TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI

KHOA CÔNG NGHỆ THÔNG TIN

🙥🕮🙧



BÁO CÁO BÀI TẬP LỚN

XỬ LÝ ẢNH

NHÓM 4

Đề tài: Tìm hiểu các phép lọc trên miền tần số, khảo sát và xây dựng ứng dụng của phép lọc trên miền tấn số đối với ảnh đa mức xám.

**Giảng viên hướng dẫn:**

**Sinh viên thực hiện:**

|  |  |  |
| --- | --- | --- |
| STT | Họ và tên | Mã sinh viên |
| 1 | Nguyễn Quỳnh Chi | 211201577 |
| 2 | Đỗ Thị Hải | 211243882 |
| 3 | Trịnh Thành Nam | 211201058 |
| 4 | Nguyễn Hoài Sơn | 211213561 |

**Lớp: Công nghệ thông tin 1 – Khoá 62**

**Hà Nội – 2023**

LỜI NÓI DẦU

Trong kỷ nguyên số, hình ảnh đã trở thành một phần không thể thiếu trong cuộc sống của chúng ta. Tuy nhiên, chất lượng hình ảnh thường bị ảnh hưởng bởi nhiễu, mờ và các yếu tố không mong muốn khác. Để cải thiện chất lượng hình ảnh và trích xuất thông tin hữu ích, các kỹ thuật xử lý ảnh đã được phát triển mạnh mẽ.

Một trong những kỹ thuật quan trọng là lọc ảnh trên miền tần số. Thay vì tác động trực tiếp lên các điểm ảnh như trong miền không gian, lọc trên miền tần số cho phép chúng ta phân tích và điều chỉnh các thành phần tần số khác nhau của hình ảnh. Bằng cách này, chúng ta có thể dễ dàng loại bỏ nhiễu có tính chọn lọc, tăng cường các chi tiết quan trọng, và thực hiện nhiều phép biến đổi phức tạp khác.

Báo cáo này sẽ trình bày chi tiết về các kỹ thuật lọc ảnh trên miền tần số, bao gồm lý thuyết nền tảng, các loại bộ lọc phổ biến, và các ứng dụng thực tế. Chúng ta sẽ cùng nhau tìm hiểu làm thế nào để thiết kế và áp dụng các bộ lọc hiệu quả để cải thiện chất lượng hình ảnh. Cuối cùng, chúng ta sẽ thảo luận về những thách thức và hướng phát triển trong tương lai của lĩnh vực này.

MỤC LỤC

[CHƯƠNG 1: GIỚI THIỆU VỀ PHÉP LỌC TRÊN MIỀN TẦN SỐ 1](#_Toc179674453)

[1.1. Giới thiệu về miền tần số 1](#_Toc179674454)

[1.1.1. Miền tần số 1](#_Toc179674455)

[1.1.2. Sự khác biệt giữa miền không gian và miền tần số 1](#_Toc179674456)

[1.1.3. Lợi ích của việc xử lý ảnh trên miền tần số 1](#_Toc179674457)

[1.2. Biến đổi Fourier trong xử lý ảnh 1](#_Toc179674458)

[1.2.1. Khái niệm 1](#_Toc179674459)

[1.2.2. Biến đổi Fourier liên tục 2](#_Toc179674460)

[1.2.3. Biến đổi Fourier rời rạc 3](#_Toc179674461)

[CHƯƠNG 2: LỌC ẢNH TRONG MIỀN TẦN SỐ 5](#_Toc179674462)

[2.1. Khái niệm 5](#_Toc179674463)

[2.2. Tầm quan trọng của lọc ảnh trên miền tần số 5](#_Toc179674464)

[2.2.1. Loại bỏ nhiễu: 5](#_Toc179674465)

[2.2.2. Nâng cao chất lượng ảnh: Tăng cường cạnh, làm mịn ảnh. 6](#_Toc179674466)

[2.2.3. Phân tích đặc trưng: 6](#_Toc179674467)

[2.3. Các bước cơ bản của lọc ảnh trong miền tần số 6](#_Toc179674468)

[2.4. Các bộ lọc cụ thể 7](#_Toc179674469)

[2.4.1. Bộ lọc thông thấp lý tưởng 7](#_Toc179674470)

[2.4.2. Bộ lọc thông thấp Butterworth 9](#_Toc179674471)

[2.4.3. Bộ lọc thông thấp Gaussian 10](#_Toc179674472)

[2.4.4. Bộ lọc thông cao lý tưởng 11](#_Toc179674473)

[2.4.5. Bộ lọc thông cao Butterworth 12](#_Toc179674474)

[2.4.6. Bộ lọc thông cao Gaussian 13](#_Toc179674475)

[CHƯƠNG 3: ỨNG DỤNG THỰC TẾ 15](#_Toc179674476)

[3.1. Ảnh đa mức xám và yêu cầu xử lý 15](#_Toc179674477)

[3.2. Áp dụng phép lọc thông thấp trong làm mịn ảnh 15](#_Toc179674478)

[3.3. Áp dụng phép lọc thông cao trong tăng cường chi tiết ảnh 15](#_Toc179674479)

[3.4. Lọc các tần số cụ thể để giảm nhiễu 16](#_Toc179674480)

[3.5. Ứng dụng trong các lĩnh vực khác 16](#_Toc179674481)

[CHƯƠNG 4: Xây dựng các thuật toán lọc ảnh 17](#_Toc179674482)

[4.1. Hàm lọc thông thấp Ideal 17](#_Toc179674483)

[4.2. Hàm lọc thông thấp Gauss 19](#_Toc179674484)

[4.3. Hàm lọc thông thấp Butterworth 23](#_Toc179674485)

[4.4. Hàm lọc thông cao Ideal 26](#_Toc179674486)

[4.5. Hàm lọc thông cao Gauss 29](#_Toc179674487)

[4.6. Hàm lọc thông cao Butterworth 33](#_Toc179674488)

[KẾT LUẬN 37](#_Toc179674489)

[TÀI LIỆU THAM KHẢO 38](#_Toc179674490)

MỤC LỤC HÌNH ẢNH

[Hình 1. Hình ảnh minh họa biến đổi Fourierl 2](#_Toc179674651)

[Hình 2. Biến đổi Fourier của ảnh 3](#_Toc179674652)

[Hình 3. Quá trình thực hiện lọc ảnh trong miền tần số 7](#_Toc179674653)

[Hình 4. Ảnh phổ của bộ lọc thông thấp lý tưởng 8](#_Toc179674654)

[Hình 5. Ảnh gốc và ảnh sau khi cho qua các bộ lọc với tần số cắt khác nhau 8](#_Toc179674655)

[Hình 6. Hình ảnh bộ lọc thông thấp Butterworth: (a) Hình ảnh 3 chiều; 9](#_Toc179674656)

[(giữa) ảnh phổ của bộ lọc; (trái) mặt cắt đứng của bộ lọc. 9](#_Toc179674657)

[Hình 7. Hình ảnh sau khi qua bộ lọc Butterworth 9](#_Toc179674658)

[Hình 8. Hình ảnh bộ lọc thông thấp Gaussian: (a) Hình ảnh 3 chiều; (giữa) 10](#_Toc179674659)

[ảnh phổ của bộ lọc; (trái) mặt cắt của bộ lọc với các tần số cắt khác nhau. 10](#_Toc179674660)

[Hình 9. Hình ảnh sau khi cho qua bộ lọc thông thấp Gaussian 11](#_Toc179674661)

[Hình 10. Hình ảnh bộ lọc thông cao lý tưởng: (a) Hình ảnh 3 chiều; 12](#_Toc179674662)

[(giữa) ảnh phổ của bộ lọc; (trái) mặt cắt đứng của bộ lọc. 12](#_Toc179674663)

[Hình 11. Hình ảnh sau khi qua bộ lọc thông cao lý tưởng 12](#_Toc179674664)

[Hình 12. Hình ảnh bộ lọc thông cao Butterword: (a) Hình ảnh 3 chiều; 13](#_Toc179674665)

[(giữa) ảnh phổ của bộ lọc; (trái) mặt cắt đứng của bộ lọc. 13](#_Toc179674666)

[Hình 13. Hình ảnh sau khi qua bộ lọc thông cao lý tưởng 13](#_Toc179674667)

[Hình 14. Hình ảnh bộ lọc thông cao Gaussian: (a) Hình ảnh 3 chiều; 14](#_Toc179674668)

[(giữa) ảnh phổ của bộ lọc; (trái) mặt cắt đứng của bộ lọc. 14](#_Toc179674669)

[Hình 15. Hình ảnh sau khi qua bộ lọc thông cao Gaussian 14](#_Toc179674670)

[Hình 16. Kết quả khởi chạy hàm lọc thông thấp Ideal 19](#_Toc179674671)

[Hình 17. Kết quả hàm lọc thông thấp Gauss 23](#_Toc179674672)

[Hình 18. Kết quả bộ lọc thông thấp Butterworth 26](#_Toc179674673)

[Hình 19. Kết quả hàm lọc thông cao Ideal 29](#_Toc179674674)

[Hình 20. Kết quả lọc thông cao Gauss 33](#_Toc179674675)

[Hình 21. Kết quả bộ lọc thông cao Butterworth 36](#_Toc179674676)

1. GIỚI THIỆU VỀ PHÉP LỌC TRÊN MIỀN TẦN SỐ
   1. Giới thiệu về miền tần số
      1. Miền tần số

Miền tần số là cách biểu diễn dữ liệu bằng cách phân tích các thành phần tần số của nó, thường sử dụng biến đổi Fourier để chuyển từ miền không gian (biểu diễn theo tọa độ) sang miền tần số (biểu diễn theo chu kỳ hoặc tần số dao động).

* + 1. Sự khác biệt giữa miền không gian và miền tần số

Trong miền không gian, ta xử lý trực tiếp trên từng điểm ảnh, còn trong miền tần số, ta xử lý dựa trên tốc độ thay đổi giá trị ảnh trên miền không gian.

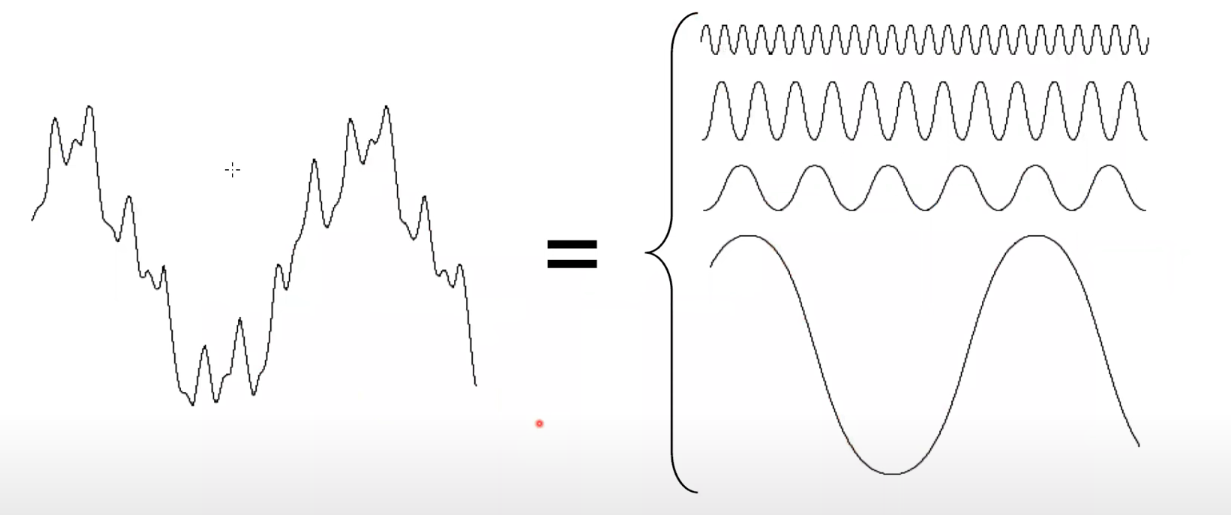
* Miền không gian: Ma trận ảnh đầu vào → Xử lý→  Ma trận ảnh đầu ra.
* Miền tần số: Ảnh vào→  Phân bố tần số→  Xử lý → Chuyển đổi ngược  → Ảnh ra.

Tổng quát hóa ta có bảng so sánh sự khác biệt:

|  |  |  |
| --- | --- | --- |
| Đặc điểm | Miền không gian | Miền tần số |
| Biểu diễn | Giá trị mỗi điểm | Thành phần tần số |
| Thông tin | Thông tin về không gian (vị trí) | Thông tin về tần số (biến đổi theo thời gian hoặc không gian) |
| Ứng dụng | Xử lý các thao tác trực tiếp trên pixel | Xử lý các thành phần tần số, lọc nhiễu, tăng cường đặc trưng |

* + 1. Lợi ích của việc xử lý ảnh trên miền tần số
* Hiệu quả tính toán cao, đặc biệt với các phép lọc tuyến tính.
* Linh hoạt trong thiết kế bộ lọc để đáp ứng các yêu cầu khác nhau.
* Dễ dàng phân tích các thành phần tần số của ảnh.
  1. Biến đổi Fourier trong xử lý ảnh
     1. Khái niệm

**Một hàm bất kỳ lặp lại có tính chu kỳ có thể biểu diễn dưới dạng tổng các hàm sine và cosine ở các tần số khác nhau – đó là chuỗi Fourier.**



1. Hình ảnh minh họa biến đổi Fourierl
   * 1. Biến đổi Fourier liên tục
        1. Hàm 1 biến

Biến đổi Fourier F(u) của một hàm số liên tục f(x) có biến x liên tục được định nghĩa bởi công thức:

Ngược lại, cho trước hàm F(u), ta có thể tìm lại f(x) bằng công thức biến đổi Fourier ngược như sau:

* + - 1. Hàm 2 biến

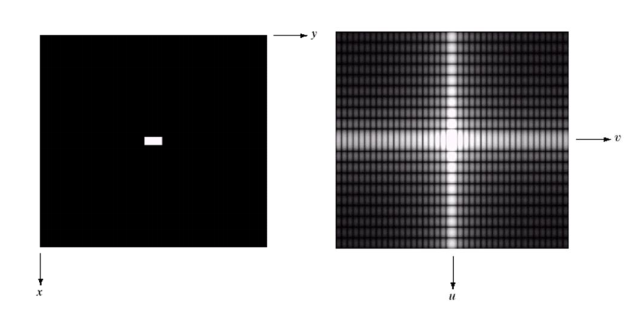
Tương tự, với hàm 2 biến, ta có cặp biến đổi Fourier thuận/nghịch như sau:

* + 1. Biến đổi Fourier rời rạc

Với hàm rời rạc 2 chiều, phép biến đổi Fourier thuận/nghịch của hàm *f(x,y)* biểu diễn mức xám của một ảnh có kích thước M x N được cho bởi công thức:

Giá trị của phép biến đổi tại giá trị (u,v) = (0,0) được cho bởi công thức:

Ta thấy rằng công thức(1) chính là giá trị trung bình mức xám của ảnh. Vì vậy có thể nói rằng hệ số 1 chiều của phép biến đổi Fourier của một bức ảnh chính là trung bình mức xám của ảnh đó.



1. Biến đổi Fourier của ảnh

Hình 2 (a) biểu diễn ảnh gốc với một hình chữ nhật kích thước 20 x 40 pixel  
màu trắng trên nền đen của bức ảnh kích thước 512 x 512 pixel. Hình 1 (b) biểu diễn các giá trị của phép biến đổi Fourier của ảnh gốc tại các giá trị (u,v). Ta thấy rằng chu  
kỳ của các giá trị 0 trên trục u dài gấp đôi chu kỳ của giá trị 0 trên trục v. Điều này tương ứng với tỷ lệ 1:2 của khối hình chữ nhật trong ảnh gốc.

1. LỌC ẢNH TRONG MIỀN TẦN SỐ
   1. Khái niệm

Lọc ảnh trong miền tần số là một phương pháp xử lý ảnh nhằm loại bỏ hoặc giữ lại các thành phần tần số cụ thể của ảnh, thường nhằm mục đích làm mịn (giảm nhiễu) hoặc tăng cường các chi tiết (làm sắc nét ảnh). Phương pháp này dựa trên ý tưởng rằng ảnh có thể được phân tích dưới dạng các tần số không gian khác nhau, và bằng cách điều chỉnh các tần số này, ta có thể thay đổi đặc tính của ảnh.

Trong miền không gian, lọc ảnh thường liên quan đến việc áp dụng các toán tử như tích chập (convolution) với các mặt nạ (mask) hoặc kernel để thao tác trực tiếp lên các điểm ảnh. Tuy nhiên, trong miền tần số, ta chuyển đổi ảnh sang miền tần số bằng cách sử dụng các phép biến đổi như Biến đổi Fourier (Fourier Transform - FT) hoặc Biến đổi Cosin rời rạc (Discrete Cosine Transform - DCT), sau đó thực hiện các phép lọc trên các thành phần tần số của ảnh.

Sau khi thực hiện lọc trong miền tần số, ảnh sẽ được biến đổi ngược lại về miền không gian để thu được kết quả cuối cùng. Kỹ thuật này có thể rất hiệu quả đối với nhiều bài toán như lọc nhiễu, tăng cường chi tiết hoặc phát hiện biên (edge detection).

* 1. Tầm quan trọng của lọc ảnh trên miền tần số

Lọc ảnh trên miền tần số là một kỹ thuật xử lý ảnh mạnh mẽ, cho phép chúng ta phân tích và điều chỉnh các thành phần tần số của một hình ảnh. Việc này mang lại nhiều lợi ích quan trọng trong các ứng dụng xử lý ảnh, bao gồm:

* + 1. Loại bỏ nhiễu:

Một trong những ứng dụng chính của lọc ảnh trên miền tần số là giảm thiểu các loại nhiễu có trong ảnh, bao gồm nhiễu Gauss (Gaussian noise) và nhiễu xung (impulse noise).

* Nhiễu Gauss: Thường xuất hiện khi chụp ảnh trong môi trường có độ sáng thấp hoặc cảm biến bị hạn chế. Lọc thông thấp trên miền tần số có thể được sử dụng để giảm thiểu các thành phần tần số cao chứa nhiều nhiễu, làm mịn ảnh và loại bỏ nhiễu Gauss.
* Nhiễu xung: Là dạng nhiễu ngẫu nhiên, thường xuất hiện dưới dạng các điểm sáng hoặc tối không mong muốn trong ảnh. Bằng cách sử dụng các bộ lọc chuyên biệt trên miền tần số, các thành phần nhiễu xung có thể được loại bỏ một cách hiệu quả mà không làm ảnh hưởng đến các chi tiết quan trọng của ảnh.
  + 1. Nâng cao chất lượng ảnh: Tăng cường cạnh, làm mịn ảnh.

Lọc ảnh trên miền tần số giúp tăng cường các đặc điểm của ảnh, đặc biệt là các cạnh và chi tiết nhỏ. Các bộ lọc như lọc thông cao (High-pass filter) giúp làm nổi bật các cạnh và chi tiết của đối tượng trong ảnh.

* Tăng cường cạnh: Khi xử lý ảnh trong miền tần số, các bộ lọc thông cao loại bỏ các thành phần tần số thấp (đại diện cho các vùng đồng nhất của ảnh) và giữ lại các thành phần tần số cao (biểu thị các cạnh và chi tiết nhỏ). Điều này giúp làm sắc nét các vùng biên trong ảnh, tạo ra hiệu ứng tăng cường cạnh rõ rệt.
* Làm mịn ảnh: Ngược lại, lọc thông thấp giúp loại bỏ các thành phần tần số cao, làm giảm độ nhiễu và làm mịn các vùng có thay đổi đột ngột trong ảnh. Điều này hữu ích trong việc tạo ra các hiệu ứng làm mờ hoặc làm mịn, giúp ảnh trông tự nhiên hơn.
  + 1. Phân tích đặc trưng:

Các đặc trưng của ảnh như hình dạng, kết cấu và chi tiết có thể được phân tích tốt hơn trên miền tần số. Phép lọc trên miền tần số giúp trích xuất các thành phần tần số cụ thể, giúp xác định các đặc điểm quan trọng trong ảnh.

* Nhận dạng đối tượng: Các phương pháp nhận dạng đối tượng sử dụng phép biến đổi Fourier để phân tích các mẫu tần số của đối tượng trong ảnh. Nhờ việc phân tách các thành phần tần số cao và thấp, các đặc trưng như hình dạng, biên dạng có thể được trích xuất rõ ràng hơn.
* Trích xuất kết cấu: Kết cấu là một thuộc tính quan trọng trong nhận dạng hình ảnh, và các thành phần tần số giúp xác định các mẫu kết cấu lặp lại. Việc áp dụng các bộ lọc trên miền tần số cho phép xác định được các kết cấu cụ thể của vật thể trong ảnh.
  1. Các bước cơ bản của lọc ảnh trong miền tần số

Cho ảnh f(x,y), biến đổi ảnh trong miền tần số bao gồm 6 bước cơ bản sau:

Nhân ảnh đầu vào *f(x,y)* với *(-1)x+y* để tập trung phổ vào vị trí trung tâm. Việc này nhờ vào tính chất dịch chuyển của phép biến đổi Fourier.

Tính *F(u,v)* của hàm trong bước 1 theo công thức:

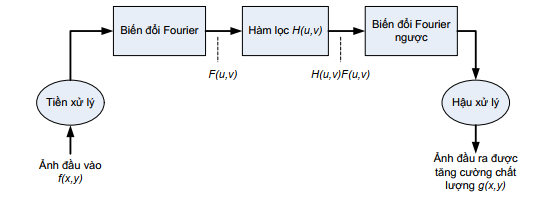
Nhân F(u,v) với hàm của bộ lọc H(u,v)

Biến đổi Fourier ngược của hàm số *F(u,v)* trong bước 3 theo công thức:

Lấy phần thực của kết quả trong bước 4

Nhân kết quả của bước 5 với *(-1)x+y.*

Để tổng kết các bước thực hiện lọc ảnh trong miền tần số ta có hình dưới đây:

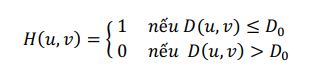


1. Quá trình thực hiện lọc ảnh trong miền tần số
   1. Các bộ lọc cụ thể

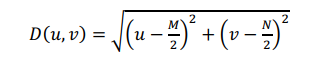
Nội dung phần này sẽ đề cập tới hai loại bộ lọc: Bộ lọc làm mịn ảnh và bộ lọc làm sắc nét ảnh. Vì cạnh và những chi tiết của ảnh tương ứng với những thành phần tần số cao khi ảnh được biến đổi sang miền tần số. Vì vậy, bộ lọc thông thấp có chức năng làm mịn ảnh trong khi bộ lọc thông cao có chức năng làm sắc nét ảnh.

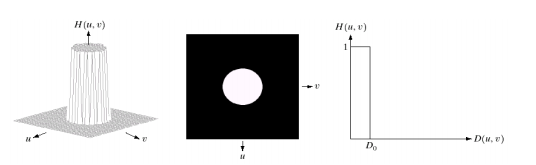
* + 1. Bộ lọc thông thấp lý tưởng

Bộ lọc thông thấp lý tưởng được coi là bộ lọc thông thấp đơn giản nhất có chức  
năng loại bỏ tất cả các thành phần tần số cao hơn tần số ngưỡng D0. Bộ lọc này có hàm biến đổi như sau:

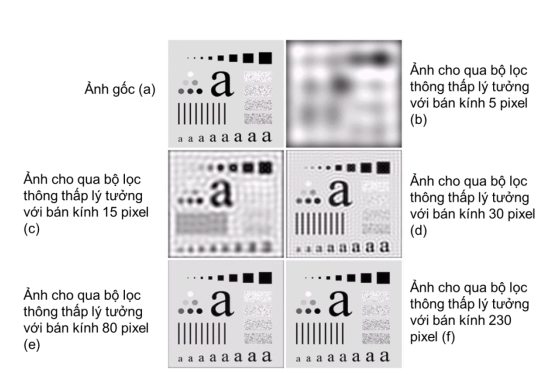


Trong đó *D0* là số dương, *D(u,v)* là khoảng cách từ điểm *(u,v)* tới tâm của ảnh phổ Fourier. Nếu ảnh có kích thước M x N thì tâm của ảnh có tọa độ *(M/2, N/2)*. Trong trường hợp này, khoảng cách từ điểm bất kỳ (u,v) tới tâm của ảnh phổ Fourier được cho bởi công thức:





1. Ảnh phổ của bộ lọc thông thấp lý tưởng



1. Ảnh gốc và ảnh sau khi cho qua các bộ lọc với tần số cắt khác nhau

Hình 2.33 mô tả kết quả của việc áp dụng bộ lọc thông thấp lý tưởng vào một bức ảnh với các tần số cắt khác nhau. Hình 2.33 (b) sử dụng đường tròn có bán kính 5 pixel tương đương với tần số cắt 92%. Việc này có nghĩa là 8% năng lượng của bức ảnh sẽ bị loại bỏ sau khi lọc. Chính vì vậy, bức ảnh trở lên mờ đi do các chi tiết trong ảnh (ứng với thành phần tần số cao) bị loại bỏ sau khi lọc. Khi bán kính càng tăng, nghĩa là càng ít năng lượng bị loại bỏ thì bức ảnh sẽ ít bị mờ (từ hình (c) tới hình (f)). Trong hầu hết các ảnh kết quả, bên cạnh hiệu ứng mờ còn có hiệu ứng vòng xuyến (ringing) thường xuất hiện bao quanh các cạnh. Đây là một nhược điểm của bộ lọc thông thấp lý tưởng khiến cho bộ lọc này ít được ứng dụng trong thực tế.

* + 1. Bộ lọc thông thấp Butterworth

Hàm chuyển đổi của bộ lọc Butterworth bậc n với tần số cắt D0 được định nghĩa như sau:

A math equation with numbers and symbols

Description automatically generated

Trong đó D(u,v) khoảng cách từ điểm có tọa độ (u,v) tới tâm của ảnh phổ. Hình 2.34 mô tả bộ lọc Butterworth. Khác với bộ lọc thông thấp lý tưởng, hàm biến đổi của bộ lọc Butterword có tần số cắt không phân chia một cách rõ rệt giữa tần số giữ lại và tần số loại bỏ. Tần số cắt D0 của bộ lọc Butterword là ngưỡng tần số qua đó giá trị H(u,v) bị suy giảm so với giá trị lớn nhất. Trong hình 2.34 (bên phải), H(u,v) = 0.5 (giảm 50% so với giá trị lớn nhất là 1) khi D(u,v) = D0.

A black square with a white light

Description automatically generated

1. Hình ảnh bộ lọc thông thấp Butterworth: (a) Hình ảnh 3 chiều;

(giữa) ảnh phổ của bộ lọc; (trái) mặt cắt đứng của bộ lọc.

A group of black and white text

Description automatically generated

1. Hình ảnh sau khi qua bộ lọc Butterworth
   * 1. Bộ lọc thông thấp Gaussian

Bộ lọc thông thấp Gaussian có chức năng làm mượt tín hiệu bằng cách loại bỏ các tần số cao và giữ lại các tần số thấp.

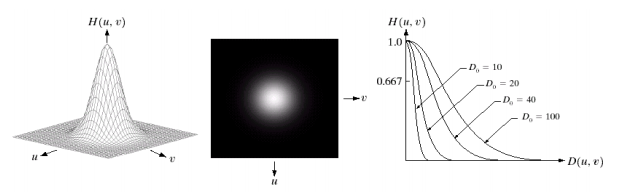
Bộ lọc thông thấp Gaussian hai chiều được cho bởi công thức:



Trong đó D(u,v) là khoảng cách từ điểm (u,v) tới tâm của biến đổi Fourier, 𝜎 là đánh giá khoảng rộng của đường cong Gaussian. Đặt 𝜎 = 𝐷0 với 𝐷0 là tần số cắt, ta có:

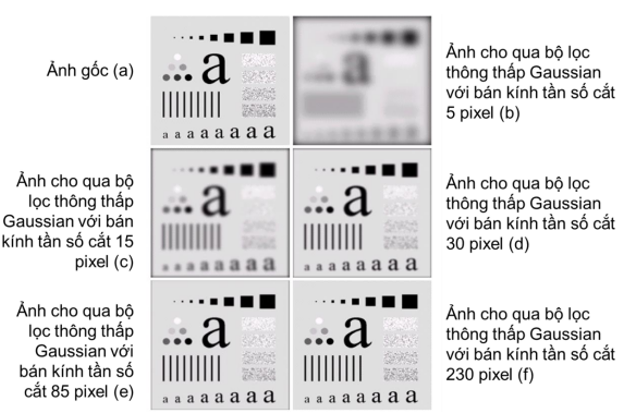


Khi D(u,v) = 𝐷0 thì bộ lọc giảm xuống ≈ 0.607 giá trị tối đa của nó.



1. Hình ảnh bộ lọc thông thấp Gaussian: (a) Hình ảnh 3 chiều; (giữa)

ảnh phổ của bộ lọc; (trái) mặt cắt của bộ lọc với các tần số cắt khác nhau.



1. Hình ảnh sau khi cho qua bộ lọc thông thấp Gaussian

Hình 5 mô tả hình ảnh sau khi qua bộ lọc thông thấp Gaussian với tần số cắt khác nhau. Ta có thể thấy rằng mức độ mờ của ảnh giảm dần khi tăng tần số cắt. Mặc dù bộ lọc Gaussian không đạt được độ mịn bằng bộ lọc Butterworth với cùng một tần số cắt ( hình 5 (b)) nhưng bộ lọc Gaussian không có hiện tượng bóng mờ như bộ lọc lý tưởng và bộ lọc Butterworth

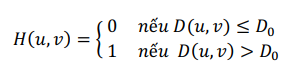
* + 1. Bộ lọc thông cao lý tưởng

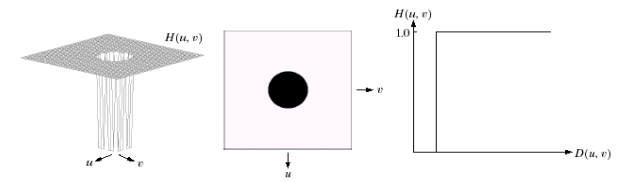
Bộ lọc thông cao có chức năng giữ lại các tần số cao và loại bỏ các tần số thấp của ảnh. Cụ thể, hàm biến đổi của bộ lọc thông cao được định như sau:



Trong đó *Hlp(u,v)* là hàm biến đổi của bộ lọc thông thấp tương ứng.

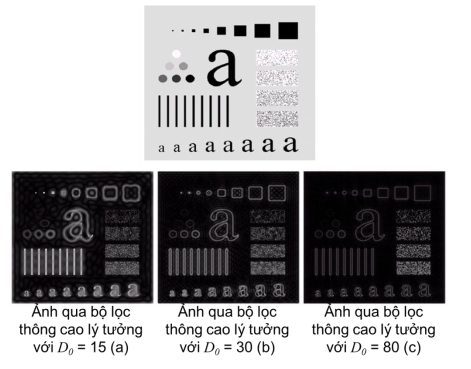
Từ định nghĩa chung về bộ lọc thông cao, ta có định nghĩa của bộ lọc thông cao lý tưởng như sau :





1. Hình ảnh bộ lọc thông cao lý tưởng: (a) Hình ảnh 3 chiều;

(giữa) ảnh phổ của bộ lọc; (trái) mặt cắt đứng của bộ lọc.



1. Hình ảnh sau khi qua bộ lọc thông cao lý tưởng
   * 1. Bộ lọc thông cao Butterworth

Hàm biến đổi của bộ lọc thông cao Butterword có dạng như sau:

A math equation with numbers and lines

Description automatically generated

Trong đó D(u, v) tính bởi công thức:

A mathematical equation with numbers and symbols

Description automatically generated

A diagram of a square with a black circle

Description automatically generated

1. Hình ảnh bộ lọc thông cao Butterword: (a) Hình ảnh 3 chiều;

(giữa) ảnh phổ của bộ lọc; (trái) mặt cắt đứng của bộ lọc.

A different type of fonts

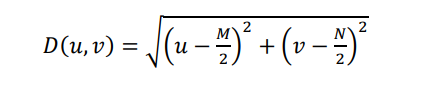
Description automatically generated with medium confidence

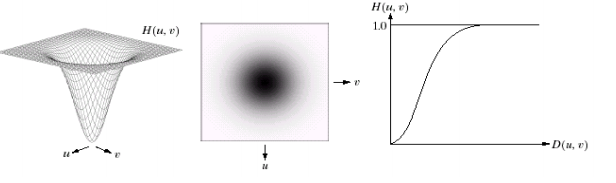
1. Hình ảnh sau khi qua bộ lọc thông cao lý tưởng
   * 1. Bộ lọc thông cao Gaussian

Hàm truyền đạt của bộ lọc thông cao Gaussian với tần số cắt Do được cho bởi công thức:



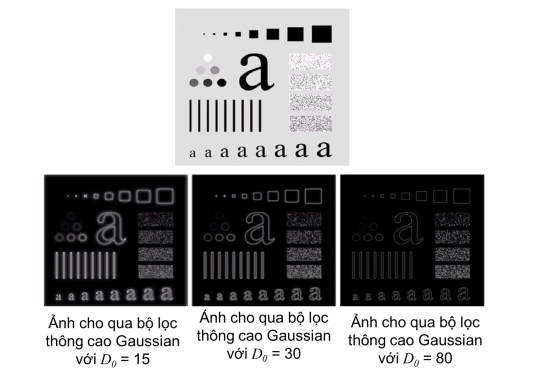
Trong đó D(u,v) được tính bởi công thức





1. Hình ảnh bộ lọc thông cao Gaussian: (a) Hình ảnh 3 chiều;

(giữa) ảnh phổ của bộ lọc; (trái) mặt cắt đứng của bộ lọc.



1. Hình ảnh sau khi qua bộ lọc thông cao Gaussian
2. ỨNG DỤNG THỰC TẾ

Việc áp dụng lọc ảnh trên miền tần số có vai trò quan trọng trong nhiều lĩnh vực của đời sống và kỹ thuật. Dưới đây là một số ứng dụng thực tế của các phép lọc trên miền tần số đối với ảnh đa mức xám.

* 1. Ảnh đa mức xám và yêu cầu xử lý

Ảnh đa mức xám là hình ảnh chỉ chứa thông tin cường độ sáng, mỗi điểm ảnh chỉ có một giá trị đại diện cho độ sáng từ đen (0) đến trắng (255). Xử lý ảnh đa mức xám yêu cầu các phương pháp để cải thiện chất lượng ảnh, làm nổi bật các chi tiết cần thiết hoặc giảm nhiễu.

* Đặc điểm của ảnh xám: Ảnh xám chỉ chứa thông tin về cường độ sáng, không chứa thông tin màu sắc, do đó các phương pháp xử lý phải tập trung vào việc làm rõ ràng biên, cạnh, và các chi tiết trong ảnh.
* Các vấn đề cần giải quyết khi xử lý ảnh xám: Nhiễu, mờ, và mất chi tiết là các vấn đề phổ biến trong ảnh xám, đòi hỏi các phương pháp lọc hiệu quả trên miền tần số để giải quyết.
  1. Áp dụng phép lọc thông thấp trong làm mịn ảnh

Lọc thông thấp được sử dụng để loại bỏ các thành phần tần số cao, thường liên quan đến nhiễu và các chi tiết sắc nét trong ảnh. Điều này giúp làm mịn các vùng lớn trong ảnh và giảm nhiễu.

* Ứng dụng trong làm mịn ảnh: Phép lọc thông thấp lý tưởng hoặc Gaussian thường được áp dụng trong quá trình làm mịn ảnh đa mức xám, giúp giảm thiểu nhiễu nhưng vẫn giữ được các cấu trúc quan trọng của hình ảnh.
* Ví dụ ứng dụng: Trong lĩnh vực y tế, lọc thông thấp giúp loại bỏ nhiễu trong các ảnh chụp X-quang hoặc MRI để tạo ra hình ảnh rõ ràng hơn cho các bác sĩ phân tích.
  1. Áp dụng phép lọc thông cao trong tăng cường chi tiết ảnh

Lọc thông cao giữ lại các tần số cao trong ảnh, giúp tăng cường các cạnh và các chi tiết nhỏ. Phương pháp này đặc biệt hữu ích khi cần làm nổi bật các đặc trưng quan trọng của hình ảnh.

* Ứng dụng trong tăng cường chi tiết: Phép lọc thông cao giúp làm rõ các cạnh và chi tiết nhỏ, đặc biệt trong các ứng dụng nhận dạng vật thể hoặc phân đoạn hình ảnh.
* Ví dụ ứng dụng: Trong ngành công nghiệp, lọc thông cao được sử dụng để kiểm tra chất lượng bề mặt sản phẩm, phát hiện các vết nứt hoặc khiếm khuyết nhỏ mà mắt thường khó nhận ra.
  1. Lọc các tần số cụ thể để giảm nhiễu

Nhiễu là một vấn đề phổ biến trong xử lý ảnh, và lọc các tần số cụ thể là một phương pháp hiệu quả để loại bỏ nhiễu mà không ảnh hưởng đến chất lượng tổng thể của ảnh. Các bộ lọc chặn dải hoặc lọc thông dải được sử dụng để loại bỏ các tần số gây nhiễu.

* Ứng dụng trong giảm nhiễu: Phép lọc chặn dải (band-stop filter) hoặc lọc thông dải (band-pass filter) được sử dụng để loại bỏ các dải tần số gây nhiễu mà vẫn giữ lại các tần số quan trọng.
* Ví dụ ứng dụng: Trong xử lý ảnh vệ tinh, lọc các tần số cụ thể giúp giảm nhiễu từ sóng điện từ hoặc các yếu tố môi trường, từ đó cải thiện chất lượng ảnh viễn thám.
  1. Ứng dụng trong các lĩnh vực khác

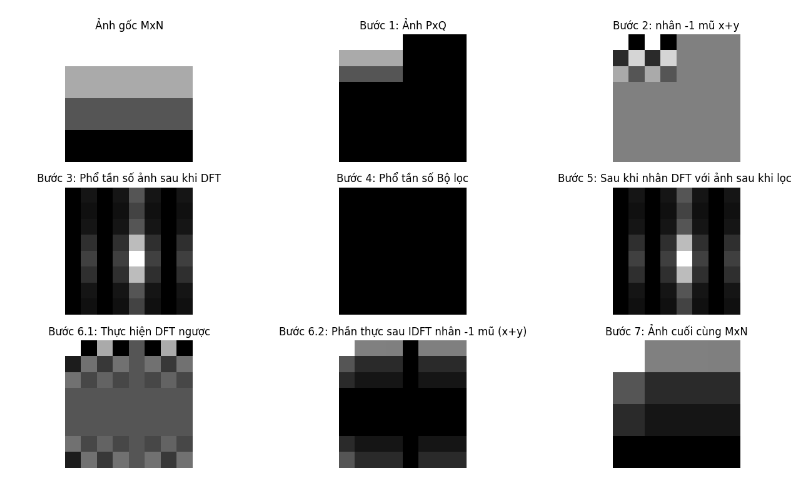
Phép lọc trên miền tần số còn được sử dụng trong nhiều lĩnh vực khác nhau, từ công nghiệp đến nghiên cứu khoa học, không chỉ để cải thiện chất lượng hình ảnh mà còn để trích xuất thông tin quan trọng từ dữ liệu hình ảnh.

* Ứng dụng trong viễn thám: Lọc ảnh vệ tinh trong lĩnh vực viễn thám giúp loại bỏ các yếu tố nhiễu từ môi trường và tăng cường chi tiết của bề mặt Trái Đất, giúp các nhà khoa học phân tích dữ liệu địa lý hiệu quả hơn.
* Ứng dụng trong đồ họa và nghệ thuật số: Lọc ảnh trên miền tần số được sử dụng để tạo ra các hiệu ứng nghệ thuật độc đáo trong đồ họa máy tính và hình ảnh kỹ thuật số, từ việc làm mờ ảnh đến tăng cường chi tiết hoặc tạo các hiệu ứng phong cách đặc biệt.

1. Xây dựng các thuật toán lọc ảnh
   1. Hàm lọc thông thấp Ideal

|  |
| --- |
| Hàm lọc thông thấp Ideal |
| from DFT\_base import \*  # Định nghĩa hàm lọc thông thấp Ideals  def lowPass\_Ideals(D0,U,V):  # H is our filter  H = np.zeros((U, V))  D = np.zeros((U, V))  U0 = int(U / 2)  V0 = int(V / 2)    # Tính khoảng cách  for u in range(U):  for v in range(V):  u2 = np.power(u, 2)  v2 = np.power(v, 2)  D[u, v] = np.sqrt( u2 +v2)  # Tính bộ lọc  for u in range(U):  for v in range(V):  if D[np.abs(u - U0), np.abs(v - V0)] <= D0:  H[u, v] = 1  else:  H[u, v] = 0  return H  if \_\_name\_\_ == "\_\_main\_\_":  # Đọc ảnh  image = cv2.imread("image/4x4pixels.png", 0)  # image = cv2.resize(src=image, dsize=(100, 100))  # Chuyển các pixel của ảnh vào mảng 2 chiều f  f = np.asarray(image)  M, N = np.shape(f) # Chiều x và y của ảnh  # Bước 1: Chuyển ảnh từ kích thước MxN vào ảnh PxQ với P= 2M và Q =2N  P, Q = 2\*M , 2\*N  shape = np.shape(f)  # Chuyển ảnh PxQ vào mảng fp  f\_xy\_p = np.zeros((P, Q))  f\_xy\_p[:shape[0], :shape[1]] = f  # Bước 2: Nhân ảnh fp(x,y) với (-1) mũ (x+y) để tạo ảnh mới  # Kết quả nhân lưu vào ma trận ảnh fpc  F\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  F\_xy\_p[x, y] = f\_xy\_p[x, y] \* np.power(-1, x + y)  # Bước 3: Chuyển đổi ảnh Fpc sang miền tần số (DFT)  # dft2d = DFT2D(F\_xy\_p)  dft\_cot = dft\_hang = np.zeros((P, Q))  # DFT chiều P - theo cột  for i in range(P):  dft\_cot[i] = DFT1D(F\_xy\_p[i])    # DFT chiều Q - theo hàng  for j in range(Q):  dft\_hang[:, j] = DFT1D(dft\_cot[:, j])  # Bước 4: Gọi hàm lowPass\_Ideals tạo bộ lọc thông thấp Ideals  H\_uv = lowPass\_Ideals(60, P, Q)  # Bước 5: Nhân ảnh sau khi DFT với ảnh sau khi lọc  G\_uv = np.multiply(dft\_hang, H\_uv)  # Bước 6:  # Bước 6.1 Thực hiện biến đổi ngược DFT  idft\_cot = idft\_hang = np.zeros((P, Q))  # chuyển đổi DFT ngược theo chiều P - theo cột  for i in range(P):  idft\_cot[i] = IDFT1D(G\_uv[i])  # Chuyển đổi DFT ngược theo chiều Q - theo hàng  for j in range(Q):  idft\_hang[:, j] = IDFT1D(idft\_cot[:, j])  # Bước 6.2: Nhân phần thực ảnh sau khi biến đổi ngược với -1 mũ (x+y)  g\_array = np.asarray(idft\_hang.real)  P, Q = np.shape(g\_array)  g\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  g\_xy\_p[x, y] = g\_array[x, y] \* np.power(-1, x + y)  # Bước 7: Rút trích ảnh kích thước MxN từ ảnh PxQ  # Và đây ảnh cuối cùng sau khi lọc  g\_xy = g\_xy\_p[:shape[0], :shape[1]] |

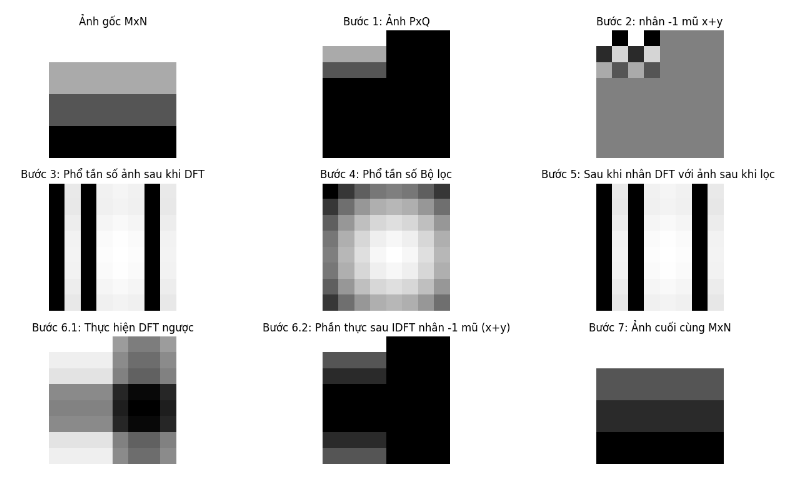
Kết quả hiện thị:



1. Kết quả khởi chạy hàm lọc thông thấp Ideal
   1. Hàm lọc thông thấp Gauss

|  |  |
| --- | --- |
| Hàm lọc thông thấp Gauss | |
| from DFT\_base import \*  # Định nghĩa hàm lọc thông thấp GaussianLP  def GaussianLP(D0,U,V):  # H cho filter  H = np.zeros((U, V))  D = np.zeros((U, V))  U0 = int(U / 2)  V0 = int(V / 2)  # Tính khoảng cách  for u in range(U):  for v in range(V):  u2 = np.power(u, 2)  v2 = np.power(v, 2)  D[u, v] = np.sqrt(u2 + v2)  # Tính bộ lọc  for u in range(U):  for v in range(V):  H[u, v] = np.exp((-D[np.abs(u - U0), np.abs(v - V0)]\*\*2)/(2\*(D0\*\*2)))  #print('H[ '+str(u)+ " , " + str(v)+' ] = ' + str(H[u,v]))  return H  if \_\_name\_\_ == "\_\_main\_\_":  #Pre-config  np.set\_printoptions(precision=2)    # Đọc ảnh  image = cv2.imread("image/4x4pixels.png", 0)  # image = cv2.resize(src=image, dsize=(50, 50))  # Chuyển các pixel của ảnh vào mảng 2 chiều f  f = np.asarray(image)  M, N = np.shape(f) # Chiều x và y của ảnh    # Bước 1: Chuyển ảnh từ kích thước MxN vào ảnh PxQ với P= 2M và Q =2N  P, Q = 2 \* M, 2 \* N  shape = np.shape(f)  # Chuyển ảnh PxQ vào mảng fp  f\_xy\_p = np.zeros((P, Q)) #creat a zero-matrix with size PxQ    f\_xy\_p[:shape[0], :shape[1]] = f  # Bước 2: Nhân ảnh fp(x,y) với (-1)^(x+y) để tạo ảnh mới  # Kết quả nhân lưu vào ma trận ảnh fpc  F\_xy\_p = np.zeros((P, Q))    for x in range(P):  for y in range(Q):  F\_xy\_p[x, y] = f\_xy\_p[x, y] \* np.power(-1, x + y)  # print("F\_xy\_p[" + str(x) + ", " + str(y) + "] = " + str(F\_xy\_p[x, y]))  # Bước 3: Chuyển đổi ảnh Fpc sang miền tần số (DFT)  dft\_cot = dft\_hang = np.zeros((P, Q),dtype=complex)    # DFT theo P - theo cột  for i in range(P):  dft\_cot[i] = DFT1D(F\_xy\_p[i])  # DFT theo Q - theo hàng  for j in range(Q):  dft\_hang[:, j] = DFT1D(dft\_cot[:, j])  # print("dft\_cot:", dft\_cot)  # print("dft\_hang:", dft\_hang)  # Bước 4: Gọi hàm GaussianLP tạo bộ lọc thông thấp Gaussian  H\_uv = GaussianLP(30, P, Q)  # np.set\_printoptions(precision=3)  # print(H\_uv)    # Bước 5: Nhân ảnh sau khi DFT với ảnh sau khi lọc  G\_uv = np.multiply(dft\_hang, H\_uv)  # print("G[u,v] = " + str(G\_uv))  # print(type(G\_uv))  # Bước 6:  # Bước 6.1 Thực hiện biến đổi ngược DFT  idft\_cot = idft\_hang = np.zeros((P, Q))  # chuyển đổi DFT ngược theo P - theo cột  for i in range(P):  idft\_cot[i] = IDFT1D(G\_uv[i])  # Chuyển đổi DFT ngược theo Q - theo hàng  for j in range(Q):  idft\_hang[:, j] = IDFT1D(idft\_cot[:, j])  print("Check")  # print("idft\_cot = ", idft\_cot)  # print("idft\_hang = ", idft\_hang)  # Bước 6.2: Nhân phần thực ảnh sau khi biến đổi ngược với -1 mũ (x+y)  g\_array = np.asarray(idft\_hang.real)    # print("Before (g\_array): ", g\_array)  P, Q = np.shape(g\_array)  g\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  g\_xy\_p[x, y] = g\_array[x, y] \* np.power(-1, x + y)  # print("After (g\_xy\_p): ", g\_xy\_p)      # Bước 7: Rút trích ảnh kích thước MxN từ ảnh PxQ  # Và đây ảnh cuối cùng sau khi lọc  g\_xy = g\_xy\_p[:shape[0], :shape[1]]    print("g\_xy: ", g\_xy)  # breakpoint() |

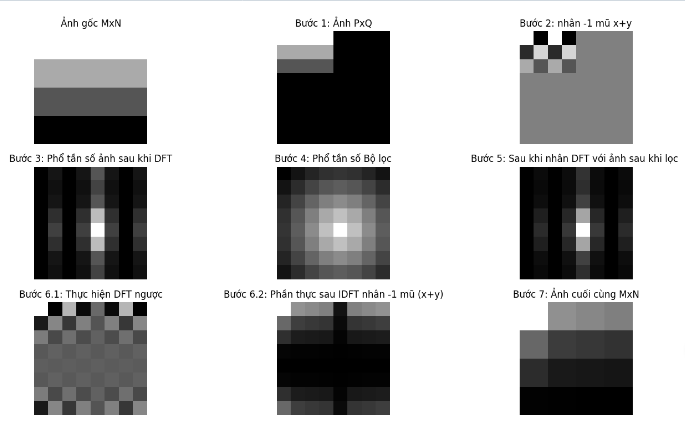
Kết quả của hàm lọc thông thấp Gauss:



1. Kết quả hàm lọc thông thấp Gauss
   1. Hàm lọc thông thấp Butterworth

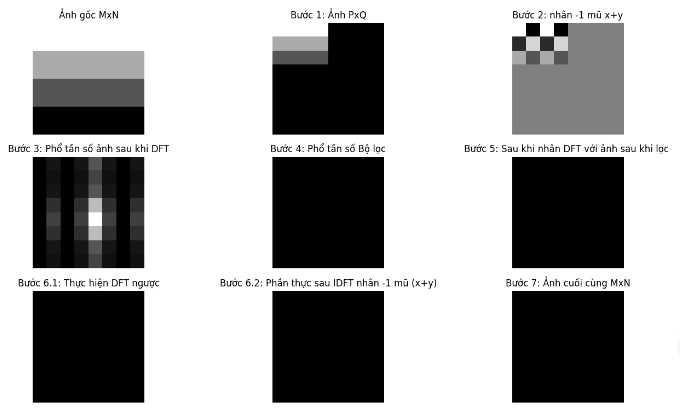
|  |
| --- |
| Hàm lọc thông thấp Butterworth |
| from DFT\_base import \*  # Định nghĩa hàm lọc thông thấp butterworthLP  def butterworthLP(D0,U,V,n):  H = np.zeros((U, V))  D = np.zeros((U, V))  U0 = int(U / 2)  V0 = int(V / 2)  # Tính khoảng cách  for u in range(U):  for v in range(V):  u2 = np.power(u, 2)  v2 = np.power(v, 2)  D[u, v] = np.sqrt(u2 + v2)  #Tính bộ lọc  for u in range(U):  for v in range(V):  H[u, v] = 1/((1+ D[np.abs(u - U0), np.abs(v - V0)]/D0)\*\*(2\*n))  return H  if \_\_name\_\_ == "\_\_main\_\_":  # Đọc ảnh  image = cv2.imread("image/4x4pixels.png", 0)  # image = cv2.resize(src=image, dsize=(100, 100))  # Chuyển các pixel của ảnh vào mảng 2 chiều f  f = np.asarray(image)  M, N = np.shape(f) # Chiều x và y của ảnh  # Bước 1: Chuyển ảnh từ kích thước MxN vào ảnh PxQ với P= 2M và Q =2N  P, Q = 2\*M , 2\*N  shape = np.shape(f)  # Chuyển ảnh PxQ vào mảng fp  f\_xy\_p = np.zeros((P, Q))  f\_xy\_p[:shape[0], :shape[1]] = f  # Bước 2: Nhân ảnh fp(x,y) với (-1) mũ (x+y) để tạo ảnh mới  # Kết quả nhân lưu vào ma trận ảnh fpc  F\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  F\_xy\_p[x, y] = f\_xy\_p[x, y] \* np.power(-1, x + y)  # Bước 3: Chuyển đổi ảnh Fpc sang miền tần số (DFT)  dft\_cot = dft\_hang = np.zeros((P, Q))  # DFT theo P - cột  for i in range(P):  dft\_cot[i] = DFT1D(F\_xy\_p[i])  # DFT theo Q - hàng  for j in range(Q):  dft\_hang[:, j] = DFT1D(dft\_cot[:, j])  # Bước 4: Gọi hàm butterworthLP tạo bộ lọc thông thấp butterworth  H\_uv = butterworthLP(30,P,Q,2)  # Bước 5: Nhân ảnh sau khi DFT với ảnh sau khi lọc  #G\_uv = np.multiply(dft2d, H\_uv)  G\_uv = np.multiply(dft\_hang, H\_uv)  # Bước 6:  # Bước 6.1 Thực hiện biến đổi ngược DFT  idft\_cot = idft\_hang = np.zeros((P, Q))  # chuyển đổi DFT ngược theo P - cột  for i in range(P):  idft\_cot[i] = IDFT1D(G\_uv[i])  # Chuyển đổi DFT ngược theo Q - hàng  for j in range(Q):  idft\_hang[:, j] = IDFT1D(idft\_cot[:, j])  # Bước 6.2: Nhân phần thực ảnh sau khi biến đổi ngược với -1 mũ (x+y)  g\_array = np.asarray(idft\_hang.real)  P, Q = np.shape(g\_array)  g\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  g\_xy\_p[x, y] = g\_array[x, y] \* np.power(-1, x + y)  # Bước 7: Rút trích ảnh kích thước MxN từ ảnh PxQ  # Và đây ảnh cuối cùng sau khi lọc  g\_xy = g\_xy\_p[:shape[0], :shape[1]] |

Kết quả hàm lọc thông thấp Butterworth:



1. Kết quả bộ lọc thông thấp Butterworth
   1. Hàm lọc thông cao Ideal

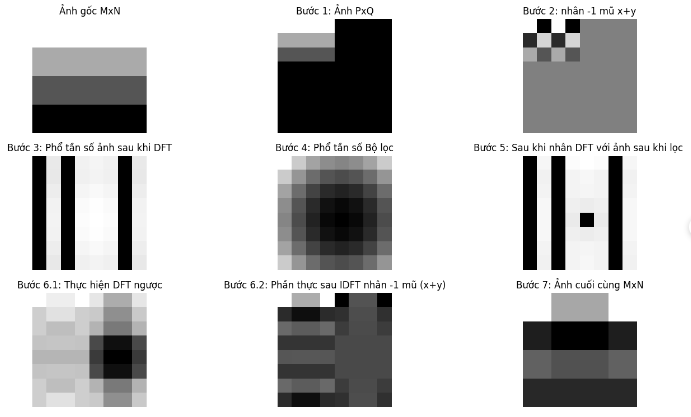
|  |
| --- |
| Hàm lọc thông cao Ideal |
| from DFT\_base import \*  # Định nghĩa hàm lọc thông cao Ideals  def HighPass\_Ideals(D0,U,V):  # H is our filter  H = np.zeros((U, V))  D = np.zeros((U, V))  U0 = int(U / 2)  V0 = int(V / 2)  # Tính khoảng cách  for u in range(U):  for v in range(V):  u2 = np.power(u, 2)  v2 = np.power(v, 2)  D[u, v] = np.sqrt(u2 + v2)  # Tính bộ lọc  for u in range(U):  for v in range(V):  if D[np.abs(u - U0), np.abs(v - V0)] <= D0:  H[u, v] = 0  else:  H[u, v] = 1  return H  if \_\_name\_\_ == "\_\_main\_\_":  # Đọc ảnh  image = cv2.imread("image/4x4pixels.png", 0)  # image = cv2.resize(src=image, dsize=(100, 100))  # Chuyển các pixel của ảnh vào mảng 2 chiều f  f = np.asarray(image)  M, N = np.shape(f) # Chiều x và y của ảnh  # Bước 1: Chuyển ảnh từ kích thước MxN vào ảnh PxQ với P= 2M và Q =2N  P, Q = 2\*M , 2\*N  shape = np.shape(f)  # Chuyển ảnh PxQ vào mảng fp  f\_xy\_p = np.zeros((P, Q))  f\_xy\_p[:shape[0], :shape[1]] = f  # Bước 2: Nhân ảnh fp(x,y) với (-1) mũ (x+y) để tạo ảnh mới  # Kết quả nhân lưu vào ma trận ảnh fpc  F\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  F\_xy\_p[x, y] = f\_xy\_p[x, y] \* np.power(-1, x + y)  # Bước 3: Chuyển đổi ảnh Fpc sang miền tần số (DFT)  dft\_cot = dft\_hang = np.zeros((P, Q))  # DFT theo chiều P - theo cột  for i in range(P):  dft\_cot[i] = DFT1D(F\_xy\_p[i])  # DFT theo chiều Q - theo hàng  for j in range(Q):  dft\_hang[:, j] = DFT1D(dft\_cot[:, j])  # Bước 4: Gọi hàm HighPass\_Ideals tạo bộ lọc thông cao Ideal  H\_uv = HighPass\_Ideals(10,P,Q)  # Bước 5: Nhân ảnh sau khi DFT với ảnh sau khi lọc  #G\_uv = np.multiply(dft2d, H\_uv)  G\_uv = np.multiply(dft\_hang, H\_uv)  # Bước 6:  # Bước 6.1 Thực hiện biến đổi ngược DFT  idft\_cot = idft\_hang = np.zeros((P, Q))  # chuyển đổi DFT ngược theo chiều P - theo cột  for i in range(P):  idft\_cot[i] = IDFT1D(G\_uv[i])  # Chuyển đổi DFT ngược theo chiều Q - theo hàng  for j in range(Q):  idft\_hang[:, j] = IDFT1D(idft\_cot[:, j])  # Bước 6.2: Nhân phần thực ảnh sau khi biến đổi ngược với -1 mũ (x+y)  g\_array = np.asarray(idft\_hang.real)  P, Q = np.shape(g\_array)  g\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  g\_xy\_p[x, y] = g\_array[x, y] \* np.power(-1, x + y)  # Bước 7: Rút trích ảnh kích thước MxN từ ảnh PxQ  # Và đây ảnh cuối cùng sau khi lọc  g\_xy = g\_xy\_p[:shape[0], :shape[1]] |



1. Kết quả hàm lọc thông cao Ideal
   1. Hàm lọc thông cao Gauss

|  |
| --- |
| Hàm lọc thông cao Gauss |
| import matplotlib.pyplot as plt  from DFT\_base import \*  # Định nghĩa hàm lọc thông cao GaussianHP  def GaussianHP(D0,U,V):  # H cho filter  H = np.zeros((U, V))  D = np.zeros((U, V))  U0 = int(U / 2)  V0 = int(V / 2)  # Tính khoảng cách  for u in range(U):  for v in range(V):  u2 = np.power(u, 2)  v2 = np.power(v, 2)  D[u, v] = np.sqrt(u2 + v2)  # Tính bộ lọc  for u in range(U):  for v in range(V):  H[u, v] = 1- np.exp((-D[np.abs(u - U0), np.abs(v - V0)] \*\* 2) / (2 \* (D0 \*\* 2)))  return H  if \_\_name\_\_ == "\_\_main\_\_":  #Pre-config  np.set\_printoptions(precision=2)    # Đọc ảnh  image = cv2.imread("image/4x4pixels.png", 0)  # image = cv2.resize(src=image, dsize=(50, 50))  # Chuyển các pixel của ảnh vào mảng 2 chiều f  f = np.asarray(image)  # print(f)    M, N = np.shape(f) # Chiều x và y của ảnh  # print(M, N)  # M, N = 4, 4      # Bước 1: Chuyển ảnh từ kích thước MxN vào ảnh PxQ với P= 2M và Q =2N  P, Q = 2 \* M, 2 \* N  shape = np.shape(f)    # print(P, Q)  # print(shape)  # Chuyển ảnh PxQ vào mảng fp  f\_xy\_p = np.zeros((P, Q))  f\_xy\_p[:shape[0], :shape[1]] = f    # print(f\_xy\_p)  # Bước 2: Nhân ảnh fp(x,y) với (-1) mũ (x+y) để tạo ảnh mới  # Kết quả nhân lưu vào ma trận ảnh fpc  F\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  F\_xy\_p[x, y] = f\_xy\_p[x, y] \* np.power(-1, x + y)  # print("F\_xy\_p[" + str(x) + ", " + str(y) + "] = " + str(F\_xy\_p[x, y]))    # print(F\_xy\_p)  # Bước 3: Chuyển đổi ảnh Fpc sang miền tần số (DFT)  dft\_cot = dft\_hang = np.zeros((P, Q),dtype=complex)  # DFT theo P - theo cột  for i in range(P):  dft\_cot[i] = DFT1D(F\_xy\_p[i])  # DFT theo Q - theo hàng  for j in range(Q):  dft\_hang[:, j] = DFT1D(dft\_cot[:, j])  # print(dft\_cot)  # print(dft\_hang)    # Bước 4: Gọi hàm GaussianLP tạo bộ lọc thông thấp Gaussian  H\_uv = GaussianHP(10, P, Q)  # print("H\_uv: ", H\_uv)  # Bước 5: Nhân ảnh sau khi DFT với ảnh sau khi lọc  G\_uv = np.multiply(dft\_hang, H\_uv)  # print("G[u,v] = " + str(G\_uv))  # Bước 6:  # Bước 6.1 Thực hiện biến đổi ngược DFT  idft\_cot = idft\_hang = np.zeros((P, Q))  # chuyển đổi DFT ngược theo P - theo cột  for i in range(P):  idft\_cot[i] = IDFT1D(G\_uv[i])  # Chuyển đổi DFT ngược theo Q - theo hàng  for j in range(Q):  idft\_hang[:, j] = IDFT1D(idft\_cot[:, j])    # print("idft\_cot: ", idft\_cot)  # print("idft\_hang: ", idft\_hang)  # Bước 6.2: Nhân phần thực ảnh sau khi biến đổi ngược với -1 mũ (x+y)  g\_array = np.asarray(idft\_hang.real)  P, Q = np.shape(g\_array)  g\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  g\_xy\_p[x, y] = g\_array[x, y] \* np.power(-1, x + y)  # print("Before (g\_array): ", g\_array)  # print("After (g\_xy\_p): ", g\_xy\_p)  # Bước 7: Rút trích ảnh kích thước MxN từ ảnh PxQ  # Và đây ảnh cuối cùng sau khi lọc  g\_xy = g\_xy\_p[:shape[0], :shape[1]]  # print("g\_xy: ", g\_xy) |

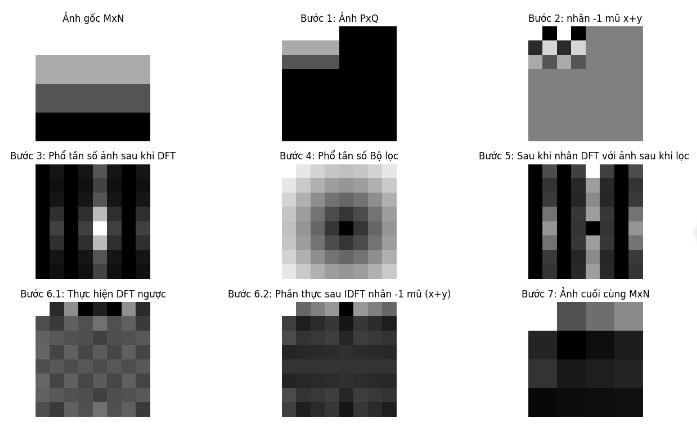
Kết quả lọc thông cao Gauss:



1. Kết quả lọc thông cao Gauss
   1. Hàm lọc thông cao Butterworth

|  |
| --- |
| Hàm lọc thông cao Butterworth |
| from DFT\_base import \*  # Định nghĩa hàm lọc thông cao butterworthHP  def butterworthHP(D0,U,V,n):  H = np.zeros((U, V))  D = np.zeros((U, V))  U0 = int(U / 2)  V0 = int(V / 2)  # Tính khoảng cách  for u in range(U):  for v in range(V):  u2 = np.power(u, 2)  v2 = np.power(v, 2)  D[u, v] = np.sqrt(u2 + v2)  # Tính bộ lọc  for u in range(U):  for v in range(V):  H[u, v] = 1 - 1 /((1 + D[np.abs(u-U0), np.abs(v-V0)]/D0)\*\*(2\*n))  return H  if \_\_name\_\_ == "\_\_main\_\_":  # Đọc ảnh  image = cv2.imread("image/4x4pixels.png", 0)  # image = cv2.resize(src=image, dsize=(100, 100))  # Chuyển các pixel của ảnh vào mảng 2 chiều f  f = np.asarray(image)  M, N = np.shape(f) # Chiều x và y của ảnh  # Bước 1: Chuyển ảnh từ kích thước MxN vào ảnh PxQ với P= 2M và Q =2N  P, Q = 2\*M , 2\*N  shape = np.shape(f)  # Chuyển ảnh PxQ vào mảng fp  f\_xy\_p = np.zeros((P, Q))  f\_xy\_p[:shape[0], :shape[1]] = f  # Bước 2: Nhân ảnh fp(x,y) với (-1) mũ (x+y) để tạo ảnh mới  # Kết quả nhân lưu vào ma trận ảnh fpc  F\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  F\_xy\_p[x, y] = f\_xy\_p[x, y] \* np.power(-1, x + y)  # Bước 3: Chuyển đổi ảnh Fpc sang miền tần số (DFT)  dft\_cot = dft\_hang = np.zeros((P, Q))  # DFT theo P - cột  for i in range(P):  dft\_cot[i] = DFT1D(F\_xy\_p[i])  # DFT theo Q - hàng  for j in range(Q):  dft\_hang[:, j] = DFT1D(dft\_cot[:, j])  # Bước 4: Gọi hàm butterworthHP tạo bộ lọc thông cao butterworth  H\_uv = butterworthHP(60,P,Q,2)  # Bước 5: Nhân ảnh sau khi DFT với ảnh sau khi lọc  G\_uv = np.multiply(dft\_hang, H\_uv)  # Bước 6:  # Bước 6.1 Thực hiện biến đổi ngược DFT  idft\_cot = idft\_hang = np.zeros((P, Q))  # chuyển đổi DFT ngược theo P - cột  for i in range(P):  idft\_cot[i] = IDFT1D(G\_uv[i])  # Chuyển đổi DFT ngược theo Q - hàng  for j in range(Q):  idft\_hang[:, j] = IDFT1D(idft\_cot[:, j])  # Bước 6.2: Nhân phần thực ảnh sau khi biến đổi ngược với -1 mũ (x+y)  g\_array = np.asarray(idft\_hang.real)  P, Q = np.shape(g\_array)  g\_xy\_p = np.zeros((P, Q))  for x in range(P):  for y in range(Q):  g\_xy\_p[x, y] = g\_array[x, y] \* np.power(-1, x + y)  # Bước 7: Rút trích ảnh kích thước MxN từ ảnh PxQ  # Và đây ảnh cuối cùng sau khi lọc  g\_xy = g\_xy\_p[:shape[0], :shape[1]] |

Kết quả phép lọc thông cao Butterworth:



1. Kết quả bộ lọc thông cao Butterworth

KẾT LUẬN

Phép lọc ảnh trên miền tần số đã được chứng minh là một công cụ mạnh mẽ và linh hoạt trong xử lý ảnh số. Bằng cách phân tích và thao tác trực tiếp lên các thành phần tần số của ảnh, chúng ta có thể đạt được những kết quả xử lý ảnh ấn tượng, từ việc loại bỏ nhiễu đến việc tăng cường các chi tiết. Tuy nhiên, việc lựa chọn loại bộ lọc phù hợp và điều chỉnh các tham số một cách hợp lý đòi hỏi sự hiểu biết sâu sắc về lý thuyết và thực tiễn.

Trong tương lai, việc nghiên cứu các thuật toán lọc nhanh hơn, hiệu quả hơn, và kết hợp lọc miền tần số với các kỹ thuật học sâu sẽ mở ra nhiều tiềm năng mới cho việc xử lý ảnh. Đồng thời, việc ứng dụng các kỹ thuật lọc trên miền tần số vào các lĩnh vực khác như xử lý tín hiệu âm thanh, xử lý video cũng là một hướng đi đầy hứa hẹn.

Qua bài báo cáo này, chúng tôi đã trình bày một cách chi tiết về cơ sở lý thuyết, các thuật toán và ứng dụng của phép lọc ảnh trên miền tần số. Chúng tôi hy vọng rằng công trình này sẽ đóng góp một phần nhỏ vào sự phát triển của lĩnh vực xử lý ảnh và cung cấp một tài liệu tham khảo hữu ích cho các nhà nghiên cứu và sinh viên.

TÀI LIỆU THAM KHẢO

Bài viết

Lecture 16: Fourier Transform, MIT OpenCourseWare, Fall 2011, <https://ocw.mit.edu/courses/6-003-signals-and-systems-fall-2011/d74bab2dfa6e465d61fed45763d50528_MIT6_003F11_lec16.pdf>

Basics of Image Processing with OpenCV in Python: Fourier Transformations, Naveed Ul Mustafa, 2023, <https://numustafa.medium.com/basics-of-image-processing-with-opencv-in-python-fourier-transformations-75ac3cf7c17b#:~:text=Fourier%20Transformation%20can%20be%20used,the%20contrast%20of%20an%20image>.

Video

Image Filtering in Frequency Domain | Image Processing II, Shree Nayar, 2021, <https://www.youtube.com/watch?v=OOu5KP3Gvx0>