao tác thường được dùng để nhấn mạnh một đối tượng trong ảnh hoặc tách đối tượng khỏi nền của ảnh. Ban đầu một ngưỡng (T – threshold) sẽ được chọn (tùy vào mục đích và ảnh cụ thể), ảnh mới sẽ được tính thao công thức bên dưới.

Công thức trên được gọi là tách ngưỡng nhị phân. Một cách khác để đạt kết quả tốt hơn gọi là tách ngưỡng linh hoạt (adaptive thresholding) trong đó việc tách ngưỡng được thực hiện với từng vùng của ảnh và với các ngưỡng khác nhau.



Hình 4. Công thức tách ngưỡng

Trong python ta áp dụng phương pháp **Otsu** để tách ngưỡng nhị phân :

**def** Tachnguong():  
 **while**(**True**):  
 *# Đọc ảnh đầu vào* img = cv2.imread(**'anh.jpg'**)  
 *# Chuyển đổi ảnh sang ảnh xám* gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 *# Xác định ngưỡng bằng phương pháp Otsu* \_, threshold\_img1 = cv2.threshold(gray\_img, 0, 255, cv2.THRESH\_BINARY + cv2.THRESH\_OTSU)  
 *# Hiển thị ảnh đầu vào và ảnh tách ngưỡng* cv2.imshow(**'Original Image'**, img)  
 cv2.imshow(**'Thresholded Image1 Otsu'**, threshold\_img1)  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:**if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

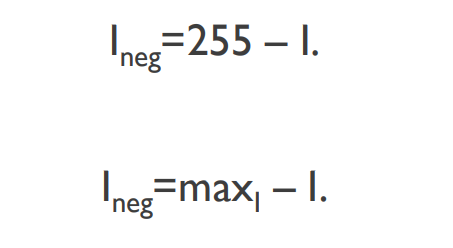
Đối số đầu tiên là ảnh xám, đối số thứ hai là giá trị ngưỡng ban đầu, đối số thứ ba là giá trị tối đa của pixel được hiển thị trên ảnh sau khi tách ngưỡng, đối số thứ tư là các tham số phương pháp tách ngưỡng, ở đây chúng ta sử dụng **cv2.THRESH\_BINARY + cv2.THRESH\_OTSU** để sử dụng phương pháp Otsu.

### • Lấy ảnh âm bản (NEGATIVE IMAGE)

Lấy ảnh âm bản (negative image) của một bức ảnh là quá trình đảo ngược giá trị của từng pixel trong ảnh. Khi lấy ảnh âm bản, các pixel có giá trị cao sẽ trở thành giá trị thấp và ngược lại, các pixel có giá trị thấp sẽ trở thành giá trị cao. Kết quả của quá trình này là tạo ra một bức ảnh mới với sự đảo ngược tông màu so với bức ảnh gốc.

Lấy ảnh âm bản thường được sử dụng trong xử lý ảnh để làm nổi bật các chi tiết của ảnh và tăng độ tương phản.

Nhằm tăng cường các chi tiết ảnh màu trắng hoặc xám bị ẩn trong các vùng tối của ảnh theo công thức sau:



Hình 5.Công thức lấy ảnh âm bản

Để lấy ảnh âm bản trong Python, ta có thể sử dụng phép toán đảo ngược giá trị pixel trên ảnh. Ta có thể thực hiện điều này bằng cách sử dụng thư viện OpenCV như sau:

**def** Layamban():  
 **while**(**True**):  
 img = cv2.imread(**'anh.jpg'**)  
 cv2.imshow(**'anh goc'**,img)  
 gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 anhamban = 255 - gray\_img  
 cv2.imshow(**'Anh am ban'**, anhamban)  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:**if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

### • Biến đổi logarit.

Biến đổi logarithmic là một trong những phép biến đổi hình ảnh phổ biến trong xử lý ảnh. Biến đổi này được sử dụng để tăng độ tương phản của ảnh và làm nổi bật các chi tiết trong vùng bóng và vùng sáng của ảnh.

Trong biến đổi logarithmic, giá trị của từng pixel được thay đổi bằng cách lấy logarit tự nhiên của giá trị đó và nhân với một hằng số. Kết quả của biến đổi này là tạo ra một bức ảnh mới có độ tương phản tốt hơn, đặc biệt là trong vùng bóng của ảnh.

Biến đổi theo hàm logarith dùng để nén vùng giá trị của một vùng hẹp các mức xám thấp sang một vùng giá trị rộng hơn theo công thức:

S = c\*log(1+I)

Trong đó:

**+ s** là giá trị của pixel mới sau khi biến đổi

**+ c** là hằng số tỷ lệ

**+ I** là giá trị của pixel trong bức ảnh gốc

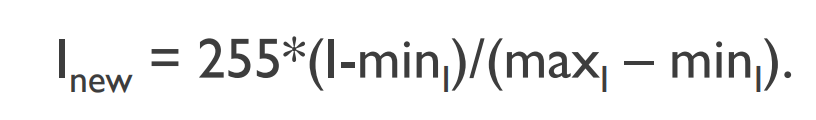
Trong python để biến đổi logarit ta sử dụng thư viện open-Cv và numpy

**def** Logarithm():  
 img = cv2.imread(**'anh.jpg'**,0)  
 **while**(**True**):  
 c = 255 / np.log(1 + np.max(img))  
 logaritthmic = c \* (np.log(img + 1))  
 logaritthmic = np.array(logaritthmic, dtype=np.uint8)  
 cv2.imshow(**'Anh xam'**, img)  
 cv2.imshow(**'Anh bien doi logarith'**, logaritthmic)  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:**if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

### • Tăng độ tương phản.

Tăng độ tương phản của ảnh trong xử lý ảnh được sử dụng để cải thiện chất lượng ảnh và làm nổi bật các chi tiết quan trọng trong ảnh. Khi một bức ảnh có độ tương phản thấp, các chi tiết trong ảnh có thể bị mờ hoặc khó nhìn rõ ràng. Bằng cách tăng độ tương phản, chúng ta có thể làm nổi bật các đối tượng, chi tiết trong ảnh và cải thiện khả năng nhận dạng của các thuật toán xử lý ảnh.

Đối với các ảnh có độ tương phản thấp thì việc tăng cường độ tương phản sẽ làm nổi rõ các chi tiết của ảnh



Hình 6. Công thức tăng độ tương phản của ảnh

Trong python để tăng độ tương phản của ảnh ta sử dụng 2 thư viện là Open-Cv và Numpy

**def** Tangdotuongphan():  
 **while**(**True**):  
 *# Đọc ảnh gốc* img = cv2.imread(**'anh.jpg'**)  
 *# Chuyển ảnh sang dạng ảnh xám* gray\_img = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  
 max = np.max(gray\_img)  
 min = np.min(gray\_img)  
 kqa = 255 \* (gray\_img - min) / (max - min)  
 cv2.imshow(**'Anh goc'**,img)  
 cv2.imshow(**'Tang do tuong phan'**,kqa)  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:**if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

## Các phép lọc không gian:

## • Mean filter

## Tác dụng:

- Phép lọc mean (hay còn gọi là bộ lọc trung bình) là một trong những phép lọc cơ bản trong xử lý ảnh. Kỹ thuật này được sử dụng để giảm nhiễu và làm mờ ảnh.

- Lọc nhiễu (noise removing)

- Làm mịn ảnh

- Việc làm mờ ảnh này có thể được sử dụng để loại bỏ các chi tiết không quan trọng và giảm độ phân giải của ảnh, giúp cho việc xử lý và nhận dạng ảnh được dễ dàng hơn. Tuy nhiên, khi áp dụng phép lọc mean quá nhiều lần hoặc với kernel quá lớn có thể dẫn đến mất mát thông tin quan trọng và ảnh hóa mờ hoàn toàn. Do đó, việc lựa chọn kernel và số lần áp dụng phép lọc cần được thực hiện cẩn thận.

Đặc điểm:

- Đơn giản, thực hiện nhanh

- Tạo ra các giá trị không thật

- Mức độ làm mịn hạn chế

Trong python để thực hiện phép lọc Mean của ảnh ta sử dụng thư viện là Open-Cv và hàm cv2.blur() :

**def** Mean():  
 **while**(**True**):  
 *# Load ảnh* img = cv2.imread(**'anh.jpg'**)  
 *# Thiết lập kích thước kernel và lọc ảnh* kernel\_size = 5  
 img\_mean = cv2.blur(img, (kernel\_size, kernel\_size))  
 *# Hiển thị ảnh gốc và ảnh đã lọc* cv2.imshow(**'Original'**, img)  
 cv2.imshow(**'Mean Filter'**, img\_mean)  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:**if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

### • Median filter.

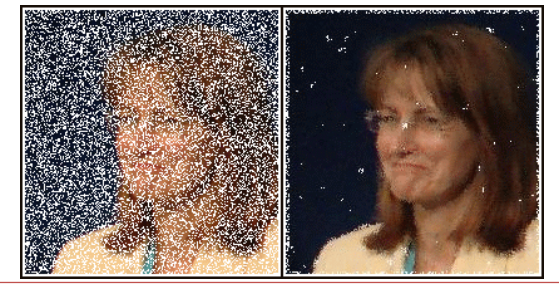
Bộ lọc trung vị (median filter) trong Python là một kỹ thuật xử lý ảnh để loại bỏ nhiễu khỏi ảnh bằng cách thay thế giá trị của mỗi pixel bằng giá trị trung vị (median) của các pixel trong vùng lân cận của nó. Kỹ thuật này được sử dụng để làm giảm nhiễu trong ảnh và giữ lại các đặc trưng cơ bản của ảnh, giúp ảnh trở nên rõ ràng và chất lượng hơn.

Bộ lọc trung vị là một trong những phương pháp xử lý nhiễu phổ biến và được sử dụng rộng rãi trong xử lý ảnh. Nó thường được sử dụng để loại bỏ các nhiễu loại đơn giản, chẳng hạn như nhiễu muối tiêu (salt-and-pepper noise), làm cho ảnh trở nên mịn hơn và tăng độ chính xác cho các phương pháp xử lý ảnh tiếp theo.

Tác dụng làm mịn có nhưng không mạnh.

Phép lọc trung vị chậm hơn so với lọc trung bình vì cần phải sắp xếp để tìm các giá trị trung vị.

Trong Python, bộ lọc trung vị có thể được thực hiện bằng cách sử dụng hàm **cv2.medianBlur()** của thư viện OpenCV hoặc bằng cách sử dụng các thư viện khác như NumPy hoặc SciPy. Khi áp dụng bộ lọc trung vị, kích thước của vùng lân cận và giá trị của pixel trung vị có thể được điều chỉnh để đạt được kết quả tốt nhất cho từng trường hợp cụ thể.



Hình. Lọc trung vị (Nhiễu muối tiêu)

Trong python để loại bỏ nhiễu muối tiêu ta sử dụng hàm medianBlur()

**def** Median():  
 **while**(**True**):  
 img = cv2.imread(**'anh.jpg'**)  
 *# Thiết lập độ nhiễu và tạo mảng ngẫu nhiên* noise\_ratio = 0.1  
 noise = np.zeros(img.shape[:2], dtype=np.uint8)  
 cv2.randu(noise, 0, 255)  
 salt = noise > 255 \* (1 - noise\_ratio)  
 pepper = noise < 255 \* noise\_ratio  
 *# Gán giá trị muối và tiêu cho ảnh* img[salt] = 255  
 img[pepper] = 0  
 median = cv2.medianBlur(img, 5)  
 *# Hiển thị ảnh* cv2.imshow(**'Salt and Pepper Noise'**, img)  
 cv2.imshow(**'Anh sau khi loc trung vi'**, median)  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:**if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

### • Gaussian filter.

Phép lọc Gauss được thực hiện bằng cách áp dụng phép nhân chập 1 nhân Gauss với ảnh input.

Bộ lọc Gauss là bộ lọc nhiễu tốt nhất, cho kết quả ảnh mịn (smooth) tốt nhất (tự nhiên nhất).

Bộ lọc Gaussian có các tham số như kích thước kernel (kernel size) và độ lệch chuẩn (standard deviation) của hàm Gaussian. Khi kích thước kernel càng lớn, thì độ mịn của ảnh càng cao và độ nhiễu càng được giảm. Khi độ lệch chuẩn càng lớn, thì hàm Gaussian sẽ giảm độ sáng của các điểm ảnh xa tâm kernel càng nhiều, từ đó làm mịn ảnh càng mạnh



Hình. Lọc Gaussian fliter

**def** Gaussian():  
 **while**(**True**):  
 *# Load ảnh đầu vào* img = cv2.imread(**'anh.jpg'**)  
  
 *# Áp dụng phép lọc Gaussian với kích thước kernel là (5, 5) và độ lệch chuẩn là 0* img\_gaussian = cv2.GaussianBlur(img, (5, 5), 0)  
  
 *# Hiển thị ảnh gốc và ảnh sau khi được áp dụng phép lọc Gaussian* cv2.imshow(**'Original Image'**, img)  
 cv2.imshow(**'Gaussian Filtered Image'**, img\_gaussian)  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:**if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

### • Linear sharpen.

Dựa vào nguyên lý xử lý tín hiệu: làm mờ ảnh là loại bỏ các tín hiệu ở tần số cao (các chi tiết sắc nét của ảnh) nên nếu áp dụng ngược (1 – bộ lọc làm mờ) thì sẽ nhận được ảnh tăng cường các tần số cao (ảnh nét hơn) và giảm bớt các tần số thấp. Công thức:



Hình9. Công thức lọc Linear sharpe

Thường γ= 0.5 và Ismooth là ảnh kết quả nhận được bằng cách áp dụng 1 bộ lọc làm mịn ảnh.

Bộ lọc làm sắt nét tuyến tính ngoài việc làm ảnh sắc nét hơn còn có tác dụng loại bỏ các quầng sáng (halos) do bị ảnh hưởng bởi điều kiện chụp nên hay được dùng trong các thiết bị thu nhận ảnh số.

Bộ lọc linear sharpen thường được sử dụng trong các ứng dụng xử lý ảnh như chỉnh sửa ảnh, phân tích hình ảnh y tế, giám sát an ninh hay chụp ảnh kỹ thuật số

Trong python ta sử dụng thư viện open-Cv:

**def** linearSharpen():  
 **while**(**True**):  
 *# Đọc ảnh* image = cv2.imread(**'anh.jpg'**)  
 anhxam = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)  
 smooth = cv2.GaussianBlur(anhxam, (5, 5), 0)  
 sharp = 1.5 \* anhxam - 0.5 \* smooth  
 *# Chuyển đổi kiểu dữ liệu của ảnh output thành uint8* sharp = np.clip(sharp, 0, 255).astype(np.uint8)  
 cv2.imshow(**'anh goc'**, anhxam)  
 cv2.imshow(**'Kqua linear'**, sharp)  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:**if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

### • Lấy ảnh gradient.

Đạo hàm của ảnh được tính bằng kết quả của phép nhân chập với các nhân dùng để tính đạo hàm: Sobel, Prewitt, Roberts (Tổng các phần tử trong nhân để lấy đạo hàm đều bằng 0)

Image gradient là kỹ thuật được sử dụng trong xử lý ảnh để tìm ra các biên của ảnh, các cạnh, và các điểm cực đại và cực tiểu của hình ảnh

Trong python ta sử dụng thư viện open-Cv và Sobel operator:

**def** Gradient():  
 **while**(**True**):  
 *# Load the image* img = cv2.imread(**'anh.jpg'**, cv2.IMREAD\_GRAYSCALE)  
  
 *# Calculate the x and y derivatives using Sobel operator* dx = cv2.Sobel(img, cv2.CV\_64F, 1, 0, ksize=3)  
 dy = cv2.Sobel(img, cv2.CV\_64F, 0, 1, ksize=3)  
  
 *# Calculate the gradient magnitude* grad\_mag = np.sqrt(dx \*\* 2 + dy \*\* 2)  
  
 *# Show the result* cv2.imshow(**'Original Image'**, img)  
 cv2.imshow(**'Gradient Magnitude'**, grad\_mag.astype(np.uint8))  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:  
 *# Xử lý sự kiện bàn phím ở đây* **if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

### • Edge detection.

Edge detection (phát hiện biên cạnh) là một kỹ thuật trong xử lý ảnh được sử dụng để tìm và nhận dạng các biên cạnh của đối tượng trong ảnh. Việc phát hiện biên cạnh thường được sử dụng để giảm nhiễu, phát hiện đối tượng và trích xuất đặc trưng trong các ứng dụng xử lý ảnh

Cụ thể, phát hiện biên cạnh được sử dụng để xác định các đường viền của các đối tượng trong ảnh, giúp phân tách chúng ra khỏi nền và nhận dạng các đối tượng dựa trên hình dáng của chúng. Nó cũng được sử dụng để giảm nhiễu bằng cách xóa bớt các chi tiết không quan trọng nhưng không thể nhận dạng được. Thêm vào đó, phát hiện biên cạnh được sử dụng trong nhiều ứng dụng như trích xuất đặc trưng, phát hiện hình dạng, phân tích vật liệu và trong các ứng dụng nhận diện khuôn mặt và nhận dạng vân tay.

**def** EdgeDetection():  
 **while**(**True**):  
 *# Đọc ảnh đầu vào và chuyển sang ảnh xám* img = cv2.imread(**'anh.jpg'**, 0)  
  
 *# Áp dụng Gaussian Blur để giảm nhiễu* img\_blur = cv2.GaussianBlur(img, (3, 3), 0)  
  
 *# Áp dụng phát hiện cạnh Canny* edges = cv2.Canny(img\_blur, 100, 200)  
  
 *# Hiển thị ảnh gốc và ảnh đã phát hiện cạnh* cv2.imshow(**'Original'**, img)  
 cv2.imshow(**'Edges'**, edges)  
  
 key = cv2.waitKeyEx(1)  
 **if** key != -1:**if** key == ord(**'q'**):  
 **break** cv2.destroyAllWindows()

## 3. Các phép lọc ảnh trong miền tần số:

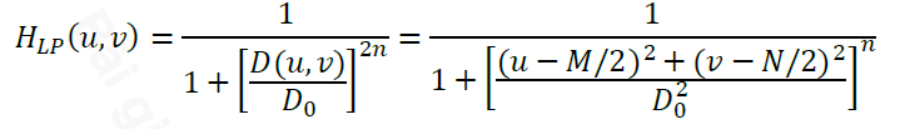
### • Lọc thông thấp.

- Lọc thông thấp là một kỹ thuật xử lý ảnh nhằm giảm bớt các tần số cao trong ảnh, từ đó làm giảm nhiễu và làm mịn ảnh

- Lọc thông thấp (lowpass filter): chỉ cho đi qua các tín hiệu tần số thấp

- Các bộ lọc thông thấp chỉ cho qua các tần số thấp và sẽ làm mờ ảnh, tác dụng là loại bỏ nhiễu. Có 3 bộ lọc thông thấp: Ideal, Butterworth, Gaussian

- Bộ lọc thông thấp Butterworth: hàm lọc H(u, v) sẽ được tính như sau:



Hình10. Công thức lọc thông thấp Butterworth

- Với D0 là bán kính cắt tần số. Công thức trên làm trơn quá trình chuyển đổi giữa các tần số thấp và tần số cao trong quá trình lọc nên giảm được hiện tượng ringing effect. Số mũ n được gọi là bậc của bộ lọc (thường có giá trị bằng 2)..

Code trong Python:

**def** butterworth\_filter(rows, cols, n, radius, highpass=**False**):x = np.linspace(-0.5, 0.5, cols) \* cols  
 y = np.linspace(-0.5, 0.5, rows) \* rows  
 radius\_mat = (x[np.newaxis, :]\*\*2 + y[:, np.newaxis]\*\*2)  
 radius2 = radius\*radius  
 **if** highpass:  
 radius\_mat = 1 / (1 + (radius2 / radius\_mat)\*\*(n))  
 **else**:  
 radius\_mat = 1 / (1 + (radius\_mat / radius2)\*\*(n))  
 **return** radius\_mat

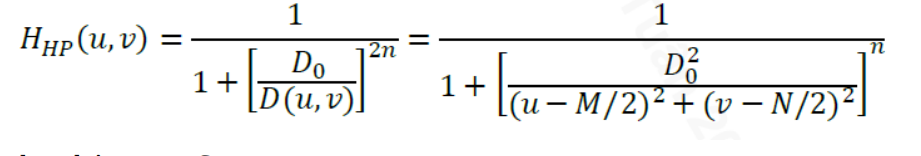
**def** Locthongthap():  
 img = cv2.imread(path,0) *# Đọc ảnh đen trắng* rows, cols = img.shape  
 *# Tạo bộ lọc Butterworth thông thấp* n = 2 *# Bậc của bộ lọc* radius = 55 *# Bán kính của bộ lọc* h = butterworth\_filter(rows, cols, n, radius)  
 *# Áp dụng bộ lọc* f = np.fft.fft2(img)  
 fshift = np.fft.fftshift(f)  
 filtered\_f = fshift \* h[:, :]  
 filtered\_img = np.abs(np.fft.ifft2(filtered\_f))  
 *# Hiển thị ảnh gốc và ảnh đã lọc* cv2.imshow(**'Original Image'**, img)  
 cv2.imshow(**'Filtered Image'**, filtered\_img.astype(np.uint8))  
 cv2.waitKey(0)  
 cv2.destroyAllWindows()

### • Lọc thông cao.

- Lọc thông cao (highpass filter): chỉ cho đi qua các tín hiệu có tần số cao

- Các tần số cao của ảnh thường tương ứng với các đường (edges) và các chi tiết (fined details) của ảnh, do đó việc lọc thông cao (loại bỏ các chi tiết tần số thấp) sẽ làm cho ảnh trở nên sắc nét hơn và làm giảm độ mờ của ảnh.

- Hàm lọc thông cao Butterworth:



Hình11. Công thức lọc thông cao Butterworth

- Ứng dụng của lọc thông cao là để làm nổi bật các chi tiết của ảnh, ví dụ như trong các ảnh chụp y học (chụp X-quang).

Trong python: Code tương tự lọc thông thấp

## Thực hiện lọc ảnh bằng bộ lọc Bilateral và NonLocalMeans.

### Bilateral

- Sử dụng để giảm nhiễu và làm mịn hình ảnh, giữ lại các cạnh và chi tiết quan trọng của hình ảnh.

- Làm mịn hình ảnh: Bộ lọc bilateral cũng có thể được sử dụng để làm mịn hình ảnh bằng cách sử dụng hàm cv2.bilateralFilter() với kích thước cửa sổ lớn hơn.

-

Code trong Python:

**def** Bilateral():  
 img = cv2.imread(path)  
 *# Sử dụng bộ lọc Bilateral với kích thước vùng lân cận là 9 và giá trị sigmaColor và sigmaSpace là 75* blurred\_img = cv2.bilateralFilter(img, 9, 75, 75)  
 *# Hiển thị ảnh gốc và ảnh sau khi được lọc* cv2.imshow(**"Original Image"**, img)  
 cv2.imshow(**"Bilateral"**, blurred\_img)  
 *# Chờ người dùng nhấn phím bất kỳ để đóng cửa sổ* cv2.waitKey(0)  
 cv2.destroyAllWindows()

### NonLocalMeans.

Bộ lọc Non-Local Means (NLM) là một phương pháp lọc hình ảnh được sử dụng để giảm nhiễu và làm mịn hình ảnh. Bộ lọc NLM được thiết kế để bảo toàn các cạnh và chi tiết quan trọng trong hình ảnh. Nó có thể giữ lại các chi tiết quan trọng và cạnh trong hình ảnh ngay cả khi áp dụng một mức độ lọc mạnh.

Khi áp dụng bộ lọc NLM lên một hình ảnh, nó sẽ tính toán sự khác biệt giữa các vùng hình ảnh và tìm kiếm các vùng tương tự trong hình ảnh để tính toán giá trị lọc. Bằng cách làm như vậy, bộ lọc NLM có thể loại bỏ các pixel nhiễu và giữ lại các chi tiết quan trọng, giúp cải thiện chất lượng hình ảnh.

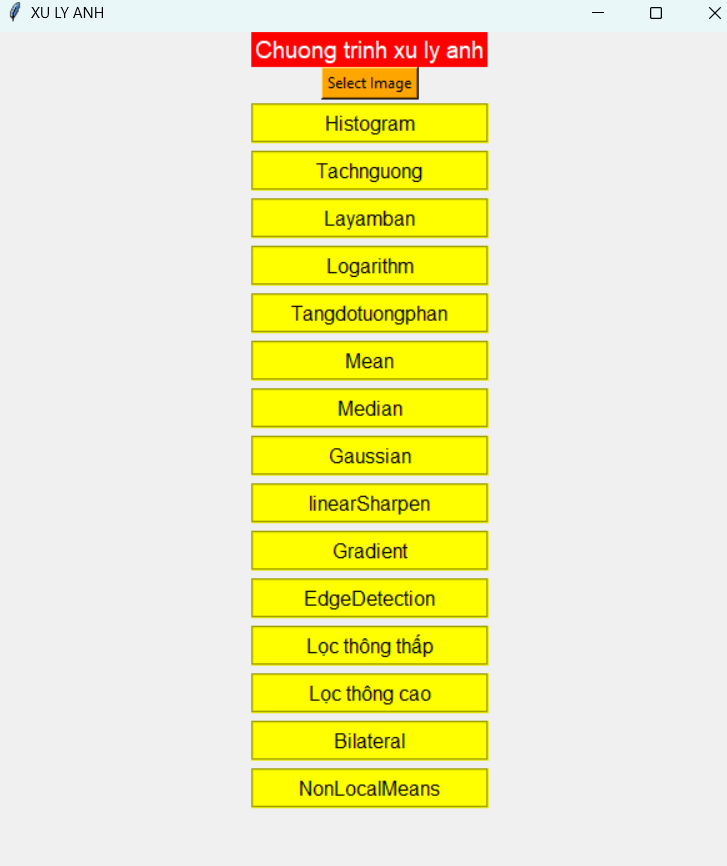
Bộ lọc NLM cũng được sử dụng trong nhiều ứng dụng khác nhau, chẳng hạn như xử lý hình ảnh y tế và xử lý hình ảnh video. Nó cũng được sử dụng trong các ứng dụng điện ảnh để loại bỏ nhiễu và tăng cường chất lượng hình ảnh.

Code trong Python:

**def** NonLocalMeans():  
 img = cv2.imread(path)  
 *# Sử dụng bộ lọc Non-local Mean với kích thước vùng lân cận là 11 và giá trị h là 10* denoised\_img = cv2.fastNlMeansDenoising(img, **None**, 11, 10)  
 *# Hiển thị ảnh gốc và ảnh sau khi được lọc* cv2.imshow(**"Original Image"**, img)  
 cv2.imshow(**"Nonlocalmeans"**, denoised\_img)  
 *# Chờ người dùng nhấn phím bất kỳ để đóng cửa sổ* cv2.waitKey(0)  
 cv2.destroyAllWindows()

# CHƯƠNG 4 : Kết luận

## 4.1 Giao diện khi chạy chương trình



Hình 12.Giao diện chương trình

## 4.2.Ưu và nhược điểm

Ưu điểm của chương trình:

* Cho phép đọc ảnh từ file và thực hiện các thao tác xử lý ảnh khác nhau, giúp người dùng dễ dàng xử lý ảnh một cách nhanh chóng và tiện lợi.
* Các thao tác xử lý ảnh được tích hợp trong chương trình đa dạng và phong phú, cho phép người dùng lựa chọn và tùy chỉnh thao tác phù hợp với nhu cầu của họ.
* Các phép lọc không gian và trong miền tần số cung cấp các phương pháp hiệu quả để làm mịn, làm sắc nét và phát hiện cạnh của ảnh.
* Các bộ lọc Bilateral và NonLocalMeans là những bộ lọc mạnh mẽ có khả năng giảm nhiễu và giữ lại các đặc trưng quan trọng của ảnh.

Nhược điểm của chương trình:

* Các thao tác xử lý ảnh trong chương trình có thể không đủ mạnh mẽ để xử lý những vấn đề phức tạp trong ảnh, như xử lý các ảnh y tế hay ảnh khoa học.
* Việc áp dụng các phép lọc ảnh và thao tác xử lý ảnh có thể tốn nhiều thời gian tính toán và tài nguyên máy tính, đặc biệt là đối với những ảnh có kích thước lớn hoặc độ phân giải cao.