**TRƯỜNG ĐẠI HỌC BÁCH KHOA HÀ NỘI**

**VIỆN ĐIỆN TỬ - VIỄN THÔNG**

----o0o----



**BÁO CÁO BÀI TẬP LỚN**

**XỬ LÝ ẢNH SỐ**

**Đề tài:**

**TÌM HIỂU VÀ XÂY DỰNG CÁC BỘ LỌC XỬ LÝ ẢNH**

**Giảng viên hướng dẫn: TS. Lê Thị Lan**

**Nhóm: 19**

|  |  |  |
| --- | --- | --- |
| **Họ và tên sinh viên** | **MSSV** |  |
| Phạm Ngọc Lâm | 20182628 |  |
| Nguyễn Huy Nam  Lê Quang Hưng | 20182695  20182562 |  |
| Nguyễn Thị Vân Anh | 20172406 |  |
|  |  |  |

Hà Nội, 7/2022

# LỜI NÓI ĐẦU

Xử lý ảnh là một lĩnh vực mang tính khoa học và công nghệ. Nó là ngành khoa học mới mẻ so với nhiều ngành khoa học khác nhưng tốc độ phát triển của nó rất nhanh, kích thích các trung tâm nghiên cứu, ứng dụng, đặc biệt là máy tính chuyên dụng cho nó. Đầu tiên là thu nhận ảnh từ camera, vệ tinh hay các bộ cảm ứng, …Tín hiệu lấy vào sẽ được số hóa thành tín hiệu số và chuyển sang giai đoạn xử lý, phân tích hay lưu trữ lại. Trong quá trình học, chúng em đặc biệt hứng thú với các phương pháp lọc số để xử lý ảnh. Chính vì vậy nhóm em đã chọn đề tài: " Tìm hiểu và xây dựng các bộ lọc xử lý ảnh”.

Đây là một đề tài rất hay, đặc biệt tạo nên nhiều ảnh với các mục đích của người dùng. Chúng em đã tìm kiếm và dưới sự hướng dẫn của cô Lê Thị Lan, từ tài liệu rất nhiều cùng với quá trình xây dựng ứng dụng, chúng em đã rút ra được rất nhiều kinh nghiệm thực tế cũng như kiến thức sâu hơn về môn học. Tuy nhiên do chưa có đầy đủ kiến thức, cũng chưa có nhiều kinh nghiệm, tài liệu về lĩnh vực này, nên đề tài của chúng em không tránh khỏi những thiếu sót mong cô góp ý để bài tập lớn của chúng em được hoàn thiện hơn.

Chúng em xin chân thành cảm ơn!

# 

# MỤC LỤC

￼

￼

￼

￼

￼

1.1 ￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

￼

**DANH MỤC HÌNH ẢNH**

[Hình 1:Hệ thống xử lý ảnh 6](#_Toc109433621)

[Hình 2: Nhân ma trận ảnh 10](#_Toc109433622)

[Hình 3:Ảnh gốc(a) và ảnh sau khi tính Gx(b) 12](#_Toc109433623)

[Hình 4:Ảnh gốc(a) và ảnh sau khi tính Gy(b) 12](#_Toc109433624)

[Hình 5:Sau khi kết hợp cả Gx và Gy 13](#_Toc109433625)

[Hình 6:Sơ đồ khối bộ lọc Sobel 14](#_Toc109433626)

[Hình 7:Ma trận lọc Gauss 15](#_Toc109433627)

[Hình 8:Đường cong Gauss chuẩn hóa với giá trị và phương sai cho mảng 2 chiều 16](#_Toc109433628)

[Hình 9:Ma trận ảnh ban đầu 18](#_Toc109433629)

[Hình 10:Kết quả 19](#_Toc109433630)

[Hình 11:Chập với nhân 3\*3,5\*5,9\*9 của bộ lọc trung vị 19](#_Toc109433631)

[Hình 12: Phân phối tuyến tính Gaussian với các giá trị sigma khác nhau 21](#_Toc109433632)

[Hình 13:Phạm vi và thông số hoạt động của Bilateral cung cấp khả năng điều khiển linh hoạt hơn Gaussian 22](#_Toc109433633)

[Hình 14:Ảnh gốc (a) và ảnh bị nhiễu (b) 23](#_Toc109433634)

[Hình 15:Ảnh sử dụng lọc 3\*3(a) và ảnh sử dụng lọc 5\*5(b) 24](#_Toc109433635)

[Hình 16:Ảnh ban đầu 33](#_Toc109433636)

[Hình 17:Ảnh qua xử lý thông qua Photo\_app 33](#_Toc109433637)

# PHÂN CÔNG CÔNG VIỆC

|  |  |  |
| --- | --- | --- |
| STT | Nhiệm Vụ | Người Thực Hiện |
| 1 |  |  |
| 2 |  |  |
| 3 |  |  |
| 4 |  |  |
| 5 |  |  |
| 6 |  |  |
| 7 |  |  |

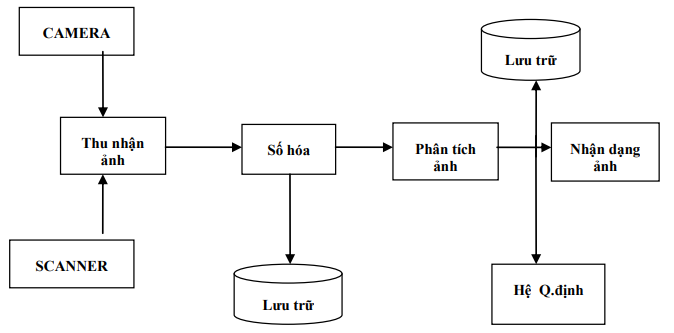
# CHƯƠNG 1: TỔNG QUAN VỀ CÁC BỘ LỌC

## 1.1 Tổng quan về xử lý ảnh

* + 1. *Hệ thống xử lý ảnh*

Xử lý ảnh là đối tượng nghiên cứu của lĩnh vực thị giác máy, là quá trình biến đổi từ một ảnh ban đầu sang một ảnh mới với các đặc tính và tuân theo ý muốn của người sử dụng. Xử lý ảnh có thể gồm quá trình phân tích, phân lớp các đối tượng, làm tăng chất lượng, phân đoạn và tách cạnh, gán nhãn cho vùng hay quá trình biên dịch các thông tin hình ảnh của ảnh. Mục đích của xử lý ảnh gồm:

* Biến đổi ảnh, làm tăng chất lượng ảnh.
* Tự động nhận dạng, đoán nhận, đánh giá các nội dung của ảnh.

Các bước cần thiết trong xử lý ảnh: 

Hình 1:Hệ thống xử lý ảnh

Trước hết là quá trình thu nhận ảnh. Ảnh có thể thu nhận được qua camera. Thường khi thu nhận ảnh qua camera là tín hiệu tương tự (loại camera ống kiểu CCIR), nhưng cũng có thể là tín hiệu số hóa (loại CCD- Charge Coupled Device). Ảnh cũng có thể thu nhận từ vệ tinh qua các bộ cảm ứng (sensor), hay ảnh tranh được quét trên scanner. Tiếp theo là quá trình số hóa (Digitalizer) để biến đổi tín hiệu tương tự sang tín hiệu rời rạc (lấy mẫu) và số hóa bằng lượng hóa, trước khi chuyển sang giai đoạn xử lý, phân tích hay lưu trữ lại. Trước hết là công việc tăng cường ảnh để nâng cao chất lượng ảnh. Do những nguyên nhân khác nhau: có thể do chất lượng thiết bị thu nhận ảnh, do nguồn sáng hay do nhiễu, ảnh có thể bị suy biến do vậy cần phải tăng cường và khôi phục lại ảnh để làm nổi bật một số đặc tính chính của ảnh, hay làm cho ảnh gần giống nhất với trạng thái gốc – trạng thái trước khi bị biến dạng. Giai đoạn tiếp theo là phát hiện các đặc tính như biên, phân vùng ảnh, trích chọn các đặc trưng... Cuối cùng tùy theo mục đích của ứng dụng, sẽ là giai đoạn nhận dạng, phân lớp hay các quyết định khác.

* + 1. C*ác vấn đề trong xử lý ảnh*

1. Một số khái niệm cơ bản

Ảnh trong thực tế là một ảnh liên tục về không gian và về giá trị độ sáng. Để có thể xử lý bằng máy tính cần thiết phải tiến hành số hóa ảnh. Trong quá trình số hóa người ta biến đổi từ tín hiệu liên tục sang tín hiệu rời rạc thông qua quá trình lấy mẫu (rời rạc hóa về không gian) và lượng hóa thành phần về giá trị mà về nguyên tắc bằng mắt thường không phân biệt được hai điểm kề nhau. Trong quá trình này người ta sử dụng khái niệm Picture element mà ta quen gọi hay viết là Pixel. Vậy 1 ảnh là một tập hợp các pixel.

Mức xám (Gray level) là kết quả sự mã hóa tương ứng với một cường độ sáng của mỗi điểm ảnh với một giá trị số - kết quả của quá trình lượng hóa. Cách mã hóa kinh điển thường dùng 16, 32 hay 64 mức. Mã hóa 256 mức là phổ dụng nhất do lý do kĩ thuật. Vì = 256 (0, 1, ..., 255), nên với 256 mức, mỗi pixel sẽ được mã hóa bởi 8 bit. Độ phân giải (Resolation) của ảnh là mật độ điểm ảnh được ấn định trên một ảnh số được hiển thị. Ảnh số là tập hợp các điểm ảnh với mức xám phù hợp dùng để mô tả ảnh gần với ảnh thật. Ảnh nhị phân là ảnh chỉ có hai mức xám 0 và 1.

Ảnh màu là ảnh số trong đó cường độ điểm ảnh là sự tổng hợp từ các màu tùy theo từng loại mà có cách biểu diễn khác nhau. Ảnh đa mức xám là ảnh có nhiều hơn hai mức xám.

1. Biểu diễn ảnh

Trong biểu diễn ảnh, người ta thường dùng các phần tử đặc trưng của ảnh là pixel. Nhìn chung có thể một hàm hai biến chứa các thông tin như biểu diễn của một ảnh. Các mô hình biểu diễn cho ta một mô tả logic hay định lượng các tính chất của hàm này. Trong biểu diễn ảnh cần chú ý đến tính trung thực hoặc các tiêu chuẩn “thông minh” để đo chất lượng ảnh hoặc tính hiệu quả của các kĩ thuật xử lý.

Một số mô hình thường được dùng trong biểu diễn ảnh: mô hình bài toán, mô hình thống kê. Trong mô hình bài toán, ảnh hai chiều được biểu diễn nhờ các hàm hai biến trực giao gọi là các hàm cơ sở. Còn mô hình thống kê, một ảnh được coi như một phần tử của một tập hợp đặc trưng bởi các đại lượng như: kỳ vọng toán học, hiệp biến, phương sai, moment.

1. Biến đổi ảnh (Image Transform)

Thuật ngữ biến đổi ảnh thường dùng để nói tới một lớp các ma trận đơn vị và các kĩ thuật dùng để biến đổi ảnh. Biến đổi ảnh nhằm làm giảm các nguyên nhân của ảnh để việc xử lý hiệu quả hơn. Như làm rõ hơn các thông tin mà ngời dùng quan tâm nhưng người dùng phải chấp nhận mất đi một số thông tin cần thiết.

1. Phân tích ảnh

Phân tích ảnh liên quan đến việc xác định các độ đo định lượng của 1 ảnh để đưa ra một mô tả đầy đủ về ảnh. Quá trình phân tích ảnh thực chất bao gồm nhiều công đoạn nhỏ. Trước hết là công việc tăng cường ảnh để nâng cao chất lượng ảnh, giai đoạn tiếp theo là phát hiện các đặc tính như phát hiện biên, phân vùng ảnh, trích chọn các đặc tính.v.v..

1. Tăng cường ảnh – khôi phục ảnh

Tăng cường ảnh là một bước quan trọng, tạo tiền đề cho xử lý ảnh. Nó gồm các kỹ thuật như: lọc độ tương phản, khử nhiễu, nổi màu…

Khôi phục ảnh là nhằm loại bỏ các suy giảm trong ảnh.

Trong bài tập lớn này chúng em thực hiện tìm hiểu các bộ lọc mục đích giúp tăng cường chất lượng của ảnh.

1. Biên

Biên là vấn đề chủ yếu trong phân tích ảnh vì các điểm trích chọn trong quá trình phân tích ảnh đều dựa vào biên. Mỗi điểm ảnh có thể là biên nếu ở đó có sự thay đổi đột ngột về mức xám. Tập hợp các điểm biên tạo thành biên hay đường bao quanh của ảnh.

1. Phân vùng

Phân vùng là bước then chốt trong xử lý ảnh. Giai đoạn này nhằm phân tích ảnh thành những thành phần có tính chất nào đó dựa theo biên hay các vùng liên thông. Tiêu chuẩn để xác định các vùng liên thông có thể là mức xám, cùng màu hay độ tương phản.

1. Nhận dạng ảnh

Nhận dạng ảnh là quá trình liên quan đến các mô tả đối tượng mà người ta muốn đặc tả nó. Quá trình nhận dạng thường đi sau quá trình trích chọn các đặc tính chủ yếu của đối tượng.

Có hai kiểu mô tả đối tượng:

* Mô tả tham số (nhận dạng theo tham số).
* Mô tả theo cấu trúc (nhận dạng theo cấu trúc).

Trên thực tế người ta đã áp dụng kỹ thuật nhận dạng khá thành công với nhiều đối tượng khác nhau như: nhận dạng ảnh vân tay, nhận dạng chữ viết.

1. Nén ảnh

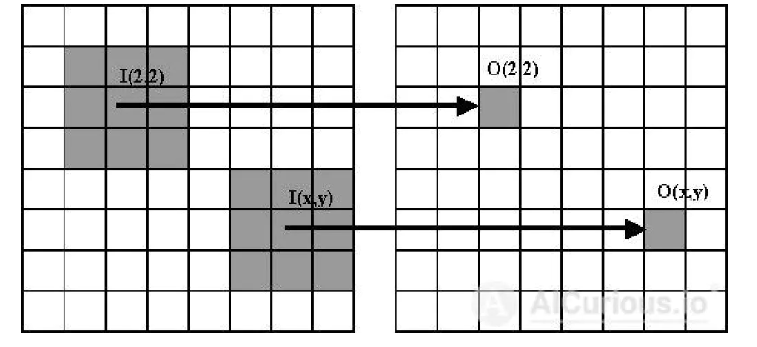
Dữ liệu ảnh cũng như các dữ liệu khác cần phải lưu trữ hay truyền đi trên mạng mà lượng thông tin để biểu diễn cho một ảnh là rất lớn. Do đó làm giảm lượng thông tin hay nén dữ liệu là một nhu cầu cần thiết.

Nén dữ liệu là quá trình làm giảm lượng thông tin “dư thừa” trong dữ liệu gốc và do vậy lượng thông tin thu được sau khi nén thường nhỏ hơn dữ liệu gốc rất nhiều.

* 1. **Tìm hiểu về các bộ lọc**

Nguyên tắc chung của các phương pháp lọc là cho ma trận ảnh nhân với một ma trận lọc (Kernel). Ma trận lọc lọc (Kernel) còn có thể được gọi là cửa số chập (trong phép nhân chập), cửa sổ lọc, mặt nạ, …

Việc nhân ảnh với ma trận lọc giống như việc trượt ma trận lọc theo hàng trên ảnh và nhân với từng vùng của ảnh, cộng các kết quả lại tạo thành kết quả của điểm ảnh trung tâm**.**



Hình 2: Nhân ma trận ảnh

Ma trận đầu vào I được nhân với ma trận lọc (phần xám ở hình trái) để tạo thành ma trận đầu ra O.

Trên thực tế, chúng ta sẽ thấy có 2 phép lọc ảnh là tương quan (correlation) và tích chập (convolution). Với phép tương quan, ma trận lọc sẽ đượt trượt đi và nhân với từng vùng của ảnh như trên. Tuy nhiên với phép tích chập, ma trận lọc sẽ được xoay 180 độ (theo cả chiều ngang và dọc) trước khi thực hiện nhân. 2 phép toán này là tương đương khi ma trận lọc đối xứng.

Với mỗi phép lọc ta có những ma trận lọc (Kernel) khác nhau, không có một quy định cụ thể nào cho việc xác định M. Kích thước ma trận M là một số lẻ. Ví dụ: 3x3, 5x5.

Khi nhân các phần tử tương ứng với nhau (giữa pixel, các điểm lân cận – các thành phần trong kernel), đối với các phần tử ở cạnh thì sẽ có một số pixel bị khuyết, lúc này, có nhiều cách giải quyết như bỏ qua, chèn thêm một (một số) hàng, cột mang giá trị 0 hoặc bằng giá trị gần nhất, hoặc tạo một đối xứng gương ở cạnh ảnh.

* + 1. *Bộ lọc Sobel*

Bộ lọc Sobel được sử dụng trong xử lý hình ảnh và thị giác máy tính chủ yếu với mục đích phát hiện cạnh. Về mặt hình thức là một phép toán vi phân để thu được độ dốc của hàm của cường độ ảnh. Sử dụng ma trận làm hạt nhân và áp dụng tích chập cho mỗi pixel trên ảnh gốc. Bộ lọc Sobel hoạt động với 2 ma trận hạt nhân, một cho các thay đổi theo chiều ngang và một cho các thay đổi theo chiều dọc. Nếu chúng ta định nghĩa Gx và Gy là hai hình ảnh chứa các đường ngang và xấp xỉ đạo hàm dọc tương ứng, các phép tính là:

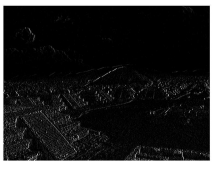
Trong đó A là ảnh nguồn ban đầu. Tọa độ x được xác định là tăng theo hướng bên phải và y tọa độ được xác định là tăng theo hướng giảm.

Tại mỗi điểm trong hình ảnh, kết quả của phép xấp xỉ gradient ngang và dọc có thể được kết hợp để thu được độ lớn của gradient, bằng cách:

Hướng của gradient được tính bằng cách sử dụng:

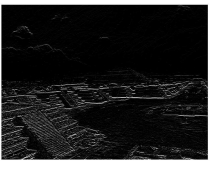
Giá trị 0 sẽ chỉ ra một cạnh thẳng đứng nhạt hơn ở bên trái cạnh.

Ví dụ 1:



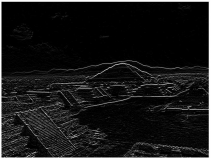
1. (b)

Hình 3:Ảnh gốc(a) và ảnh sau khi tính Gx(b)



1. (b)

Hình 4:Ảnh gốc(a) và ảnh sau khi tính Gy(b)



Hình 5: Sau khi kết hợp cả Gx và Gy

Ví dụ 2:

|  |  |  |  |
| --- | --- | --- | --- |
| 3 | 1 | 4 | 2 |
| 3 | 2 | 5 | 1 |
| 2 | 3 | 4 | 5 |
| 3 | 4 | 5 | 6 |

|  |  |  |
| --- | --- | --- |
| -1 | 0 | 1 |
| -2 | 0 | 2 |
| -1 | 0 | 1 |

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 1 |
| 0 | 0 | 0 |
| -1 | -2 | -1 |

Ta có:

Trong một số trường hợp gradient được tính đơn giản: G=10 với

Thuật toán Sobel có thể được tóm tắt trong bốn bước:

* Chuyển hình ảnh thành thang độ xám
* Chuyển đổi hình ảnh màu xám với bộ lọc Sobel-x
* Chuyển đổi hình ảnh màu xám với bộ lọc Sobel-y
* Tính toán độ lớn và hướng của gradient

Sơ đồ khối:

Diagram

Description automatically generated

Hình 6:Sơ đồ khối bộ lọc Sobel

### *1.2.2 Bộ lọc Gauss (Gaussian Filter)*

Bộ lọc Gauss được cho là bộ lọc hữu ích nhất, được thực hiện bằng cách nhân chập ảnh đầu vào với một ma trận lọc Gauss sau đó cộng chúng lại để tạo thành ảnh đầu ra.

* **Nguyên lý**

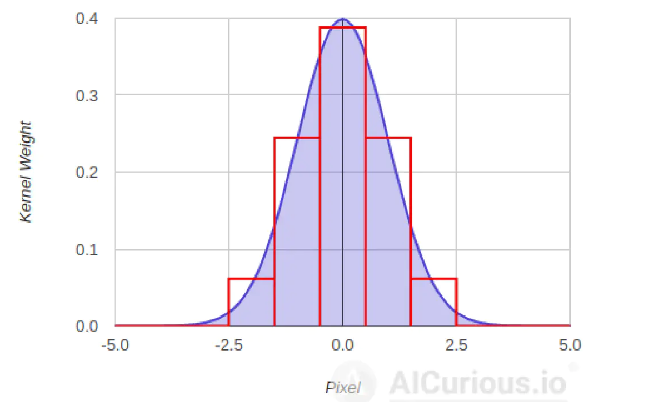
Ý tưởng chung là giá trị mỗi điểm ảnh sẽ phụ thuộc nhiều vào các điểm ảnh ở gần hơn là các điểm ảnh ở xa. Trọng số của sự phụ thuộc được lấy theo hàm Gauss (cũng được sử dụng trong quy luật phân phối chuẩn).

Phương trình hàm Gaussian dùng trong không gian một chiều và hai chiều:

(1 chiều)

(2 chiều)

Trong đó x và y là tọa độ theo hai trục đứng và ngang còn σ là phương sai chuẩn của phân bố Gaussian hay là giá trị quyết định độ lệch giữa các điểm trên bề mặt Gaussian.Trong phân bố Gaussian chuẩn thì giá trị kỳ vọng μ =0.



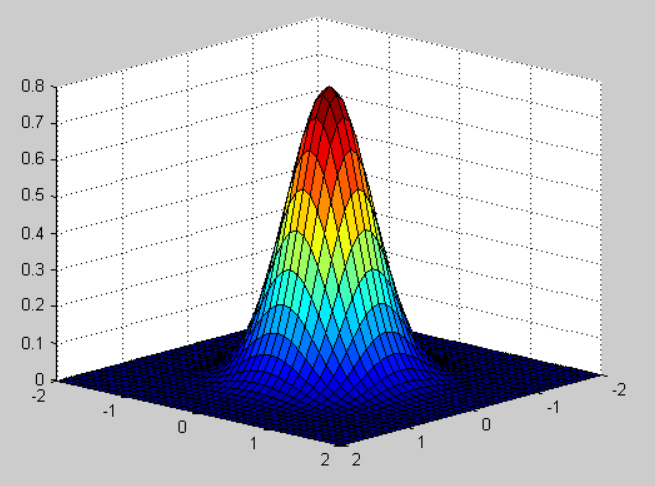
Hình 7: Ma trận lọc Gauss

Giả sử ảnh là một chiều. Điểm ảnh ở trung tâm sẽ có trọng số lớn nhất. Các điểm ảnh ở càng xa trung tâm sẽ có trọng số giảm dần khi khoảng cách từ chúng tới điểm trung tâm tăng lên. Như vậy điểm càng gần trung tâm sẽ càng đóng góp nhiều hơn vào giá trị điểm trung tâm.

Trên thực tế, việc lọc ảnh dựa trên hàm Gauss 2 chiều (ngang và dọc). Phân phối chuẩn 2 chiều có thể biểu diễn dưới dạng:

*Chú ý:* Tham số quyết định tác dụng của bộ lọc Gauss lên ảnh. Độ lớn của ma trận lọc (kernel) cần được lựa chọn cho đủ rộng.

Giá trị mới của mỗi Pixel sau khi tính tích chập với nhân đại diện cho hàm Gaussian có thể coi là trung bình lượng giá trị của các pixel xung quanh nó. Ta thấy rằng giá trị lượng giá của phần tử trung tâm kernel tương ứng với pixel đang xét là lớn nhất, giá trị này sẽ nhỏ hơn đối với các phần tử tương ứng với những pixel kế cận một cách đối xứng và tỉ lệ thuận với khoảng cách của phần tử này với trung tâm. Tính chất này giúp giữ lại đường viền và biên cũng như làm mờ một cách đồng bộ hơn so với các phương pháp khác.



Hình 8:Đường cong Gauss chuẩn hóa với giá trị và phương sai cho mảng 2 chiều

Từ hình 8 ta thấy khoảng tập trung của phân bố trong khoảng (-1.341,1.341) là 99.7%. Trong lý thuyết, hàm Gaussian tại mỗi điểm trên hình là khác 0. Điều này có nghĩa là Gaussian Kernel nên có kích thước bằng với hình ảnh và giá trị tại mỗi phần tử luôn khác 0. Tuy nhiên trong thực hành, do việc tính tóan dựa trên xấp xỉ rời rạc (Discrete Approximation) cho nên giá trị của các phần tử trên bề mặt Gaussian, tập trung trong khoảng 3 σ quanh tâm phân phối chuẩn chiếm 99.7% ở khoảngg cách lớn hơn 3σ so với trung tâm gần như không đáng kể (tiệm cận 0). Do vậy các phân tán Gaussian ngòai bán kính này sẽ bị bỏ qua, đó cũng là lý do mà thông thường Gaussian kernel có kích thước giới hạn 3, 5, 7... tùy vào giá trị phương sai chuẩn được chọn. Khoảng cách giữa hai điểm gần nhau trong Gaussian Kernel là σ.

### *1.2.3 Bộ lọc Median*

Bộ lọc trung vị đã được sử dụng trong nhiều năm trong quá trình xử lý hình ảnh, do tính đơn giản và khả năng loại bỏ các "đốm" trong hình ảnh trong khi vẫn bảo toàn thông tin rìa. Điều này trái ngược với các kỹ thuật lọc tuyến tính cho những hình ảnh này có xu hướng làm mờ hình ảnh. Các bộ lọc trung vị như vậy có thể hiệu quả và rất đơn giản để triển khai cho các ứng dụng một chiều.

* **Nguyên lý**

Khác với bộ lọc trung bình, nó không xảy ra bất cứ phép tính toán gì, nó chỉ đơn giản là chỉ ra đâu là giá trí nằm ở vị trí trung bình trong các giá trị đầu vào được thiết lập bởi khung lọc rồi lấy giá trị đó thế vào tâm của khung lọc trong ảnh gốc để tạo ra giá trị ảnh mới. Hiệu quả của phép lọc có khi không rõ ràng, khó có thể nhận ra được sự thật đằng sau phép biến đổi vì nó chỉ thay đổi một số điểm ảnh thôi chứ không phải toàn bộ các điểm ảnh nhưng nó có ưu thế là sự sửa chữa mạnh mẽ một số điểm bất thường trên ảnh mà các bộ lọc khác không làm được hoặc dễ dàng bỏ qua.

Bộ lọc trung vị được thực hiện như sau:

Bước 1. Liệt kê các giá trị pixel trong cửa sổ ảnh thành mảng một chiều.

Bước 2. Sắp xếp tất cả các giá trị pixel theo trật tự tăng dần hoặc giảm dần.

Bước 3. Thay thế giá trị pixel đang xem xét với giá trị trung vị(giữa) của mảng.

Ví dụ 1:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 123 | 125 | 126 | 130 | 140 |
| 122 | 125 | 124 | 127 | 135 |
| 118 | 120 | 150 | 125 | 134 |
| 119 | 115 | 119 | 123 | 133 |
| 111 | 116 | 110 | 120 | 130 |

Các giá trị điểm ảnh: 123,125,126,122,125,124,118,120,150

Sắp xếp các điểm ảnh theo thứ tự tăng dần:

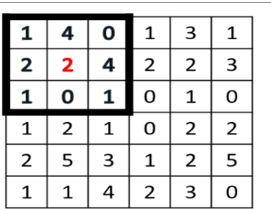
Với các giá trị 118,120,122,123,124,125,125,126, 150 ta thấy Median value:124.

Giá trị trung bình ở vị trí trung tâm sẽ thay đổi.

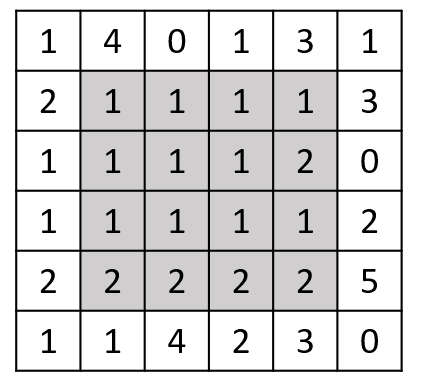
Kết quả:

|  |  |  |
| --- | --- | --- |
| 123 | 125 | 126 |
| 122 | 124 | 124 |
| 118 | 120 | 150 |

Ví dụ 2:



Hình 9: Ma trận ảnh ban đầu



Hình 10:Kết quả

Ví dụ 3:



Hình 11:Chập với nhân 3\*3,5\*5,9\*9 của bộ lọc trung vị

Ưu điểm:

* Không giảm độ tương phản giữa các bước, vì các giá trị đầu ra có sẵn chỉ bao gồm các giá trị hiện diện trong vùng lân cận (không có giá trị trung bình) .
* Lọc trung vị không làm thay đổi ranh giới, như có thể xảy ra với các bộ lọc làm mịn thông thường (vấn đề phụ thuộc vào độ tương phản) .
* Vì giá trị trung bình ít nhạy cảm hơn giá trị trung bình đối với các giá trị cực trị (giá trị ngoại lệ), các giá trị cực trị đó được loại bỏ hiệu quả hơn.

### *1.2.4 Bộ lọc Bilateral (bộ lọc song phương)*

Mặc dù làm mờ cũng được xem là một cách hiệu quả để giảm nhiễu trong hình ảnh, nhưng thường không nên làm mờ toàn bộ hình ảnh, vì các chi tiết quan trọng và các cạnh sắc nét có thể bị mất. Trong những trường hợp như vậy, bộ lọc song phương có thể hiệu quả hơn.

– Phương pháp này lọc một cách có lựa chọn để làm mờ các pixel có cường độ tương tự trong một vùng lân cận. Các cạnh được giữ nguyên.

– Phương pháp này cho phép kiểm soát không chỉ kích thước không gian của bộ lọc mà còn cả mức độ bao gồm các pixel lân cận trong đầu ra được lọc. Điều này được thực hiện, dựa trên sự thay đổi về cường độ màu của chúng và cũng như khoảng cách từ pixel được lọc.

Lọc song phương về cơ bản áp dụng phương pháp làm mờ Gaussian 2D (có trọng số) cho hình ảnh, đồng thời xem xét sự thay đổi về cường độ của các pixel lân cận để giảm thiểu việc làm mờ gần các cạnh (mà muốn giữ nguyên). Điều này có nghĩa là dạng của mặt nạ phụ thuộc vào nội dung hình ảnh cục bộ, tại mọi vị trí pixel.

Gaussian có thể được xây dựng như sau:

Trong đó:

Ví dụ: Một điểm ảnh sáng có giá trị lớn hơn một điểm ảnh tối liền kề mặc dù hai giá trị điểm ảnh này khá khác nhau. Kết quả là, các cạnh hình ảnh bị mờ vì các pixel trên các điểm không liên tục được tính trung bình với nhau. Do đó, một pixel có trên một pixel khác chỉ phụ thuộc vào khoảng cách của chúng trong hình ảnh, không phụ thuộc vào các giá trị hình ảnh thực tế.



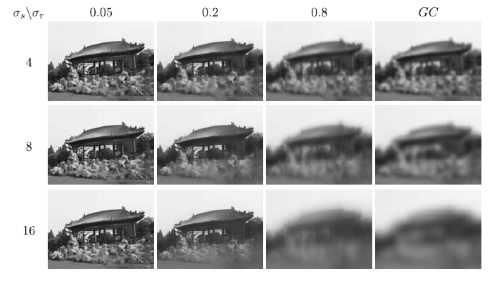
Hình 12: Phân phối tuyến tính Gaussian với các giá trị sigma khác nhau

là kết quả tại pixel p và RHS về cơ bản là tổng trên tất cả các pixel q được tính trọng số bởi hàm Gaussian. là cường độ tại pixel q.

Bộ lọc Bilateral sẽ được xây dựng:

Khi:

Ở đây, hệ số chuẩn hóa và trọng số phạm vi là các thuật ngữ mới được thêm vào phương trình trước đó. biểu thị phạm vi không gian của hạt nhân, tức là kích thước của vùng lân cận, và biểu thị biên độ tối thiểu của một cạnh. Nó đảm bảo rằng chỉ những pixel có giá trị cường độ tương tự như của pixel trung tâm mới được xem xét để làm mờ, trong khi các thay đổi cường độ sắc nét vẫn được duy trì. Giá trị của càng nhỏ , cạnh càng sắc nét. Khi có xu hướng đến vô cùng, phương trình có xu hướng mờ Gaussian.



Hình 13: Phạm vi và thông số hoạt động của Bilateral cung cấp khả năng điều khiển linh hoạt hơn Gaussian

Ngay sau khi một trong hai trọng lượng song phương đạt đến giá trị gần bằng không, không có hiện tượng làm mịn nào xảy ra. Do đó, việc tăng sigma không gian sẽ không làm mờ cạnh miễn là phạm vi sigma nhỏ hơn biên độ cạnh. Ví dụ, các đường viền trên mái nhà là sắc nét đối với các cài đặt phạm vi nhỏ và vừa phải , và độ sắc nét đó độc lập với cài đặt không gian. Các giá trị cường độ hình ảnh gốc nằm trong khoảng [0,1].

### *1.2.5 Mean filter*

Mean filtering là phương pháp lọc khá đơn giản, trực quan và dễ thực hiện việc làm mịn hình ảnh, nghĩa là giảm cường độ biến đổi giữa một pixel và pixel kế tiếp. Nó thường được dùng để giảm nhiễu trong ảnh. Thuật toán này cũng nổi tiếng là làm mờ hộp và thực hiện dưới dạng tích chập với mặt nạ làm mờ trung bình.

* **Nguyên lý:**

Ý tưởng của bộ lọc này đơn giản chỉ là thay thế từng giá trị pixel trong ảnh với các giá trị trung bình của các pixel “hàng xóm”, bao gồm chính nó. Điều này có tác dụng loại bỏ các giá trị pixel không đại diện cho môi trường xung quanh. Mean filtering thường được coi như là convolution filter, chúng giống nhau ở chỗ cùng dựa trên một hạt nhân đại diện cho tất cả các hình dạng kích thước của cả khối được lấy mẫu. Thường thì người ta sử dụng hạt nhân 3\*3 hình vuông. Hạt nhân 5\*5 cũng có thể được sử dụng và nó làm mịn nặng hơn 3\*3. Giống như toán tử Gaussian, bộ lọc này cũng có thể phân tách được, vì vậy thường ở kích thước lớn hơn, tốt hơn nên chạy nó một lần cho các hàng và một lần cho các cột.

|  |  |  |
| --- | --- | --- |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |
| 1/9 | 1/9 | 1/9 |

Ví dụ:



1. (b)

Hình 14:Ảnh gốc (a) và ảnh bị nhiễu (b)



1. (b)

Hình 15:Ảnh sử dụng lọc 3\*3(a) và ảnh sử dụng lọc 5\*5(b)

Rõ ràng khi sử dụng bộ lọc có kích thước 3\*3 thì nhiễu giảm đi nhưng ảnh “mềm” hơn. Nếu chúng ta tăng kích thước bộ lọc lên 5\*5 thì ảnh sẽ có rất ít nhiễu và ít các thành phần tần số cao (hình 6.b). Ta thấy ảnh trong hình 6.b ảnh ko giảm nhiễu nhiều nhưng độ mờ thì giảm đi đáng kể.

*Nhận xét:*

* Một pixel có giá trị khác nhiều với các giá trị khác xung quanh sẽ ảnh hưởng đến tất cả điểm ảnh trong khối của nó.
* Với các khối ở cạnh, góc bộ lọc sẽ phải nội suy ra giá trị cho các điểm ảnh, từ đó sẽ làm cho ảnh bị mờ góc cạnh, rất không tốt cho ảnh có yêu cầu hình ảnh sắc nhọn ở đầu ra.
* Cả 2 vấn đề có thể được giải quyết với median filter, nhưng bù lại nó có thời gian tính toán nhanh hơn.

# CHƯƠNG 2: XÂY DỰNG CÁC BỘ LỌC XỬ LÝ ẢNH

Trong báo cáo này chúng em trình bày việc xây dựng 5 bộ lọc xử lý ảnh và thiết kế một Photo\_app dựa trên bộ lọc Sobel

## 2.1 Xây dựng các hàm triển khai các bộ lọc

### *2.1.1 Bộ lọc Sobel*

|  |
| --- |
| def Sobel():  # Here we read the image and bring it as an array  filename = os.path.basename(fname[0])  path = os.path.dirname(fname[0])  name, type = os.path.splitext(filename)  im = Image.open(fname[0])  new\_path = path + name + ".png"  im.save(new\_path)  original\_image = imread(new\_path)  # Next we apply the Sobel filter in the x and y directions to then calculate the output image  dx,dy=ndimage.sobel(original\_image,axis=0),ndimage.sobel(original\_image, axis=1)  sobel\_filtered\_image = np.hypot(dx, dy) # is equal to (dx ^ 2 + dy ^ 2) ^ 0.5  sobel\_filtered\_image = sobel\_filtered\_image / np.max(sobel\_filtered\_image) # normalization step  # Display  plt.subplot(121), plt.imshow(original\_image), plt.title('Original')  plt.xticks([]), plt.yticks([])  plt.subplot(122), plt.imshow(sobel\_filtered\_image), plt.title('Sobel Filter')  plt.xticks([]), plt.yticks([])  plt.show() |

### *2.1.2 Bộ lọc Gaussian*

|  |
| --- |
| def Gaussian(self):  # Load and blur image  # filename = os.path.basename(fname[0])  path = os.path.dirname(fname[0])  img = cv.imread(path)  blur = cv.GaussianBlur(img, (5, 5), 0)  # Convert color from bgr (OpenCV default) to rgb  img\_rgb = cv.cvtColor(img, cv.COLOR\_BGR2RGB)  blur\_rgb = cv.cvtColor(blur, cv.COLOR\_BGR2RGB)  # Display  plt.subplot(221), plt.imshow(img\_rgb), plt.title('Gauss Noise')  plt.xticks([]), plt.yticks([])  plt.subplot(222), plt.imshow(blur\_rgb), plt.title('Gauss Noise - Blurred')  plt.xticks([]), plt.yticks([])  plt.show() |

### *2.1.3 Bộ lọc Median*

|  |
| --- |
| def Median(self):  # Load and blur image  path = os.path.dirname(fname[0])  img = cv.imread(path)  blur = cv.medianBlur(img, 5)  # Convert color from bgr (OpenCV default) to rgb  img\_rgb = cv.cvtColor(img, cv.COLOR\_BGR2RGB)  blur\_rgb = cv.cvtColor(blur, cv.COLOR\_BGR2RGB)  # Display  plt.subplot(221), plt.imshow(img\_rgb), plt.title('Median')  plt.xticks([]), plt.yticks([])  plt.subplot(222), plt.imshow(blur\_rgb), plt.title('Median - Blurred')  plt.xticks([]), plt.yticks([])  plt.show() |

### *2.1.4 Bộ lọc Mean Filter*

|  |
| --- |
| def Normalized(self):  # Load and blur image  path = os.path.dirname(fname[0])  img = cv.imread(path)  blur = cv.blur(img, (5, 5))  # Convert color from bgr (OpenCV default) to rgb  img\_rgb = cv.cvtColor(img, cv.COLOR\_BGR2RGB)  blur\_rgb = cv.cvtColor(blur, cv.COLOR\_BGR2RGB)  # Display  plt.subplot(221), plt.imshow(img\_rgb), plt.title('Normalized')  plt.xticks([]), plt.yticks([])  plt.subplot(222), plt.imshow(blur\_rgb), plt.title('Normalized - Blurred')  plt.xticks([]), plt.yticks([])  plt.show() |

### *2.1.5 Bộ lọc Bilateral*

|  |
| --- |
| def Bilateral(self):  # Load and blur image  path = os.path.dirname(fname[0])  img = cv.imread(path)  blur = cv.bilateralFilter(img, 9, 75, 75)  # Convert color from bgr (OpenCV default) to rgb  img\_rgb = cv.cvtColor(img, cv.COLOR\_BGR2RGB)  blur\_rgb = cv.cvtColor(blur, cv.COLOR\_BGR2RGB)  # Display  plt.subplot(221), plt.imshow(img\_rgb), plt.title('Bilateral')  plt.xticks([]), plt.yticks([])  plt.subplot(222), plt.imshow(blur\_rgb), plt.title('Bilateral - Blurred')  plt.xticks([]), plt.yticks([])  plt.show() |

### *2.1.6 Xây dựng lớp gán Nhãn*

|  |
| --- |
| class Label(QLabel):  def \_\_init\_\_(self):  super(Label, self).\_\_init\_\_()  self.pixmap\_width: int = 1  self.pixmapHeight: int = 1  def setPixmapNew(self, pm: QPixmap) -> None:  self.pixmap\_width = pm.width()  self.pixmapHeight = pm.height()  self.updateMargins()  super(Label, self).setPixmap(pm)  def resizeEvent(self, a0: QResizeEvent) -> None:  self.updateMargins()  super(Label, self).resizeEvent(a0)  def updateMargins(self):  if self.pixmap() is None:  return  pixmapWidth = self.pixmap().width()  pixmapHeight = self.pixmap().height()  if pixmapWidth <= 0 or pixmapHeight <= 0:  return  w, h = self.width(), self.height()  if w <= 0 or h <= 0:  return  if w \* pixmapHeight > h \* pixmapWidth:  m = int((w - (pixmapWidth \* h / pixmapHeight)) / 2)  self.setContentsMargins(m, 0, m, 0)  else:  m = int((h - (pixmapHeight \* w / pixmapWidth)) / 2)  self.setContentsMargins(0, m, 0, m) |

### *2.1.7 Lớp chỉnh sửa ảnh class EditWindow*

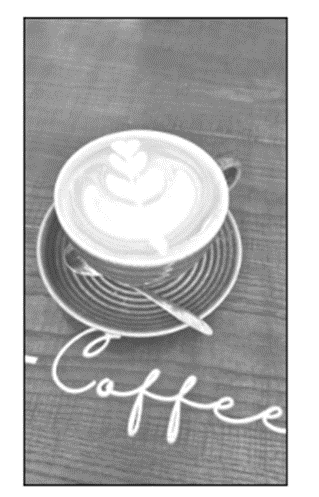
|  |
| --- |
| class EditWindow(QDialog):  def \_\_init\_\_(self):  super(EditWindow, self).\_\_init\_\_()  loadUi("edit\_window.ui", self)  self.show\_img()  self.btn\_back.clicked.connect(self.back)  self.Sobel.clicked.connect(Sobel)  self.Invert.clicked.connect(self.Invert)  # self.exposure.valueChanged.connect(self.show\_value)  self.exposure.valueChanged.connect(self.Exposure\_gfc)  self.btn\_cancel.clicked.connect(self.cancel)  def back(self):  back = MainWindow()  @@ -162,51 +163,85 @@ def show\_img(self):  self.display.setPixmap(QPixmap(fname[0]).scaled(1041, 721, QtCore.Qt.KeepAspectRatio))  self.label\_name.setText(fname[0])  def cancel(self):  cacel()  self.back()  def show\_value(self):  new\_value = str(self.exposure.value())  self.display.setText(new\_value)  def Invert(self):  img = cv2.imread(fname[0], cv2.IMREAD\_UNCHANGED)  img1 = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  # Create zeros array to store the stretched image  new\_img = np.zeros((img1.shape[0], img1.shape[1]), dtype='uint8')  for i in range(img1.shape[0]):  for j in range(img1.shape[1]):  new\_img[i, j] = 255 - img1[i, j]  new\_path = path + name + "\_tmp.png"  f = open(new\_path, 'wb')  (new\_img)  self.display.setPixmap(QPixmap(new\_path).scaled(1041, 721, QtCore.Qt.KeepAspectRatio))  def Exposure\_gfc(self):  # Read the image  img = cv2.imread(fname[0], cv2.IMREAD\_UNCHANGED)  img1 = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)  k = 50  # Create zeros array to store the stretched image  new\_img = np.zeros((img1.shape[0], img1.shape[1]), dtype='uint8')  for i in range(img1.shape[0]):  for j in range(img1.shape[1]):  v = img1[i, j] + k  if v > 255:  new\_img[i, j] = 255  elif v < 0:  new\_img[i, j] = 0  else:  new\_img[i, j] = v  new\_path = path + name + "\_tmp.png"  f = open(new\_path, 'wb')  f.write(new\_img)  self.display.setPixmap(QPixmap(new\_path).scaled(1041, 721, QtCore.Qt.KeepAspectRatio)) |

## 2.2 Xây dựng Photo\_app

Dựa trên Sobel Filter:

|  |
| --- |
| def SobelFilter(image, gauss\_ksize=5, dx=1, dy=1, threshold=60):  # GaussKernel = createGaussianKernel(gauss\_ksize,1)  X = np.array([[1, 0, -1], [2, 0, -2], [1, 0, -1]])  Y = np.array([[1, 2, 1], [0, 0, 0], [-1, -2, -1]])  # kernel\_x = convolute(GaussKernel,X)  sobel\_x = convolute(image, X)  # kernel\_y = convolute(GaussKernel,Y)  sobel\_y = convolute(image, Y)  if dx == 1 and dy == 0:  return sobel\_x  if dx == 0 and dy == 1:  return sobel\_y  # abs\_sobel\_x = np.absolute(sobel\_x)  # img\_sobel\_x = np.uint8(abs\_sobel\_x)  # abs\_sobel\_y = np.absolute(sobel\_y)  # img\_sobel\_y = np.uint8(abs\_sobel\_y)  # sobelxy = img\_sobel\_x + img\_sobel\_y  # sobelxy = np.sqrt(np.square(sobel\_x) + np.square(sobel\_y))  abs\_grad\_x = cv2.convertScaleAbs(sobel\_x)#|src(I)∗alpha+beta|  abs\_grad\_y = cv2.convertScaleAbs(sobel\_y)  # sobel\_xy = cv.addWeighted(abs\_grad\_x, 0.5, abs\_grad\_y, 0.5, 0)  alpha = 0.5  beta = 0.5  gamma = 0  sobel\_xy = abs\_grad\_x \* alpha + abs\_grad\_y \* beta + gamma  # sobel = cv.addWeighted(abs\_sobel\_x, 0.5, abs\_sobel\_y, 0.5, 0)  # [src1, alpha, src2, beta, gamma[, dst[, dtype]]  # dst = src1\*alpha + src2\*beta + gamma  # void cv::addWeighted ( InputArray src1, double alpha, InputArray src2, double beta,double gamma,OutputArray dst,int dtype = -1)  # sobel = sobel\_x + sobel\_y  # sobel = image + sobel  # Loại bỏ những pixel yếu để tăng độ sắc nét của cạnh  # sobelxy = np.float64(sobel\_xy)  # for i in range(sobelxy.shape[0]):  # for j in range(sobelxy.shape[1]):  # if sobelxy[i][j] < threshold:  # sobelxy[i][j] = 0  # else:  # sobelxy[i][j] = 255  return sobel\_xy |

## 2.3 Kết quả



Hình 16:Ảnh ban đầu



Hình 17:Ảnh quả xử lý thông qua Photo\_app

# KẾT LUẬN

Sau khi nhóm em tìm hiểu và hoàn thiện đề tài, nhóm đã tìm hiểu được nguyên lý thực hiện của các bộ lọc, xây dựng chương trình tạo ra các bộ lọc và tạo được một Photo\_app dựa trên kiến thức về các bộ lọc và đặc biệt ở đây là sử dụng bộ lọc Sobel để thực hiện. Tuy nhiên bên cạnh đó nhóm em vẫn còn một vài những hạn chế về phần code tự triển khai theo ý hiểu và tham khảo một số code trên mạng nên không tránh khỏi những sai sót, ứng dụng còn đơn giản và chưa đưa vào sử dụng.

Qua đây nhóm chúng em xin chân thành cảm ơn cô Lê Thị Lan đã hướng dẫn, giúp đỡ bọn em trong quá trình học cũng như thực hiện đề tài. Chúng em mong muốn nhận được những góp ý của cô để đề tài của chúng em hoàn thiện hơn.

# TÀI LIỆU THAM KHẢO

[1] “Nhập môn xử lý ảnh( NXB Khoa Học Kỹ Thuật 2007)” \_ Lương Mạnh Bá

[2] “Lọc ảnh (Image Filtering)” \_AlCurious.io*,* [*https://aicurious.io/posts/2018-09-29-loc-anh-image-filtering*](https://aicurious.io/posts/2018-09-29-loc-anh-image-filtering)

[3] ” Image filtering and morphology” \_ E.R.Davies, [*https://doi.org/10.1016/B978-0-12-809284-2.00003-4*](https://doi.org/10.1016/B978-0-12-809284-2.00003-4)

[4] Online available: *https://binarymindset.com/image-processing-with-median-filter/*