

ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC SÀI GÒN



BÁO CÁO TỔNG KẾT
ĐỒ ÁN MÔN HỌC
PHÂN TÍCH VÀ XỬ LÝ ẢNH

Xử lý ảnh trên miền tần số
(Image Enhancement in the
Frequency Domain)

Giảng viên hướng dẫn: PGS.TS Phạm Thế Bảo

Thành phố Hồ Chí Minh, tháng 11 năm 2025

ỦY BAN NHÂN DÂN THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC SÀI GÒN

BÁO CÁO TỔNG KẾT
ĐỒ ÁN MÔN HỌC
PHÂN TÍCH VÀ XỬ LÝ ẢNH

Xử lý ảnh trên miền tần số
(Image Enhancement in the Frequency Domain)

MSV	Họ và tên
3122480001	Trần Đức Anh
3122480034	Nguyễn Thành Nam
3122480042	Bùi Tấn Phát

Thành phố Hồ Chí Minh, tháng 11 năm 2025

Lời cảm ơn

Tiểu luận này được hoàn thành tại trường Đại Học Sài Gòn dưới sự hướng dẫn của *PGS.TS Phạm Thế Bảo*. Em xin bày tỏ lòng biết ơn chân thành và sâu sắc về sự tận tâm và nhiệt tình của Thầy trong suốt quá trình tác giả thực hiện đề tài.

Xin cảm ơn Phòng Đào tạo và Khoa Toán - Ứng dụng trường Đại học Sài Gòn, gia đình, bạn bè và các thầy cô đã tạo nhiều điều kiện thuận lợi, giúp em hoàn thành đồ án môn học này.

Tp. HCM, tháng 11 năm 2025

Các tác giả

Mục lục

Lời cảm ơn	i
Mục lục	ii
Danh mục các từ viết tắt	iv
Danh mục các bảng, hình vẽ	v
Lời nói đầu	1
1 Xử lý ảnh trong miền tần số	2
1.1 Giới thiệu và đôi nét về miền tần số	2
1.2 Khác biệt giữa miền không gian và miền tần số	4
1.3 Khái niệm chuỗi Fourier và biến đổi Fourier	4
1.3.1 Biến đổi Fourier cho hàm liên tục	6
1.3.1.1 Biến đổi Fourier cho hàm một biến số	6
1.3.1.2 Biến đổi Fourier cho hàm hai biến số	8
1.3.2 Biến đổi Fourier cho hàm rời rạc - DFT	10
1.3.2.1 Hàm một biến số	10
1.3.2.2 Hàm hai biến số	11
1.3.3 Biến đổi Fourier nhanh(FFT)	11
1.3.3.1 Hàm một biến số	11
1.3.3.2 Hàm hai biến số	13
1.3.4 Định lý tích chập	13
1.4 Xử lý ảnh trong miền tần số	14
1.4.1 Khái niệm về xử lý ảnh trong miền tần số	14

1.4.2 Các bộ lọc băng thông trong miền tần số	16
1.4.3 Làm mờ ảnh trong miền tần số	16
1.4.3.1 Lọc thông thấp Ideal	19
1.4.3.2 Lọc thông thấp Gauss	25
1.4.3.3 Lọc thông thấp Butterworth	27
1.4.4 Ứng dụng phép lọc làm mờ ảnh	31
1.4.5 Làm sắc ảnh trong miền tần số	34
1.4.5.1 Lọc thông cao Ideal	35
1.4.5.2 Lọc thông cao Gauss	38
1.4.5.3 Lọc thông cao Butterworth	44
1.4.6 Ứng dụng phép lọc sắc nét ảnh	51
1.4.7 Lọc chặn	52
Kết luận	57
Tài liệu tham khảo	58

Danh mục các từ viết tắt

DFT	Discrete Fourier Transforms
IDFT	Inverse Discrete Fourier Transforms

Danh mục các bảng, hình vẽ

Hình, bảng Trang

Hình 1.1	4
Hình 1.2	6
Hình 1.3	6
Hình 1.4	8
Hình 1.5	9
Hình 1.6	9
Hình 1.7	9
Hình 1.8	10
Hình 1.9	14
Hình 1.10	15
Hình 1.11	20
Hình 1.12	20
Hình 1.13	22
Hình 1.14	22
Hình 1.16	25

Hình 1.17	26
Hình 1.18	26
Hình 1.19	27
Hình 1.20	28
Hình 1.21	30
Hình 1.22	30
Hình 1.23	31
Hình 1.24	31
Hình 1.25	32
Hình 1.26	33
Hình 1.27	34
Hình 1.28	35
Hình 1.29	36
Hình 1.30	37
Hình 1.31	38
Hình 1.32	38
Hình 1.33	39
Hình 1.34	40
Hình 1.35	41
Hình 1.36	41
Hình 1.37	45
Hình 1.39	47

Hình 1.40	49
Hình 1.41	50
Hình 1.42	51
Hình 1.43	52
Hình 1.44	53
Hình 1.45	55
Hình 1.46	56
Hình 1.47	56

Lời nói đầu

Xử lý ảnh là một lĩnh vực quan trọng trong khoa học máy tính và kỹ thuật, đóng vai trò thiết yếu trong nhiều ứng dụng thực tế như nhận dạng khuôn mặt, phân tích hình ảnh y tế, xử lý video, và trí tuệ nhân tạo. Trong bối cảnh đó, việc xử lý ảnh trên miền tần số mang lại những lợi thế đặc biệt so với xử lý trong miền không gian, đặc biệt là khả năng lọc và tăng cường ảnh một cách hiệu quả thông qua các phép biến đổi Fourier.

Báo cáo này tập trung vào chủ đề "Xử lý ảnh trên miền tần số (Image Enhancement in the Frequency Domain)", nhằm trình bày các phương pháp cơ bản và ứng dụng của việc lọc ảnh trong miền tần số. Chúng em sẽ tìm hiểu các khái niệm nền tảng như biến đổi Fourier, chuỗi Fourier, và các bộ lọc thông thấp, thông cao, cũng như ứng dụng thực tế của chúng trong việc làm mờ, làm sắc nét ảnh và loại bỏ nhiễu.

Cấu trúc của báo cáo được tổ chức như sau: Chương 1 giới thiệu về miền tần số, khái niệm biến đổi Fourier và các tính chất của nó. Chương 2 trình bày chi tiết các phương pháp lọc ảnh trong miền tần số, bao gồm lọc thông thấp và thông cao với các loại bộ lọc phổ biến như Ideal, Gauss và Butterworth. Chương 3 kết luận và tổng hợp lại những nội dung chính.

Hy vọng rằng báo cáo này sẽ cung cấp một vài tri thức hữu ích về xử lý ảnh trong miền tần số, góp phần vào việc nâng cao kiến thức và kỹ năng trong lĩnh vực phân tích và xử lý ảnh.

Tp. HCM, tháng 11 năm 2025

Các tác giả

Chương 1

Xử lý ảnh trong miền tần số

1.1. Giới thiệu và đôi nét về miền tần số

Trong bài tiểu luận này chúng tôi nghiên cứu về xử lý ảnh trên miền tần số. Sử dụng công cụ toán học để tính toán và ứng dụng cho xử lý ảnh/ lọc ảnh trên miền tần số. Ảnh đầu vào của các phép xử lý sẽ là ảnh xám của 1 bức hình bình thường, sau đó dùng các phương pháp xử lý ảnh trên miền tần số để làm sắc nét ảnh hơn hoặc làm mờ ảnh đi để bảo mật.

Khái niệm lọc ảnh dễ dàng nhận thấy trong miền tần số. Giúp tăng cường ảnh $f(x, y)$. Ảnh tăng cường được xây dựng nên thông qua tích chập sau:

$$\underbrace{g(m, n)}_{\substack{\text{Ảnh tăng cường} \\ PSS}} = \underbrace{h(x, y)}_{\substack{\text{Hàm} \\ \text{biến đổi}}} * \underbrace{f(x, y)}_{\substack{\text{Ảnh}}},$$

Ta cũng có thể cài đặt tăng cường ảnh trong miền tần số và thiết kế hàm tương ứng như:

$$\underbrace{\hat{g}(u, v)}_{\substack{\text{Ảnh tăng cường} \\ \text{Hàm biến đổi}}} = \underbrace{\hat{h}(u, v)}_{\substack{\text{Hàm} \\ \text{biến đổi}}} * \underbrace{\hat{f}(u, v)}_{\substack{\text{Ảnh}}},$$

với các kí hiệu hàm trên là các kết quả của các biến đổi Fourier.

Ta sẽ tìm hiểu khái niệm miền tần số trong không gian 1 chiều.

1. Chu kỳ của $\cos t$ là 2π giây, nghĩa là tín hiệu sẽ được lặp lại sau mỗi 2π giây.
2. Chu kỳ của $\cos(2\pi t)$ là 1 giây.

3. Ta ký hiệu chu kỳ là T .
4. Đơn vị đo cho tần số là Hz (Herts), ký hiệu là f , tương ứng với số chu kỳ xảy ra trong 1 giây

$$f = \frac{1}{T}$$

5. Hàm $\cos(2\pi \cdot ft)$ có chu kỳ $1/T$ và tần số f .
6. Đơn vị đo cho tần số góc là rad/s , ký hiệu là ω

$$\omega = \frac{2\pi}{T}$$

Ví dụ: Giả sử một bức ảnh có các điểm ảnh thỏa hàm số

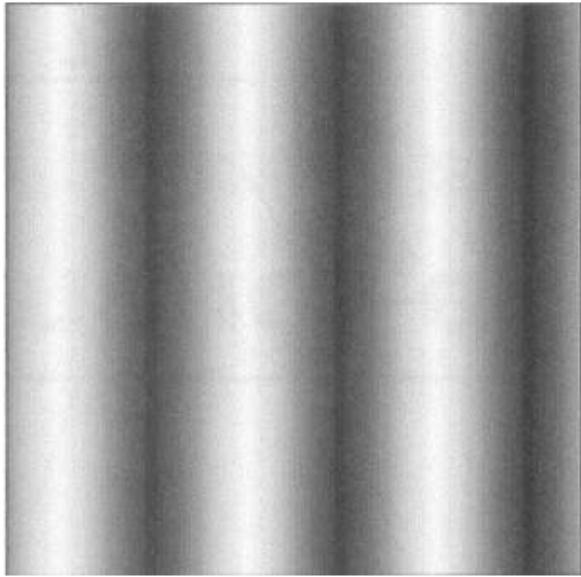
$$f(x, y) = 128 + A \sin \left(\frac{22\pi ux}{N-1} + \phi \right)$$

Khi đó, ta biết được ảnh có mức xám trung bình là 128, biên độ $A \in [1, 127]$, độ rộng của ảnh là N, ϕ là pha và u là tần số không gian (số vòng hàm sin "vừa với độ rộng của hình, chia cho $N \rightarrow$ tần số không gian trong 1 vòng đơn vị trên điểm ảnh).

Cho hàm

$$f(x, y) = 128 + 127 \sin \left(\frac{2\pi \cdot 3x}{100-1} + 0 \right).$$

Ta được ảnh tương ứng của hàm số $f(x, y)$ là



Hình 1.1: Ảnh $f(x, y)$

1.2. 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 của điểm ảnh trên miền không gian.

1. Miền không gian: Ma trận ảnh đầu vào \rightarrow Xử lý \rightarrow Ma trận ảnh đầu ra.
2. Miền tần số: Ảnh vào \rightarrow Phân bố tần số \rightarrow Xử lý \rightarrow Chuyển đổi ngược \rightarrow Ảnh ra.

Miền tần số không gian có thể tạo ra mối quan hệ chu kỳ rõ ràng trong miền không gian, trong miền tần số, một số toán tử xử lý ảnh sẽ trở nên hiệu quả hơn.

Trong nhiều trường hợp, người ta dùng chuyển đổi Fourier để chuyển ảnh từ miền không gian sang miền tần số.

1.3. Khái niệm chuỗi Fourier và biến đổi Fourier

Chuỗi Fourier (Fourier series) được nhà Toán học người Pháp tên Jean Baptiste Joseph Fourier đưa ra vào thế kỷ 19. Ông khẳng định rằng với bất kỳ hàm số $f(t)$

tuần hoàn với chu kỳ T đều có thể biểu diễn được dưới dạng tổng của các hàm số sine và cosine với những tần số khác nhau, mỗi hàm số nhân với một hệ số tương ứng. Khi đó, ta gọi tổng các chuỗi hàm số sine và cosine này là chuỗi Fourier.

Giống như chuỗi Taylor, chuỗi Fourier là một dạng đặc biệt của khai triển hàm số. Với chuỗi Taylor, ta quan tâm đến tập các hàm số đặc biệt dạng như sau:

$$1, x, x^2, x^3, \dots \quad (\text{a})$$

còn với chuỗi Fourier, ta quan tâm đến các dạng hàm đặc biệt lượng giác:

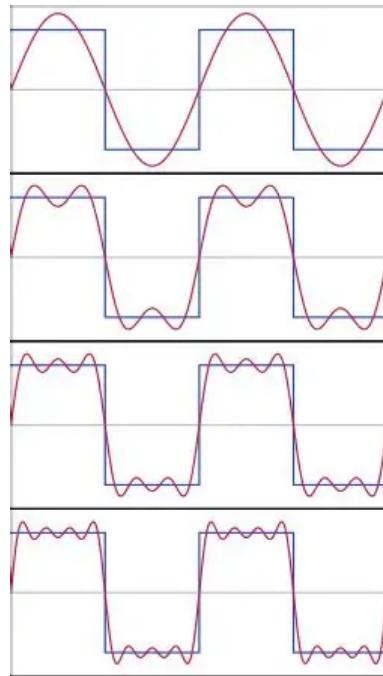
$$1, \cos x, \cos 2x, \cos 3x, \dots, \sin x, \sin 2x, \sin 3x, \dots \quad (\text{b})$$

Do đó, khai triển chuỗi Fourier của một hàm số f sẽ có dạng tổng các hàm lượng giác sin và cos cùng các hệ số:

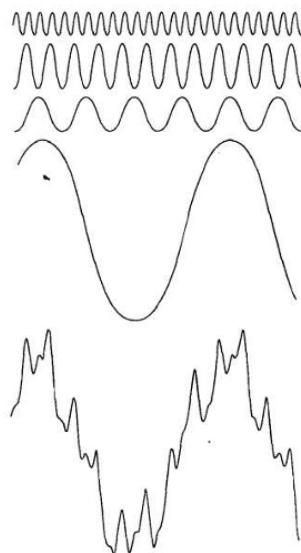
$$f(x) = a_0 + \sum_{n=1}^{+\infty} a_n \cos nx + b_n \sin nx, \quad (\text{c})$$

với

$$\begin{aligned} a_0 &= \frac{1}{2\pi} \int_{-\pi}^{\pi} f(x) dx, \\ a_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \cos nx dx, \quad n = 1, 2, \dots \\ b_n &= \frac{1}{\pi} \int_{-\pi}^{\pi} f(x) \sin nx dx, \quad n = 1, 2, \dots \end{aligned}$$



Hình 1.2: Minh họa về chuỗi Fourier và các dạng hàm sóng tương ứng từ trên xuống với chỉ số n càng lớn.



Hình 1.3: Minh họa chuỗi Fourier

1.3.1 Biến đổi Fourier cho hàm liên tục

1.3.1.1 Biến đổi Fourier cho hàm một biến số

Lấy f là hàm trơn từng khúc tuần hoàn chu kỳ T . Dạng phức của chuỗi Fourier của f là:

Định lý 1.1. (*Dạng phức của chuỗi Fourier*)

$$f(t) = \sum_{n=-\infty}^{\infty} c_n e^{i \frac{2\pi n}{T} t}, \quad (1.3.1)$$

với hệ số Fourier c_n được cho bởi

$$c_n = \frac{1}{T} \int_{-\frac{T}{2}}^{\frac{T}{2}} f(t) e^{-i \frac{2\pi n}{T} t} dt, \quad n = 0, \pm 1, \pm 2, \pm 3, \dots \quad (1.3.2)$$

Phương trình (1.3.1) là cách khai triển hàm số sin và cos theo công thức Euler trong trường số phức:

$$e^{i\varphi} = \cos \varphi + i \sin \varphi \quad (1.3.3)$$

Đối với những hàm số không có tính tuần hoàn, nhưng diện tích dưới đường cong của hàm số đó là hữu hạn, ta có thể biểu diễn hàm số đó dưới dạng tích phân của hàm sin và cos nhân với hàm trọng số. Biểu thức thu được gọi là biến đổi Fourier. Ta xác định phương trình biến đổi Fourier của một hàm số liên tục $f(t)$ có biến t liên tục như sau:

$$\mathcal{F}\{f(t)\}(\mu) = \int_{-\infty}^{+\infty} f(t) e^{-2\pi i \mu t} dt \quad (1.3.4)$$

với μ là biến liên tục. Vì khi lấy xong tích phân sẽ mất t nên ta chỉ còn lại μ , vậy ta sẽ ký hiệu lại phương trình biến đổi Fourier cho rõ ràng hơn như sau:

$$\hat{f}(\mu) = \int_{-\infty}^{+\infty} f(t) e^{-2\pi i \mu t} dt \quad (1.3.5)$$

Ngược lại, cho trước $\hat{f}(\mu)$, ta có thể tìm lại $f(t)$ bằng cách sử dụng chuyển đổi ngược Fourier (inverse Fourier transform), $f(t) = \mathcal{F}^{-1}\{\hat{f}(\mu)\}$, viết là:

$$f(t) = \int_{-\infty}^{+\infty} \hat{f}(\mu) e^{2\pi i \mu t} d\mu, \quad (1.3.6)$$

sử dụng công thức Euler, ta viết lại phương trình (1.3.5) như sau:

$$\hat{f}(\mu) = \int_{-\infty}^{+\infty} f(t) [\cos(2\pi \mu t) - i \sin(2\pi \mu t)] dt. \quad (1.3.7)$$

Biến còn lại khi lấy tích phân là μ , chính là tần số của hàm lượng giác nên miền của chuyển đổi Fourier là miền tần số.

1.3.1.2 Biến đổi Fourier cho hàm hai biến số

Ta có công thức biến đổi Fourier và biến đổi Fourier ngược cho hàm hai biến như sau:

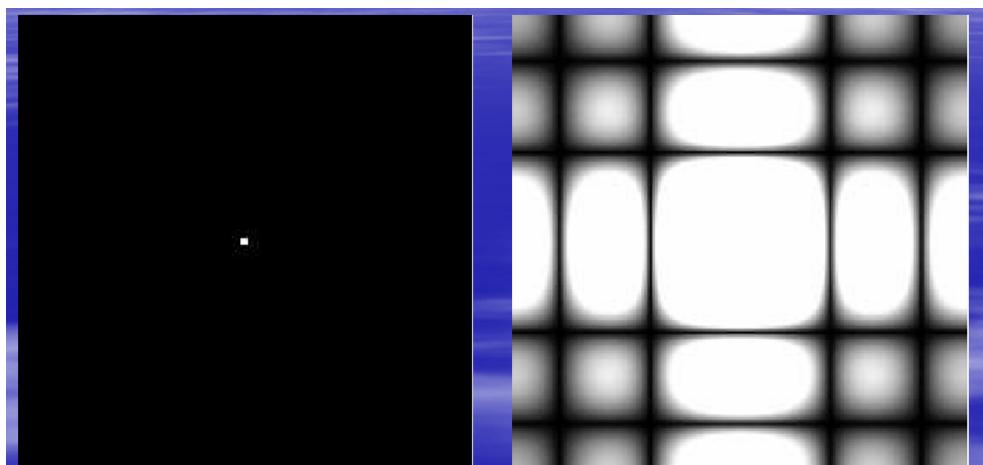
Biến đổi Fourier thuận:

$$\mathcal{F}(u, v) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) e^{-2\pi i(ux+vy)} dx dy \quad (1.3.8)$$

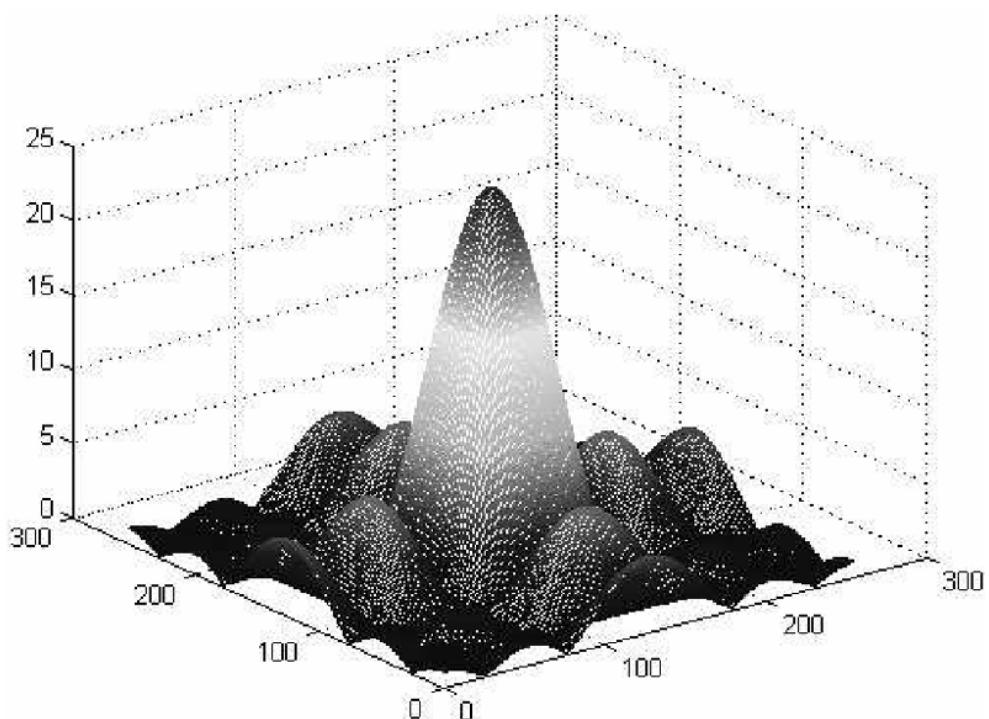
Biến đổi Fourier ngược:

$$\mathcal{F}^{-1}(u, v) = f(x, y) = \int_{-\infty}^{+\infty} \int_{-\infty}^{+\infty} f(x, y) e^{2\pi i(ux+vy)} du dv \quad (1.3.9)$$

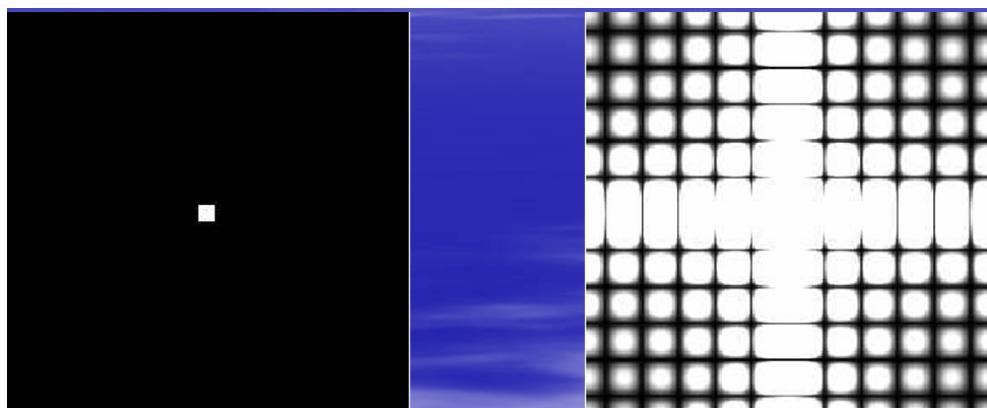
Một số ví dụ về biến đổi Fourier với ảnh.



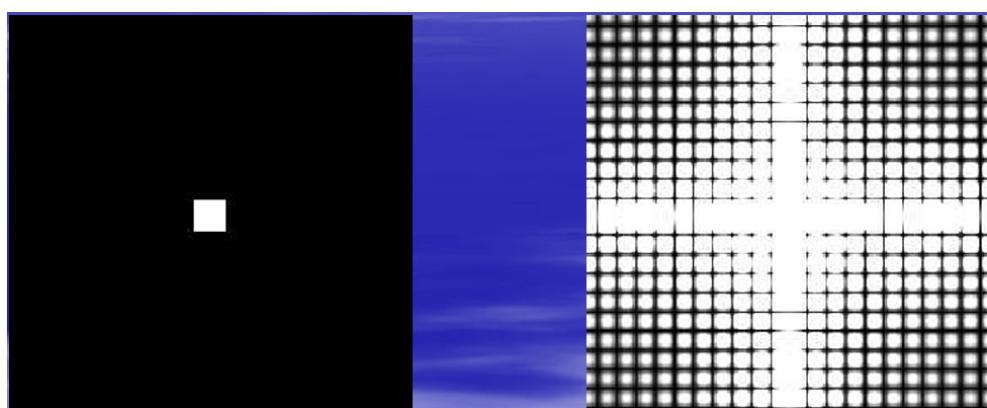
Hình 1.4: Ảnh $f(x, y)$ (bên trái) và Phổ độ lớn $\hat{f}(u, v)$ (bên phải)



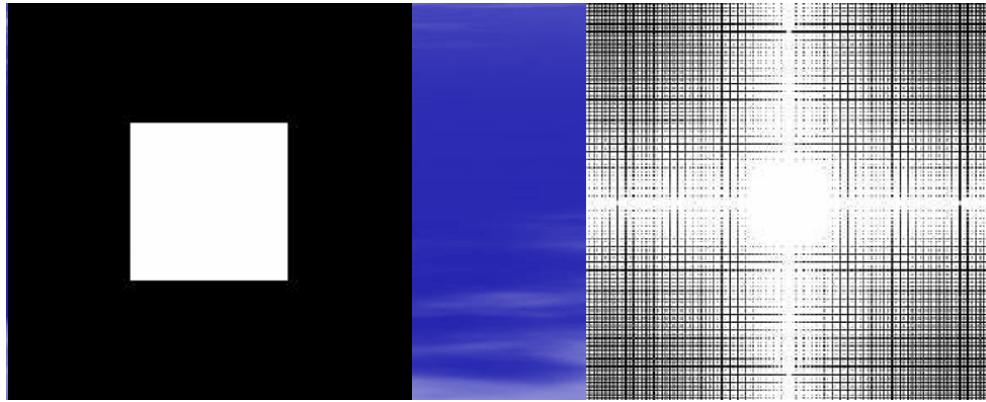
Hình 1.5: $\hat{f}(u, v)$ trong không gian 3 chiều



Hình 1.6: Ảnh- $f(x, y)$ (bên trái) và Phổ độ lớn $\hat{f}(u, v)$ (bên phải)



Hình 1.7: Ảnh- $f(x, y)$ (bên trái) và Phổ độ lớn $\hat{f}(u, v)$ (bên phải)



Hình 1.8: Ảnh- $f(x, y)$ (bên trái) và Phổ độ lớn $\hat{f}(u, v)$ (bên phải)

Nhận xét: Ta thấy khi kích thước đối tượng tăng lên trong miền không gian thì tương ứng "kích thước" - mức độ biến động lên xuống trong miền tần số giảm. Nhưng có xu hướng dao động rộng hơn với nhịp độ rất nhỏ. Hay nói cách khác là các tần số thấp nhiều hơn hẳn ảnh kích thước nhỏ.

1.3.2 Biến đổi Fourier cho hàm rời rạc - DFT

Vì các điểm ảnh là các điểm dữ liệu rời rạc nên ta áp dụng biến đổi Fourier, ta cần xây dựng một công thức sử dụng cho các biến rời rạc.

1.3.2.1 Hàm một biến số

Giả sử ra có bộ dữ liệu dãy f_n , (với $n = 1, 2, 3, \dots, N$) là dãy-vector-ảnh 1-D, ta xác định DFT(Discrete Fourier Transforms) cho f_n như sau:

$$f_n = \frac{1}{N} \sum_{i=1}^N \hat{f}(k) e^{-\frac{ik2\pi n}{N}}, \quad n = \overline{1, N}. \quad (1.3.10)$$

Biến đổi Fourier ngược(IDFT) là

$$\hat{f}(k) = \sum_{n=1}^N f_n e^{\frac{ik2\pi n}{N}}, \quad n = \overline{1, N}. \quad (1.3.11)$$

Lưu ý:

- $\hat{f}(k)$ là phức dù cho f_n là thực.
- Ta cài đặt trực tiếp DFT thì có độ phức tạp là $O(N^2)$.
- Ta có thể dùng FFT được nói đến sau đây để cài đặt DFT hiệu quả hơn. Nhưng đây không phải là một biến đổi mới.

1.3.2.2 Hàm hai biến số

Giả sử hàm $f(x, y), x = 1, \dots, M, y = 1, \dots, N$ là ảnh đầu vào rời rạc $M \times N$, DFT cho hàm hai biến (DFT 2-D) này là:

$$\hat{f}(u, v) = \sum_{x=1}^M \sum_{y=1}^N f(x, y) e^{-2\pi i \left(\frac{ux}{M} + \frac{vy}{N}\right)} \quad (1.3.12)$$

$$\mathcal{F}^{-1}\{\hat{f}(u, v)\} = f(x, y) = \frac{1}{MN} \sum_{u=1}^M \sum_{v=1}^N \hat{f}(u, v) e^{2\pi i \left(\frac{ux}{M} + \frac{vy}{N}\right)} \quad (1.3.13)$$

1.3.3 Biến đổi Fourier nhanh(FFT)

1.3.3.1 Hàm một biến số

Để chuyển ảnh từ miền không gian sang miền tần số bằng cách sử dụng chuyển đổi Fourier thông thường đòi hỏi chi phí lớn ($O(N^2)$ với N là số điểm ảnh).

Thuật toán FFT (Fast Fourier Transforms) thông dụng nhất là thuật toán do J.W.Cooley và John Tukey đề xuất, tính chuyển đổi Fourier cho các giá trị rời rạc bằng cách sử dụng đệ quy tính các giá trị ở vị trí chẵn lẻ.

$$\mathcal{X}_k = \underbrace{\sum_{m=1}^{\frac{N}{2}} x_{2m} e^{-\frac{2\pi i}{N} (2m)k}}_{\text{DFT cho phần chẵn}} + \underbrace{\sum_{m=1}^{\frac{N}{2}} x_{2m+1} e^{-\frac{2\pi i}{N} (2m+1)k}}_{\text{DFT cho phần lẻ}}. \quad (1.3.14)$$

$$\mathcal{X}_k = \sum_{m=1}^{\frac{N}{2}} x_{2m} e^{-\frac{2\pi i}{N/2} mk} + e^{-\frac{2\pi i}{N} k} \sum_{m=1}^{\frac{N}{2}} x_{2m+1} e^{-\frac{2\pi i}{N/2} mk}. \quad (1.3.15)$$

Ta có thể viết lại X_k cho gọn hơn như sau:

$$\mathcal{X}_k = E_k + e^{-\frac{2\pi i}{N} k} O_k, \quad (1.3.16)$$

với E_k, O_k lần lượt là tổng các phần chẵn và phần lẻ.

Do tính tuần hoàn có chu kỳ của DFT nên

$$E_{k+\frac{N}{2}} = E_k \quad (1.3.17)$$

và

$$O_{k+\frac{N}{2}} = O_k \quad (1.3.18)$$

Khi đó, ta viết lại phương trình (1.3.14) và (1.3.15) thành

$$\mathcal{X}_k = \begin{cases} E_k + e^{-\frac{2\pi i}{N}k} O_k, & 0 \leq k < \frac{N}{2} \\ E_{k-N/2} + e^{-\frac{2\pi i}{N}k} O_{k-N/2}, & \frac{N}{2} \leq k < N \end{cases} \quad (1.3.19)$$

Mặt khác, $e^{-\frac{2\pi i}{N}k}$ hình thành từ:

$$e^{-\frac{2\pi i}{N}(k+N/2)} = e^{-\frac{2\pi i}{N}k - \pi i} = e^{-\pi i} e^{-\frac{2\pi i}{N}k} = -e^{-\frac{2\pi i}{N}k}$$

Khi đó, ta có thể giảm khối lượng tính toán xuống một nửa. Với $0 \leq k < N/2$, từ phương trình (1.3.19) ta được:

$$\mathcal{X}_k = E_k + e^{-\frac{2\pi i}{N}k} O_k \quad (1.3.20)$$

$$\mathcal{X}_{k+N/2} = E_k - e^{-\frac{2\pi i}{N}k} O_k \quad (1.3.21)$$

Listing 1.1: Thuật toán FFT để quy dạng ditfft2

```

1 import cmath
2
3 def ditfft2(x, N, s):
4     """
5         x: list day so (co the la phuc)
6         N: kich thuoc FFT (phai la luy thua cua 2)
7         s: buoc nhay (stride)
8     """
9     if N == 1:
10         return [x[0]] # Truong hop co so
11
12     # Tinh FFT phan chan va le
13     X_even = ditfft2(x[0::2*s], N//2, 2*s) # (x0, x2s, x4s, ...)
14     X_odd = ditfft2(x[s::2*s], N//2, 2*s) # (xs, x3s, x5s, ...)
15
16     X = [0] * N
17     for k in range(N//2):
18         t = X_even[k]
19         exp_term = cmath.exp(-2j * cmath.pi * k / N) * X_odd[k]
20         X[k] = t + exp_term
21         X[k + N//2] = t - exp_term

```

22	
23	<code>return X</code>

1.3.3.2 Hàm hai biến số

Từ ý tưởng của hàm một biến, ta có thể tính được FFT của hàm hai biến bằng cách tính theo một chiều với mỗi giá trị của biến x (theo từng cột). Sau đó tính ngược lại theo y (theo từng hàng) với giá trị thu được ở trên.

Trong Python, ta có thể sử dụng hàm `fft2` để đưa ảnh sang chuỗi Fourier và dùng `ifft2` để trả ngược lại ảnh lúc đầu.

1.3.4 Định lý tích chập

Trong xử lý ảnh hoặc tăng cường ảnh trong miền không gian, người ta sử dụng khái niệm về tích chập , trong đó ảnh sau khi được xử lý bằng tích chập của ảnh ban đầu và bộ lọc ảnh tạo thành. Về mặt Toán học, ta giả sử hai hàm số liên tục, ta có định nghĩa tích chập $*$ của hai hàm số này như bên dưới.

$$f(t) * g(t) = \int_{-\infty}^{+\infty} f(\tau)g(t - \tau)d\tau. \quad (1.3.22)$$

Ta áp dụng tích chập cho hàm hai biến sau khi biến đổi Fourier như sau:

$$\mathcal{F}\{f(t) * g(t)\} = \int_{-\infty}^{+\infty} \left[\int_{-\infty}^{+\infty} f(\tau)g(t - \tau)d\tau \right] e^{-2\pi i \mu t} dt, \quad (1.3.23)$$

từ đây ta có:

$$\begin{aligned} \mathcal{F}\{f(t) * g(t)\} &= \int_{-\infty}^{+\infty} f(\tau) \left[\int_{-\infty}^{+\infty} g(t - \tau)e^{-2\pi i \mu t} d\tau \right] dt \\ &= \int_{-\infty}^{+\infty} f(\tau)e^{-2\pi i \mu t} d\tau \int_{-\infty}^{+\infty} g(u)e^{-2\pi i \mu u} du \\ &= \mathcal{F}(f)(\mu)\mathcal{F}(g)(\mu) = \hat{f}(\mu)\hat{g}(\mu). \end{aligned}$$

Vậy giả sử $\mathcal{F}\{f(t)\} = \hat{f}(\mu), \mathcal{F}\{g(t)\} = \hat{g}(\mu)$, khi đó:

$$\mathcal{F}\{f(t) * g(t)\} = \hat{f}(\mu)\hat{g}(\mu). \quad (1.3.24)$$

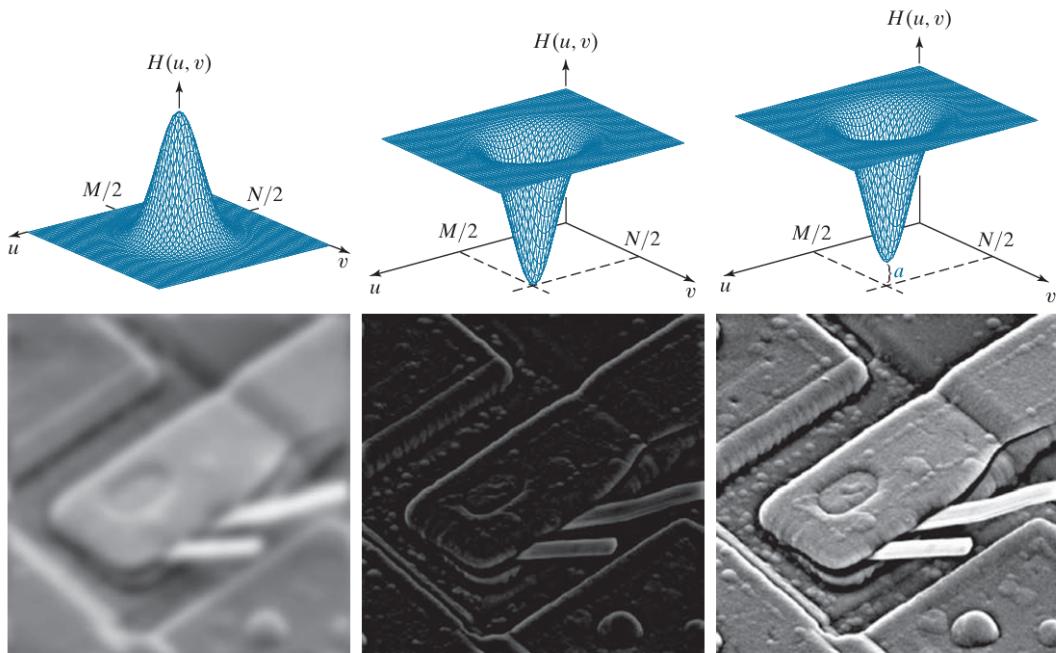
Áp dụng cho trường hợp hai chiều, giả sử $f(x, y)$ là ảnh đầu vào, $h(x, y)$ là bộ lọc ảnh, $\mathcal{F}\{f(x, y)\} = \hat{f}(\mu, \eta)$, $\mathcal{F}\{g(x, y)\} = \hat{g}(\mu, \eta)$, khi đó

$$\mathcal{F}\{f(x, y) * g(x, y)\} = \hat{f}(\mu, \eta)\hat{g}(\mu, \eta). \quad (1.3.25)$$

1.4. Xử lý ảnh trong miền tần số

1.4.1 Khái niệm về xử lý ảnh trong miền tần số

Khi chụp ảnh, ta có thể thu được các ảnh có tần số thấp (sự thay đổi mức xám của ảnh ít, ví dụ như ảnh một bức tường) hay ảnh có tần số cao (ví dụ như biên của vật thể). Vì vậy, ta cần có một bộ lọc $H(u, v)$ lọc bỏ các tần số cao và giữ lại tần số thấp (được gọi là lọc thông thấp) làm cho ảnh mờ đi hoặc loại bỏ các tần số thấp giữ lại tần số cao (được gọi là lọc thông cao) làm tăng cường chi tiết vật thể và đồng thời giảm độ tương phản. Hình ảnh mô tả điều này bên dưới



Hình 1.9: Ảnh phía trên là đồ thị hàm tần số ứng với ảnh phía dưới.

Để biết được ảnh trong miền tần số như thế nào ta quan sát ảnh sau:



Hình 1.10: Ảnh gốc (bên trái) và ảnh trong miền tần số khi dùng hàm `fft2` trong Python để biến đổi (ảnh bên phải)

Dịnh lý tích chập cho ta mối quan hệ giữa miền không gian và miền tần số, cụ thể, thông qua tích chập, một ảnh trong miền không gian có thể chuyển qua miền tần số và ngược lại.

Quy trình lọc ảnh trong miền tần số như sau:

Ảnh → Biến đổi Fourier → Lọc → Biến đổi Fourier ngược → Ảnh

1. Bước 1: Xử lý ảnh trong miền không gian, tức tăng hoặc giảm độ sáng của ảnh.
2. Bước 2: Lấy DFT của ảnh.
3. Bước 3: Canh giữa DFT, tức mang DFT từ góc ảnh ra giữa ảnh, trong Matlab ta sử dụng hàm `shifft` để thực hiện.
4. Bước 4: Thực hiện tích chập với hàm lọc.
5. Bước 5: Trượt DFT từ giữa ảnh ra góc.
6. Bước 6: Lấy chuyển đổi ngược IDFT, tức chuyển ảnh từ miền tần số sang miền không gian.

1.4.2 Các bộ lọc băng thông trong miền tần số

Khái niệm bộ lọc trong miền tần số tương tự như khái niệm mặt nạ trong miền không gian.

Sau khi chuyển ảnh sang miền không gian, ta áp dụng một số bộ lọc trong quy trình lọc ảnh nhằm làm mờ ảnh, giảm nhiễu ảnh hoặc làm rõ-làm nét ảnh.

Các bộ lọc thông dụng:

1. Lọc thông thấp/cao Ideal (Ideal low/high pass filter).
2. Lọc thông thấp/cao Gaus (Gaussian low/high pass filter).
3. Lọc thông thấp/cao Butterworth (Butterworth low/high pass filter).

Ta sẽ đi qua một số ứng dụng của bộ lọc này.

1.4.3 Làm mờ ảnh trong miền tần số

Ta có thể làm mờ ảnh bằng cách giảm hoặc loại bỏ các thành phần tần số cao trong ảnh, vì các tần số cao tương ứng với các chi tiết nhỏ của ảnh và các biên cảnh ảnh.

Phương pháp này tương đương với việc dùng bộ lọc thông thấp (Low pass filter) trong miền tần số.

Khi ta biến đổi ảnh $f(x, y)$ sang miền tần số, ta được:

$$\hat{f}(u, v) = \mathcal{F}\{f(x, y)\} = \sum_{x=1}^M \sum_{y=1}^N f(x, y) e^{-2\pi i(\frac{ux}{M} + \frac{vy}{N})}. \quad (1.4.1)$$

Sau đó, ta nhân với một bộ lọc thông thấp $\hat{h}(u, v)$:

$$\hat{g}(u, v) = \hat{h}(u, v) \cdot \hat{f}(u, v), \quad (1.4.2)$$

cuối cùng, ta biến đổi ngược để thu được ảnh mờ :

$$g(x, y) = \mathcal{F}^{-1}\{\hat{g}(u, v)\} \quad (1.4.3)$$

Ảnh thu được $g(x, y)$ là phiên bản đã được làm mờ của ảnh gốc $f(x, y)$, do các thành phần tần số cao trong phỏ bị giảm hoặc loại bỏ.

Tính chất của phép làm mờ ảnh

Phép làm mờ ảnh bằng bộ lọc thông thấp có những tính chất sau:

- **Tất cả các giá trị trong mặt nạ mờ đều dương:** Các giá trị của mặt nạ $h(x, y)$ thỏa mãn

$$h(x, y) > 0, \quad \forall (x, y).$$

Điều này đảm bảo rằng mỗi điểm ảnh mới là một tổ hợp lồi của các điểm ảnh lân cận, giúp ảnh mờ đi nhưng không bị méo màu.

- **Tổng các giá trị trong mặt nạ bằng 1:** Để duy trì mức sáng trung bình của ảnh, ta cần chuẩn hóa mặt nạ sao cho

$$\sum_x \sum_y h(x, y) = 1.$$

Nếu tổng này khác 1, ảnh sau khi làm mờ có thể bị tối hoặc sáng hơn so với ảnh gốc.

- **Làm giảm biên cạnh bằng mặt nạ mờ:** Các biên ảnh tương ứng với tần số cao. Khi nhân với bộ lọc thông thấp, các thành phần này bị triệt tiêu, dẫn đến ảnh mờ hơn:

$$g(x, y) = f(x, y) * h(x, y),$$

trong đó ký hiệu $*$ biểu thị phép chập (convolution). Theo định lý chập, trong miền tần số ta có:

$$\mathcal{F}\{g(x, y)\} = \mathcal{F}\{f(x, y) * h(x, y)\} = \hat{f}(u, v) \cdot \hat{h}(u, v).$$

- **Khi kích thước mặt nạ tăng, ảnh càng mượt:** Mặt nạ càng lớn \Rightarrow loại bỏ nhiều tần số cao hơn \Rightarrow ảnh càng bị mờ mạnh và mượt hơn.

Một số bộ lọc thông thấp

Các bộ lọc thông thấp $H(u, v)$ thường được sử dụng trong miền tần số gồm:

- **Bộ lọc thông thấp lý tưởng (Ideal-ILPF)**

$$H(u, v) = \begin{cases} 1, & \text{nếu } D(u, v) \leq D_0, \\ 0, & \text{nếu } D(u, v) > D_0, \end{cases}$$

trong đó $D(u, v) = \sqrt{(u - u_0)^2 + (v - v_0)^2}$, thường chọn $(u_0, v_0) = (0, 0)$, D_0 là tần số cắt-tần số chặc cụt.

- **Bộ lọc thông thấp Butterworth (BLPF):**

$$H(u, v) = \frac{1}{1 + \left(\frac{D(u, v)}{D_0}\right)^{2n}},$$

trong đó

- n là thứ tự lọc.
- $D(u, v) = \sqrt{(u - u_0)^2 + (v - v_0)^2}$ (thường chọn $(u_0, v_0) = (0, 0)$).
- D_0 là tần số cắt, hay tần số chặc cụt, còn được kí hiệu là r_0 .

- **Bộ lọc thông thấp Gauss (GLPF):**

$$H(u, v) = e^{-\frac{D(u, v)^2}{2\sigma^2}},$$

trong đó σ điều khiển độ mờ của ảnh. Bộ lọc Gauss có đặc tính mượt, không gây rung biên, và được sử dụng phổ biến nhất.

Một số bộ lọc thông cao

- **Bộ lọc thông cao lý tưởng (Ideal-IHPF)**

$$H(u, v) = \begin{cases} 0, & \text{nếu } D(u, v) \leq D_0, \\ 1, & \text{nếu } D(u, v) > D_0, \end{cases}$$

trong đó $D(u, v) = \sqrt{(u - u_0)^2 + (v - v_0)^2}$, thường chọn $(u_0, v_0) = (0, 0)$.

- **Bộ lọc thông cao Butterworth (BHPF)**

$$H(u, v) = \frac{1}{1 + \left[\frac{D_0}{D(u, v)} \right]^{2n}},$$

trong đó:

- n là thứ tự lọc.
- D_0 là tần số cắt.

- **Bộ lọc thông cao Gauss (GHPF)**

$$H(u, v) = 1 - e^{-\frac{D(u, v)^2}{2\sigma^2}},$$

trong đó:

- Tham số σ đo mức độ lan truyền hay phân tán của đường cong Gauss.

Kết luận

Làm mờ ảnh trong miền tần số là quá trình loại bỏ các tần số cao bằng cách nhân phỏ ảnh với bộ lọc thông thấp. Khi giảm tần số cao nhiều hơn (tức là giảm D_0 hoặc σ), ảnh sẽ càng mờ và mượt. Ngược lại, nếu chọn ngưỡng lớn, ảnh chỉ bị làm mờ nhẹ và vẫn giữ được các chi tiết biên.

1.4.3.1 Lọc thông thấp Ideal

Là phép lọc hai chiều qua tất cả tần số cắt mà không làm giảm nó trong bán kính đường tròn D_0 từ tâm phép lọc và "cắt" hết tần số bên ngoài hình tròn.

$$H(u, v) = \begin{cases} 1, & \text{nếu } D(u, v) \leq D_0, \\ 0, & \text{nếu } D(u, v) > D_0, \end{cases} \quad (1.4.4)$$

trong đó $D(u, v) = \sqrt{(u - u_0)^2 + (v - v_0)^2}$, thường chọn $(u_0, v_0) = (0, 0)$.

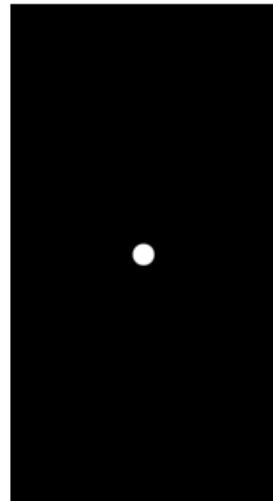
Ta sẽ lọc ảnh sau



Hình 1.11: Ảnh mặt trăng



Bo loc thong thap ly tuong (D0=30.0)



Hình 1.12: Ảnh gốc và ảnh bộ lọc

Code gợi ý:

```
1 | import numpy as np
```

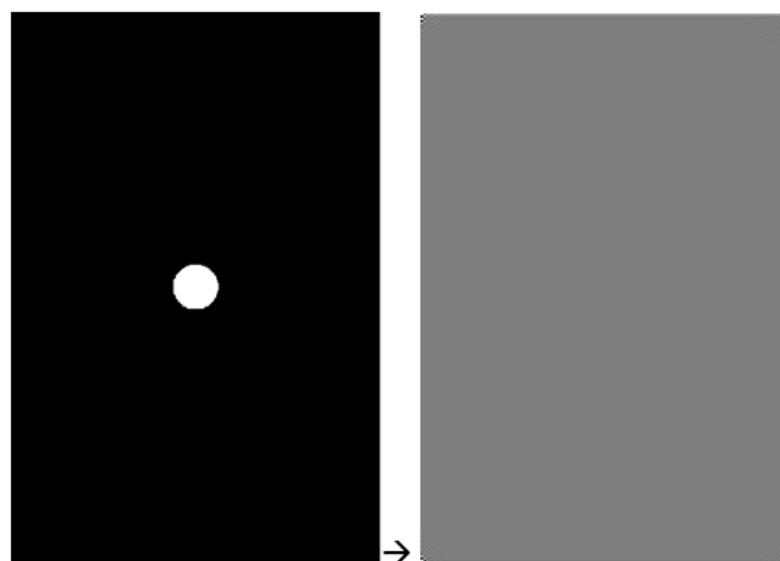
```

2  from PIL import Image
3  import matplotlib.pyplot as plt
4
5  # Doc anh va chuyen sang kieu float (tuong duong double)
6  # Dung r' de tranh loi escape ky tu trong duong dan Windows
7  f = Image.open(r'C:\ImageProcessing\Doan-xulyanh\img\moon.jpg').
8      convert('L')
9  f = np.array(f, dtype=float)
10
11 # Kich thuoc anh
12 P, Q = f.shape
13
14 # Ban kinh bo loc
15 D0 = 30.0
16
17 # Tao bo loc thong thap ly tuong (Ideal Lowpass)
18 h = np.zeros((P, Q))
19 for i in range(P):
20     for j in range(Q):
21         if (i - P / 2)**2 + (j - Q / 2)**2 <= D0**2:
22             h[i, j] = 1
23
24 # Hien thi anh goc va bo loc
25 plt.figure(figsize=(10, 4))
26 plt.subplot(1, 2, 1)
27 plt.imshow(f, cmap='gray')
28 plt.title('Anh goc')
29 plt.axis('off')
30
31 plt.subplot(1, 2, 2)
32 plt.imshow(h, cmap='gray')
33 plt.title(f'Bo loc thong thap ly tuong (D0={D0})')
34 plt.axis('off')
35
36 plt.show()
37
38 # Thong tin kiem tra
39 print("Kich thuoc anh:", f.shape)
40 print("Bo loc h da tao xong voi ban kinh D0 =", D0)

```



Hình 1.13: Chuyển ảnh sang miền tần số



Hình 1.14: Chuyển từ miền không gian sang miền tần số



Hình 1.15: Ảnh mờ sau khi dùng lọc thông thấp Ideal

```

1  from PIL import Image
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import os
5
6  def load_image_grayscale(path):
7      """Doc anh muc xam."""
8      if os.path.exists(path):
9          img = Image.open(path).convert('L')
10         arr = np.array(img).astype(np.float32)
11         print(f"Da mo file: {path} (shape={arr.shape})")
12         return arr
13     else:
14         raise FileNotFoundError(f"Khong tim thay file: {path}")
15
16 def ideal_lowpass_filter(image, D0):
17     """Bo loc thong thap iDeal trong mien tan so."""
18     M, N = image.shape
19     F = np.fft.fft2(image)
20     Fshift = np.fft.fftshift(F)
21
22     u = np.arange(0, M)
23     v = np.arange(0, N)
24     U, V = np.meshgrid(u, v, indexing='ij')
25     D = np.sqrt((U - M//2)**2 + (V - N//2)**2)
26     H = (D <= D0).astype(float)
27
28     Gshift = Fshift * H
29     G = np.fft.ifftshift(Gshift)

```

```

30     img_back = np.fft.ifft2(G)
31     img_back = np.real(img_back)
32
33     # chuan hoa 0-255
34     img_back = img_back - img_back.min()
35     img_back = img_back / img_back.max() * 255.0
36     return img_back
37
38 # ----- tham so -----
39 input_path = "C:\ImageProcessing\Doan-xulyanh\img\moon.jpg" # anh
40             dau vao
41 D0 = 40 # ban kinh cat (cang lon cang it mo)
42
43 # -----
44
45 img = load_image_grayscale(input_path)
46 result = ideal_lowpass_filter(img, D0)
47
48 out_path = "output_moon_ideal_lp.png"
49 Image.fromarray(np.clip(result, 0, 255).astype(np.uint8)).save(
50     out_path)
51 print(f"Da luu ket qua: {out_path}")
52
53 # hien thi
54 plt.figure(figsize=(12,5))
55 plt.subplot(1,2,1)
56 plt.title("Anh goc")
57 plt.axis('off')
58 plt.imshow(img, cmap='gray')
59
60 plt.subplot(1,2,2)
61 plt.title(f"Bo loc Ideal LP (D0={D0})")
62 plt.axis('off')
63 plt.imshow(result, cmap='gray')
64 plt.show()

```

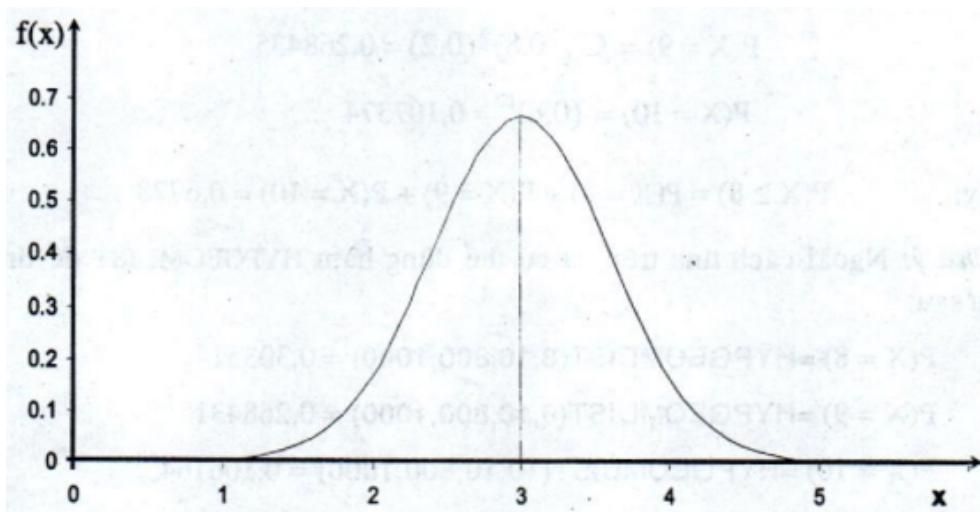
Như vậy với phép lọc thông thấp Ideal, ta đã làm mờ ảnh, làm ảnh giảm các tần số cao, ví dụ như độ nét biên của hình mặt trăng phía trên sau khi lọc đã giảm và các chi tiết cũng mờ đi.

1.4.3.2 Lọc thông thấp Gauss

Đặc trưng cho nhiễu đó là hàm mật độ xác suất thể hiện sự phân bố của nhiễu. Ta sử dụng hàm phân phối Gauss làm bộ lọc nhằm làm mờ ảnh và giảm nhiễu. Trong trường hợp 1 chiều, phân phối Gauss có công thức:

$$G(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2}{2\sigma^2}} \quad (1.4.5)$$

với σ là độ lệch chuẩn của phân phối, Ta giả sử phân phối này có trung bình là 0.



Hình 1.16: Đồ thị phân phối Gauss

Khi xử lý ảnh, ta sẽ sử dụng hàm phân phối Gauss cho 2 chiều, hình thành bằng tích của 2 hàm Gauss 1 chiều x và y

$$G(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2 + y^2}{2\sigma^2}} \quad (1.4.6)$$

Bộ lọc thông thấp Gauss có dạng:

$$G(u, v) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{D^2(u, v)}{2\sigma^2}} \quad (1.4.7)$$

với $D(u, v)$ là khoảng cách từ điểm (u, v) đến tâm hình $(0, 0)$.

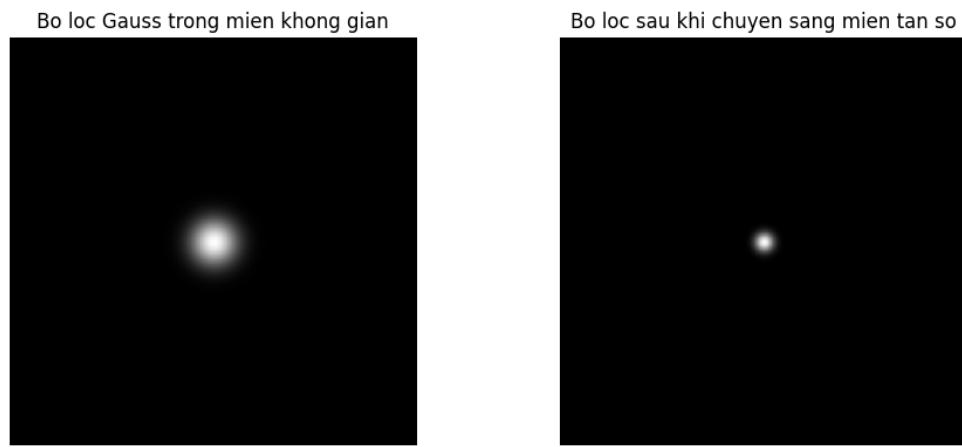
Ta sẽ xử lý ảnh sau với phép lọc thông thấp Gauss



Hình 1.17: Ảnh mặt trăng

khởi tạo bộ lọc

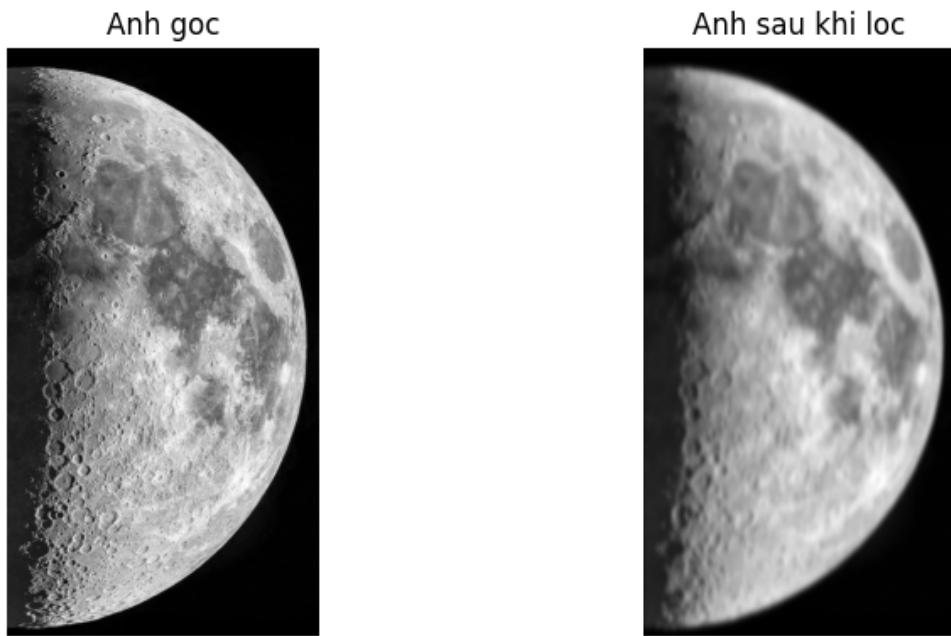
Ảnh của bộ lọc ở miền không gian chuyển sang miền tần số là



Hình 1.18: Bộ lọc Gauss trong miền không gian (trái) sang miền tần số (phải)

Sử dụng thuật toán 1.5 - 3, ta tính tích chập của ảnh gốc trong miền tần số và bộ lọc trong miền tần số, sao đó lấy biến đổi ngược.

Ta được ảnh sau khi lọc là:



Hình 1.19: Ảnh mặt trăng sau bộ lọc thông thấp Gauss

Ảnh sau khi lọc không xảy ra hiệu ứng chuông ở biên mặt trăng như phép lọc thông thấp Ideal. Phép lọc thông thấp Gauss này cho ảnh mượt hơn.

1.4.3.3 Lọc thông thấp Butterworth

Bộ lọc này bao gồm các tính chất của lọc thông thấp Ideal và lọc thông thấp Gauss. Bộ lọc thông thấp Butterworth có dạng sau:

$$H(u, v) = \frac{1}{1 + \left[\frac{D_0}{D(u, v)} \right]^{2n}},$$

trong đó:

- n là thứ tự lọc.

- D_0 là tần số cắt.

Từ dạng của bộ lọc, ta có những tính chất sau

- Với $D(u, v) \ll D_0, H \approx 1$
- Với $D(u, v) \gg D_0, H \approx 0$

- Với $D(u, v) = D_0, H = \frac{1}{2}$

Bây giờ, ta sử dụng bộ lọc này để lọc ảnh Mặt Trăng



Hình 1.20

Khởi tạo bộ lọc có kích thước $P \times Q$ (bằng với kích thước hình ban đầu)

```

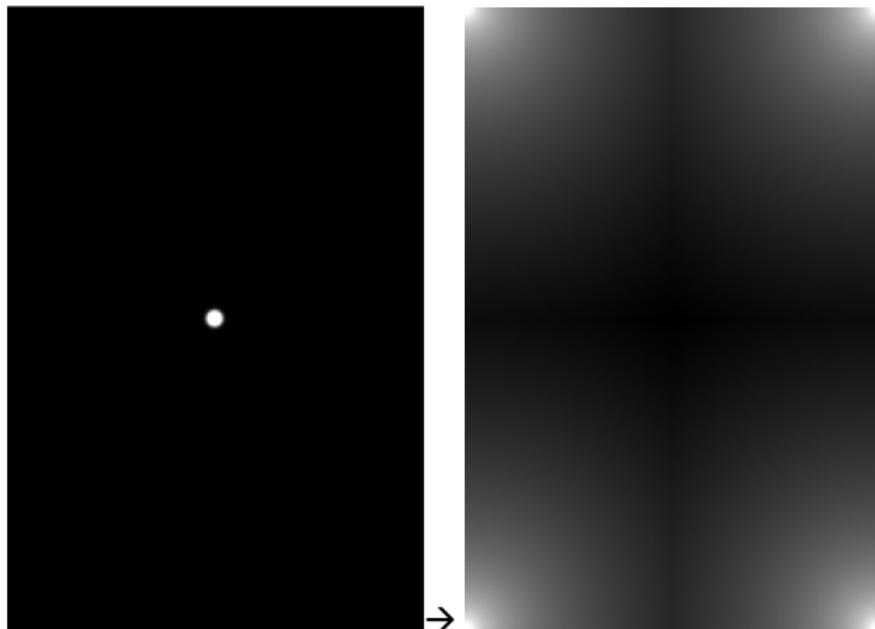
1      import numpy as np
2 from PIL import Image
3 import matplotlib.pyplot as plt
4
5 # Doc anh xam va chuyen sang kieu float
6 f = Image.open(r'C:\ImageProcessing\Doan-xulyanh\img\moon.jpg').
7     convert('L')
8 f = np.array(f, dtype=float)
9
10 # Kich thuoc anh
11 P, Q = f.shape
12
13 # Tham so bo loc Butterworth
14 D0 = 10.0 # tan so cat
    
```

```

14 n = 5          # bac bo loc
15
16 # Tao bo loc thong thap Butterworth
17 h = np.zeros((P, Q))
18 for i in range(P):
19     for j in range(Q):
20         u = np.sqrt((i - P / 2)**2 + (j - Q / 2)**2)
21         h[i, j] = 1 / (1 + (u / D0)**(2 * n))
22
23 # Ap dung bo loc trong mien tan so
24 F = np.fft.fft2(f)
25 F_shift = np.fft.fftshift(F)
26 H = np.fft.fftshift(h)
27 G = H * F_shift
28 g = np.abs(np.fft.ifft2(np.fft.ifftshift(G)))
29
30 # Hien thi anh goc, bo loc va anh sau khi loc
31 plt.figure(figsize=(12, 4))
32 plt.subplot(1, 3, 1)
33 plt.imshow(f, cmap='gray')
34 plt.title('Anh goc')
35 plt.axis('off')
36
37 plt.subplot(1, 3, 2)
38 plt.imshow(h, cmap='gray')
39 plt.title(f'Bo loc Butterworth (D0={D0}, n={n})')
40 plt.axis('off')
41
42 plt.subplot(1, 3, 3)
43 plt.imshow(g, cmap='gray')
44 plt.title('Anh sau khi loc')
45 plt.axis('off')
46
47 plt.tight_layout()
48 plt.show()
49
50 print("Kich thuoc anh:", f.shape)
51 print("Bo loc Butterworth da tao xong voi D0 =", D0, "va cap n =", n)

```

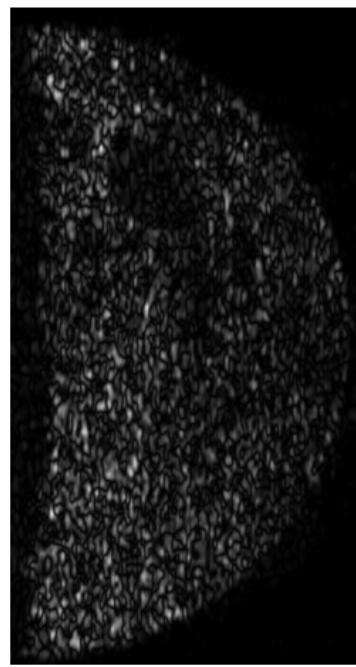
Khi đó, ta được ảnh bộ lọc trong miền không gian và miền tần số là



Hình 1.21: Bộ lọc miền không gian sang miền tần số

Ta được ảnh sau khi lọc là:

Anh sau khi làm mờ (Butterworth)



Hình 1.22: Ảnh sau lọc

Ảnh bị mờ và hầu như mất các chi tiết cần có của ảnh gốc đi. Biến động khá nhiều và hầu như giữ lại thông số ảnh khá ít.

Một số ví dụ khác.



Hình 1.23



Hình 1.24

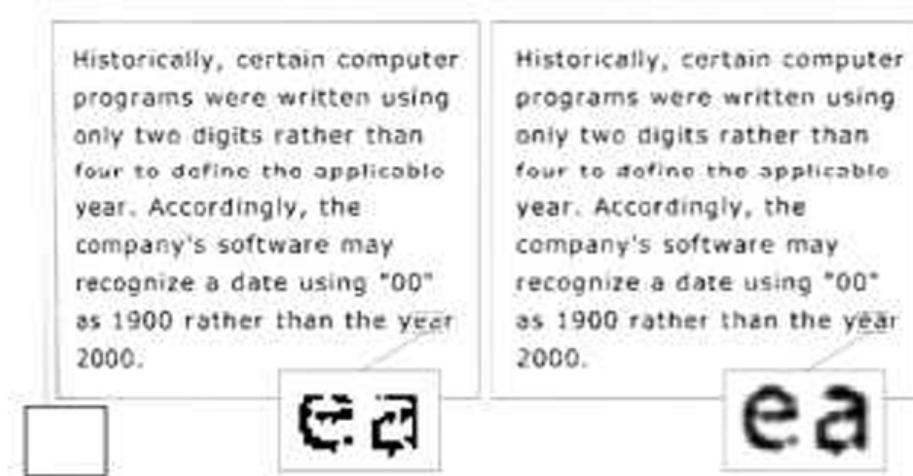
1.4.4 Ứng dụng phép lọc làm mờ ảnh

Phép lọc này có nhiều ứng dụng như:

- Nhận dạng ký tự trong trí thức máy.
- Dùng trong công nghiệp in ấn.
- Xử lý ảnh trên vệ tinh hoặc trên không trung.
- ...

Hình ảnh dưới đây cho thấy một đoạn văn có độ phân giải thấp, một vài ký tự bị đứt gãy.

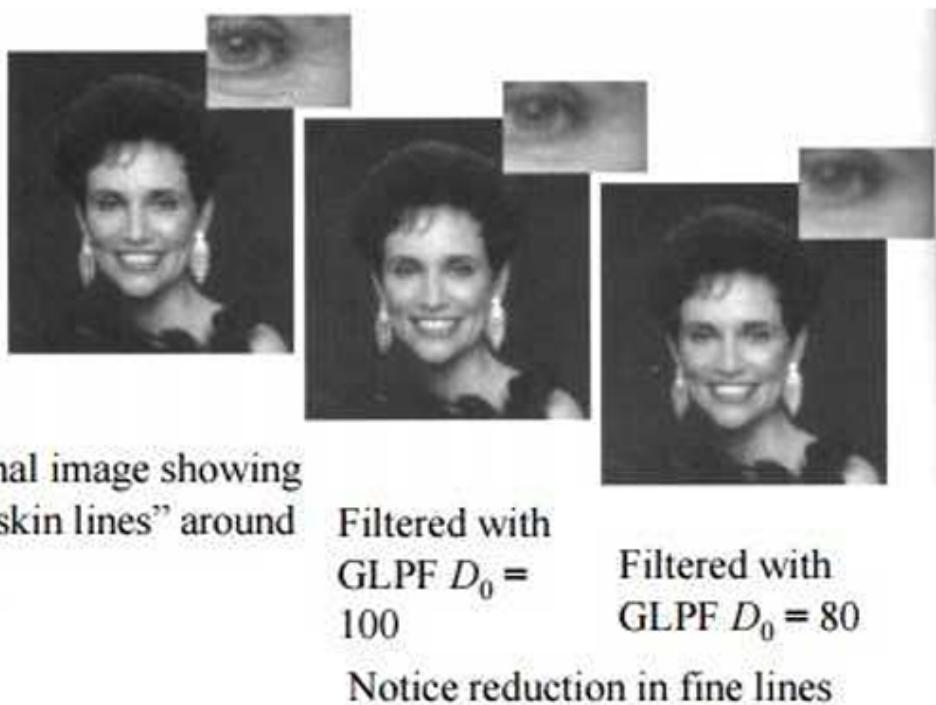
Sử dụng phép lọc thông thấp Gauss, ta được ảnh đoạn văn rõ nét hơn.



Hình 1.25: Ảnh trái: đoạn văn có ký tự "ea" đứt gãy. Ảnh phải: ký tự "ea" liền nét bằng lọc thông thấp Gauss

Mặc dù mắt người dễ dàng nhận diện nét chữ đứt gãy nhưng hệ thống nhận dạng của máy thì không thấy như vậy. Một cách giải quyết đó là ta bắc cầu các mảnh nhỏ lại với nhau bằng cách làm mờ chúng. Hình bên phải cho thấy các kí tự rất nét bằng bộ lọc thông thấp Gauss với $D_0 = 80$.

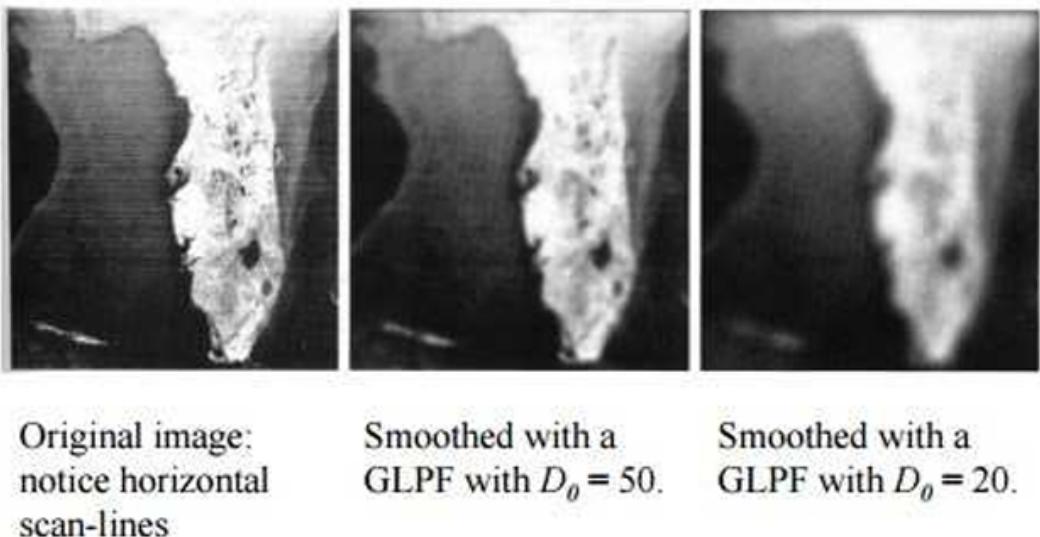
Lọc thông thấp còn được áp dụng trong công nghiệp in ấn với nhiều hàm số dùng để "sơ chế" ảnh, bao gồm ảnh chưa xác định rõ hình dạng. Hình ảnh dưới đây là một ứng dụng của lọc thông thấp, giúp ảnh mượt hơn, dễ nhìn hơn.



Hình 1.26: Áp dụng lọc thông thấp để mắt trông trẻ hơn, ít nếp nhăn hơn (từ trái sang)

Với ảnh người, mục tiêu hướng đến là làm giảm độ nét của đường da mảnh và các nhược điểm nhỏ. Trong hình nền, ta đã làm giảm độ nét đường da quanh mắt, giúp cho ảnh nhìn mượt mà hơn.

Ảnh sau cho ta 2 ứng dụng của lọc thông thấp trên cùng một hình nhưng với các đối tượng khác nhau. Ảnh bên trái có độ phân giải bức xạ rất cao, đó là ảnh Vịnh Mexico (tối) và Florida (sáng) do vệ tinh NOAA chụp lại. Biên của các vật thể trong nước được chụp bởi vòng lặp dòng. Ảnh này dùng để minh họa ảnh viễn thám với cảm biến có xu hướng đưa ra các dòng quét rõ ràng theo hướng quét cảnh.



Hình 1.27: Dùng lọc thông thấp để giảm độ phân giải bức xạ (trái sang phải)

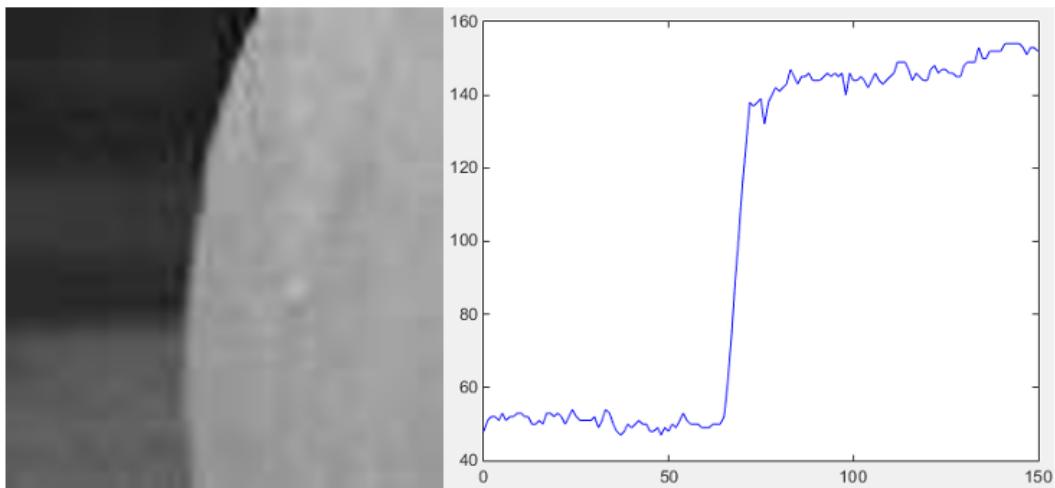
Lọc thông thấp cho kết quả thô nhưng đây là cách đơn giản để làm giảm hiệu ứng đường sọc ngang này. Hình giữa sử dụng lọc thông thấp Gauss với $D_0 = 50$. Làm giảm các hiệu ứng đường quét này giúp đơn giản hóa việc phát hiện các đặc tính như các biên giữa dòng nước biển.

Ảnh bên phải là kết quả của phép lọc thông thấp Gauss với $D_0 = 20$. Ảnh này cho thấy vật thể bị làm mờ nhiều chi tiết nhất có thể, để lại các đặc tính nhận dạng lớn. Cụ thể, cách lọc này là một phần của bước chuẩn bị cho hệ thống phân tích hình ảnh nhằm tìm kiếm các đặc tính ở rìa ảnh.

Phép lọc thông thấp giúp đơn giản hóa khi phân tích bằng cách tính trung bình các đặc tính nhỏ hơn đặc tính mong muốn.

1.4.5 Làm sắc ảnh trong miền tần số

Ta có thể làm mờ ảnh bằng cách làm giảm các tần số cao khi sử dụng chuyển đổi Fourier. Nay, để làm sắc các biên ảnh, ta sẽ lọc các tần số cao và làm giảm các phần có tần số thấp bởi vì đường biên vật thể trong ảnh có độ biến thiên tần số lớn, ứng với các tần số cao.



Hình 1.28: Ảnh phải là đồ thị tần số khi đi quan đường biên vật thể

Khi đi qua đường biên, độ biến thiên tần số lớn.

Cho bộ lọc thông thấp, ta thu được bộ lọc thông cao bằng công thức:

$$H_{HP}(u, v) = 1 - H_{LP}(u, v) \quad (1.4.8)$$

với $H_{HP}(u, v)$ là hàm chuyển đổi lọc thông thấp. Công thức này có nghĩa rằng khi lọc thông thấp. Công thức này có nghĩa rằng khi lọc thông thấp làm giảm tần số thì lọc thông cao bỏ qua điều đó và ngược lại.

Bây giờ, ta sẽ tìm hiểu một số bộ lọc thông cao.

1.4.5.1 Lọc thông cao Ideal

Lọc thông cao Ideal có dạng như sau:

$$H(u, v) = \begin{cases} 0 & \text{nếu } D(u, v) \leq D_0 \\ 1 & \text{nếu } D(u, v) > D_0 \end{cases} \quad (1.4.9)$$

Ta thấy rằng dạng của lọc thông cao Ideal ngược với dạng của lọc thông thấp Ideal. Ta dùng phép lọc này để lọc hình sau:



Hình 1.29: Ảnh quả dưa hấu

Khởi tạo bộ lọc

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from PIL import Image
4
5 # 1. Doc anh goc (anh xam)
6 img = Image.open(r'C:\ImageProcessing\Doan-xulyanh\img\duahau.png',
7     ).convert('L')
8 f = np.array(img, dtype=float)
9 P, Q = f.shape
10
11 # 2. Tao bo loc thong cao ly tuong (Ideal High-pass)
12 D0 = 30.0 # ban kinh vong tron
13
14 h = np.zeros((P, Q))
15 for i in range(P):
16     for j in range(Q):
17         D = np.sqrt((i - P / 2)**2 + (j - Q / 2)**2)
18         if D > D0:
19             h[i, j] = 1
20         else:
21             h[i, j] = 0
22
23 # Dich tam bo loc ve giua (mien tan so)
24 H = np.fft.fftshift(h)
25
26 # 3. Bien doi Fourier anh va ap dung bo loc
27 F = np.fft.fft2(f)
28 F_shift = np.fft.fftshift(F)

```

```

28 G = H * F_shift
29
30 # 4. Bien doi nguoc de lay anh sau khi loc
31 G_ishift = np.fft.ifftshift(G)
32 filtered_img = np.abs(np.fft.ifft2(G_ishift))
33
34 # 5. Hien thi ket qua
35 plt.figure(figsize=(12, 5))
36
37 # Bo loc trong mien khong gian
38 plt.subplot(1, 3, 1)
39 plt.imshow(h, cmap='gray')
40 plt.title('Bo loc thong cao (mien khong gian)')
41 plt.axis('off')
42
43 # Bo loc trong mien tan so
44 plt.subplot(1, 3, 2)
45 plt.imshow(H, cmap='gray')
46 plt.title('Bo loc thong cao (mien tan so)')
47 plt.axis('off')
48
49
50 plt.tight_layout()
51 plt.show()

```

Ta được hình ảnh bộ lọc từ miền không gian sang miền tần số như sau:

Bo loc thong cao (mien khong gian)



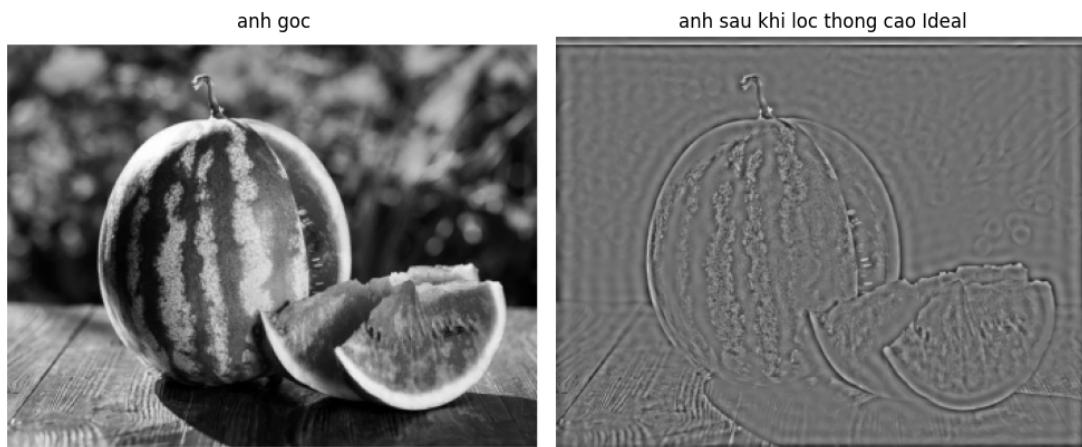
Bo loc thong cao (mien tan so)



Hình 1.30: Bộ lọc miền không gian (trái) sang miền tần số (phải)

Ta tính tích chập của ảnh gốc trong miền tần số và bộ lọc trong miền tần số, sau đó lấy chuyển đổi ngược.

Ta được ảnh sau khi lọc là:

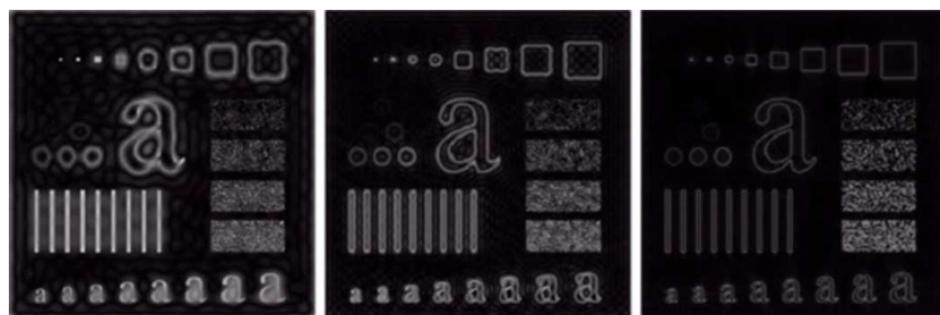


Hình 1.31: Ảnh quả dưa hấu sau khi lọc

Nhìn vào phép lọc này ta có thể thấy phép lọc này có hiệu ứng chuông cho ảnh sau lọc.

Phép lọc này biến các giá trị tần số khác biên trở nên gần nhau, đồng thời làm nổi bật tần số ở biên quả bí ngô.

Ảnh sau đây minh họa hiệu ứng chuông khi sử dụng phép lọc thông cao Ideal với giá trị D_0 (từ trái sang) lần lượt là 15, 30 và 80



Hình 1.32: Hiệu ứng chuông trong ảnh sử dụng bộ lọc thông cao Ideal

1.4.5.2 Lọc thông cao Gauss

Phép lọc thông cao Gauss 2 chiều (2-D) có dạng:

$$\hat{g}(u, v) = 1 - e^{-\frac{\hat{d}^2(u, v)}{2\sigma^2}}. \quad (1.4.10)$$

Ta dùng phép lọc này để lọc ảnh quả dưa hấu sau:



Hình 1.33: Ảnh quả dưa hấu

Khởi tạo bộ lọc trong python với matplotlib:

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from PIL import Image
4
5 # 1 Doc anh goc
6 img = Image.open("img/duahau.png").convert("L")
7 f = np.array(img)
8 P, Q = f.shape
9
10 # 2 Tao bo loc thong cao Gauss
11 sig = 40
12 b = 2 * sig * sig
13
14 h = np.zeros((P, Q))
15 for i in range(P):
16     for j in range(Q):
17         D = (i - P/2)**2 + (j - Q/2)**2
18         h[i, j] = 1 - np.exp(-D / b)
19
20 # Dich tam bo loc ve giua
21 H = np.fft.fftshift(h)
22
23 # 3 Bien doi Fourier anh va ap bo loc
24 F = np.fft.fft2(f)
25 F_shift = np.fft.fftshift(F)
26 G = H * F_shift
27
28 # 4 Bien doi nguoc de lay anh sau loc

```

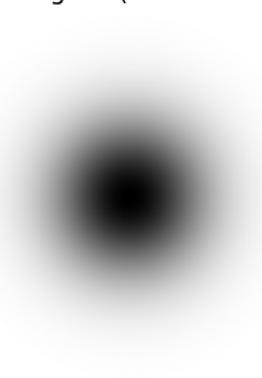
```

29 G_ishift = np.fft.ifftshift(G)
30 filtered_img = np.abs(np.fft.ifft2(G_ishift))
31
32 # 5 Hien thi ket qua
33 plt.figure(figsize=(10, 6))
34
35 # Bo loc trong mien khong gian
36 plt.subplot(1, 3, 1)
37 plt.imshow(h, cmap='gray')
38 plt.title("Bo loc thong cao mien khong gian")
39 plt.axis('off')
40
41 # Bo loc trong mien tan so
42 plt.subplot(1, 3, 2)
43 plt.imshow(H, cmap='gray')
44 plt.title("Bo loc thong cao mien tan so")
45 plt.axis('off')
46
47
48 plt.tight_layout()
49 plt.show()

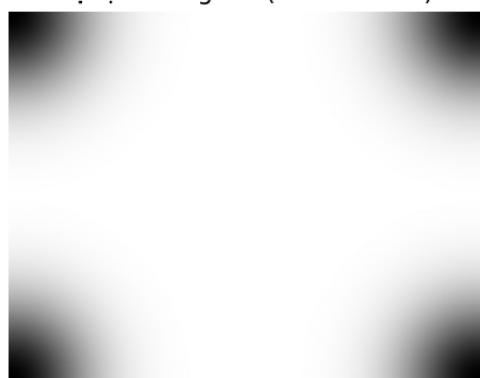
```

Ta được ảnh bộ lọc trong miền không gian và miền tần số như sau:

Bộ lọc thông cao(miền không gian)



Bộ lọc thông cao(miền tần số)



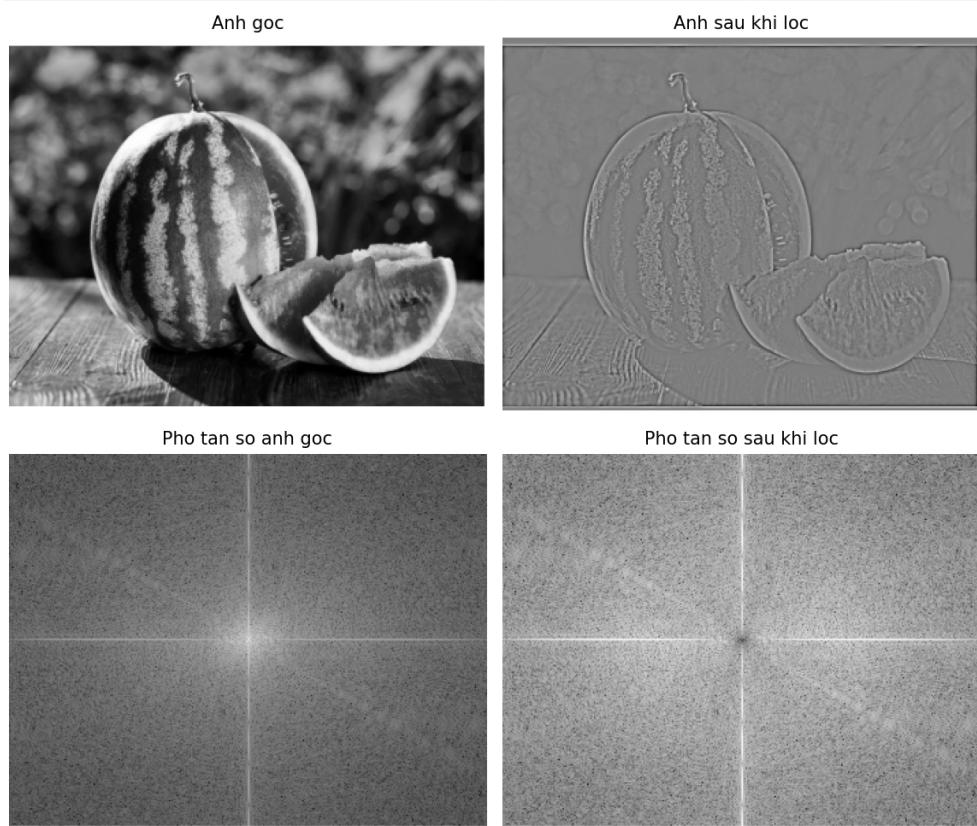
Hình 1.34: Ảnh lọc thông cao Gauss miền không gian(bên trái) sang miền tần số(bên phải)

Anh sau khi lọc



Hình 1.35: Ảnh quả dưa hấu sau khi lọc thông cao Gauss

Ta quan sát phổ miền tần số của ảnh trước và sau khi lọc thông cao Gauss:



Hình 1.36: So sánh phổ tần số của ảnh trước và sau khi lọc

Bảng 1.1: Tổng hợp đặc điểm và tác dụng của lọc thông cao Gauss

Nội dung	Mô tả
Ảnh gốc	Chứa nhiều tàn số thấp, vùng sáng tối mượt và ít chi tiết
Ảnh sau lọc	Giữ lại chi tiết và biên, loại bỏ vùng phẳng và chậm biến
Độ mịn	Giảm mạnh sau khi lọc, hình ảnh trở nên sắc nét hơn
Độ sắc nét	Tăng rõ rệt, các biên vật thể nổi bật hơn
Phổ tàn số ảnh gốc	Tập trung năng lượng ở trung tâm (tàn số thấp chiếm ưu thế)
Phổ tàn số sau lọc	Trung tâm tối đi, năng lượng chuyển ra vùng rìa (tàn số cao)
Tác dụng chính	Nhấn mạnh chi tiết, làm rõ cấu trúc và biên của vật thể
Cơ chế hoạt động	Loại bỏ thành phần tàn số thấp, giữ lại tàn số cao trong phổ Fourier
Ưu điểm	Dường cong Gauss trơn, không gây nhiễu hoặc hiện tượng chuông
Ứng dụng	Tăng cường ảnh, phát hiện biên, làm sắc nét ảnh trong xử lý ảnh số

Code gợi ý

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from PIL import Image
4
5 # === a Doc anh goc (grayscale) ===
6 img = Image.open("img/duahau.png").convert("L")    # anh xam
7 f = np.array(img, dtype=np.float64)
8 P, Q = f.shape
9
10 # === b Tao bo loc thong cao Gauss ===
11 sigma = 40.0                                     # do rong sigma
12 b = 2 * sigma * sigma
13
14 # tao luoi toa do
15 u = np.arange(P)
16 v = np.arange(Q)
17 U, V = np.meshgrid(u, v, indexing='ij')

```

```

18
19 # khoang cach D(u,v) den tam
20 D2 = (U - P/2)**2 + (V - Q/2)**2
21
22 # ham loc thong cao Gauss
23 H = 1 - np.exp(-D2 / b)
24
25 # === c Bien doi Fourier anh goc ===
26 F = np.fft.fft2(f)
27 F_shift = np.fft.fftshift(F)
28
29 # === d Nhan pho anh voi bo loc ===
30 G_shift = F_shift * H
31
32 # === e Bien doi nguoc Fourier ===
33 G = np.fft.ifftshift(G_shift)
34 g = np.fft.ifft2(G)
35 g_real = np.real(g)
36
37 g_min, g_max = g_real.min(), g_real.max()
38 g_norm = (g_real - g_min) / (g_max - g_min) * 255
39 g_uint8 = np.clip(g_norm, 0, 255).astype(np.uint8)
40
41 # === g Pho tan so ===
42 # dung log(1 + |F|) de nhin ro hon
43 mag_F = np.log(1 + np.abs(F_shift))
44 mag_G = np.log(1 + np.abs(G_shift))
45
46 # === h Hien thi ket qua ===
47 plt.figure(figsize=(14,8))
48
49 plt.subplot(2,3,1)
50 plt.title("Anh goc")
51 plt.imshow(f, cmap='gray')
52 plt.axis('off')
53
54 # plt.subplot(2,3,2)
55 # plt.title("Bo loc thong cao Gauss (H)")
56 # plt.imshow(H, cmap='gray')
57 # plt.axis('off')
58
59 plt.subplot(2,3,2)
60 plt.title("Anh sau khi loc")

```

```

61 plt.imshow(g_uint8, cmap='gray')
62 plt.axis('off')
63
64 plt.subplot(2,3,4)
65 plt.title("Pho tan so anh goc")
66 plt.imshow(mag_F, cmap='gray')
67 plt.axis('off')
68
69 # plt.subplot(2,3,5)
70 # plt.axis('off')
71
72 plt.subplot(2,3,5)
73 plt.title("Pho tan so sau khi loc")
74 plt.imshow(mag_G, cmap='gray')
75 plt.axis('off')
76
77 plt.tight_layout()
78 plt.show()

```

1.4.5.3 Lọc thông cao Butterworth

Phép lọc thông cao Butterworth trong không gian 2 chiều có cấp n và tần số cắt D_0 xác định bởi công thức

$$H(u, v) = \frac{1}{1 + \left[\frac{D_0}{D(u, v)} \right]^{2n}}, \quad (1.4.11)$$

trong đó:

- n là thứ tự lọc.
- D_0 là tần số cắt.

Với $D(u, v)$ là khoảng cách từ tâm hình ảnh đến tọa độ (u, v) .

Ta dùng bộ lọc này để lọc hình quả bí ngô sau:



Hình 1.37

Khởi tạo bộ lọc

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from PIL import Image
4
5 # 1 doc anh goc
6 img = Image.open('C:\ImageProcessing\Doan-xulyanh\img\pumpkin.png')
    ).convert("L")
7 f = np.array(img)
8 P, Q = f.shape
9
10 # 2 tao bo loc thong cao Butterworth
11 D0 = 30      # tan so cat
12 n = 2        # cap cua bo loc
13
14 h = np.zeros((P, Q))
15 for i in range(P):
16     for j in range(Q):
17         D = np.sqrt((i - P/2)**2 + (j - Q/2)**2)
18         if D == 0:
19             h[i, j] = 0
20         else:
21             h[i, j] = 1 / (1 + (D0 / D)**(2 * n))
22
23 # dich tam bo loc ve giua

```

```

24 H = np.fft.fftshift(h)
25
26 # 3 bien doi Fourier anh va ap bo loc
27 F = np.fft.fft2(f)
28 F_shift = np.fft.fftshift(F)
29 G = H * F_shift
30
31 # 4 bien doi nguoc de lay anh sau loc
32 G_ishift = np.fft.ifftshift(G)
33 filtered_img = np.abs(np.fft.ifft2(G_ishift))
34
35 # 5 hien thi ket qua
36 plt.figure(figsize=(10, 6))
37
38 # bo loc trong mien khong gian
39 plt.subplot(1, 3, 1)
40 plt.imshow(h, cmap='gray')
41 plt.title("bo loc Butterworth trong mien khong gian")
42 plt.axis('off')
43
44 # bo loc trong mien tan so
45 plt.subplot(1, 3, 2)
46 plt.imshow(H, cmap='gray')
47 plt.title("bo loc Butterworth trong mien tan so")
48 plt.axis('off')
49
50
51 plt.tight_layout()
52 plt.show()

```

Ta được ảnh của bộ lọc trong miền không gian và miền tần số như sau:

bo loc Butterworth trong mien khong gian
bo loc Butterworth trong mien tan so



Hình 1.38: Ảnh bộ lọc miền không gian (trái) sang miền tần số (phải)

Ta tính tích chập của ảnh gốc trong miền tần số và bộ lọc trong miền tần số, sau đó lấy chuyển đổi ngược.

Ảnh với tần số cắt $D_0 = 30$

anh sau khi loc Butterworth highpass



Hình 1.39: Ảnh sau lọc với tần số cắt $D_0 = 30$

Code gợi ý

```

1 import numpy as np
2 import matplotlib.pyplot as plt
3 from PIL import Image
4
5 # === a doc anh goc (grayscale) ===
6 img = Image.open("C:\ImageProcessing\Doan-xulyanh\img\pumpkin.png")
    ).convert("L")
7 f = np.array(img, dtype=np.float64)
8 P, Q = f.shape
9
10 # === b tao bo loc thong cao Butterworth ===
11 D0 = 30          # tan so cat
12 n = 2            # cap cua bo loc
13
14 # tao luoi toa do
15 u = np.arange(P)
16 v = np.arange(Q)
17 U, V = np.meshgrid(u, v, indexing='ij')
18
19 # khoang cach D(u,v) den tam

```

```

20 D = np.sqrt((U - P/2)**2 + (V - Q/2)**2)
21
22 # ham loc thong cao Butterworth
23 H = 1 / (1 + (D0 / (D + 1e-5))**(2 * n)) # cong 1e-5 de tranh
24 chia 0
25
26 # === c bien doi Fourier anh goc ===
27 F = np.fft.fft2(f)
28 F_shift = np.fft.fftshift(F)
29
30 # === d nhan pho anh voi bo loc ===
31 G_shift = F_shift * H
32
33 # === e bien doi nguoc Fourier ===
34 G = np.fft.ifftshift(G_shift)
35 g = np.fft.ifft2(G)
36 g_real = np.real(g)
37
38 g_min, g_max = g_real.min(), g_real.max()
39 g_norm = (g_real - g_min) / (g_max - g_min) * 255
40 g_uint8 = np.clip(g_norm, 0, 255).astype(np.uint8)
41
42 # === g hien thi anh sau khi loc ===
43 plt.figure(figsize=(6,6))
44 plt.title("anh sau khi loc Butterworth highpass")
45 plt.imshow(g_uint8, cmap='gray')
46 plt.axis('off')
47 plt.show()

```

Ảnh với tần số cắt $D_0 = 70$

anh sau khi loc Butterworth highpass

Hình 1.40: Ảnh sau lọc với tần số cắt D_0

```

1 # butterworth_highpass_duahau.py
2 # loc thong cao Butterworth cho anh duahau.png va chi hien thi anh
3     sau khi loc
4
5 import numpy as np
6 import matplotlib.pyplot as plt
7 from PIL import Image
8
9 # === a doc anh goc (grayscale) ===
10 img = Image.open("C:\ImageProcessing\Doan-xulyanh\img\pumpkin.png")
11     ).convert("L")
12 f = np.array(img, dtype=np.float64)
13 P, Q = f.shape
14
15 # === b tao bo loc thong cao Butterworth ===
16 D0 = 70          # tan so cat
17 n = 2            # cap cua bo loc
18
19 # tao luoi toa do
20 u = np.arange(P)
21 v = np.arange(Q)
22 U, V = np.meshgrid(u, v, indexing='ij')
23
24 # khoang cach D(u,v) den tam
25 D = np.sqrt((U - P/2)**2 + (V - Q/2)**2)

```

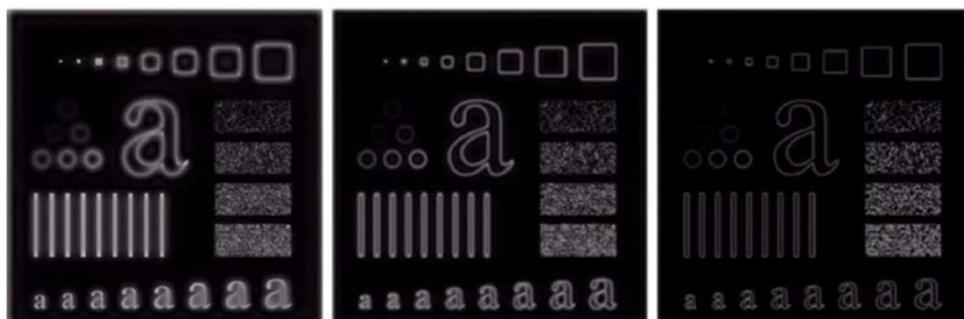
```

25 # ham loc thong cao Butterworth
26 H = 1 / (1 + (D0 / (D + 1e-5))**(2 * n)) # cong 1e-5 de tranh
    chia 0
27
28 # === c bien doi Fourier anh goc ===
29 F = np.fft.fft2(f)
30 F_shift = np.fft.fftshift(F)
31
32 # === d nhan pho anh voi bo loc ===
33 G_shift = F_shift * H
34
35 # === e bien doi nguoc Fourier ===
36 G = np.fft.ifftshift(G_shift)
37 g = np.fft.ifft2(G)
38 g_real = np.real(g)
39
40 # === f chuan hoa ===
41 g_min, g_max = g_real.min(), g_real.max()
42 g_norm = (g_real - g_min) / (g_max - g_min) * 255
43 g_uint8 = np.clip(g_norm, 0, 255).astype(np.uint8)
44
45 # === g hien thi anh sau khi loc ===
46 plt.figure(figsize=(6,6))
47 plt.title("anh sau khi loc Butterworth highpass")
48 plt.imshow(g_uint8, cmap='gray')
49 plt.axis('off')
50 plt.show()

```

Ảnh sau khi lọc có xảy ra hiệu ứng chuông nhưng nhẹ hơn lọc thông cao Ideal, phổ tần số và ảnh cho thấy phần biên có tần số cao hơn hẳn những phần khác.

Hình ảnh sau là ví dụ của phép lọc thông cao Butterworth cấp 2 với các mức D_0 lần lượt (trái sang phải) là 15, 30 và 80



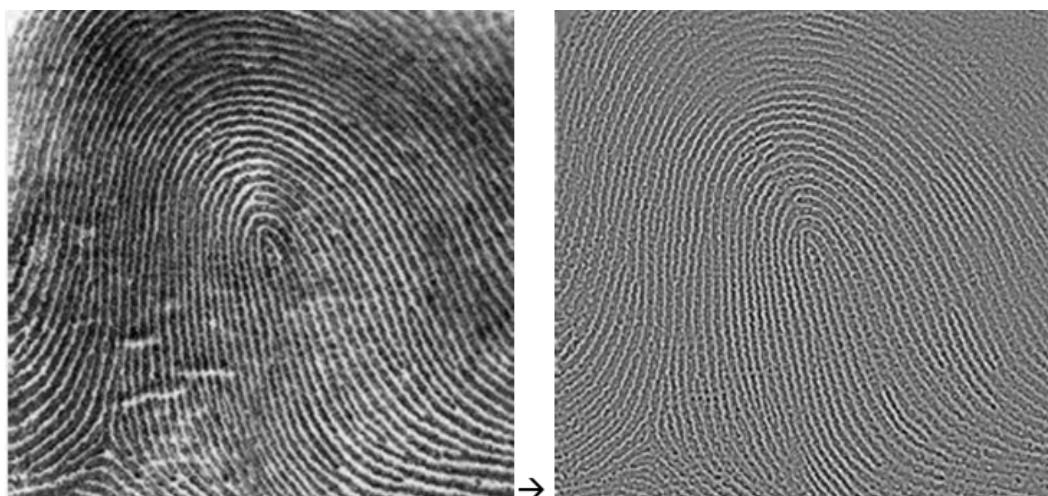
Hình 1.41: Hiệu ứng chuông trong ảnh sử dụng bộ lọc thông cao Butterworth

1.4.6 Ứng dụng phép lọc sắc nét ảnh

Các phép lọc ảnh tần số cao này còn được sử dụng để nhận diện dấu vân tay con người. Một yếu tố quan trọng để máy móc nhận diện được vân tay đó là làm sao để tăng cường các đường vân và giảm thiểu đi các vết bẩn trên vân.

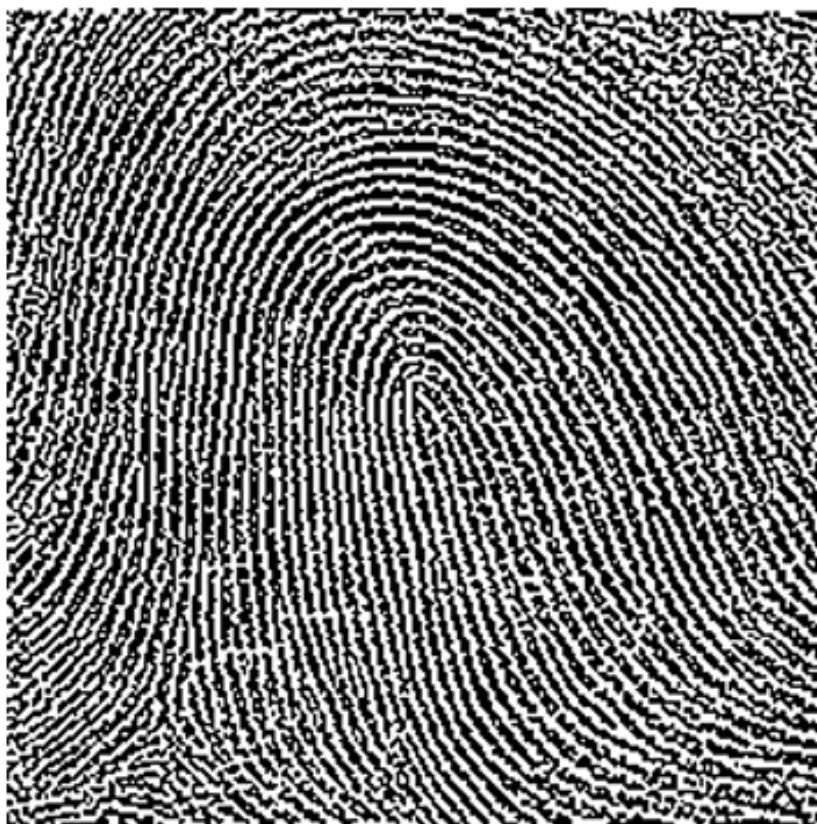
Để tăng cường độ sắc của đường vân ta chú tâm vào yếu tố quan trọng chính là đường vân tay chứa các tần số cao, không thay đổi khi ta sử dụng phép lọc thông cao. Mặt khác phép lọc này còn giảm các thành phần chứa tần số thấp ví dụ như phông nền hoặc các vết bẩn. Vì vậy, ta thu được ảnh tăng cường bằng cách giảm tất cả đặc trưng khác ngoại trừ đặc trưng có tần số cao.

Hình ảnh vân tay này là sử dụng phép lọc thông cao Butterworth cấp 4 và tần số cắt là 50.



Hình 1.42: (Trái): ảnh vân tay có nhiều vết mờ, bẩn, đường vân có nơi không rõ nét và nứt gãy - (Phải): ảnh vân tay rõ nét sau khi sử dụng phép lọc thông cao Butterworth

Để có thể quan sát, ta sẽ biến đổi ảnh bên phải theo quy tắc điểm ảnh là giá trị âm sẽ có màu đen, còn dương thì sẽ có màu trắng



Hình 1.43: Biến đổi ảnh phải phía trên, điểm ảnh có giá trị âm sẽ có màu đen, có giá trị dương là màu trắng

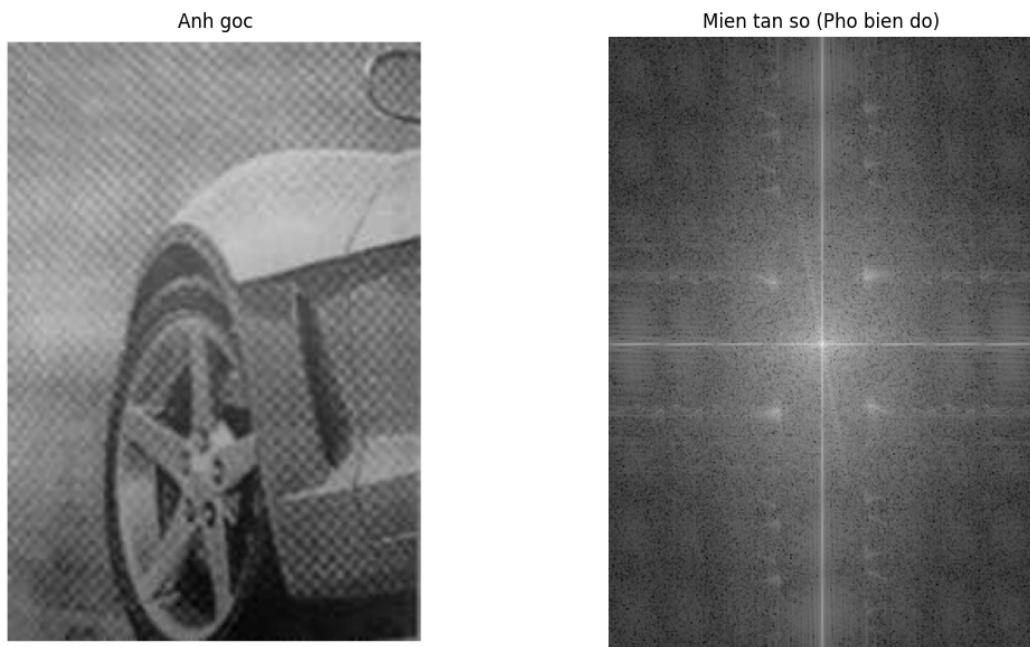
Sử dụng phép lọc thông cao, ta loại bỏ những vết bẩn có trong ảnh ta đang xét và thu về ảnh vân tay rõ nét, thuận lợi hơn nếu công tác trong điều tra vụ án.

1.4.7 Lọc chặn

Đây là một ứng dụng của lọc ảnh trong miền tần số. Phép lọc chặn loại bỏ (hoặc đi qua) các tần số riêng biệt nào đó, ví dụ như nhiều chu kỳ ứng với các nhánh hay các đường trong miền tần số, ta sẽ thiết kế một bộ lọc có tần số 0 tại những vị trí đó sẽ loại bỏ các tần số nhiều.

Ví dụ về nhiều chu kỳ như khám ảnh khi kết hợp nhiều ảnh lại để tạo khám, nhiều dòng quét khi dùng máy ảnh quét, hay nhiều bán sắc (kiểu gợn sóng) của bức ảnh trong tờ báo dưới đây.

Chuyển ảnh vào miền tần số, ta được ảnh sau



Hình 1.44: Ảnh xe hơi trên tờ báo và ảnh sau khi chuyển sang miền tần số

Code gợi ý

```

1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
4
5 def image_to_frequency_domain(image_path):
6     """
7         Chuyen anh sang mien tan so su dung FFT
8     """
9
10    # Doc anh va chuyen sang grayscale neu can
11    img = cv2.imread(image_path)
12    if img is None:
13        print("Khong the doc anh. Kiem tra lai duong dan!")
14        return None, None, None
15
16    if len(img.shape) == 3:
17        img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
18    else:
19        img_gray = img
20
21    # Thuc hien Fast Fourier Transform (FFT)
22    f = np.fft.fft2(img_gray)
23
24    # Dich tan so zero vao trung tam
25    fshift = np.fft.fftshift(f)

```

```

25
26     # Tinh magnitude spectrum (pho bien do)
27     magnitude_spectrum = 20 * np.log(np.abs(fshift) + 1) # +1 de
28         tranh log(0)
29
30
31 # Su dung ham
32 image_path = r"C:\ImageProcessing\Doan-xulyanh\img\locChan\
33         anhXehoi.png"
34 original_img, freq_domain, fshift = image_to_frequency_domain(
35         image_path)
36
37 # Kiem tra xem anh co doc thanh cong khong
38 if original_img is not None:
39     # Hien thi ket qua
40     plt.figure(figsize=(12, 6))
41
42     plt.subplot(1, 2, 1)
43     plt.imshow(original_img, cmap='gray')
44     plt.title('Anh goc')
45     plt.axis('off')
46
47     plt.subplot(1, 2, 2)
48     plt.imshow(freq_domain, cmap='gray')
49     plt.title('Mien tan so (Pho bien do)')
50     plt.axis('off')
51
52     plt.tight_layout()
53     plt.show()
54 else:
55     print("Co loi xay ra khi xu ly anh!")

```

Ta có thể thấy trên ảnh có những đỉnh nhỏ, các đỉnh này tương ứng với dạng nhiễu chu kì của ảnh bên miền không gian.

Các bước lọc chặn:

1. Nhìn vào các phô $|\hat{f}(u, v)|$ của ảnh nhiễu $f(x, y)$, tìm vị trí tần số có liên quan đến nhiễu.
2. Tạo ảnh mặt nạ $\hat{m}(u, v)$ với vết khuyết (các số 0) tại vị trí đó, những vị trí còn

lại có giá trị bằng 1.

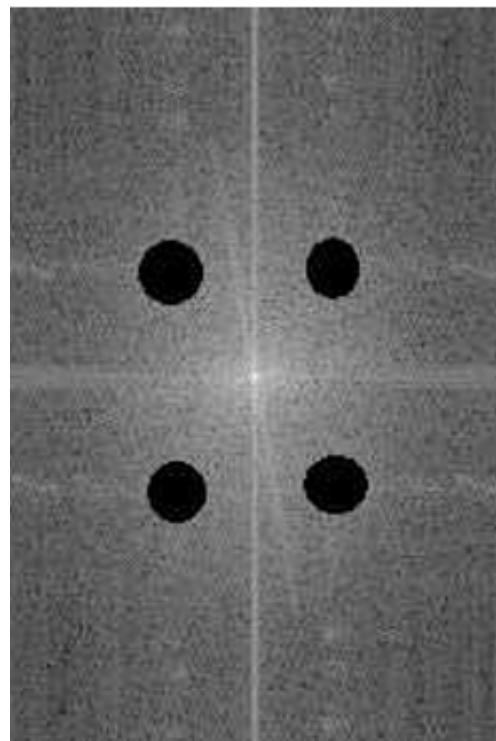
3. Lấy tích mặt nạ với ảnh ban đầu đã được chuyển đổi, các giá trị 0 sẽ làm mất các tần số nhiễu:

$$\hat{g}(u, v) = \hat{m}(u, v) \hat{f}(u, v).$$

4. Lấy biến đổi Fourier ngược để thu về ảnh khôi phục:

$$g(x, y) = \mathcal{F}^{-1}(\hat{g}(u, v)).$$

Ta xác định vị trí có vết khuyết trong ảnh miền tần số sau:



Hình 1.45

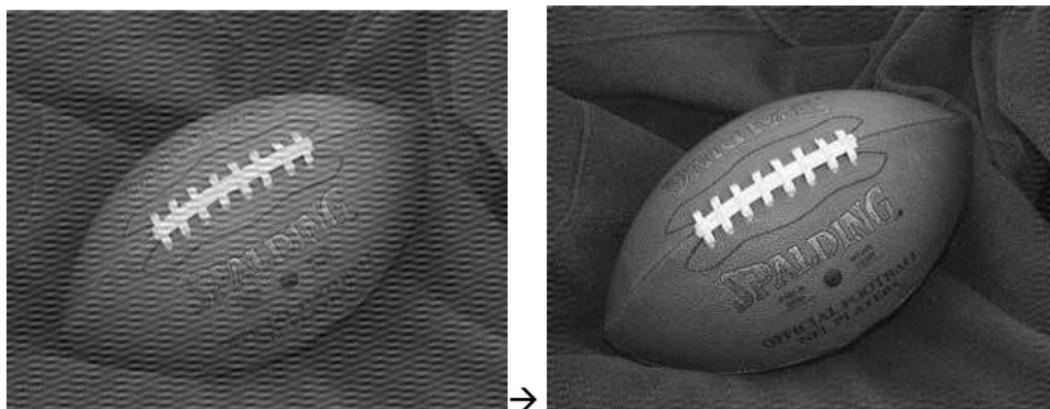
Lấy tích chập, chuyển đổi ngược, ta được kết quả:



Hình 1.46

Như vậy ta đã loại bỏ các đường quết của ảnh ban đầu.

Một ví dụ sau về phép lọc ảnh



Hình 1.47: Ảnh trái là ảnh có đường kẻ sọc ngang, dọc và ảnh phải là ảnh dùng phép lọc chẵn để thu được ảnh rõ ràng hơn

Kết luận

Bài báo cáo này đạt được các vấn đề sau đây:

- Sau khi tìm hiểu về lọc ảnh trong miền tần số, ta thấy phép lọc này có chức năng lọc những tần số thấp và cao trong ảnh.

1. Với thành phần có tần số cao liên quan đến biên vật thể trong ảnh, làm rõ biên. Các phép lọc trong cao làm giảm các thành phần số và bỏ qua thành phần tần số cao.
2. Với thành phần có thành số thấp liên quan đến vùng mượt trong ảnh, làm mờ ảnh. Cho phép lọc thông thấp giảm các thành phần tần số cao và bỏ qua thành phần tần số thấp.

Ngoài ra, phép lọc chặn giúp chúng ta loại bỏ các nhiễu lặp trong ảnh, làm giảm các tần số chọn trước (và một vài lân cận) và bỏ qua các tần số nhiễu khác.

Quy trình chung khi lọc trong miền tần số:

1. Dùng thuật toán FFT chuyển ảnh $f(x, y)$ sang miền tần số thành ảnh $\hat{f}(u, v)$.
2. Tạo bộ lọc $h(x, y)$ cùng kích thước với ảnh cần lọc, dùng FFT chuyển sang miền tần số thành $\hat{h}(u, v)$.
3. Thực hiện phép tính $\hat{g}(u, v) = \hat{h}(u, v)\hat{f}(u, v)$.
4. Lấy chuyển số đổi ngược IFFT của $\hat{g}(u, v)$, ta được hình $g(x, y)$ là hình sau khi lọc.

Tài liệu tham khảo

- [1] Phạm Thanh Bảo - *Bài giảng Phân tích và xử lý ảnh*, Trường Đại học Sài Gòn(2025).
- [2] Võ Hoàng Trọng - *LOC TRONG MIỀN TẦN SỐ VÀ LOC CONTOURLET*, ĐH KHTN(2015).
- [3] Md Zobaer Islam - *Image Enhancement in Frequency Domain* - PhD student, Oklahoma State University.
- [4] Rafael C. Gonzalez(University of Tennessee), Richard E. Woods(Interappitics) - *Digital Image Processing (Fourth edition)*(2018), NXB Pearson.
- [5] Nguyễn Định Cường - *image processing Lecture*(2020).
- [6] Vidya Manian - *Image Enhancement in the frequency domain.*

