

ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH
TRƯỜNG ĐẠI HỌC BÁCH KHOA
KHOA KHOA HỌC VÀ KĨ THUẬT MÁY TÍNH



XỬ LÝ ẢNH SỐ VÀ THỊ GIÁC MÁY TÍNH

TIỂU LUẬN
PHƯƠNG PHÁP XỬ LÝ TỐI ƯU
NHIỀU LOẠI NHIỀU

Giảng viên: Nguyễn Đức Dũng
Sinh viên: Phạm Bá Nhật Khang 2211460



Mục lục

1 Mô tả vấn đề	2
2 Phân tích các giải pháp hiện nay	2
2.1 Lọc Gaussian	2
2.2 Lọc trung vị	5
2.3 Lọc trung bình	6
2.4 Các phương pháp sử dụng học máy	8
3 Hướng tiếp cận xử lý	10
4 Thực hành thí nghiệm	13
4.1 Chuẩn bị dữ liệu ảnh nhiễu	13
4.2 Thí nghiệm phương pháp Fast Non-Local Means	15
5 Đánh giá kết quả	18
6 Kết luận	24
7 Mở rộng	28
8 Mã Python cho thí nghiệm	28
9 Dữ liệu hình ảnh cho thí nghiệm	28



1 Mô tả vấn đề

Trong lĩnh vực xử lý ảnh số và thị giác máy tính, vấn đề nhiễu ở ảnh luôn là một điều không thể tránh khỏi bởi có nhiều yếu tố khác nhau tác động đến quá trình thu nhận tín hiệu. Các yếu tố đó có thể bao gồm ảnh hưởng của môi trường (nhiệt độ, độ ẩm, ...), ánh sáng chụp không đều, cảm biến thiết bị gặp lỗi và nhiều nguyên nhân khác. Một bức ảnh nhiễu sẽ làm mờ, che lấp đi các chi tiết và đặc trưng, điều này sẽ ảnh hưởng nghiêm trọng đến chất lượng của ảnh và hiệu quả của các thuật toán thị giác máy tính sử dụng ảnh như dữ liệu đầu vào để tính toán.

Một phương pháp xử lý nhiễu tốt sẽ phải cân bằng, tối ưu được giữa mức độ giảm nhiễu và mức độ giữ chi tiết ảnh. Một số phương pháp sẽ cho ra kết quả giảm nhiễu tốt nhưng phải đánh đổi bằng sự mất nét ở các chi tiết và ngược lại. Vì thế, với mỗi loại nhiễu và mục tiêu khác nhau khác nhau, các nhà nghiên cứu đã tìm ra các phương pháp lọc nhiễu đặc trưng tương ứng có thể tối ưu tốt được giữa hai yếu tố phải đánh đổi. Ví dụ đối với loại nhiễu Gaussian nhẹ và trung bình, phương pháp lọc Gaussian sẽ đơn giản và hiệu quả nhất do nó hoạt động dựa trên chính phân phối Gaussian của nhiễu và với loại nhiễu Salt and Pepper, phương pháp lọc nhiễu median sẽ khử nhiễu tốt mà vẫn giữ lại được các chi tiết trong ảnh.

Tuy nhiên, trong thực tế, ta phải xử lý nhiều bức ảnh với nhiều loại nhiễu khác nhau, vì thế ta không thể áp dụng các phương pháp khử nhiễu đặc trưng như cho từng loại nhiễu được. Để giải quyết bài toán này, ta cần tìm ra một phương pháp có thể xử lý nhiều loại nhiễu một cách hiệu quả và cho phép làm mờ các chi tiết ảnh ở mức độ phù hợp, chấp nhận được.

2 Phân tích các giải pháp hiện nay

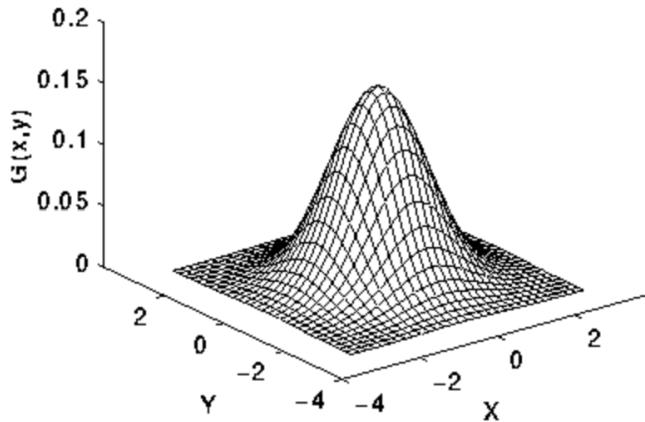
2.1 Lọc Gaussian

Phương pháp lọc Gaussian hoạt động dựa trên phép tích chập (convolution) giữa bức ảnh và một kernel Gaussian. Các trọng số của kernel được tính toán dựa trên hàm phân phối Gaussian hai chiều như sau:

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

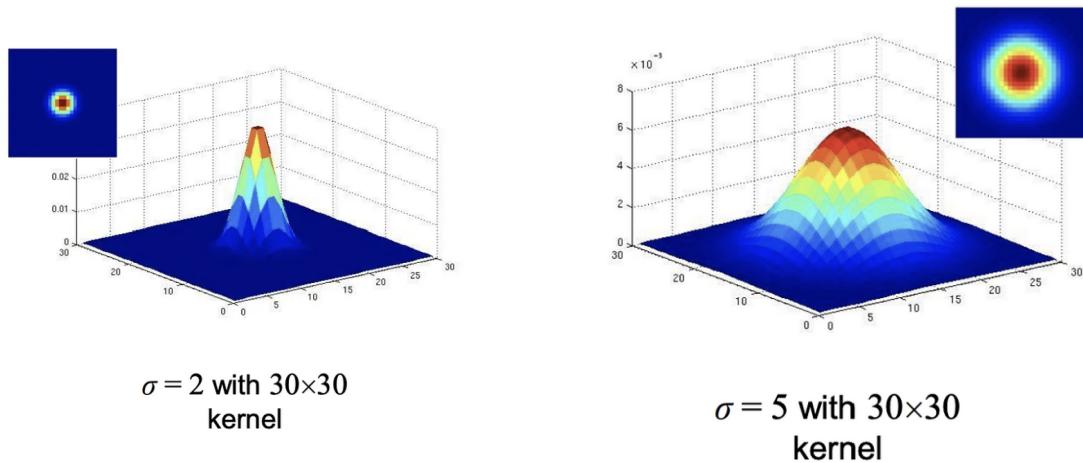
Trong đó:

- (x, y) : là tọa độ của các trọng số trong kernel Gaussian, x là chỉ số hàng và y là chỉ số cột. Trung tâm của kernel Gaussian sẽ có tọa độ là $(0, 0)$.
- σ : là độ lệch chuẩn của phân phối Gaussian, biểu thị mức độ phân tán của hàm phân phối. Khi σ càng nhỏ, phân phối sẽ có xu hướng tập trung, co cụm hơn ở tâm và khi σ càng lớn, phân phối sẽ trải rộng ra xung quanh nhiều hơn.



Hình 1: Đồ thị 2D phân phối Gaussian với $\sigma = 1$

Trong phương pháp lọc nhiễu Gaussian, σ đóng vai trò điều chỉnh mức độ mạn yếu của bộ lọc. Một bộ lọc có σ nhỏ sẽ có các trọng số trong kernel tập trung chủ yếu ở xung quanh tâm và sẽ cho kết quả làm mờ nhẹ. Ngược lại, bộ lọc có σ lớn sẽ có các trọng số phân bổ đều hơn trên kernel và khi đó tác động làm mờ sẽ mạnh hơn.



Hình 2: Đồ thị 2D phân phối Gaussian với $\sigma = 2$ và $\sigma = 5$

Ưu điểm của bộ lọc Gaussian:



Hình 3: Bức ảnh bị làm mờ bởi bộ lọc Gaussian với 3 mức σ khác nhau

- Bộ lọc Gaussian có tác dụng giảm nhiễu hiệu quả, đặc biệt đối với loại nhiễu Gaussian. Bức ảnh sẽ nên mịn và mượt hơn khi các điểm ảnh có giá trị cao hoặc thấp bất thường so với các điểm ảnh xung quanh đã được hiệu chỉnh bởi bộ lọc.
- Bộ lọc Gaussian có cơ chế hoạt động đơn giản, dễ hiện thực và áp dụng. Chi phí tính toán của bộ lọc không cao nên giúp tiết kiệm tài nguyên và thời gian khi tính toán.

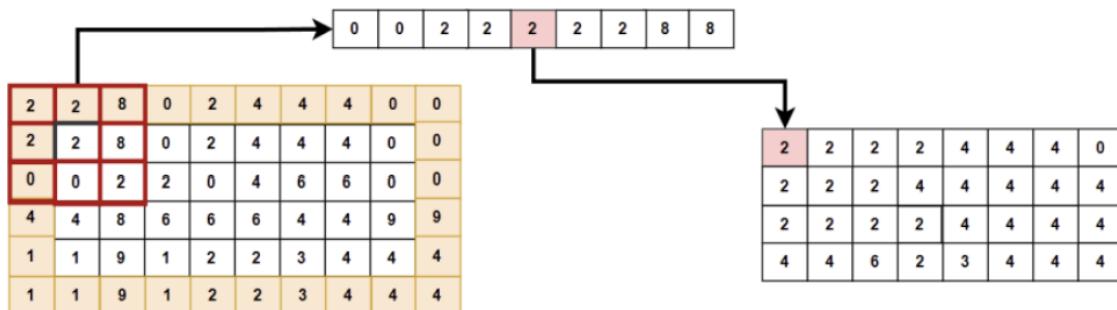
Nhược điểm của bộ lọc Gaussian:

- Bộ lọc Gaussian có thể gây mất các chi tiết và đường biên của bức ảnh khi có σ lớn.
- Bộ lọc Gaussian không thể xử lý loại nhiễu "Salt and Pepper"

Đối với loại nhiễu "Salt and Pepper", bộ lọc Gaussian không thể xử lý tốt được là loại nhiễu này có các giá trị tại điểm ảnh bị nhiễu rất khác (cực đại hoặc cực tiểu) so với các điểm ảnh xung quanh trong khi cơ chế hoạt động của bộ lọc Gaussian là tính toán lấy giá trị trung bình có trọng số của các điểm ảnh lân cận cho điểm ảnh cần tính. Các điểm ảnh lân cận của một điểm ảnh đang cần tính có thể bao gồm các điểm bị nhiễu "Salt and Pepper" với các giá trị rất khác biệt, điều này dẫn tới việc tính toán ra giá trị không phù hợp và kết quả có thể làm mất đi các chi tiết quan trọng của ảnh mà vẫn không loại được hết nhiễu. Ngoài ra, nhiễu "Salt and Pepper" có vị trí xuất hiện ngẫu nhiên trên ảnh trong khi bộ lọc Gaussian chỉ xử lý tốt đối với nhiễu phân phối theo Gaussian nên hiệu quả xử lý của bộ lọc không cao khi phải xử lý loại nhiễu này.

2.2 Lọc trung vị

Lọc trung vị (median filter) là một phép lọc sẽ thay đổi giá trị của điểm ảnh được lọc bằng giá trị trung vị của các điểm ảnh lân cận, xung quanh. Bộ lọc sẽ có một kernel được trượt trên từng điểm ảnh của ảnh, bên trong kernel sẽ chứa các giá trị của các điểm ảnh xung quanh. Khi tính toán trên một điểm ảnh, các giá trị trong kernel sẽ được sắp xếp theo thứ tự thành một chuỗi và giá trị ở giữa chuỗi sẽ được chọn làm giá trị lấp vào điểm ảnh đó.



Hình 4: Cơ chế hoạt động của bộ lọc trung vị

Ưu điểm của bộ lọc trung vị

- Xử lý nhiễu "Salt and Pepper" hiệu quả vì cơ chế của bộ lọc là lấy giá trị trung vị nên các giá trị khác thường lớn (cực đại hay cực tiểu) không làm ảnh hưởng đến kết quả tính toán.
- Các đường nét, chi tiết biên được giữ lại tương đối ổn.
- Cơ chế hoạt động đơn giản nên giúp giảm thiểu chi phí tính toán và tài nguyên khi hiện thực.

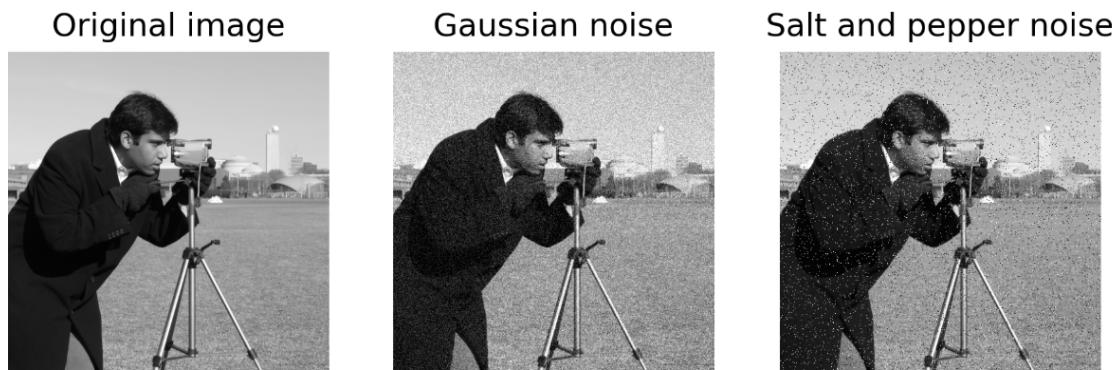
Nhược điểm của bộ lọc trung vị

- Lọc trung vị không tối ưu cho các loại nhiễu khác, ví dụ như nhiễu Gaussian.
- Khi bức ảnh bị nhiễu quá nhiều, dày bộ lọc sẽ loại bỏ được nhiều tốt và gây mờ, mất đi các chi tiết của bức ảnh.

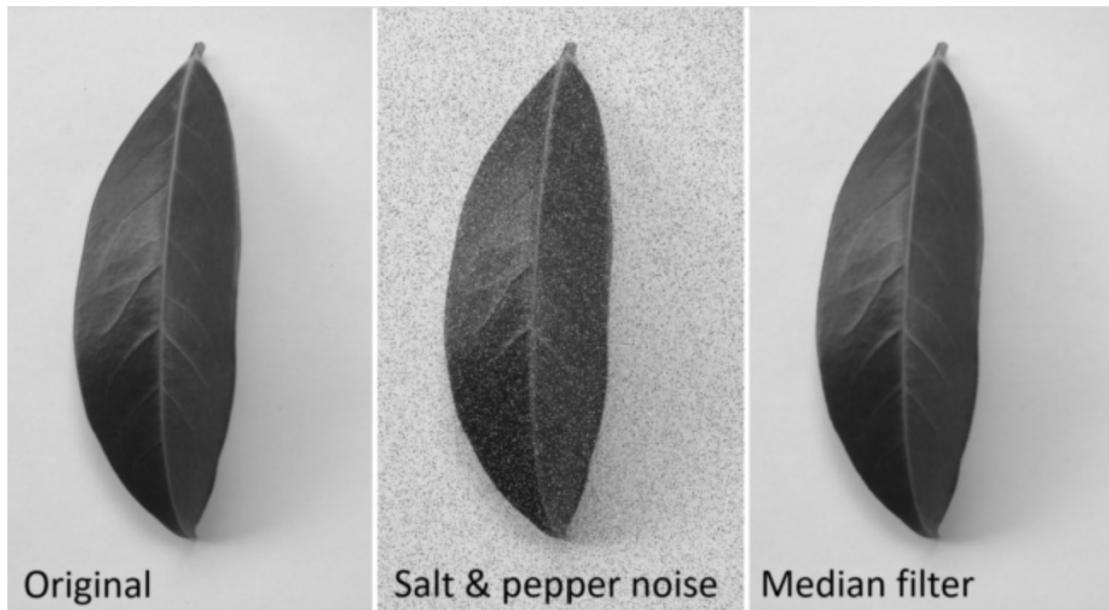
Đối với loại nhiễu Gaussian, bộ lọc trung vị không thể hoạt động tốt được vì loại nhiễu này có phân phối Gaussian, các điểm ảnh bị nhiễu sẽ có độ sai lệch không quá lớn so với các điểm ảnh xung quanh. Khi sử dụng bộ lọc trung vị, giá trị trung vị trong kernel sẽ không có khác biệt lớn so với giá trị điểm ảnh cần thay đổi do các



giá trị trong kernel tương đối không quá cách biệt. Các giá trị bị nhiễu Gaussian sẽ không được loại bỏ sau khi sử dụng bộ lọc trung vị.



Hình 5: Minh họa về nhiễu Gaussian và nhiễu Salt and Pepper



Hình 6: Áp dụng bộ lọc trung vị lên nhiễu Salt and Pepper

2.3 Lọc trung bình

Lọc trung bình là phương pháp lọc sẽ dùng một kernel trượt trên các điểm ảnh của bức ảnh như bộ lọc Gaussian và trung vị, tuy nhiên, cơ chế tính toán toán của nó là sẽ lấy giá trị trung bình của các giá trị trong kernel để điền vào điểm ảnh cần tính. Mỗi điểm ảnh trong bức ảnh sẽ được tính lại bằng giá trị trung bình của các điểm ảnh lân cận xung quanh.

Ưu điểm của bộ lọc trung bình

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

3x3

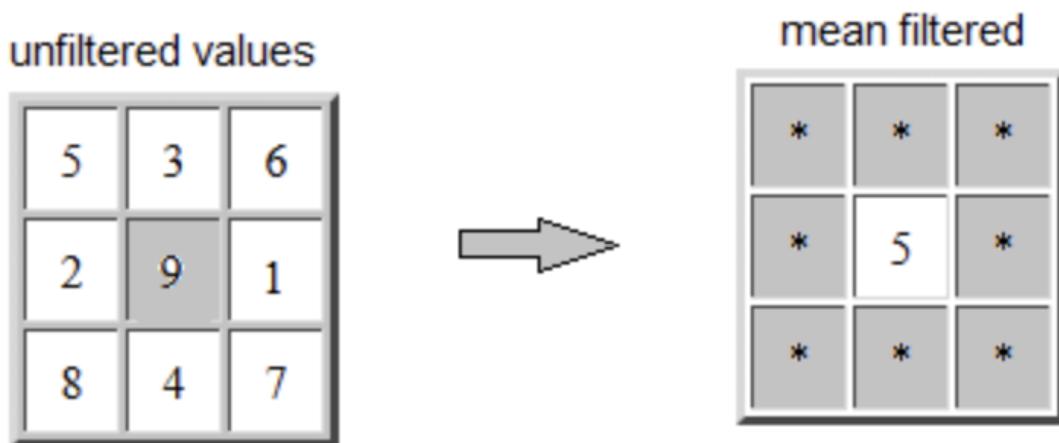
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

5x5

1/n²	1/n²	...	1/n²
1/n²	1/n²	...	1/n²
1/n²	1/n²	...	1/n²
1/n²	1/n²	...	1/n²

nxn

Hình 7: Các kernel của bộ lọc trung bình



$$5 + 3 + 6 + 2 + 9 + 1 + 8 + 4 + 7 = 45 \\ 45 / 9 = 5$$

Hình 8: Cơ chế hoạt động của bộ lọc trung bình

- Bộ lọc trung bình lọc nhiễu, làm mượt ảnh tốt với các loại nhiễu có phân phối đồng đều, ví dụ như nhiễu Gaussian.
- Bộ lọc có cơ chế hoạt động đơn giản nên không tốn nhiều tài nguyên và dễ hiện thực.

Nhược điểm của bộ lọc trung bình

- Bộ lọc trung bình làm mờ các chi tiết và đường biên của bức ảnh do mọi điểm ảnh đều được thay thế bởi giá trị trung bình của vùng lân cận nên các chi tiết có xu hướng mờ đi, hòa vào hình nền.
- Bộ lọc trung bình không hoạt động hiệu quả đối với loại nhiễu "Salt and Pepper" do các giá trị khác biệt lớn của loại nhiễu này ảnh hưởng đến việc

tính toán giá trị trung bình của bộ lọc và cho ra kết quả sai lệch.



Hình 9: Nhiễu Salt and Pepper được xử lý bởi bộ lọc trung bình và bộ lọc trung vị

2.4 Các phương pháp sử dụng học máy

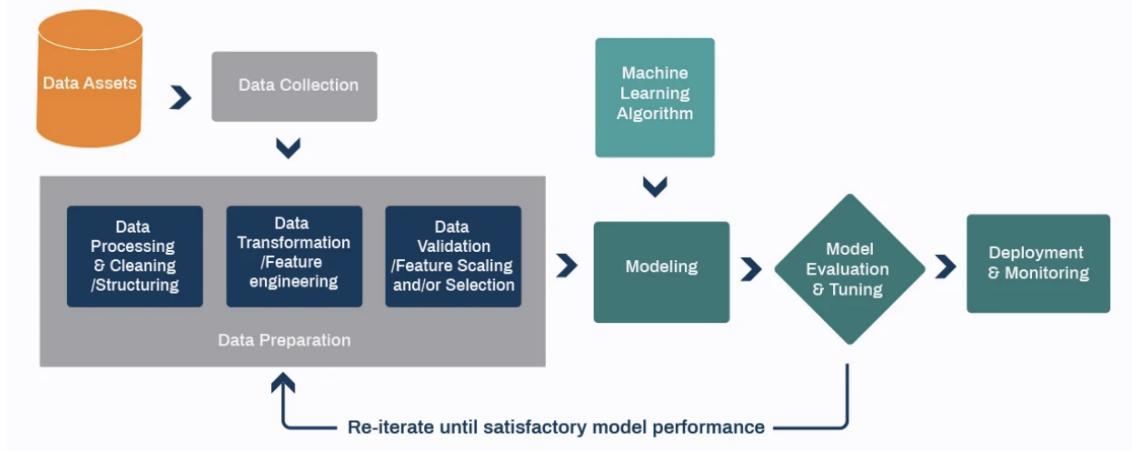
Bên cạnh các phương pháp truyền thống xử lý nhiễu như bộ lọc Gaussian, bộ lọc trung bình hay bộ lọc trung vị, bài toán xử lý nhiễu có thể được giải quyết bằng hướng tiếp cận học máy. Phương pháp học máy sẽ tập trung tìm ra mối quan hệ giữa những bức ảnh bị nhiễu và các bức ảnh không có nhiễu tương ứng để huấn luyện ra một mô hình tối ưu có thể loại bỏ được nhiễu. Hướng tiếp cận này cho phép mô hình có thể tự học và xử lý được nhiều loại nhiễu khác nhau, từ các loại nhiễu đơn giản tới các loại nhiễu phức tạp như nhiễu phi tuyến tính, điều mà các phương pháp truyền thống khó có thể đạt được. Nhiều mô hình học máy đã được áp dụng trong việc xử lý vấn đề nhiễu ở ảnh, từ các mô hình đơn giản tới các mô hình học sâu phức tạp.

Các mô hình đơn giản có thể kể đến như mô hình hồi quy tuyến tính (Linear Regression), mô hình KNN (K-Nearest Neighbors) hay mô hình Random Forest. Trong khi đó, các mô hình học sâu, phức tạp hơn có thể được áp dụng như mô hình sử dụng CNN cho xử lý nhiễu là DnCNN (Denoising Convolutional Neural Network) hay mô hình GANs (Generative Adversarial Networks), tiêu biểu là các mô hình Noise2Noise và Pix2Pix.

Ưu điểm của các phương pháp học máy:

- Mô hình học máy có khả năng tự học dựa trên dữ liệu đầu vào, điều này giúp các nhà phát triển giảm công sức thiết kế thủ công cho các loại nhiễu khác nhau.

Steps to Implement Machine Learning



Hình 10: Quy trình tổng quát cho xây dựng mô hình học máy

- Phương pháp học máy cho phép xử lý nhiều loại nhiễu khác nhau và nhiễu hỗn hợp nhiều loại nhờ sự linh hoạt trong quá trình học dữ liệu đầu vào. Vì thế, loại nhiễu phi tuyến cũng có thể được xử lý trong khi các phương pháp cổ điển khó làm được.
- Các mô hình học máy có độ chính xác cao và độ hiệu quả tốt khi được huấn luyện với dữ liệu đầu vào đủ lớn.

Nhược điểm của các phương pháp học máy:

- Các mô hình học máy phụ thuộc lớn vào dữ liệu đầu vào. Nếu dữ liệu không đủ lớn, đủ đa dạng để bao quát toàn bộ các trường hợp, mô hình sẽ có hiệu quả thấp trong việc xử lý các loại nhiễu khác nhau, đặc biệt là các loại nhiễu ít được đề cập trong dữ liệu đầu vào. Ngoài ra, dữ liệu đầu vào không được xử lý tốt sẽ ảnh hưởng lên chất lượng của mô hình vì chúng dễ bị ảnh hưởng bởi các dữ liệu ngoại lai, dữ liệu gây nhiễu.
- Việc thu thập dữ liệu đủ lớn và đủ đa dạng có thể khó khăn và tốn kém. Ngoài ra, quá trình tiền xử lý dữ liệu đầu vào đòi hỏi nhiều công sức và chi phí của các nhà phát triển.
- Việc cài đặt và triển khai mô hình học máy yêu cầu lượng tài nguyên tính toán lớn và thời gian huấn luyện lâu, các phần cứng như GPU phải đủ mạnh để có thể giảm thời gian huấn luyện mô hình. Điều này gây khó khăn cho việc tích hợp mô hình lên các ứng dụng, thiết bị có tài nguyên hạn chế và yêu cầu thời gian xử lý ngắn.



3 Hướng tiếp cận xử lý

Các phương pháp xử lý nhiễu cổ điển đơn giản như bộ lọc trung vị, trung bình hay bộ lọc Gaussian chỉ giải quyết được một số loại nhiễu đặc trưng của phương pháp mà không có sự linh động trong việc xử lý tốt các loại nhiễu khác. Trong khi đó, các phương pháp sử dụng hướng tiếp cận học máy có thể xử lý tốt nhiều loại nhiễu khác nhau, tuy nhiên chúng lại có độ phức tạp cao, đòi hỏi nhiều nguồn tài nguyên, dữ liệu huấn luyện lớn và thời gian phát triển lâu. Để cân bằng được giữa hai yếu tố là khả năng linh hoạt trong xử lý nhiều loại nhiễu và ít tiêu tốn tài nguyên, chúng ta sẽ chọn cách tiếp cận dùng phương pháp lọc nhiễu Non-local means.

Phương pháp Non-local means có tính tương đồng với bộ lọc trung bình là giá trị thay thế cho điểm ảnh bị nhiễu là trung bình có trọng số của các điểm ảnh khác. Tuy nhiên, phương pháp này tính trung bình có trọng số cho điểm ảnh bị nhiễu trên tất cả các điểm ảnh khác của ảnh, tùy thuộc vào sự tương đồng với điểm ảnh bị nhiễu mà mỗi điểm ảnh sẽ có các trọng số khác nhau. Giá trị mới cho điểm ảnh bị nhiễu sẽ được tính bởi công thức sau:

$$NL[v](i) = \sum_{j \in I} w(i, j)v(j)$$

Với

$$0 \leq w(i, j) \leq 1$$

và

$$\sum_j w(i, j) = 1$$

Trong đó:

- $v(i)$: giá trị của điểm ảnh i
- $w(i, j)$: trọng số của điểm ảnh j với điểm ảnh i

Mỗi trọng số $w(i, j)$ giữa hai điểm ảnh i và j sẽ được tính dựa trên sự tương đồng của hai khối xung quanh (patch) các điểm ảnh đó thông qua phép tính khoảng cách Euclidean, công thức của trọng số $w(i, j)$ như sau:

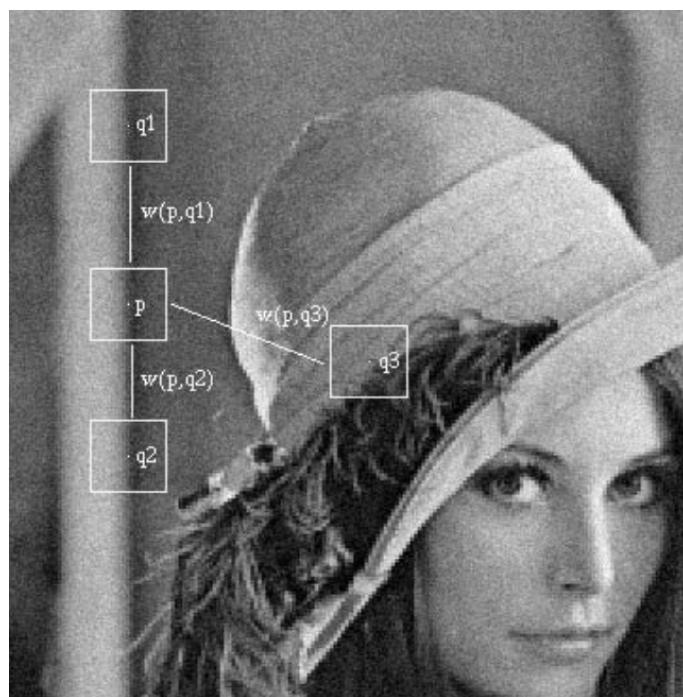
$$w(i, j) = \frac{1}{Z(i)} e^{-\frac{\|v(\mathcal{N}_i) - v(\mathcal{N}_j)\|_{2,a}^2}{h^2}},$$

Trong đó, $Z(i)$ là hằng số chuẩn hóa nhằm đảm bảo ràng buộc tổng các trọng số của các điểm ảnh j so với điểm ảnh i bằng giá trị 1.

$$Z(i) = \sum_j e^{-\frac{\|v(\mathcal{N}_i) - v(\mathcal{N}_j)\|_{2,a}^2}{h^2}}.$$

Trong công thức tính trọng số $w(i, j)$:

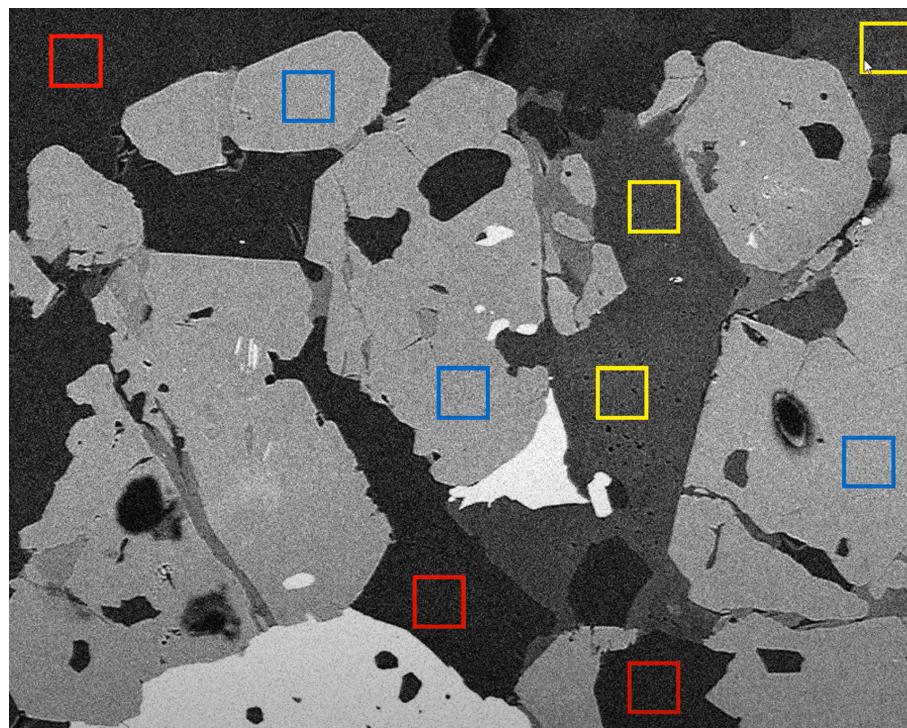
- N_i : là khối lân cận vuông với kích thước cố định có tâm là điểm ảnh i .
- a : là độ lệch chuẩn của Gaussian kernel.
- h : là tham số điều chỉnh độ mạnh yếu của bộ lọc, h càng lớn thì lọc càng mạnh và ngược lại



Hình 11: Các vùng tương đồng sẽ cho ra trọng số lớn như $w(p, q1)$ và $w(p, q2)$, trong khi $q3$ ít tương đồng với p nên trọng số $w(p, q3)$ sẽ nhỏ

Ưu điểm của Non-Local Means:

- Chi tiết ảnh được giữ lại tốt do thuật toán tìm sự tương đồng trên toàn bức ảnh thay vì vùng lân cận.
- Phương pháp này có thể xử lý tốt các loại nhiễu như nhiễu Gaussian, nhiễu Poisson, nhiễu Uniform và tương đối ổn ở các loại nhiễu như Salt and Pepper, nhiễu Speckle.



Hình 12: Minh họa về các vùng tương đồng

- Bộ lọc hoạt động hiệu quả trên những bức ảnh có cấu trúc lặp lại và giữ lại chi tiết các cấu trúc tốt.
- Thuật toán có khả năng tùy chỉnh vào như thay đổi tham số h , kích thước vuông bao quanh (patch) và phạm vi tìm kiếm để có thích ứng tốt cho nhiều loại nhiễu.

Nhược điểm của Non-Local Means:

- Phương pháp có độ phức tạp lớn vì tại đi tìm sự tương đồng trên toàn bộ bức ảnh.
- Thuật toán sẽ hoạt động kém hiệu quả hơn khi nhiễu quá nhiều và dày, các vùng tương đồng sẽ khó được tìm kiếm thấy.
- Bộ lọc nhạy cảm với các tham số điều chỉnh như h và kích thước khối (patch). Nếu giá trị h quá lớn sẽ làm bức ảnh quá mịn, dẫn tới mất đi chi tiết ảnh và ngược lại.

Một trong những nhược điểm lớn của thuật toán Non-Local Means là có độ phức tạp lớn, điều này sẽ dẫn tới tiêu tốn nhiều thời gian và tài nguyên cho việc tính toán, tuy nhiên, nó vẫn nhỏ hơn nhiều so với các hướng tiếp cận xử dụng học máy như DnCNN hay GANs. Dù vậy, chúng ta có thể giảm thiểu độ phức tạp và tăng



tốc độ tính toán bằng cách sử dụng phương pháp được tối ưu hơn là Fast Non-Local Means.

Để tối ưu hóa phương pháp Non-Local Means, phiên bản cải tiến này đã áp dụng một số kỹ thuật như sau:

- Thay vì tìm kiếm các vùng tương đồng trên toàn bộ bức ảnh, phương pháp này sẽ chỉ tìm kiếm trong một khu vực lân cận bị giới hạn (search window). Điều này giúp làm giảm độ phức tạp và thời gian trong quá trình đi tìm vùng tương đồng.
- Các phép tính tổng bình phương trong công thức tính trọng số sẽ được thực hiện bằng ảnh tích phân (Integral Image) để tăng tốc độ tính toán.
- Trong khi tính toán các trọng số $w(i, j)$, phương pháp cải tiến này sẽ sử dụng nội suy thay vì tính toán chính xác để giảm độ phức tạp cho thuật toán.
- Phương pháp sẽ tận dụng việc tính toán song song để có thể xử lý nhiều điểm ảnh cùng lúc.

4 Thực hành thí nghiệm

Để kiểm tra tính hiệu quả của phương pháp Fast Non-Local Means trong việc xử lý các loại nhiễu khác nhau, chúng ta sẽ tiến hành thí nghiệm áp dụng bộ lọc này trên ba loại nhiễu phổ biến là nhiễu Gaussian, nhiễu Salt and Pepper và nhiễu Speckle. Dữ liệu hình ảnh dùng cho thí nghiệm sẽ là hai mươi tám hình trắng đen được lần lượt gắn với ba loại nhiễu trên. Tiếp theo, chúng ta sẽ áp dụng bộ lọc Fast Non-Local Means lên từng loại nhiễu và sẽ đánh giá tính hiệu quả của phương pháp bằng hai thang đo là PSNR và SSIM. Thí nghiệm này sẽ được hiện thực bằng ngôn ngữ Python trên Jupyter Notebook.

4.1 Chuẩn bị dữ liệu ảnh nhiễu

Đầu tiên, ta sẽ áp dụng nhiễu Gaussian lên các bức ảnh trắng đen với trung bình (mean) bằng 0 và độ lệch chuẩn (standard deviation) bằng 10. Các thông số này sẽ tạo ra nhiễu Gaussian ở mức độ vừa phải, nhưng đủ dày để ta có thể thử nghiệm đánh giá cho phương pháp fast Non-Local Means.

```
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_a.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_b.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_c.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_d.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_e.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_f.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_g.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_h.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_i.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_j.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_k.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_l.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_m.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_n.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_o.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_p.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_q.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_r.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_s.jpg
Processed and saved: ./Image_Dataset/Gaussian_Images\GaussianNoise_Image_t.jpg
Add Gaussian Noise successfully!
```

Hình 13: Kết quả gắn nhiễu Gaussian lên ảnh gốc



Hình 14: Ảnh được thêm nhiễu Gaussian so với ảnh gốc - 1



Hình 15: Ảnh được thêm nhiễu Gaussian so với ảnh gốc - 2

Tương tự như vậy, chúng ta sẽ gắn nhiễu Salt and Pepper vào các bức ảnh trắng đen gốc với mật độ nhiễu là 0.005, nghĩa là sẽ có 0.5% số điểm ảnh trong ảnh sẽ bị biến thành nhiễu (mang giá trị cực đại hoặc cực tiểu). Ví dụ nếu một bức ảnh có khoảng 1,000,000 điểm ảnh thì trong số đó sẽ có 5000 điểm ảnh bị nhiễu.

```
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_a.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_b.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_c.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_d.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_e.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_f.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_g.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_h.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_i.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_j.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_k.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_l.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_m.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_n.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_o.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_p.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_q.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_r.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_s.jpg
Processed and saved: ./Image_Dataset/Salt_and_Pepper_Images/Salt_and_Pepper_Image_t.jpg
Add Salt and Pepper noise successfully!
```

Hình 16: Kết quả gắn nhiễu Salt and Pepper lên ảnh gốc



Hình 17: Ảnh được thêm nhiễu Salt and Pepper so với ảnh gốc - 1

Tiếp theo, chúng ta sẽ thêm vào các bức ảnh trắng đen gốc với loại nhiễu Speckle với thông số trung bình (mean) bằng 0 và độ lệch chuẩn (standard deviation) bằng 0.05. Các thông số này sẽ cho ra nhiễu Speckle vừa và nhẹ nhưng đủ để ta có thể nhận thấy và dùng để đánh giá thuật toán Fast Non-Local Means.

4.2 Thí nghiệm phương pháp Fast Non-Local Means

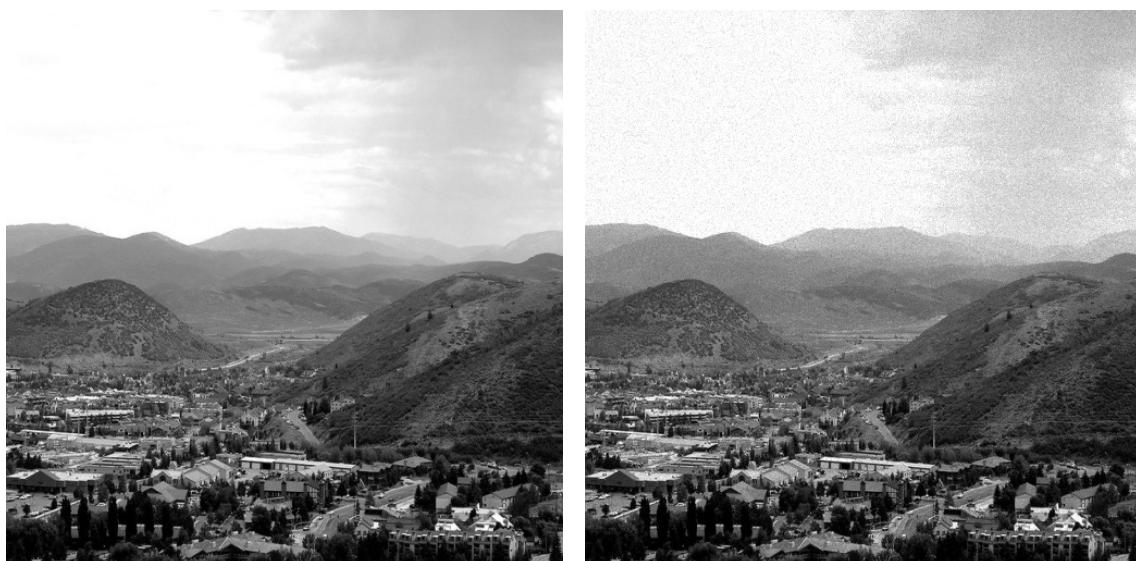
Sau khi đã chuẩn bị dữ liệu hình ảnh nhiễu gồm 20 tấm ảnh nhiễu Gaussian, 20 tấm ảnh nhiễu Salt and Pepper và 20 tấm ảnh nhiễu Speckle, chúng ta sẽ tiến hành



Hình 18: Ảnh được thêm nhiễu Salt and Pepper so với ảnh gốc - 2

```
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_a.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_b.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_c.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_d.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_e.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_f.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_g.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_h.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_i.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_j.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_k.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_l.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_m.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_n.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_o.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_p.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_q.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_r.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_s.jpg
Processed and saved: ../Image_Dataset/Specle/Images/SpecleNoise_Image_t.jpg
Add Speckle noise successfully!
```

Hình 19: Kết quả gắn nhiễu Speckle lên ảnh gốc



Hình 20: Ảnh được thêm nhiễu Speckle so với ảnh gốc - 1



Hình 21: Ảnh được thêm nhiễu Speckle so với ảnh gốc - 2

chạy thuật toán Fast Non-Local Means để lọc nhiễu các hình ảnh và đánh giá kết quả đạt được.

Để hiện thực thuật toán Fast Non-Local Means, chúng ta sẽ sử dụng thư viện cv2 (thư viện OpenCV cho Python). Đây là một thư viện mạnh mẽ hỗ trợ nhiều tác vụ trong lĩnh vực xử lý ảnh và thị giác máy tính. Thư viện OpenCV đã cung cấp một hàm đã hiện thực sẵn thuật toán Fast Non-Local Means là **fastNlMeansDenoising()** để cho ta có thể sử dụng.

```
def fast_nl_means_denoising(image_path, h=10):
    image = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
    if image is None:
        raise ValueError("Image not found or invalid image path")
    denoised_image = cv2.fastNlMeansDenoising(image, None, h, 7,
                                                51)
    return denoised_image
```

Hàm **fastNlMeansDenoising()** sẽ nhận đầu vào là giá trị h , kích thước khối bao quanh (patch) và kích thước của sổ tìm kiếm (searchWindowSize). Giá trị h sẽ là giá trị quyết định độ mạnh yếu của bộ lọc, giá trị h càng lớn thì thuật toán sẽ lọc nhiễu càng mạnh và có thể gây mất nhiều chi tiết hơn và ngược lại. Kích thước của sổ tìm kiếm cũng ảnh hưởng nhiều đến hiệu suất của bộ lọc, nó chính là vùng để tìm kiếm các khối tương đồng, cửa sổ càng lớn thì phạm vi tìm kiếm càng lớn và hiệu quả làm mượt, giữ chi tiết càng tốt, tuy nhiên phải đánh đổi bằng tốc độ xử lý chậm hơn và độ phức tạp lớn hơn.



Trong phần thí nghiệm này, hàm **fastNLMeansDenoising()** có h mang giá trị 13, kích thước khối (patch) là 7 và kích thước của sổ tìm kiếm (searchWindowSize) là 51 nên thuật toán này sẽ vừa đủ mạnh để có thể xử lý các loại nhiễu ồn mà không làm tăng thời gian tính toán và làm mất chi tiết ảnh quá nhiều.

Thuật toán Fast Non-Local Means sẽ được chạy trên từng tập dữ liệu của các ảnh nhiễu Gaussian, nhiễu Salt and Pepper và nhiễu Speckle. Thời gian xử lý từng bức ảnh và thời gian xử lý trung bình một tấm ở từng loại sẽ được ghi lại trong mỗi quá trình chạy. Đầu tiên, chúng ta sẽ chạy thuật toán trên ảnh nhiễu Gaussian, rồi tới ảnh nhiễu Salt and Pepper và cuối cùng là trên nhiễu Speckle.

Sau khi chạy thuật toán trên ba loại nhiễu khác nhau, với mỗi loại nhiễu có 20 tấm ảnh, ta có được tổng thời gian chạy và thời gian chạy trung bình ở mỗi lượt chạy như sau:

	Nhiễu Gaussian	Nhiễu Salt and Pepper	Nhiễu Speckle
Tổng thời gian chạy	27.9048s	29.0964s	27.5015s
Thời gian chạy trung bình	1.3952s	1.4548s	1.3751s

Bảng 1: Bảng thời gian chạy trên các loại nhiễu khác nhau

```
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_a.jpg | Time taken: 1.4637 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_b.jpg | Time taken: 1.4048 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_c.jpg | Time taken: 1.5731 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_d.jpg | Time taken: 1.3347 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_e.jpg | Time taken: 1.3672 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_f.jpg | Time taken: 1.3196 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_g.jpg | Time taken: 1.3654 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_h.jpg | Time taken: 1.3171 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_i.jpg | Time taken: 1.3472 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_j.jpg | Time taken: 1.2718 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_k.jpg | Time taken: 1.3522 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_l.jpg | Time taken: 1.6996 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_m.jpg | Time taken: 1.4298 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_n.jpg | Time taken: 1.5845 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_o.jpg | Time taken: 1.2725 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_p.jpg | Time taken: 1.3431 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_q.jpg | Time taken: 1.3446 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_r.jpg | Time taken: 1.3234 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_s.jpg | Time taken: 1.3307 seconds
Processed and saved: ./Image_Dataset/FastNLM_Gaussian_Images\FastNLM_GaussianNoise_Image_t.jpg | Time taken: 1.4597 seconds
Total processing time: 27.9048 seconds
Average processing time: 1.3952 seconds
Apply Fast Non-Local Means Successfully
```

Hình 22: Thời gian chạy Fast Non-Local Means trên ảnh nhiễu Gaussian

5 Dánh giá kết quả

Dựa trên kết quả thời gian chạy của thuật toán Fast Non-Local Means ở [Bảng 1](#), ta có thể nhận thấy khi xử lý loại nhiễu Salt and Pepper, thuật toán có thời gian chạy tổng và thời gian trung bình nhiều nhất trong khi hai loại nhiễu Gaussian và nhiễu



```
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_a.jpg | Time taken: 2.0374 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_b.jpg | Time taken: 1.6198 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_c.jpg | Time taken: 1.8288 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_d.jpg | Time taken: 1.2951 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_e.jpg | Time taken: 1.4966 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_f.jpg | Time taken: 1.4513 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_g.jpg | Time taken: 1.3295 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_h.jpg | Time taken: 1.2492 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_i.jpg | Time taken: 1.3359 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_j.jpg | Time taken: 1.3429 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_k.jpg | Time taken: 1.4656 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_l.jpg | Time taken: 1.6968 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_m.jpg | Time taken: 1.4631 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_n.jpg | Time taken: 1.3706 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_o.jpg | Time taken: 1.2866 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_p.jpg | Time taken: 1.3214 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_q.jpg | Time taken: 1.3303 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_r.jpg | Time taken: 1.3476 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_s.jpg | Time taken: 1.3063 seconds
Processed and saved: ../Image_Dataset/FastNLM_SaltPepper_Images\FastNLM_Salt_and_Pepper_Image_t.jpg | Time taken: 1.5217 seconds
Total processing time: 29.0964 seconds
Average processing time: 1.4548 seconds
Apply Fast Non-Local Means Successfully
```

Hình 23: Thời gian chạy Fast Non-Local Means trên ảnh nhiễu Salt and Pepper

```
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_a.jpg | Time taken: 1.6899 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_b.jpg | Time taken: 1.5349 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_c.jpg | Time taken: 1.3738 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_d.jpg | Time taken: 1.3846 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_e.jpg | Time taken: 1.4703 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_f.jpg | Time taken: 1.3404 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_g.jpg | Time taken: 1.3224 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_h.jpg | Time taken: 1.2789 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_i.jpg | Time taken: 1.2684 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_j.jpg | Time taken: 1.2441 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_k.jpg | Time taken: 1.4262 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_l.jpg | Time taken: 1.5246 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_m.jpg | Time taken: 1.3714 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_n.jpg | Time taken: 1.2917 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_o.jpg | Time taken: 1.3194 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_p.jpg | Time taken: 1.3367 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_q.jpg | Time taken: 1.4860 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_r.jpg | Time taken: 1.2492 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_s.jpg | Time taken: 1.2824 seconds
Processed and saved: ../Image_Dataset/FastNLM_Speckle_Images\FastNLM_SpeckeNoise_Image_t.jpg | Time taken: 1.3063 seconds
Total processing time: 27.5015 seconds
Average processing time: 1.3751 seconds
Apply Fast Non-Local Means Successfully
```

Hình 24: Thời gian chạy Fast Non-Local Means trên ảnh nhiễu Speckle

Speckle có thời gian chạy trung bình gần tương đương nhau. Điều này có thể giải thích là do loại nhiễu Salt and Pepper có các điểm nhiễu mang giá trị cực đại hoặc cực tiểu làm cho các vùng trong ảnh giảm đi độ tương đồng, nên việc tìm kiếm các vùng ấy của thuật toán trở nên khó hơn. Trong khi đó, nhiễu Gaussian có các giá trị nhiễu liên tục, làm cho các điểm nhiễu ít có sự khác biệt và có sự tương đồng lớn hơn với vùng xung quanh so với nhiễu Salt and Pepper. Điều này giúp cho việc tìm kiếm các vùng tương đồng của thuật toán trở nên thuận lợi hơn nên tiết kiệm được thời gian tính toán. Tương tự, nhiễu Speckle cũng tạo ra các điểm nhiễu ít khác biệt hơn so với nhiễu Salt and Pepper nên quá trình tìm vùng tương đồng ít gấp khó khăn hơn so với nhiễu kia. Dánh giá trên thời gian chạy, ta có thể nhận thấy độ phức tạp và tiêu tốn tài nguyên của thuật toán này tương đối lớn hơn so với các hướng tiếp cận cổ điển khác như bộ lọc Gaussian, bộ lọc trung bình và bộ lọc trung vị.



Tiếp theo, chúng ta sẽ đánh giá hiệu quả chạy của thuật toán Fast Non-Local Means trên ba loại nhiễu dựa trên các thước đo là PSNR và SSIM. Đầu tiên chúng ta sẽ tính thông số PSNR trên từng loại nhiễu rồi đến tính thông số SSIM để đánh giá.

Thông số PSNR (Peak Signal-to-Noise Ratio) là một thước đo được sử dụng để đánh giá chất lượng của hình ảnh sau khi được lọc nhiễu dựa trên sự khác biệt giữa hình ảnh gốc và hình ảnh đã được khử nhiễu. Đơn vị đo của PSNR là decibel (dB), giá trị đo càng cao thì mức độ giống của ảnh được khử nhiễu và ảnh gốc càng cao, nghĩa là thuật toán lọc nhiễu hoạt động càng tốt và ngược lại. Các ngưỡng giá trị PSNR thông dụng để loại mức độ tốt, khá của thuật toán như sau:

- **30-50 dB:** Chất lượng ảnh phục hồi tốt (gần giống ảnh gốc)
- **20-30 dB:** Chất lượng trung bình (xảy ra vài lỗi hoặc còn tương đối nhiễu).
- **Dưới 20 dB:** Chất lượng kém, ảnh bị biến dạng hoặc nhiễu nghiêm trọng.

Kết quả đánh giá PSNR trên các ảnh được lọc ở ba loại nhiễu như sau:

	Nhiễu Gaussian	Nhiễu Salt and Pepper	Nhiễu Speckle
PSNR trung bình	32.97 dB	34.38 dB	33.26 dB

Bảng 2: Bảng kết quả thông số đo PSNR

Dựa vào kết quả PSNR, ta có thể nhận thấy thuật toán Fast Non-Local Means hoạt động tương đối tốt trên cả ba loại nhiễu là Gaussian, Salt and Pepper và Speckle. Cả ba thông số PSNR trung bình đều trên ngưỡng 30 dB, trong đó đối với loại nhiễu Gaussian là 32.97 dB, nhiễu Salt and Pepper là 34.38 dB và nhiễu Speckle là 33.26 dB.

Tiếp theo, ta sẽ đánh giá thuật toán Fast Non-Local Means dựa trên thang đo SSIM. Tương tự như PSNR, ta sẽ đánh giá hiệu quả thuật toán bằng SSIM dựa trên mức độ tương đồng giữa ảnh gốc và ảnh được lọc nhiễu. SSIM (Structural Similarity Index) là một chỉ số dùng để đánh giá sự tương đồng giữa hai hình ảnh và nó sẽ phản ánh tốt cảm nhận của con người về chất lượng hình ảnh. Thang đo này sẽ đánh giá sự tương đồng dựa trên ba yếu tố là: độ sáng (luminance), độ tương phản (contrast) và cấu trúc của ảnh (structure). Đo lường về độ sáng sẽ kiểm tra sự tương đồng về mức độ sáng của hai bức hình và đánh giá độ tương phản sẽ kiểm tra về sự tương đồng giữa sự thay đổi độ sáng của các điểm ảnh. Về yếu tố cấu trúc, thang

Average PSNR: 32.97 dB

	Original Image	Denoised Image	PSNR (dB)
0	Image_a.jpg	FastNLM_GaussianNoise_Image_a.jpg	30.899638
1	Image_b.jpg	FastNLM_GaussianNoise_Image_b.jpg	34.385090
2	Image_c.jpg	FastNLM_GaussianNoise_Image_c.jpg	32.611741
3	Image_d.jpg	FastNLM_GaussianNoise_Image_d.jpg	33.563184
4	Image_e.jpg	FastNLM_GaussianNoise_Image_e.jpg	32.248715
5	Image_f.jpg	FastNLM_GaussianNoise_Image_f.jpg	31.526584
6	Image_g.jpg	FastNLM_GaussianNoise_Image_g.jpg	30.988323
7	Image_h.jpg	FastNLM_GaussianNoise_Image_h.jpg	35.015713
8	Image_i.jpg	FastNLM_GaussianNoise_Image_i.jpg	34.528309
9	Image_j.jpg	FastNLM_GaussianNoise_Image_j.jpg	32.376993
10	Image_k.jpg	FastNLM_GaussianNoise_Image_k.jpg	33.488788
11	Image_l.jpg	FastNLM_GaussianNoise_Image_l.jpg	35.613352
12	Image_m.jpg	FastNLM_GaussianNoise_Image_m.jpg	33.073123
13	Image_n.jpg	FastNLM_GaussianNoise_Image_n.jpg	31.617164
14	Image_o.jpg	FastNLM_GaussianNoise_Image_o.jpg	32.885699
15	Image_p.jpg	FastNLM_GaussianNoise_Image_p.jpg	31.842117
16	Image_q.jpg	FastNLM_GaussianNoise_Image_q.jpg	34.261268
17	Image_r.jpg	FastNLM_GaussianNoise_Image_r.jpg	31.437106
18	Image_s.jpg	FastNLM_GaussianNoise_Image_s.jpg	35.625950
19	Image_t.jpg	FastNLM_GaussianNoise_Image_t.jpg	31.404209

Hình 25: Kết quả PSNR trên ảnh khử nhiễu Gaussian

đo SSIM sẽ đánh giá về mức độ tương đồng về cấu trúc hình học và kết cấu ảnh. Giá trị của SSIM sẽ chạy từ -1 đến 1, giá trị càng gần đến 1 sẽ biểu thị hai ảnh có độ tương đồng cao và bằng 1 khi hai ảnh giống hệt nhau.

Kết quả đánh giá PSNR trên các ảnh được lọc ở ba loại nhiễu như sau:



Average PSNR: 34.38 dB

	Original Image	Denoised Image	PSNR (dB)
0	Image_a.jpg	FastNLM_Salt_and_Pepper_Image_a.jpg	32.974588
1	Image_b.jpg	FastNLM_Salt_and_Pepper_Image_b.jpg	34.983855
2	Image_c.jpg	FastNLM_Salt_and_Pepper_Image_c.jpg	33.381014
3	Image_d.jpg	FastNLM_Salt_and_Pepper_Image_d.jpg	34.493463
4	Image_e.jpg	FastNLM_Salt_and_Pepper_Image_e.jpg	34.531302
5	Image_f.jpg	FastNLM_Salt_and_Pepper_Image_f.jpg	32.853095
6	Image_g.jpg	FastNLM_Salt_and_Pepper_Image_g.jpg	34.961214
7	Image_h.jpg	FastNLM_Salt_and_Pepper_Image_h.jpg	35.677883
8	Image_i.jpg	FastNLM_Salt_and_Pepper_Image_i.jpg	34.922752
9	Image_j.jpg	FastNLM_Salt_and_Pepper_Image_j.jpg	33.117564
10	Image_k.jpg	FastNLM_Salt_and_Pepper_Image_k.jpg	34.663660
11	Image_l.jpg	FastNLM_Salt_and_Pepper_Image_l.jpg	36.097048
12	Image_m.jpg	FastNLM_Salt_and_Pepper_Image_m.jpg	34.842388
13	Image_n.jpg	FastNLM_Salt_and_Pepper_Image_n.jpg	33.512284
14	Image_o.jpg	FastNLM_Salt_and_Pepper_Image_o.jpg	33.696324
15	Image_p.jpg	FastNLM_Salt_and_Pepper_Image_p.jpg	32.883958
16	Image_q.jpg	FastNLM_Salt_and_Pepper_Image_q.jpg	35.017155
17	Image_r.jpg	FastNLM_Salt_and_Pepper_Image_r.jpg	33.632110
18	Image_s.jpg	FastNLM_Salt_and_Pepper_Image_s.jpg	35.402047
19	Image_t.jpg	FastNLM_Salt_and_Pepper_Image_t.jpg	35.930674

Hình 26: Kết quả PSNR trên ảnh khử nhiễu Salt and Pepper

	Nhiễu Gaussian	Nhiễu Salt and Pepper	Nhiễu Speckle
SSIM trung bình	0.8597	0.7455	0.8581

Bảng 3: Bảng kết quả thông số đo SSIM



Average PSNR: 33.26 dB

	Original Image	Denoised Image	PSNR (dB)
0	Image_a.jpg	FastNLM_SpeckeNoise_Image_a.jpg	31.162272
1	Image_b.jpg	FastNLM_SpeckeNoise_Image_b.jpg	34.584426
2	Image_c.jpg	FastNLM_SpeckeNoise_Image_c.jpg	32.565330
3	Image_d.jpg	FastNLM_SpeckeNoise_Image_d.jpg	33.793956
4	Image_e.jpg	FastNLM_SpeckeNoise_Image_e.jpg	32.495887
5	Image_f.jpg	FastNLM_SpeckeNoise_Image_f.jpg	31.831044
6	Image_g.jpg	FastNLM_SpeckeNoise_Image_g.jpg	31.754301
7	Image_h.jpg	FastNLM_SpeckeNoise_Image_h.jpg	35.178619
8	Image_i.jpg	FastNLM_SpeckeNoise_Image_i.jpg	34.256414
9	Image_j.jpg	FastNLM_SpeckeNoise_Image_j.jpg	32.409370
10	Image_k.jpg	FastNLM_SpeckeNoise_Image_k.jpg	33.991055
11	Image_l.jpg	FastNLM_SpeckeNoise_Image_l.jpg	35.620325
12	Image_m.jpg	FastNLM_SpeckeNoise_Image_m.jpg	33.055001
13	Image_n.jpg	FastNLM_SpeckeNoise_Image_n.jpg	32.320779
14	Image_o.jpg	FastNLM_SpeckeNoise_Image_o.jpg	32.844359
15	Image_p.jpg	FastNLM_SpeckeNoise_Image_p.jpg	31.971011
16	Image_q.jpg	FastNLM_SpeckeNoise_Image_q.jpg	34.645791
17	Image_r.jpg	FastNLM_SpeckeNoise_Image_r.jpg	31.798349
18	Image_s.jpg	FastNLM_SpeckeNoise_Image_s.jpg	35.544793
19	Image_t.jpg	FastNLM_SpeckeNoise_Image_t.jpg	33.368277

Hình 27: Kết quả PSNR trên ảnh khử nhiễu Speckle

Dựa vào kết quả đo SSIM trên các ảnh được khử nhiễu, ta có thể thấy thuật toán Fast Non - Local Means hoạt động tốt đối với loại nhiễu Gaussian và nhiễu Speckle với các thông số SSIM trung bình tương ứng lần lượt là 0.8597 và 0.8581 tiệp cận



sát với giá trị 1. Trong khi đó, đối với loại nhiễu Salt and Pepper, thuật toán hoạt động ít hiệu quả hơn khi giá trị SSIM là 0.7455, thấp hơn giá trị SSIM của nhiễu Gaussian và nhiễu Speckle. Điều này có thể giải thích là do nhiễu Salt and Pepper có các giá trị nhiễu cực đại hoặc cực tiểu, làm cho mức độ tương đồng giữa các vùng giảm nên các giá trị trọng số để tính trung bình giữa các vùng có thể không chuẩn xác, dẫn tới chất lượng lọc nhiễu bị ảnh hưởng. Tuy nhiên, giá trị 0.7455 SSIM cũng là một giá trị tương đối ổn, điều này cho thấy thuật toán Fast Non-Local Means có khả năng xử lý nhiễu Salt and Pepper với mức độ hiệu quả chấp nhận được, tuy không hiệu quả bằng như với nhiễu Gaussian hay nhiễu Speckle.

Tổng kết lại các kết quả thí nghiệm, ta có được bảng các thông số đo như sau:

	Nhiễu Gaussian	Nhiễu Salt and Pepper	Nhiễu Speckle
Tổng thời gian chạy	27.9048s	29.0964s	27.5015s
Thời gian chạy trung bình	1.3952s	1.4548s	1.3751s
PSNR trung bình	32.97 dB	34.38 dB	33.26 dB
SSIM trung bình	0.8597	0.7455	0.8581

Bảng 4: Tổng kết các giá trị thí nghiệm

6 Kết luận

Sau quá trình tìm hiểu và thực hiện thí nghiệm, ta có thể kết luận rằng thuật toán Fast Non - Local Means có thể thực hiện xử lý nhiễu tốt đối với các loại nhiễu như Gaussian và nhiễu Speckle và có khả năng xử lý nhiễu Salt and Pepper ở mức chấp nhận được. Tuy nhiên, thuật toán này cũng chỉ hoạt động hiệu quả đối với các loại nhiễu nhẹ và trung bình, nó có thể loại nhiễu được các loại nhiễu nặng, tuy nhiên, ta phải đánh đổi đi sự mất chi tiết ảnh nhiều hơn và thời gian xử lý lâu hơn. Dánh giá tổng quát, đối với mật độ nhiễu lớn, thuật toán này không đạt hiệu quả tốt như mật độ nhiễu vừa và nhẹ nếu như phải cân bằng giữa hai yếu tố là mức độ giảm nhiễu và sự mất mát chi tiết ảnh.

So sánh với các thuật toán cổ điển khác, phương pháp Fast Non - Local Means hoạt động hiệu quả hơn ở khía cạnh có thể xử lý nhiều loại nhiễu khác nhau, không phải chỉ xử lý các loại nhiễu đặc trưng với mỗi phương pháp. Tuy nhiên, xét ở khía cạnh độ phức tạp thuật toán và thời gian xử lý, ta có thể nhận thấy hướng tiếp cận này đòi hỏi nhiều thời gian xử lý hơn và có sự phức tạp về mặt tính toán hơn.

So sánh với các phương pháp tiếp cận theo hướng học máy, phương pháp Fast



Average SSIM: 0.8597

	Original Image	Denoised Image	SSIM
0	Image_a.jpg	FastNLM_GaussianNoise_Image_a.jpg	0.813988
1	Image_b.jpg	FastNLM_GaussianNoise_Image_b.jpg	0.900470
2	Image_c.jpg	FastNLM_GaussianNoise_Image_c.jpg	0.817705
3	Image_d.jpg	FastNLM_GaussianNoise_Image_d.jpg	0.870751
4	Image_e.jpg	FastNLM_GaussianNoise_Image_e.jpg	0.846728
5	Image_f.jpg	FastNLM_GaussianNoise_Image_f.jpg	0.740295
6	Image_g.jpg	FastNLM_GaussianNoise_Image_g.jpg	0.887860
7	Image_h.jpg	FastNLM_GaussianNoise_Image_h.jpg	0.920767
8	Image_i.jpg	FastNLM_GaussianNoise_Image_i.jpg	0.861948
9	Image_j.jpg	FastNLM_GaussianNoise_Image_j.jpg	0.809034
10	Image_k.jpg	FastNLM_GaussianNoise_Image_k.jpg	0.911251
11	Image_l.jpg	FastNLM_GaussianNoise_Image_l.jpg	0.894699
12	Image_m.jpg	FastNLM_GaussianNoise_Image_m.jpg	0.915359
13	Image_n.jpg	FastNLM_GaussianNoise_Image_n.jpg	0.783517
14	Image_o.jpg	FastNLM_GaussianNoise_Image_o.jpg	0.887384
15	Image_p.jpg	FastNLM_GaussianNoise_Image_p.jpg	0.805824
16	Image_q.jpg	FastNLM_GaussianNoise_Image_q.jpg	0.909024
17	Image_r.jpg	FastNLM_GaussianNoise_Image_r.jpg	0.842392
18	Image_s.jpg	FastNLM_GaussianNoise_Image_s.jpg	0.857857
19	Image_t.jpg	FastNLM_GaussianNoise_Image_t.jpg	0.916455

Hình 28: Kết quả SSIM trên ảnh khử nhiễu Gaussian

Non - Local Means sẽ có hiệu quả lọc các loại nhiễu khác nhau tương đối không tốt bằng và giữa các loại nhiễu cũng có mức độ hiệu quả không đồng đều. Tuy nhiên, độ phức tạp hay mức độ tiêu tốn tài nguyên thì hướng tiếp cận này lại tối ưu hơn



Average SSIM: 0.7455

	Original Image	Denoised Image	SSIM
0	Image_a.jpg	FastNLM_Salt_and_Pepper_Image_a.jpg	0.756936
1	Image_b.jpg	FastNLM_Salt_and_Pepper_Image_b.jpg	0.713063
2	Image_c.jpg	FastNLM_Salt_and_Pepper_Image_c.jpg	0.707147
3	Image_d.jpg	FastNLM_Salt_and_Pepper_Image_d.jpg	0.750408
4	Image_e.jpg	FastNLM_Salt_and_Pepper_Image_e.jpg	0.696146
5	Image_f.jpg	FastNLM_Salt_and_Pepper_Image_f.jpg	0.674516
6	Image_g.jpg	FastNLM_Salt_and_Pepper_Image_g.jpg	0.799935
7	Image_h.jpg	FastNLM_Salt_and_Pepper_Image_h.jpg	0.752807
8	Image_i.jpg	FastNLM_Salt_and_Pepper_Image_i.jpg	0.701817
9	Image_j.jpg	FastNLM_Salt_and_Pepper_Image_j.jpg	0.717810
10	Image_k.jpg	FastNLM_Salt_and_Pepper_Image_k.jpg	0.809550
11	Image_l.jpg	FastNLM_Salt_and_Pepper_Image_l.jpg	0.771203
12	Image_m.jpg	FastNLM_Salt_and_Pepper_Image_m.jpg	0.747450
13	Image_n.jpg	FastNLM_Salt_and_Pepper_Image_n.jpg	0.700373
14	Image_o.jpg	FastNLM_Salt_and_Pepper_Image_o.jpg	0.767514
15	Image_p.jpg	FastNLM_Salt_and_Pepper_Image_p.jpg	0.711682
16	Image_q.jpg	FastNLM_Salt_and_Pepper_Image_q.jpg	0.750295
17	Image_r.jpg	FastNLM_Salt_and_Pepper_Image_r.jpg	0.738642
18	Image_s.jpg	FastNLM_Salt_and_Pepper_Image_s.jpg	0.792951
19	Image_t.jpg	FastNLM_Salt_and_Pepper_Image_t.jpg	0.850086

Hình 29: Kết quả SSIM trên ảnh khử nhiễu Salt and Pepper

và nhanh hơn.

Kết luận lại, phương pháp Fast Non - Local Means phù hợp cho các bài toán đòi hỏi phải xử lý nhiều loại nhiễu khác nhau nhưng có nguồn tài nguyên giới hạn và



Average SSIM: 0.8581

	Original Image	Denoised Image	SSIM
0	Image_a.jpg	FastNLM_SpeckeNoise_Image_a.jpg	0.798858
1	Image_b.jpg	FastNLM_SpeckeNoise_Image_b.jpg	0.911383
2	Image_c.jpg	FastNLM_SpeckeNoise_Image_c.jpg	0.812446
3	Image_d.jpg	FastNLM_SpeckeNoise_Image_d.jpg	0.869760
4	Image_e.jpg	FastNLM_SpeckeNoise_Image_e.jpg	0.841544
5	Image_f.jpg	FastNLM_SpeckeNoise_Image_f.jpg	0.734099
6	Image_g.jpg	FastNLM_SpeckeNoise_Image_g.jpg	0.876539
7	Image_h.jpg	FastNLM_SpeckeNoise_Image_h.jpg	0.920767
8	Image_i.jpg	FastNLM_SpeckeNoise_Image_i.jpg	0.858819
9	Image_j.jpg	FastNLM_SpeckeNoise_Image_j.jpg	0.803664
10	Image_k.jpg	FastNLM_SpeckeNoise_Image_k.jpg	0.921798
11	Image_l.jpg	FastNLM_SpeckeNoise_Image_l.jpg	0.891936
12	Image_m.jpg	FastNLM_SpeckeNoise_Image_m.jpg	0.912091
13	Image_n.jpg	FastNLM_SpeckeNoise_Image_n.jpg	0.793572
14	Image_o.jpg	FastNLM_SpeckeNoise_Image_o.jpg	0.881704
15	Image_p.jpg	FastNLM_SpeckeNoise_Image_p.jpg	0.797617
16	Image_q.jpg	FastNLM_SpeckeNoise_Image_q.jpg	0.915052
17	Image_r.jpg	FastNLM_SpeckeNoise_Image_r.jpg	0.833432
18	Image_s.jpg	FastNLM_SpeckeNoise_Image_s.jpg	0.855472
19	Image_t.jpg	FastNLM_SpeckeNoise_Image_t.jpg	0.931656

Hình 30: Kết quả SSIM trên ảnh khử nhiễu Speckle

yêu cầu về thời gian xây dựng, xử lý không quá lâu. Áp dụng phương pháp này sẽ phải cân bằng sự đánh đổi nhất định giữa hiệu quả thuật toán và độ phức tạp, thời



gian xử lý, tùy vào từng hoàn cảnh, bài toán áp dụng khác nhau mà phương pháp này có thể được cấu hình để đáp ứng các yêu cầu cụ thể.

7 Mở rộng

Dể giải quyết bài toán xử lý nhiều loại nhiễu khác nhau và hạn chế độ phức tạp, tiêu tốn tài nguyên, chúng ta có thể tiếp cận theo hai hướng sau. Hướng thứ nhất là chúng ta vẫn tiếp tục khai thác các thuật toán cổ điển nhằm đảm bảo về mặt đơn giản tính toán và ít tiêu tốn tài nguyên nhưng sẽ kết hợp nhiều thuật toán lại với nhau thay vì chỉ sử dụng một loại để xử lý. Ví dụ ở thuật toán Fast Non - Local Means, phương pháp này vẫn chưa hoạt động tối ưu đối với loại nhiễu Salt and Pepper, chúng ta có thể thêm các bước tiền xử lý như áp dụng bộ lọc trung vị để xử lý trước. Điều này sẽ giúp tăng độ hiệu quả của giải pháp nhưng không tăng quá nhiều về độ phức tạp xử lý. Hướng tiếp cận thứ hai ta có thể phát triển là tối ưu hóa các thuật toán học máy nhằm giảm độ phức tạp của chúng nhưng vẫn mang lại hiệu quả cao trong lọc nhiễu, tuy nhiên, phương pháp này đòi hỏi nhiều công sức nghiên cứu và phát triển.

8 Mã Python cho thí nghiệm

Mã Python của thí nghiệm: [Computer Vision Assignment Github Repository](#).

9 Dữ liệu hình ảnh cho thí nghiệm

Các hình ảnh được dùng cho thí nghiệm được lấy tại nguồn sau: [Kaggle Data](#).



Tài liệu tham khảo

- A. Buades, J.-M. M., B. Coll. (2005). A non-local algorithm for image denoising. *IEEE Transactions on Image Processing*. <https://doi.org/10.1109/CVPR.2005.38>
- Fiveable. (2025). Noise reduction techniques. <https://library.fiveable.me/computer-vision-and-image-processing/unit-10/noise-reduction-techniques/study-guide/YbweeSU8WR0OPHQV>
- GeeksforGeeks. (2025). What are the different image denoising techniques in computer vision? <https://www.geeksforgeeks.org/what-are-the-different-image-denoising-techniques-in-computer-vision/>
- Venkateswarlu Karnati, S. D., Mithun Uliyar. (2009). Fast non-local algorithm for image denoising. *IEEE Transactions on Image Processing*. <https://doi.org/10.1109/ICIP.2009.5414044>