

**Tài liệu hướng dẫn học
matlab dành cho môn xử lý
ảnh rất hay**

CHƯƠNG I:

TÌM HIỂU CƠ BẢN VỀ MATLAB VÀ GUI

I) Cơ bản về Matlab:

1) Giới thiệu chung về Matlab:

Matlab là một công cụ tính toán trong kỹ thuật, đặc biệt là các bài toán về ma trận. Matlab còn cung cấp các toolboxes chuyên dụng để giải quyết những vấn đề cụ thể như xử lý ảnh, xử lý số tín hiệu, neuron, mô phỏng...

Matlab cung cấp Image Processing toolbox, chuyên về xử lý ảnh. Có thể nói Matlab là một công cụ lợi hại giúp cho việc thực hiện các giải thuật xử lý ảnh nhanh chóng và dễ hiểu.

2) Khởi động Matlab:

2.1) Mở chương trình:

- Click vào biểu tượng  để mở chương trình.

2.2) Nhập lệnh trong Matlab:

- Cửa sổ Command line hiện ra, đây là nơi chúng ta sẽ nhập lệnh và Matlab đưa ra kết quả.

- Lệnh sẽ thực hiện ngay và thể hiện kết quả thực thi trên màn hình.

Vd: `>> 5+3`

`ans =`

`8`

- Trong nhiều trường hợp ta không muốn thể hiện kết quả thì sau khi gõ lệnh phải thêm dấu “;”.

2.3) Sử dụng công cụ giúp đỡ:

- Đánh lệnh `help ten_lenh` để biết được cách sử dụng và công dụng của `ten_lenh`.

Ví dụ: `help convert`

- Nếu không biết chính xác tên lệnh là gì, ta có thể dùng lệnh `lookfor`.

- Ví dụ: `lookfor convert` sẽ cho ta danh sách các lệnh có từ `convert` trong phần trợ giúp.

3) Phép toán với vector và ma trận:

3.1) Các toán tử:

-Trong Matlab, không cần khai báo biến. Matlab phân biệt biến chữ hoa và thường
Ví dụ: A và a là hai biến khác nhau

-Các phép toán số học: +,-,*./,\(chia đảo), ^.

-Các toán tử quan hệ :< , <= , > , >= , == , ~=

-Các toán tử logic : & , | (or) , ~ (not)

-Các hằng : pi 3.14159265

i số ảo

j tương tự i

eps sai số 2-52

inf vô cùng lớn

NaN Not a number

3.2) Làm việc với vector và ma trận:

-Trong Matlab, tất cả các đối tượng đều xem là ma trận. Một chữ số là một ma trận 1×1 .
Một vector là ma trận một hàng hay một cột.

Ví dụ: >> a=[5 10 2;10 2 4; 2 4 5]

a =

5 10 2

10 2 4

2 4 5

-Chỉ số : Phần tử ở hàng i cột j của ma trận có kí hiệu là $A(i,j)$. Tuy nhiên ta cũng có thể tham chiếu tới phần tử của mảng nhờ một chỉ số $A(k)$. Ví dụ: $A(6)$ là tham chiếu của $A(3,2)$.

-Toán tử “:” là một toán tử quan trọng, xuất hiện ở nhiều dạng khác nhau

Ví dụ: >> 5:10

ans =

5 6 7 8 9 10

>> 1:2:10

ans=

1 3 5 7 9

- $A(:,j)$ để trích ra cột thứ j của A
- $A(i,:)$ để trích ra hàng thứ i
- $A(k:l,m:n)$ trích ra ma trận con của A
- $V(i:j)$ trích ra một phần vector V

Ví dụ: >> A=[2 4 6; 1 3 5; 3 1 4];

>> A(3,:)

ans =

3 1 4

-Chuyển vị: Dùng dấu ‘ để tạo ma trận chuyển vị

Ví dụ: >>a=[1 3; 2 4]

a =

1 3

2 4

>> a‘

ans =

1 2

3 4

-Phép toán số học với ma trận:

- Cộng, trừ ma trận:

>> a=[2 3];

>> b=[1 2];

>> a+b

ans =

3 5

- Nhân hai ma trận:

>> a*b'

ans =

8

>> a.*b

ans =

2 6

- Chia các thành phần của ma trận này cho một ma trận khác:

>> a./b

ans =

2.0000 1.5000

- Lũy thừa của ma trận:

>> a.^2

ans =

4 9

>> c=[1 2; 3 4];

>> c^2

ans =

7 10

15 22

-Ma trận đặc biệt:

- zeros(m,n): ma trận toàn 0

- eye(n):ma trận đơn vị

- ones(m,n): Ma trận toàn 1

4) Lập trình trong matlab:

4.1) Biểu thức điều kiện: Gần giống trong C

-If, else , elseif.

-switch(chỉ thực thi duy nhất một nhóm lệnh)

4.2) Vòng lặp:

-for, while.

5) Hàm m-file:

- Hàm m-file là một chương trình con do chúng ta yêu cầu các đối số ngõ vào và có thể trả về đối số ngõ ra
- Cú pháp:

```
function[outputArgs]=function_name(inputArgs)
```

-Chú thích(đặt sau dấu %, chú thích sẽ hiện ra khi dùng lệnh help)

-Các lệnh.

-return;

Phải lưu lại với tên giống tên hàm.

Ví dụ: Vẽ hàm sau trong khoảng [-10,10]

Giải:

```
function f = function1(x)
```

```
y = 1./((x-0.3).^2+0.01)+1./((x-0.9).^2+0.04)-6 ;
```

Lưu lại với tên function1.m

```
>>fplot('function1',[-10,10]);
```

6) Vẽ hình trong Matlab:

-Matlab cung cấp nhiều hàm để biểu diễn đồ thị 2D và 3D.

- plot: vẽ đồ thị 2D
- plot3: vẽ đồ thị 3D

- loglog: vẽ đồ thị các trục là logarit
- semilogx, semilogy: vẽ đồ thị với 1 trục là logarit

-Sử dụng hàm **figure** để tạo nhiều cửa sổ hình vẽ

-Sử dụng lệnh **subplot** để vẽ nhiều hình trên một cửa sổ

-Hàm chú thích hình vẽ:

- title: Nhãn hình vẽ
- xlabel, ylabel,zlabel: nhãn các trục.
- legend: thêm chú thích vào hình vẽ

Ví dụ:

```
x = -pi:.1:pi;
y = sin(x);
>>plot(x,y)
xlabel('t = 0 to 2\pi','FontSize',16)
ylabel('sin(t)','FontSize',16)
title('Giá trị của sin từ zero đến 2\pi','FontSize',16)
```

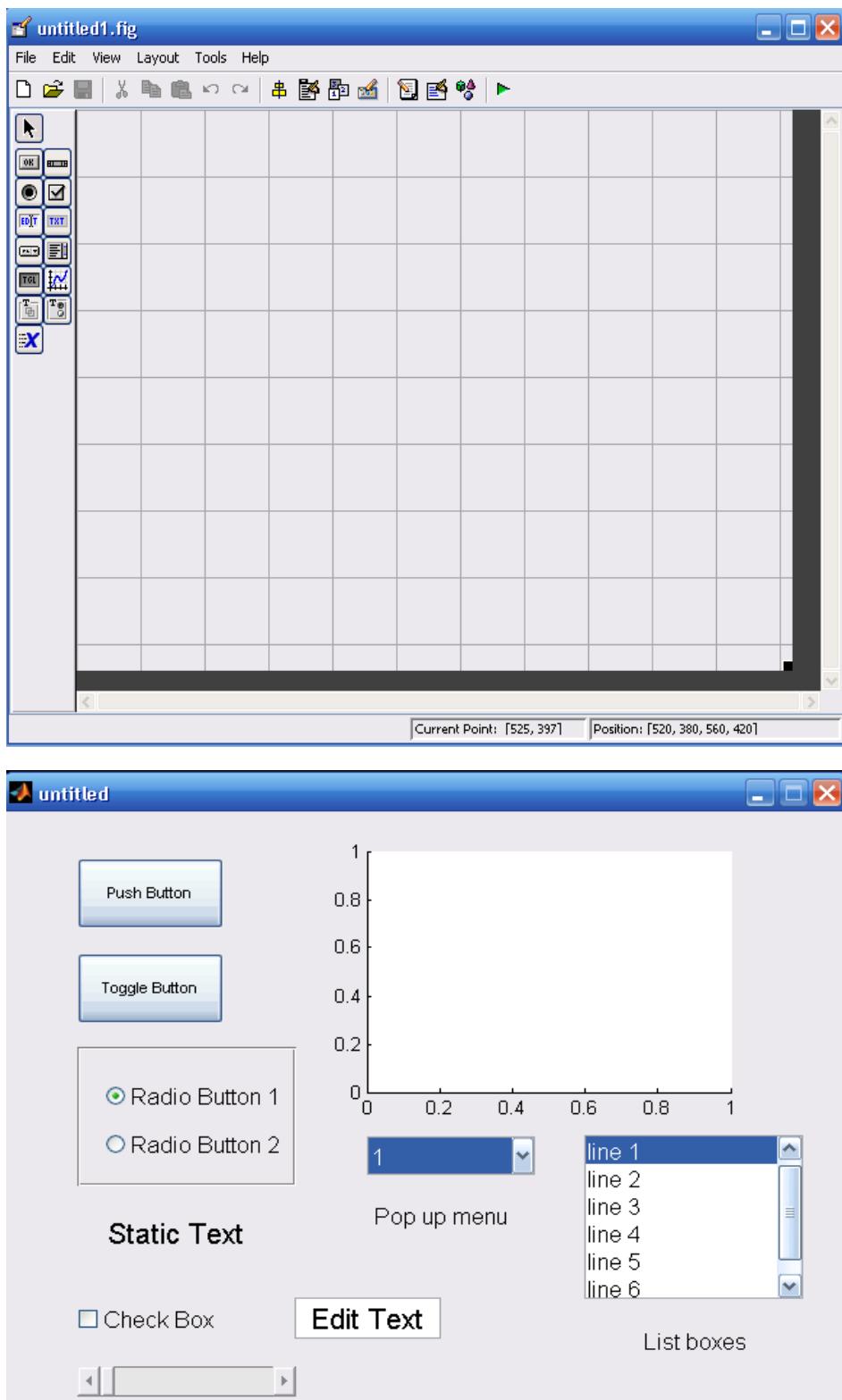
II) Matlab GUI:

1) Giới thiệu:

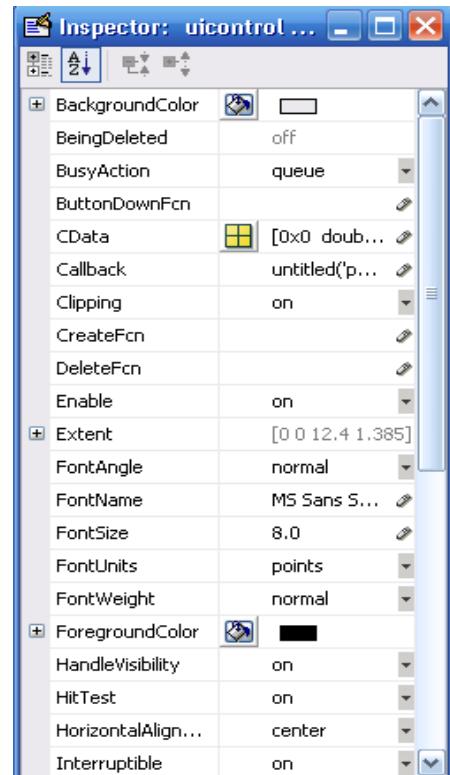
- Gui-là giao diện bằng hình ảnh của chương trình
- Gui-bao gồm các nút nhấn, hộp liệt kê, thanh trượt, menu... chúng cung cấp cho người dùng sử dụng một môi trường làm việc thân thiện để họ tập trung vào các ứng dụng của chương trình hơn là đi tìm hiểu cách thức làm việc của chương trình đó.



- Để mở công cụ tạo Gui : File → New → GUI
- Khi lưu giao diện vừa tạo, Matlab sẽ tạo ra hai file có cùng tên nhưng khác phần mở rộng:
 - File có phần mở rộng .fig chứa nội dung của giao diện
 - File có phần mở rộng .m chứa những đoạn mã liên quan đến giao diện



- Khi thiết kế bất cứ thành phần nào của Gui ta cần thiết lập thuộc tính cho thành phần đó.
- Để thiết lập các thuộc tính ta có thể chọn mục “Property Inspector” trên thanh công cụ hoặc right-click vào đối tượng và chọn mục “Inspector Properties”
- Hai thuộc tính quan trọng mà ta cần xác lập là “String Property” và “Tag Property”.
 - String property : dòng ký tự xuất hiện trên đối tượng.
 - Tag property : tên của đối tượng.
- Khi click chuột vào 1 đối tượng, Matlab sẽ gọi hàm tương ứng với đối tượng đó. Tên của hàm chính là tên của đối tượng cộng với “_Callback”



2) Các hàm thường được sử dụng trong Gui:

- Set : Thay đổi giá trị của các thuộc tính của một đối tượng giao diện
set(handles.TextBox,'String',str)
- Get : Truy xuất giá trị của thuộc tính của một đối tượng giao diện
get(handles.EditBox,'String')
- Ngoài ra còn có các hàm như axes, guide, num2str(), str2num()...

CHƯƠNG II:

CƠ BẢN VỀ ẢNH VÀ CÁC HÀM

XỬ LÝ ẢNH CƠ BẢN TRONG MATLAB

I) Các kiểu ảnh trong Matlab:

1) Ảnh Index:

Ảnh được biểu diễn bởi hai ma trận, một ma trận dữ liệu ảnh X và một ma trận màu (còn gọi là bản đồ màu) map. Ma trận dữ liệu có thể thuộc kiểu uint8, uint16 hoặc double. Ma trận màu là một ma trận kích thước $m \times 3$ gồm các thành phần thuộc kiểu double có giá trị trong khoảng [0 1]. Mỗi hàng của ma trận xác định thành phần red, green, blue của một màu trong tổng số m màu được sử dụng trong ảnh. Giá trị của một phần tử trong ma trận dữ liệu ảnh cho biết màu của điểm ảnh đó nằm ở hàng nào trong ma trận màu.

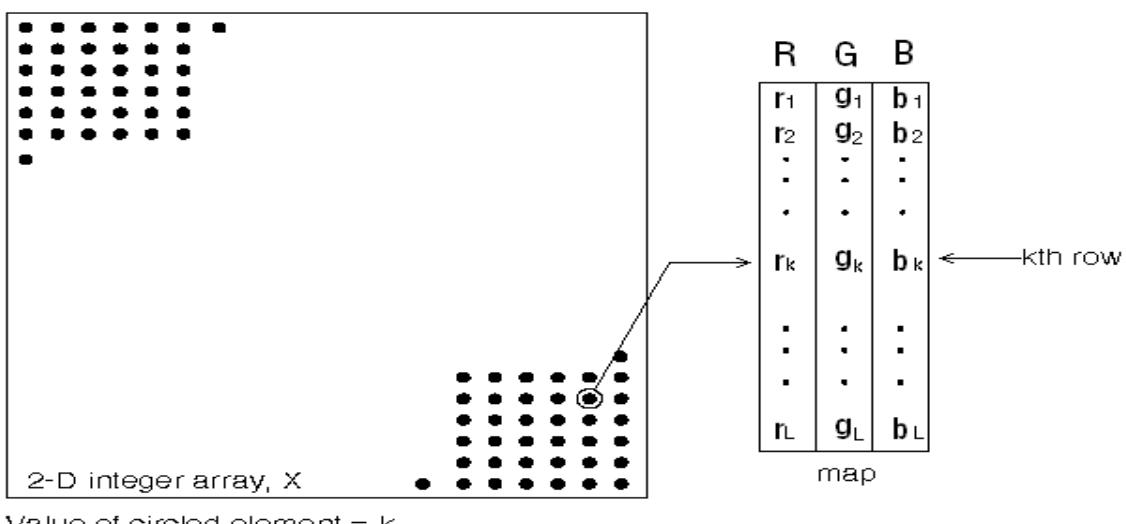




Figure 1.19: An indexed colour image

2) Ảnh grayscale:

Mỗi ảnh được biểu diễn bởi một ma trận hai chiều, trong đó giá trị của mỗi phần tử cho biết độ sáng (hay mức xám) của điểm ảnh đó. Ma trận này có thể một trong các kiểu uint8, uint16 hoặc double. Ảnh biểu diễn theo kiểu này còn gọi là ảnh ‘trắng đen’.

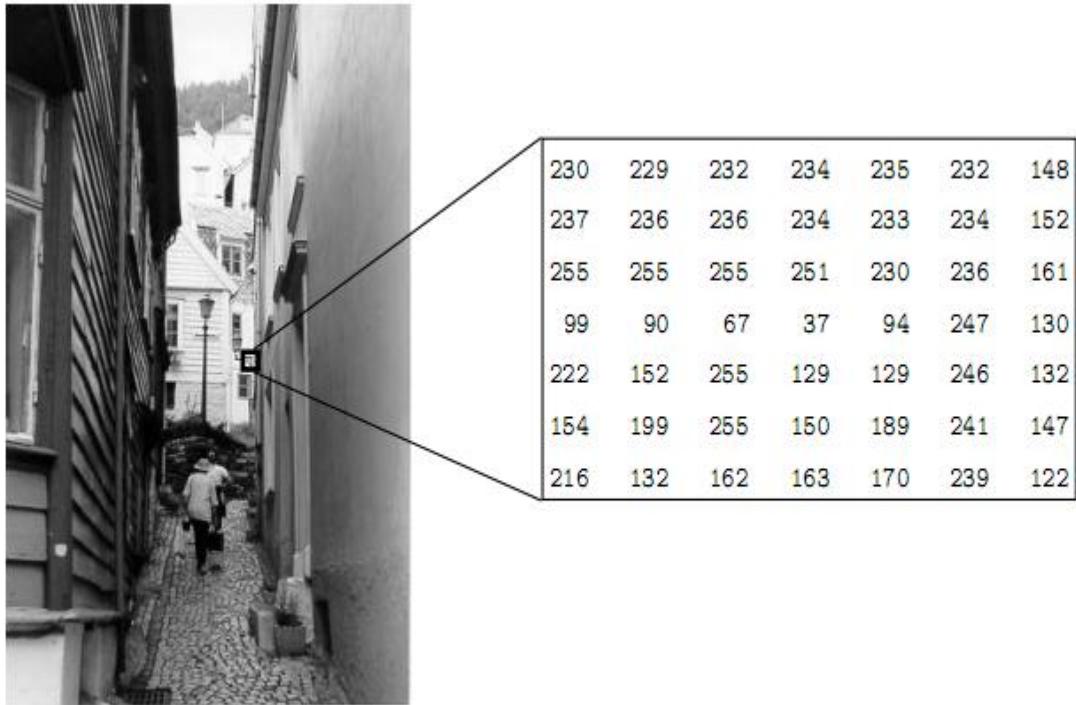


Figure 1.17: A greyscale image

3) Ảnh nhị phân:

Ảnh được biểu diễn bởi một ma trận hai chiều thuộc kiểu logical. Mỗi điểm ảnh chỉ có thể nhận một trong hai giá trị là 0 (đen) hoặc 1 (trắng)

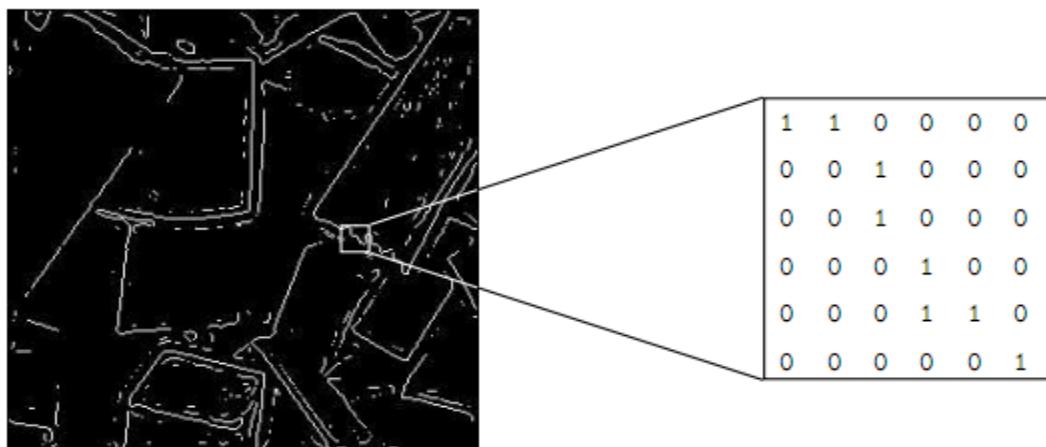
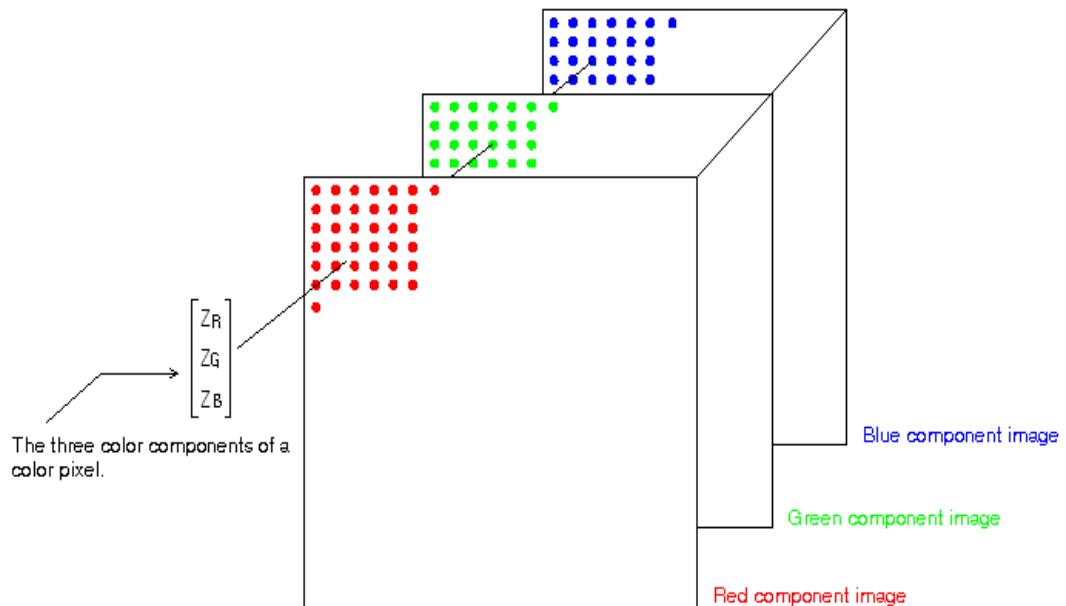
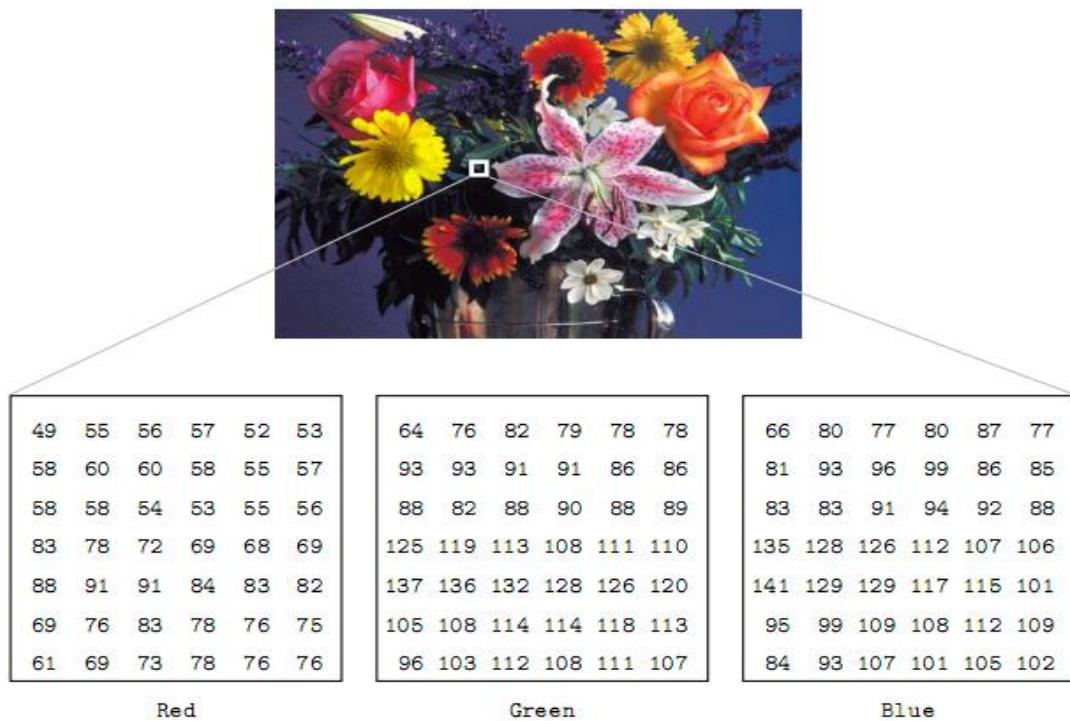


Figure 1.16: A binary image

4) Ảnh RGB:

Còn gọi là ảnh “truecolor” do tính trung thực của nó. Ảnh này được biểu diễn bởi một ma trận ba chiều kích thước $m \times n \times 3$, với $m \times n$ là kích thước ảnh theo pixels. Ma trận này định nghĩa các thành phần màu red, green, blue cho mỗi điểm ảnh, các phần tử của nó có thể thuộc kiểu uint8, uint16 hoặc double.





II) Các phép biến đổi ảnh:

1) Biến đổi Fourier:

Phép biến đổi Fourier biểu diễn ảnh dưới dạng tổng của các lũy thừa phức của các thành phần biến độ, tần số, pha khác nhau của ảnh.

Nếu $f(m,n)$ là một hàm của hai biến không gian rời rạc m và n , thì biến đổi Fourier hai chiều của $f(m,n)$ được định nghĩa :

$$F(w_1, w_2) = \sum_{m=-\infty}^{+\infty} \sum_{n=-\infty}^{+\infty} f(m, n) e^{-j m w_1} e^{-j n w_2}$$

Nếu $f(m,n)$ biểu diễn độ sáng của ảnh X ở vị trí pixel (m,n) thì $F(w_1, w_2)$ chính là biến đổi Fourier của ảnh X.

Do các dữ liệu trên máy tính được lưu trữ dưới dạng rời rạc, cụ thể là dữ liệu ảnh được tổ chức theo đơn vị pixel nên phép biến đổi Fourier cũng được rời rạc hóa thành biến đổi Fourier rời rạc (DFT). Giả sử hàm $f(m,n)$ chỉ khác 0 trong miền $(0 \leq m \leq M-1, 0 \leq n \leq N-1)$, các phép biến đổi DFT thuận và nghịch kích thước $M \times N$ được định nghĩa như sau :

$$F(p, q) = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m, n) e^{-j(2\pi/M)pm} e^{-j(2\pi/N)qn} \quad (0 \leq p \leq M-1, 0 \leq q \leq N-1)$$

$$f(m,n) = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} F(p,q) e^{j(2\pi/M)pm} e^{j(2\pi/N)qn} \quad (0 \leq m \leq M-1, 0 \leq n \leq N-1)$$

$F(p,q)$ gọi là các hệ số của biến đổi DFT.

Với các ứng dụng trong xử lý ảnh, chúng ta chỉ quan tâm đến các hàm **fft2** và **ifft2**.

Cú pháp : $F = \text{fft2}(X, \text{Mrows}, \text{Ncols})$
 $f = \text{ifft2}(F, \text{mrows}, \text{ncols})$

Nếu ảnh ban đầu có kích thước nhỏ hơn thì Matlab tự động thêm vào các zero pixel trước khi biến đổi.

Sau khi thực hiện biến đổi DFT bằng **fft2**, thành phần DC sẽ nằm ở góc trên bên trái của ảnh. Ta có thể dùng hàm **fftshift** để dịch thành phần DC này về trung tâm của ảnh.

2) Phép biến đổi DCT:

Biến đổi DCT (Discrete Cosine Transform) biểu diễn ảnh dưới dạng tổng của các cosine của các thành phần biên độ và tần số khác nhau của ảnh. Hầu hết các thông tin về ảnh chỉ tập trung trong một vài hệ số của biến đổi DCT, trong khi các hệ số còn lại chứa rất ít thông tin. Biến đổi DCT 2 chiều của một ma trận A kích thước $M \times N$ là:

$$B_{pq} = \alpha_p \alpha_q \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} A_m \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N} \quad \text{với } 0 \leq p \leq M-1 \\ 0 \leq q \leq N-1$$

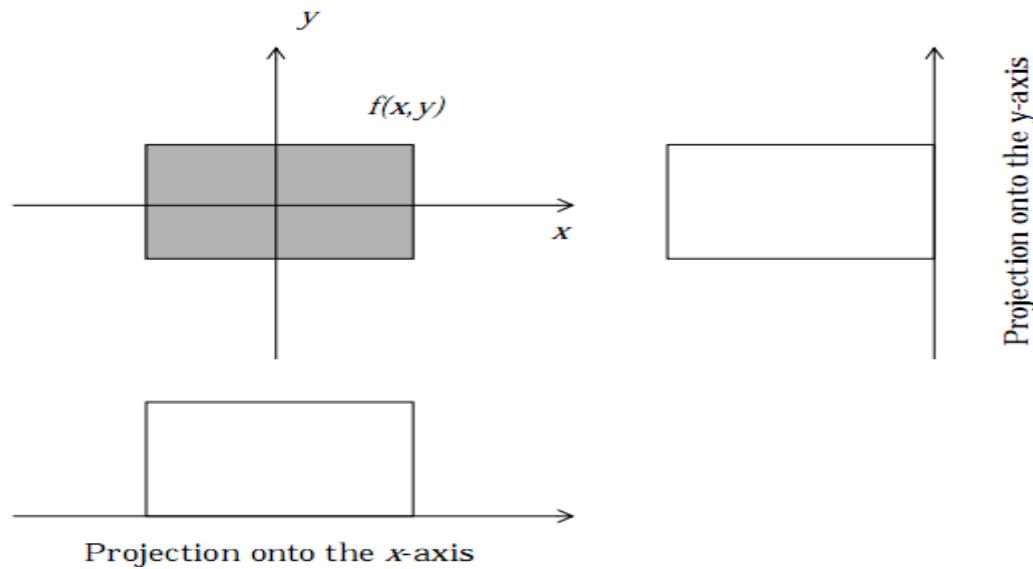
$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N} \quad \text{với } 0 \leq m \leq M-1 \\ 0 \leq n \leq N-1$$

Phép biến đổi DCT thuận và nghịch được thực hiện bằng các hàm **dtc2** và **idtc2**. Các hàm này sử dụng giải thuật dựa theo FFT để tăng tốc độ tính toán.

Cú pháp : $B = \text{dtc2}(A, M, N)$
 $A = \text{idtc2}(B, M, N)$

3) Biến đổi Radon:

Phép biến đổi Radon được thực hiện bởi hàm **radon** trong Matlab, biểu diễn ảnh dưới dạng các hình chiêu của nó dọc theo các hướng xác định. Hình chiêu của một hàm hai biến $f(x,y)$ là tập hợp các tích phân đường. Hàm **radon** tính các tích phân đường từ nhiều điểm nguồn dọc theo các đường dẫn song song, gọi là các tia chiêu, theo một hướng xác định nào đó. Các tia chiêu này nằm cách nhau 1 pixel. Để biểu diễn toàn bộ ảnh, hàm **radon** sẽ lấy nhiều hình chiêu song song của ảnh từ các góc quay khác nhau bằng cách xoay các điểm nguồn quanh trung tâm của ảnh.

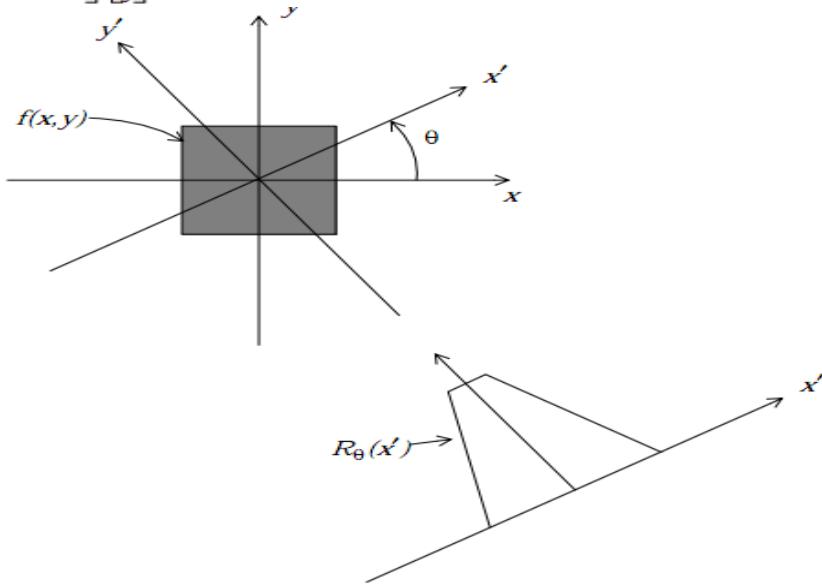


Biến đổi **Radon** của $f(x,y)$ tương ứng với góc quay θ là tích phân đường của f dọc theo trục y' :

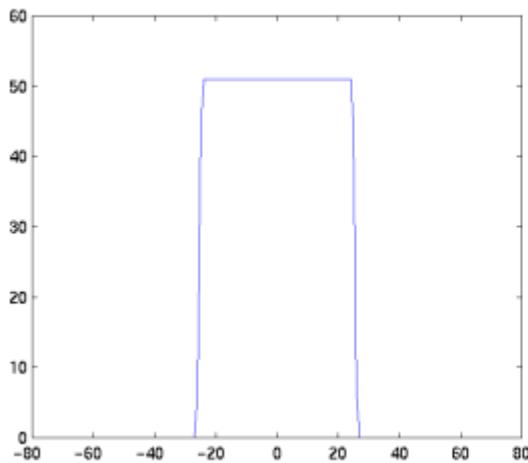
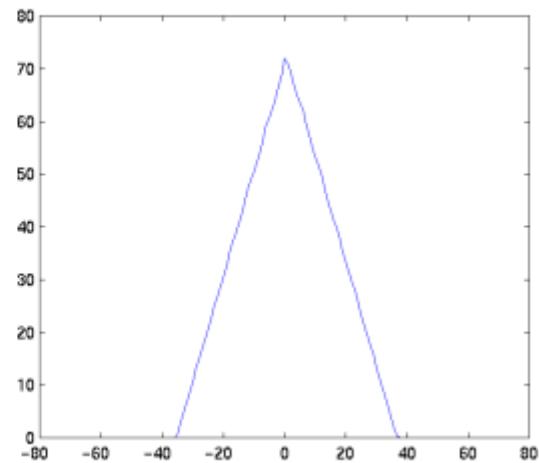
$$R_\theta(x') = \int_{-\infty}^{\infty} f(x' \cos \theta - y' \sin \theta, x' \sin \theta + y' \cos \theta) dy'$$

where

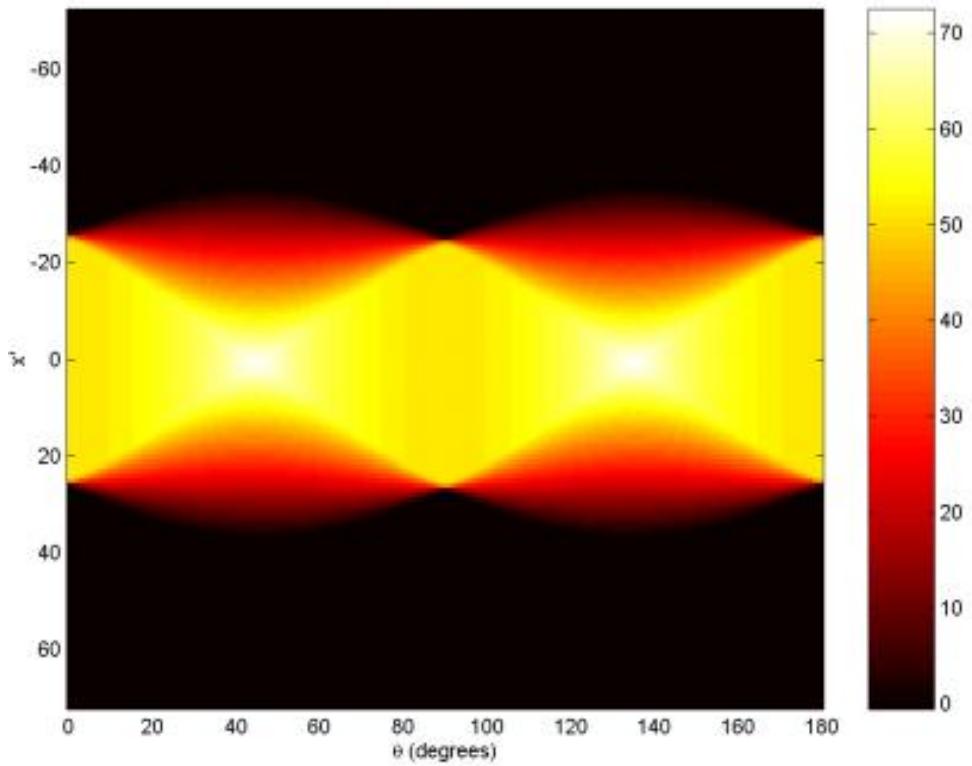
$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$



Trong Matlab, biến đổi **Radon** được tính bằng hàm **Radon** có cú pháp như sau:
 $[R, Xp] = \text{radon}(I, \theta)$

 $R_{0^\circ}(x')$  $R_{45^\circ}(x')$ 

Phép biến đổi **Radon** với nhiều góc thường được hiển thị dưới dạng ảnh.

 $R_\theta(x')$ 

III) Các hàm xử lý cơ bản:

1) Đọc và ghi dữ liệu ảnh:

✓ Hàm **imread** đọc các file ảnh với bất kỳ các định dạng ảnh đã biết hiện nay và lưu lại dưới dạng một ma trận biểu diễn ảnh trong Matlab. Cú pháp : **A=imread(filename,fmt)**

✓ Hàm **imwrite** cho phép lưu một ảnh biểu diễn bằng một ma trận trong Matlab thành một file ảnh dưới một trong các định dạng đã biết. Cú pháp : **imwrite(A,filename,fmt)**

✓ Hàm **imfinfo** dùng để xem các thông số của một file ảnh nào đó. Cú pháp : **imfinfo(filename,fmt)**

Các thông tin được cung cấp bởi hàm **imfinfo** là : filename, filemoddate, filesize, format, formatversion, width, height, bitdepth, colortype.

2) Chuyển đổi giữa các kiểu dữ liệu, kiểu ảnh:

2.1) Chuyển đổi giữa các kiểu dữ liệu ảnh:

Matlab cung cấp sẵn các hàm thực hiện chuyển kiểu cho các ma trận biểu diễn ảnh, bao gồm : **im2double**, **im2uint8** và **im2uint16**.

Tuy nhiên, khi thực hiện chuyển kiểu giữa các dữ liệu ảnh cần lưu ý một số điều sau:

- ✓ Khi chuyển từ một kiểu dữ liệu dùng nhiều bit sang một kiểu dữ liệu dùng ít bit hơn thì một số thông tin chi tiết về bức ảnh ban đầu sẽ bị mất.
- ✓ Không phải lúc nào cũng có thể chuyển đổi kiểu dữ liệu đối với kiểu ảnh indexed, vì các giá trị của ma trận ảnh xác định một địa chỉ trong bản đồ màu chứ không phải là giá trị màu, do đó không thể lượng tử hóa được.

2.2) Chuyển đổi giữa các kiểu ảnh:

- ✓ **Dither** : dither(RGB,map)
dither(I)
- ✓ **Gray2ind** : [X,Map] = gray2ind(I,N)
[X,Map] = gray2ind(BW,N)
- ✓ **Grayslice** : x=grayslice(I,N)
x=grayslice(I,V)
- ✓ **Im2bw** : bw=im2bw(I,level)
bw=im2bw(x,map,level)
bw=im2bw(rgb,level)
- ✓ **Ind2gray** : i=ind2gray(x,map)
- ✓ **Ind2rgb** : rgb=ind2rgb(x,map)
- ✓ **Mat2gray** : i=mat2gray(a,[amin amax])
- ✓ **Rgb2gray** : i=rgb2gray(rgb)
- ✓ **Rgb2ind** : [x, map]=rgb2ind(rgb,n)

```
x=rgb2ind(rgb,map)
[x,map]=rgb2ind(rgb,tol)
```

3) Các phép toán số học cơ bản đối với dữ liệu ảnh:

Các phép toán số học cơ bản trên các dữ liệu ảnh bao gồm các phép cộng, trừ, nhân và chia. Tuy nhiên, Matlab chỉ hỗ trợ các phép toán này trên kiểu double nên cần thực hiện chuyển đổi kiểu trước khi thực hiện. Để giảm bớt thao tác này, trong IPT có cung cấp các hàm thực hiện các phép toán số học trên ảnh mà có thể chấp nhận bất kỳ kiểu dữ liệu ảnh nào và trả về kết quả thuộc cùng kiểu với các toán hạng. Các hàm này cũng xử lý các dữ liệu tràn một cách tự động.

- ✓ **Imabsdiff** : $z = \text{imabsdiff}(x, y)$
- ✓ **Imadd** : $z = \text{imadd}(x, y, \text{out_class})$
- ✓ **Imcomplement** : $\text{im2} = \text{imcomplement}(\text{im})$
- ✓ **Imdivide** : $z = \text{imdivide}(x, y)$
- ✓ **Imlincomb** : $z = \text{imlincomb}(k1, a1, k2, a2, \dots, kn, an, k, \text{out_class})$
- ✓ **Immultiply** : $z = \text{immultiply}(x, y)$
- ✓ **Imsubtract** : $z = \text{imsubtract}(x, y)$

4) Các hàm hiển thị ảnh trong Matlab:

Matlab cung cấp hai hàm hiển thị cơ bản là **image** và **imagesc**. Ngoài ra trong IPT cũng có hai hàm hiển thị ảnh khác, đó là **imview** và **imshow**.

- ✓ Hàm **image(X,Y,C)** hiển thị hình ảnh biểu diễn bởi ma trận C kích thước $M \times N$ lên trực tọa độ hiện hành. X, Y là các vector xác định vị trí các pixel C(1,1) và C(M,N) trong hệ trực hiện hành.
- ✓ Hàm **imagesc** có chức năng tương tự như hàm **image**, ngoại trừ việc dữ liệu ảnh sẽ được co giãn để sử dụng toàn bộ bản đồ màu hiện hành.
- ✓ Hàm **imview** cho phép hiển thị hình ảnh trên một cửa sổ riêng, nền Java, gọi là Image Viewer. Image Viewer cung cấp các công cụ dò tìm và xác định các giá trị pixel một cách linh hoạt.
- ✓ Hàm **imshow** cũng tạo một đối tượng đồ họa thuộc loại image và hiển thị ảnh trên một figure. Hàm imshow sẽ tự động thiết lập các giá trị của các đối tượng image, axes và figure để thể hiện hình ảnh.

5) Các phép biến đổi hình học:

5.1) Phép nội suy ảnh:

Nội suy là quá trình ước lượng giá trị của một điểm nằm giữa hai pixel có giá trị đã biết. IPT cung cấp 3 phương pháp nội suy ảnh : nội suy theo các lân cận gần nhất, nội suy song tuyến tính và nội suy bicubic. Cả 3 phương pháp đều thực hiện theo nguyên tắc chung : để

xác định giá trị của một pixel ảnh nội suy, ta tìm một điểm trong ảnh ban đầu tương ứng với pixel đó, sau đó giá trị của pixel ở ảnh mới sẽ được tính bằng trung bình có trọng số của một tập các pixel nào đó ở lân cận của điểm vừa xác định, trong đó trọng số của các pixel phụ thuộc vào khoảng cách tới điểm này.

- ✓ Phương pháp lân cận gần nhất (nearest neighbor) : pixel mới sẽ được gán giá trị của pixel chưa điểm tương ứng của nó trong ảnh ban đầu
- ✓ Phương pháp song tuyến tính (bilinear interpolation) : pixel mới sẽ được gán là trung bình có trọng số của các pixel trong một lân cận kích thước 2×2 .
- ✓ Phương pháp bicubic, pixel mới sẽ được gán là trung bình có trọng số của các pixel trong một lân cận kích thước 4×4 .

5.2) Thay đổi kích thước ảnh:

Hàm **imresize** cho phép người sử dụng thay đổi kích thước ảnh. Ngoài kích thước ảnh mới, người sử dụng còn có thể xác định phương pháp nội suy sẽ dùng và loại bộ lọc dùng để chống aliasing.

Cú pháp : **b=imresize(a,m, method)** :tạo ảnh gấp m lần ảnh a.
b=imresize(a,[mrows ncols],method)
b=imresize(a,[mrows ncols],method,N)
b=imresize(a,[mrows ncols],method,h)

5.3) Phép quay ảnh:

Để thực hiện phép quay ảnh, ta có thể sử dụng hàm **imrotate**. Ngoài hai thông số cơ bản là ảnh gốc và góc quay, người sử dụng cũng có thể xác định phương pháp nội suy sẽ dùng và kích thước của ảnh mới.

Cú pháp : **b=imrotate(a,angle,method,Bbox)**

5.4) Trích xuất ảnh:

Khi cần trích xuất một phần ảnh gốc, ta dùng hàm **imcrop**.

- ✓ Xác định cụ thể vị trí của phần ảnh cần trích xuất (dưới dạng hình chữ nhật)

Cú pháp : **x2=imcrop(x,map,rect)** % indexed
x2=imcrop(a,rect) % grayscale or RGB

trong đó **rect=[Xmin Ymin width height]**

- ✓ Sử dụng mouse để chọn phần ảnh cần trích xuất.

Ta không cần cung cấp thông số rect, khi thực hiện hàm này, con trỏ sẽ chuyển sang dạng chữ thập, người dùng sẽ kéo chuột để chọn phần ảnh cần trích xuất sau đó thả chuột.

CHƯƠNG III:

NÂNG CAO CHẤT LƯỢNG ẢNH

I) Mở đầu:

Nâng cao chất lượng ảnh số là quá trình xử lý trên ảnh ban đầu để tạo ra kết quả là một bức ảnh tốt hơn xét theo một tiêu chí cụ thể. Ví dụ xử lý để nâng cao chất lượng của ảnh chụp X-quang sẽ khác với việc nâng cao chất lượng của ảnh chụp của một vệ tinh địa tĩnh.

Có nhiều phương pháp nhằm tăng cường chất lượng của ảnh, nhưng tập trung vào hai nhánh chính là xử lý ảnh trong miền không gian và xử lý ảnh trong miền tần số. Trong miền không gian, ảnh được xử lý trực tiếp trên các pixels. Miền tần số sử dụng biến đổi Fourier để xử lý.

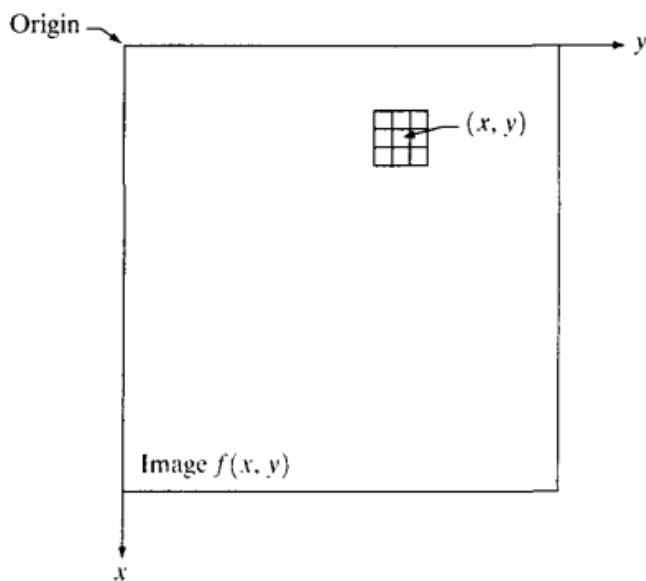
II) Xử lý ảnh trong miền không gian:

1) Giới thiệu:

Miền không gian là tập hợp các pixels trong một bức ảnh. Chúng ta sẽ tiến hành xử lý trực tiếp trên các pixels này. Quá trình xử lý này có thể được mô tả thông qua biểu thức sau:

$$g(x,y) = T[f(x,y)]$$

Với $f(x,y)$ là ảnh gốc, $g(x,y)$ là ảnh sau xử lý, và T là phép toán biến đổi, dựa trên các điểm ảnh xung quanh (x,y) .



Các điểm ảnh xung quanh có thể có các kích cỡ khác nhau, có thể là dạng vuông hoặc chữ nhật, trong đó điểm ảnh cần xử lý ở vị trí trung tâm. Trên hình là một khung có kích thước 3×3 .

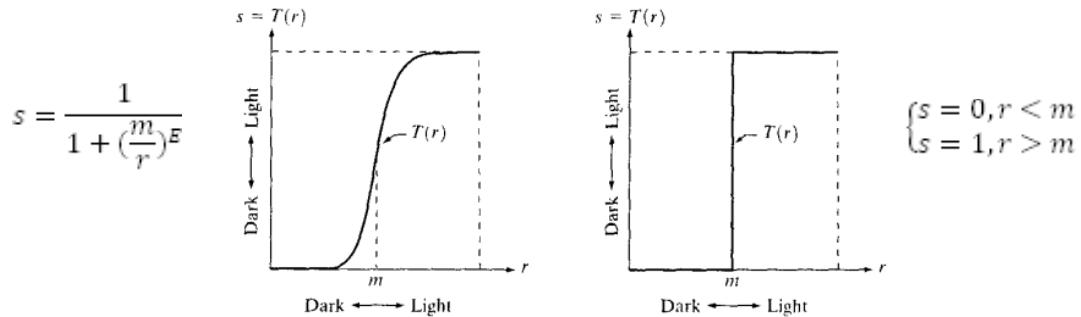
Tùy mục đích cụ thể mà ta dùng các phép biến đổi khác nhau.

2) Phép biến đổi mức xám:

Trong phép biến đổi này, giá trị $g(x,y)$ chỉ phụ thuộc vào giá trị của $f(x,y)$, và T trở thành hàm biến đổi mức xám. Ta có biểu thức đơn giản sau:

$$s = T(r)$$

Với r là mức xám ban đầu tại (x,y) , s là mức xám sau biến đổi tại (x,y) .
Ví dụ: Xét hai phép biến đổi mức xám sau:



Với hình a, phép biến đổi cho ta ảnh sau xử lý có độ tương phản cao hơn so với ảnh ban đầu. Các giá trị mức xám $r < m$ qua phép biến đổi được nén lại gần mức 0 (tối hơn), tương tự với các giá trị $r > m$ nhưng được nén lại gần mức 1 (sáng hơn) làm ảnh sau xử lý có độ tương phản cao.

Phép biến đổi ở hình b nhằm biến 1 ảnh grayscale thành 1 ảnh nhị phân. Ta xét mức ngưỡng m , với $r < m$ được xét thành mức 0, và $r > m$ xét thành mức 1.

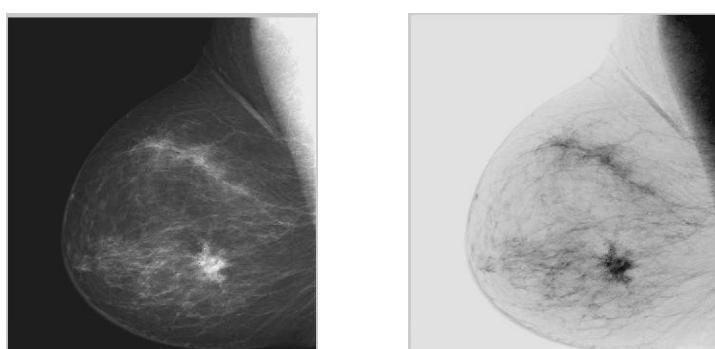
2.1) Một số phép biến đổi mức xám cơ bản:

a) Ảnh âm bản:

Với 1 ảnh có các giá trị mức xám nằm trong khoảng $[0, L-1]$, ta có:

$$s = L-1-r$$

Ta sử dụng phép biến đổi này trong trường hợp muốn làm nổi bật các chi tiết có màu sáng ở trong một vùng tối, đặc biệt với các bức ảnh có vùng tối lớn.



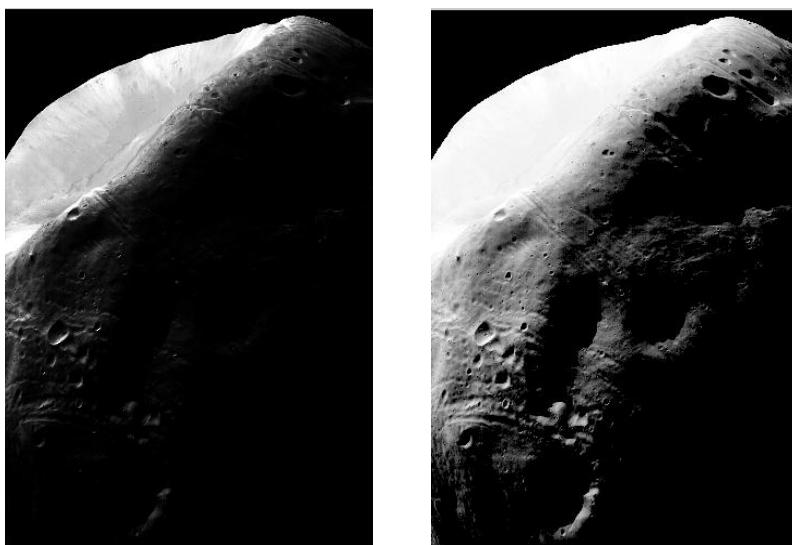
Hình trên là hình chụp một mô ngực, với ảnh bên trái là ảnh gốc và bên phải là ảnh âm bản. Ta có thể dễ dàng thấy được việc phân tích sẽ dễ dàng hơn với ảnh âm bản.

b) Phép biến đổi log:

Biểu thức:

$$s=c^*\log(1+r)$$

Các giá trị r mức thấp dải hẹp qua phép biến đổi sẽ tạo ra dải rộng hơn, trong khi đó các giá trị r mức cao sẽ nén lại thành 1 dải hẹp ở ngõ ra. Phép biến đổi này nhằm mục đích tăng chi tiết hóa ở vùng tối.

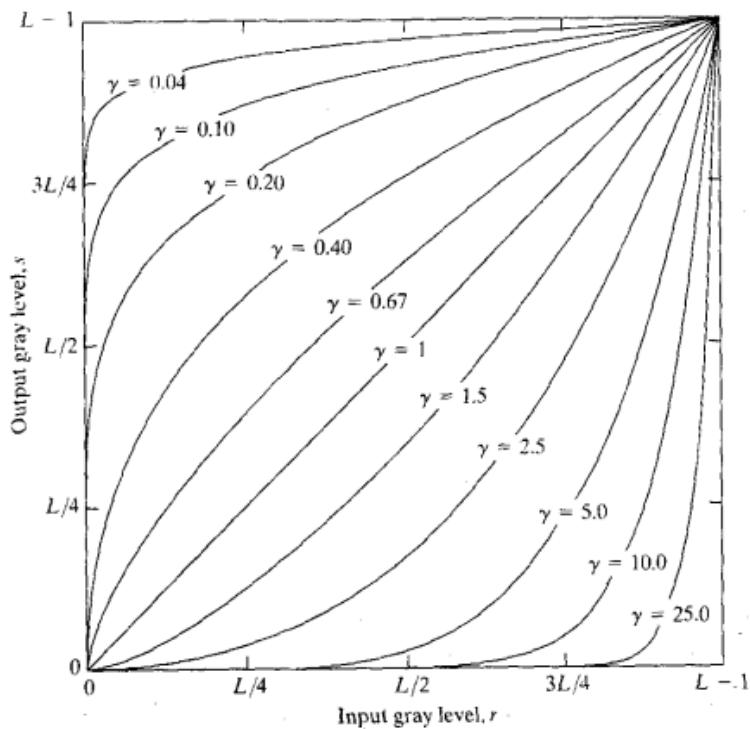


Ảnh trước và sau xử lý dụng phép biến đổi log, $c=0.8$

c) Biến đổi theo quy tắc lũy thừa:

Biểu thức:

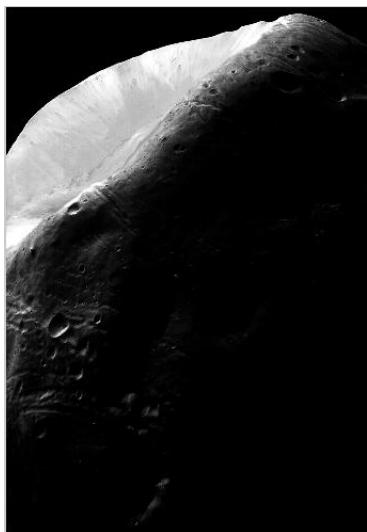
$$s = cr^\gamma$$



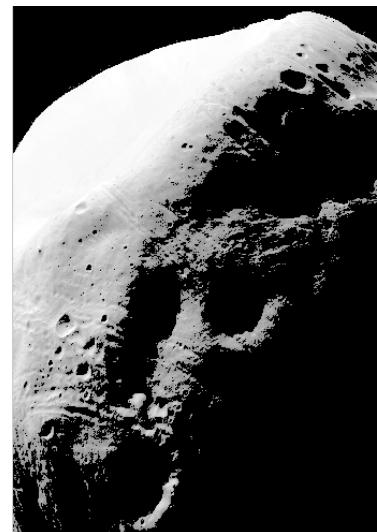
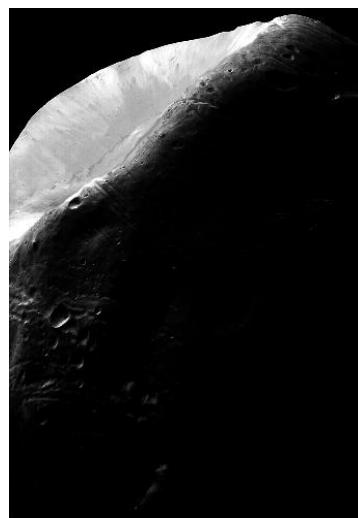
- Với $\gamma < 1$, phép biến đổi tương tự với hàm log, nhưng giá trị của γ có thể thay đổi được, trong khi hàm log là cố định. Với phép biến đổi này, các giá trị r mức thấp dải hẹp qua phép biến đổi sẽ tạo ra dải rộng hơn, trong khi đó các giá trị r mức cao sẽ nén lại thành 1 dải hẹp ở ngõ ra.

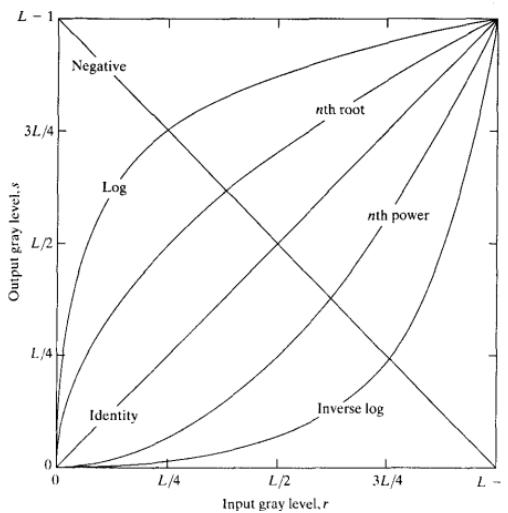
- Với $\gamma=1$, phép biến đổi là một hàm tuyến tính giữa ngõ vào và ngõ ra. Đặc biệt khi $c= \gamma=1$, ảnh ra và ảnh vào là giống nhau.

- Với $\gamma > 1$, ta có phép biến đổi ngược so với $\gamma < 1$



Ảnh gốc

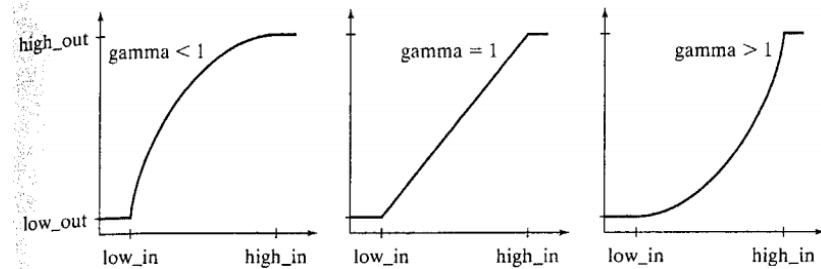
 $\gamma = 0.4$  $\gamma = 0.1$  $\gamma = 1.1$  $\gamma = 3$



Ta nhận thấy $\gamma < 1$ làm tăng độ tương phản của hình ảnh.

Trong Matlab cũng có các hàm giúp biến đổi mức xám của ảnh grayscale.

```
g=imadjust(f,[low_in high_in], [low_out high_out],gamma)
```



[low_in, high_in], [low_out, high_out] nằm trong khoảng [0,1]. Các giá trị nhỏ hơn hoặc bằng low_in sẽ được gán bằng low_out và cũng tương tự với các giá trị high_in và high_out.. Giá trị mặc định của gamma=1.

Ví dụ: `>>g=imadjust(f,[0.5 0.75],[0 1],2);`

Ta cũng có thể dùng hàm imadjust để tạo ảnh âm bản:

```
>>g=imadjust(f,[0 1],[1 0]);
```

Ngoài ra ta có thể dùng hàm imcomplement để tạo ảnh âm bản: `>>g=imcomplement(f);`

2.2) Xử lý histogram:

Histogram của 1 ảnh grayscale có L mức xám khác nhau là một hàm rời rạc, có biểu thức $h(r_k)=n_k$, trong đó r_k là giá trị mức xám thứ k trong đoạn $[0,L-1]$ và n_k là số pixels có giá trị mức xám là r_k . Ví dụ với đoạn $[0,255]$, $r_0=0$, $r_1=1\dots$

Histogram thường được chuẩn hóa. Với n là tổng số pixels của ảnh, histogram chuẩn hóa được tính qua biểu thức:

$$p(r_k) = \frac{h(r_k)}{n} = \frac{n_k}{n}$$

Ta có thể xem $p(r_k)$ là hàm mật độ xác suất của r_k , cho biết khả năng xuất hiện tương ứng của từng giá trị mức xám.

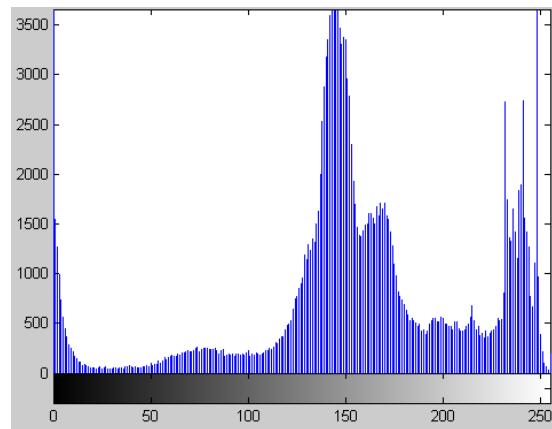
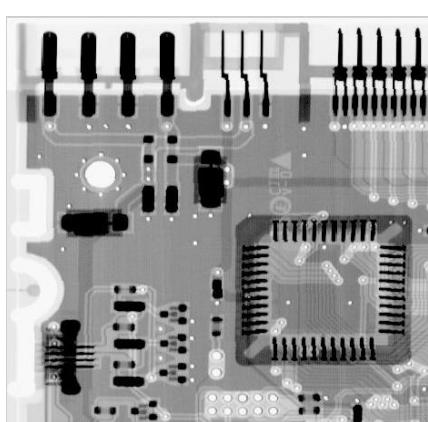
Trong Matlab, ta có thể sử dụng hàm imhist để làm việc với histogram

```
>>h=imhist(f,b)
```

f là ảnh ban đầu, b là số đoạn biểu diễn(mặc định giá trị là 256). Với b là 1 giá trị nhỏ hơn, giả sử như 2, thì thang cường độ sẽ chia làm 2 khoảng : 0 đến 127, 128 đến 255, trong đó h(1) là số pixels có giá trị trong đoạn $[0,127]$ và h(2) là số pixels có giá trị trong đoạn $[127,255]$.

Nếu không có thông số đầu ra: >>imhist(f) cho ta đồ thị histogram của ảnh.

Ngoài ra ta có thể có được hàm $p(r_k)$ qua biểu thức:



```
>>p=imhist(f,b)/numel(f);
```

Với numel là tổng số pixels có trong ảnh f.

- **Cân bằng histogram:**

Giả sử ta có phép biến đổi sau: $s = T(r) = \int_0^r p_r(w)dw$

Ta có được hàm mật độ xác suất của s:

$$p_s(s) = \begin{cases} 1, & s \text{ thuộc } [0,1] \\ 0 & \text{tại vị trí khác} \end{cases}$$

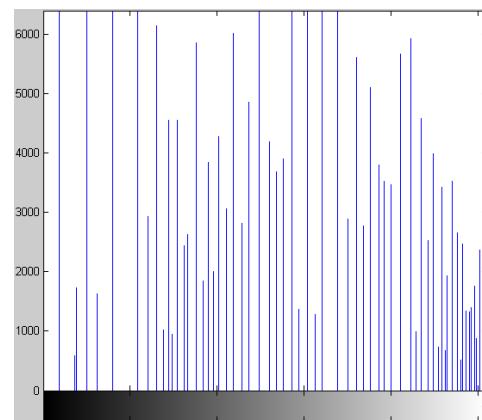
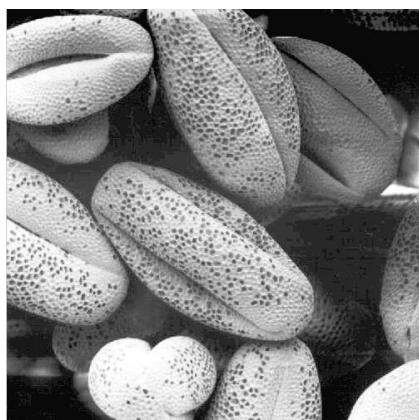
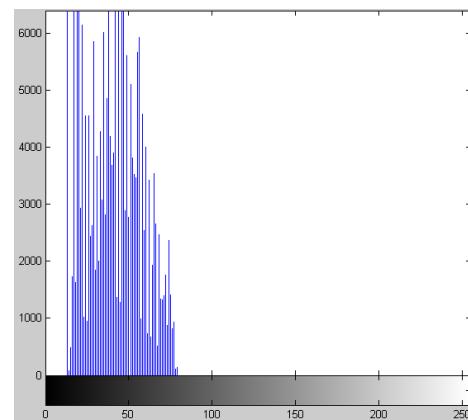
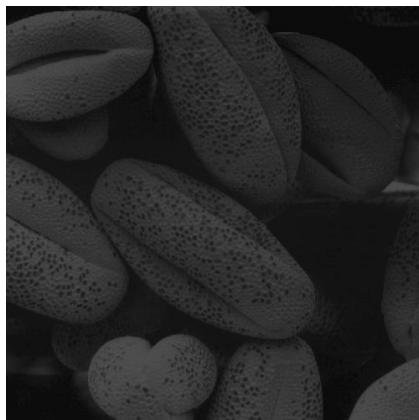
Do p_r là hàm rời rạc, ta có: $s_k = T(s_k) = \sum_{i=0}^k p_r(r_i) = \sum_{i=0}^k n_i/n$

Với phép biến đổi này, ảnh sau xử lý sẽ có biểu đồ histogram gần giống với ảnh ban đầu nhưng trải rộng trên toàn đoạn $[0,1]$, làm cho dải động lớn hơn và độ tương phản cao hơn

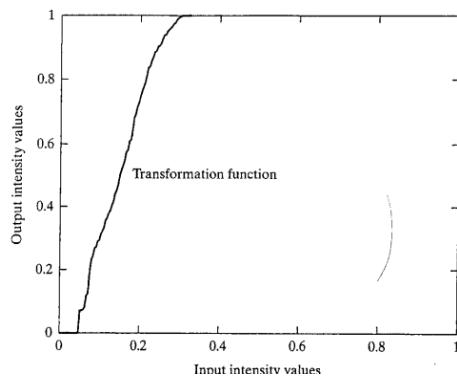
Matlab cung cấp cho ta hàm histeq để thực hiện cân bằng histogram.

`g=histeq(f, nlev)`

Với f là ảnh vào và nlev là số mức cường độ của ảnh ra. Giá trị mặc định của nlev là 64, thường ta chọn là 256 để cùng mức với histogram ảnh ban đầu.



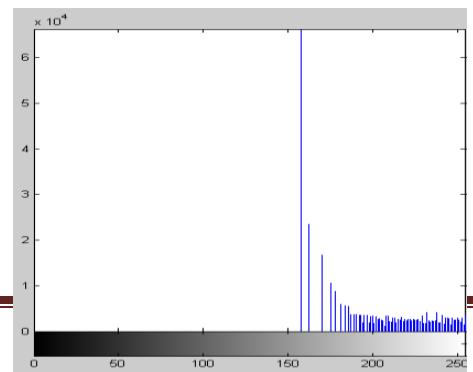
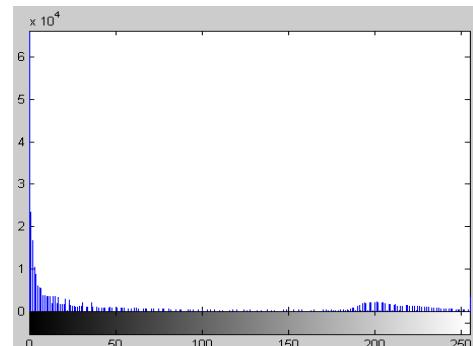
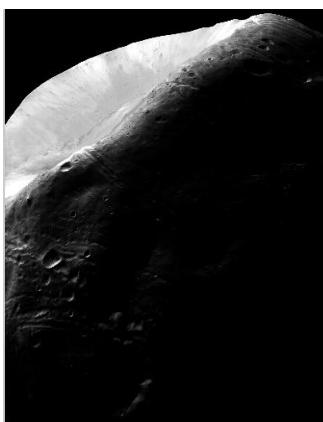
Trên là ảnh trước và sau cân bằng histogram cùng với biểu đồ histogram tương ứng

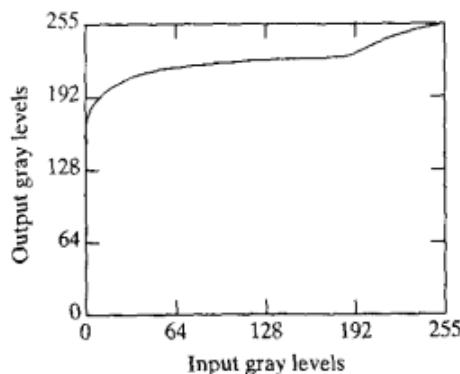


Hình dưới cho ta biết giá trị của s thay đổi theo r. Ta nhận thấy sau khi thực hiện cân bằng histogram, dải hẹp của mức xám ảnh ban đầu được mở ra toàn khoảng của ảnh ra.

- **Phối hợp histogram:**

Trong phân trước, chúng ta thấy được cân bằng histogram là một hàm có tính thích nghi với ảnh đầu vào. Phương pháp cân bằng histogram tương đối đơn giản, có thể đoán trước histogram của ảnh sau xử lý và có thể cho ảnh ra có chất lượng được nâng cao rõ rệt. Tuy nhiên, histogram sau cân bằng lại cố định với mỗi ảnh đầu vào và không thể thay đổi được. Trong một số trường hợp, việc cân bằng histogram không cho kết quả như ý, hình ảnh sau cân bằng histogram không được cải thiện hoặc không đáp ứng được yêu cầu. Xét một ví dụ cụ thể sau:





Hình trên là ảnh ban đầu và sau khi xử lý dùng cân bằng histogram. Ta thấy rằng ảnh ban đầu có vùng tối chiếm diện tích rất lớn, do đó histogram có sự tập trung cường độ về phía phần tối. Sau khi cân bằng histogram, do sự tập trung lớn của các thành phần nằm gần 0 của histogram ảnh gốc nên histogram ảnh sau tập trung cường độ về phía nửa trên, làm chất lượng ảnh không được cải thiện, mà còn giảm độ tương phản của ảnh.

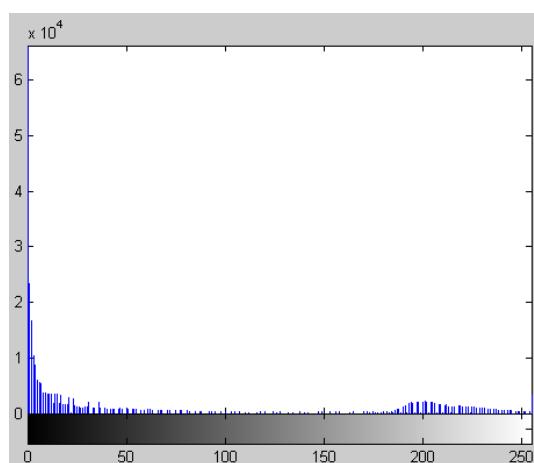
Ta có thể giải quyết vấn đề này bằng cách thực hiện một quá trình gọi là phôi hợp histogram(histogram matching). Phôi hợp histogram là quá trình biến đổi mức xám của ảnh để histogram của ảnh mới có một hình dạng cụ thể định trước.

Ta đã biết phép biến đổi $s = T(r) = \int_0^r p_r(w)dw$ cho ta kết quả là mức xám s trải rộng trên toàn miền. Giả sử ta có một biến z sao cho $H(z) = s = \int_0^z p_z(t)dt$. Khi đó z phải thỏa phương trình $z = H^{-1}(s) = H^{-1}[T(r)]$. $T(r)$ có thể tìm được từ ảnh ban đầu. Nếu ta có $p_z(z)$ đã biết thì ta có thể tìm thấy H^{-1} .

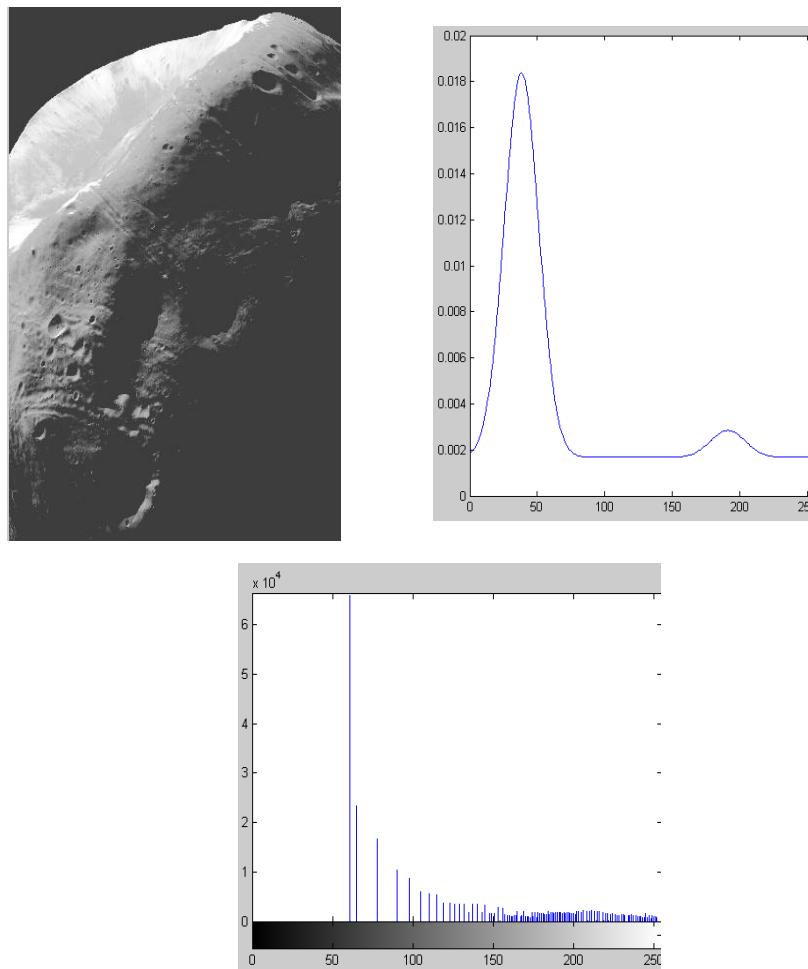
Trong Matlab ta dùng hàm histeq để thực hiện phôi hợp histogram.

`>>g=hiseq(f,hgram)`

hgram là một vector gồm các số nguyên đếm số giá trị mức xám với mỗi khoảng chia trong dải mức xám $[0, G]$.



Trở lại với ví dụ trên. Ta thấy histogram của ảnh ban đầu có các thành phần mức xám tập trung ở hai phần. Phần đầu chiếm tỉ lệ lớn tập trung gần 0, và một phần nhỏ hơn nằm về phía bên phải của histogram. Ta đã biết rằng do sự tập trung mức xám tại gần 0 nên histogram ảnh ra không trải trên toàn miền, vì thế ta có thể dùng phối hợp histogram để giảm sự tập trung này mà vẫn giữ được hình dạng tổng thể ban đầu của histogram gốc.



Ảnh sau phối hợp histogram có độ tương phản tốt hơn. Ta có thể thấy rằng các thành phần mức xám trải đều trên khoảng [0,255].

3) Lọc ảnh không gian:

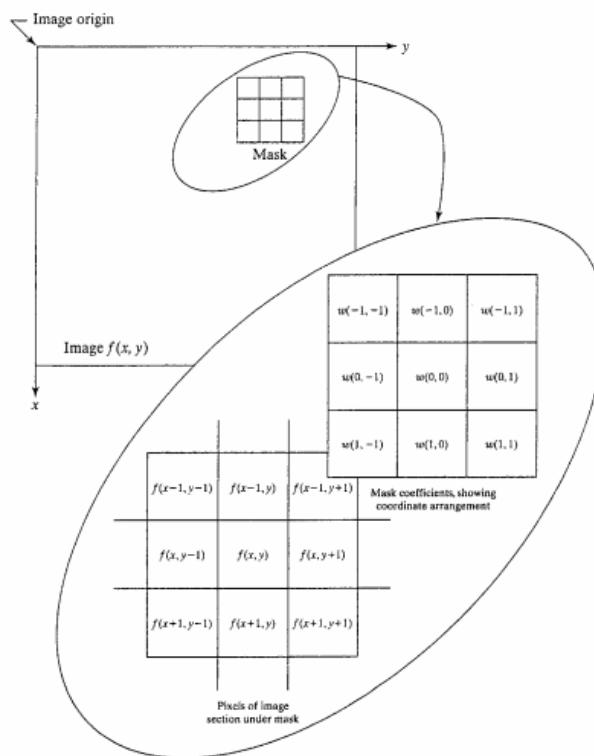
Nhiều thường xuất hiện trên ảnh do nhiều nguyên nhân khác nhau. Để giảm nhiễu và nâng cao chất lượng ảnh ta sử dụng những phương pháp lọc khác nhau, phù hợp với mỗi loại nhiễu cụ

thể. Trong chương này ta sẽ xét đến lọc ảnh trong không gian. Chương tiếp theo sẽ đề cập đến lọc ảnh trong miền tần số.

Lọc không gian(spatial filtering) cũng là một quá trình xử lý trên các điểm ảnh, dựa trên một phép toán với các điểm ảnh xung quanh. Phương pháp lọc ảnh bao gồm các bước: (1)xác định điểm ảnh trung tâm (x,y); (2) thực hiện các phép toán với các điểm xung quanh (x,y); (3) kết quả ta được đáp ứng của quá trình lọc tại (x,y); (4) lặp lại các bước trên với tất cả các điểm ảnh khác.

3.1)Lọc tuyến tính:

Lọc tuyến tính là phương pháp lọc trong đó **mức xám** mỗi pixel của ảnh mới là **tổ hợp tuyến tính** của các **mức xám** của các **pixels lân cận**,tức là **mỗi pixel lân cận** sẽ được **nhân** với một **hệ số** tương ứng rồi được **cộng lại** để được **đáp ứng** tại **điểm ảnh trung tâm**. Nếu vùng **lân cận** có kích thước **$m \times n$** thì ta có **$m \times n$** **hệ số** tương ứng. Trong Matlab,các hệ số này được sắp xếp trong một ma trận kích thước **$m \times n$** ,gọi là **bộ lọc**. Cơ chế lọc được thực hiện bằng cách di chuyển trung tâm của mặt nạ qua lần lượt từng điểm ảnh và thực hiện tính tổng các tích của mức xám các điểm ảnh xung quanh với hệ số bộ lọc. Kích thước bộ lọc là lẻ. Kích thước nhỏ nhất có ý nghĩa là **3×3** .



Giả thiết rằng $m=2a+1$, $n=2b+1$, với $a,b \geq 1$. Ta có biểu thức tính mức xám tại (x,y) :

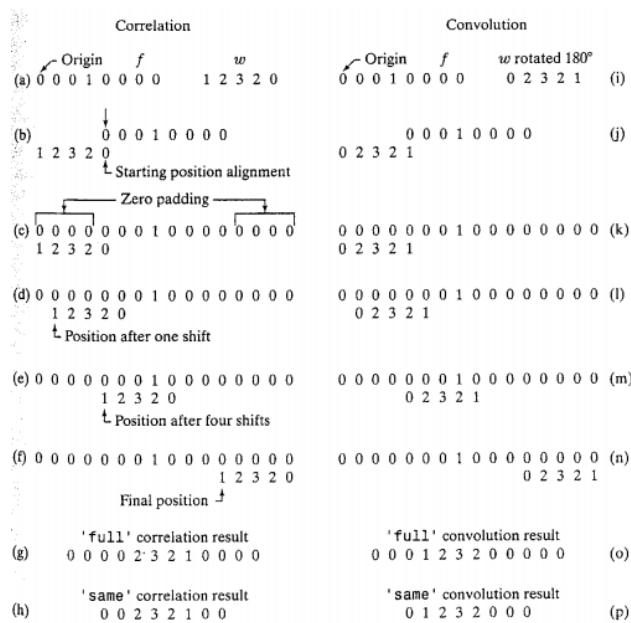
$$g(x, y) = \sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)$$

Có hai khái niệm khi chúng ta thực hiện phương pháp lọc tuyến tính, đó là tương quan(correlation) và tích chập(convolution). Tương quan là quá trình dịch bộ lọc qua từng điểm ảnh như ta đã đề cập. Tích chập cũng sử dụng quá trình tương tự, ngoại trừ bộ lọc w quay 180° trước khi tiến hành dịch bộ lọc.

Một vấn đề khác ta cũng cần quan tâm là tiến hành lọc tại các điểm nằm gần biên ảnh. Với một bộ lọc vuông $n \times n$, tại vị trí cách biên một khoảng cách $\frac{n-1}{2}$ bộ lọc sẽ có biên trùng khớp với biên ảnh, nhưng đối với các điểm ảnh nằm gần biên thì một hoặc một số hàng hoặc cột của ma trận lọc sẽ nằm bên ngoài ảnh. Có nhiều giải pháp để giải quyết vấn đề này. Một giải pháp đơn giản là ta chỉ tiến hành xử lý tại các điểm có khoảng cách không nhỏ hơn $\frac{n-1}{2}$ so với biên ảnh.

Kết quả là ảnh sau lọc có kích thước nhỏ hơn so với ảnh gốc nhưng toàn bộ điểm ảnh đều được xử lý. Trong trường hợp cần ảnh sau xử lý có cùng kích thước với ảnh gốc, một giải pháp là tiến hành xử lý các điểm ảnh ở gần biên với các hệ số bộ lọc phủ trong ảnh và bỏ qua các hệ số nằm bên ngoài ảnh. Một giải pháp khác là thêm một số mức xám vào ảnh gốc, gọi là đệm(padding) để mặt nạ phủ toàn bộ ảnh. Miếng đệm có thể là một số hàng và cột có giá trị 0(hoặc một hằng số nào đó), hoặc thêm các hàng và cột lặp lại các giá trị mức xám trên biên ảnh, hoặc đối xứng với các điểm ảnh bên trong qua biên ảnh. Điều dễ thấy là kích thước mặt nạ càng lớn, ảnh sau lọc sẽ có độ sai lệch càng lớn so với ảnh gốc, do đó để ảnh không bị biến dạng thì cách duy nhất là thực hiện lọc đối với các điểm ảnh có kích thước không nhỏ hơn $\frac{n-1}{2}$.

Xét một ví dụ sau:



Ta có một hàm f và một mặt nạ w . Ta tiến hành thêm miếng đệm vào f , cụ thể là các số 0 để mặt nạ w quét toàn bộ các điểm của f . Đối với phép toán tương quan ta tiến hành dịch mặt nạ w theo từng điểm của f , tại mỗi điểm ta tiến hành cộng các tích số của hai hàm f và w . Đối với tích chập, ta quay w một góc 180° rồi tiến hành như trên. Kết quả cuối cùng được thể hiện, chia ra làm hai dạng full và same. Dạng full là kết quả của quá trình tính toán như trên, trong khi đó dạng same cho ta kết quả có cùng kích thước với f .

Xét 1 ví dụ tương tự nhưng là mảng 2 chiều:

	Padded f		
Origin of $f(x, y)$	0 (a)	0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 (b)	
Initial position for w	1 2 3 0 0 0 0 0 0 0 4 5 6 0 0 0 0 0 0 0 7 8 9 0 1 0 (c)	'full' correlation result 0 9 8 7 0 0 0 0 0 0 0 6 5 4 0 0 0 0 0 0 0 3 2 1 0 (d)	'same' correlation result 0 0 0 0 0 0 9 8 7 0 0 6 5 4 0 0 3 2 1 0 0 0 0 0 0 (e)
Rotated w	9 8 7 0 0 0 0 0 0 0 6 5 4 0 0 0 0 0 0 0 3 2 1 0 1 0 (f)	'full' convolution result 0 1 2 3 0 0 0 0 0 0 0 4 5 6 0 0 0 0 0 0 0 7 8 9 0 (g)	'same' convolution result 0 0 0 0 0 0 1 2 3 0 0 4 5 6 0 0 7 8 9 0 0 0 0 0 0 (h)

Matlab cung cấp cho ta hàm imfilter để thực hiện lọc tuyến tính. Cú pháp của hàm như sau:

```
>>g=imfilter(f,w,filtering_mode, boundary_options, size_options)
```

Trong đó `f` là ảnh gốc, `g` là ảnh sau xử lý, các thông số tùy định: `filtering_mode` có gồm ‘corr’ thực hiện phép toán tương quan và ‘conv’ thực hiện phép chập, mặc định là ‘corr’, `size_options` có thể là ‘same’ và ‘full’ giống như cách thực hiện ví dụ trên, mặc định là ‘same’.

Thông số boundary_options cho ta cách thức chèn đệm(padding). Matlab cho ta 4 cách thức:

P: Biên ảnh gốc sẽ được thêm padding có giá trị mức xám là P, mặc định là 0

‘replicate’: các mức xám bên ngoài lặp lại giá trị của biên.

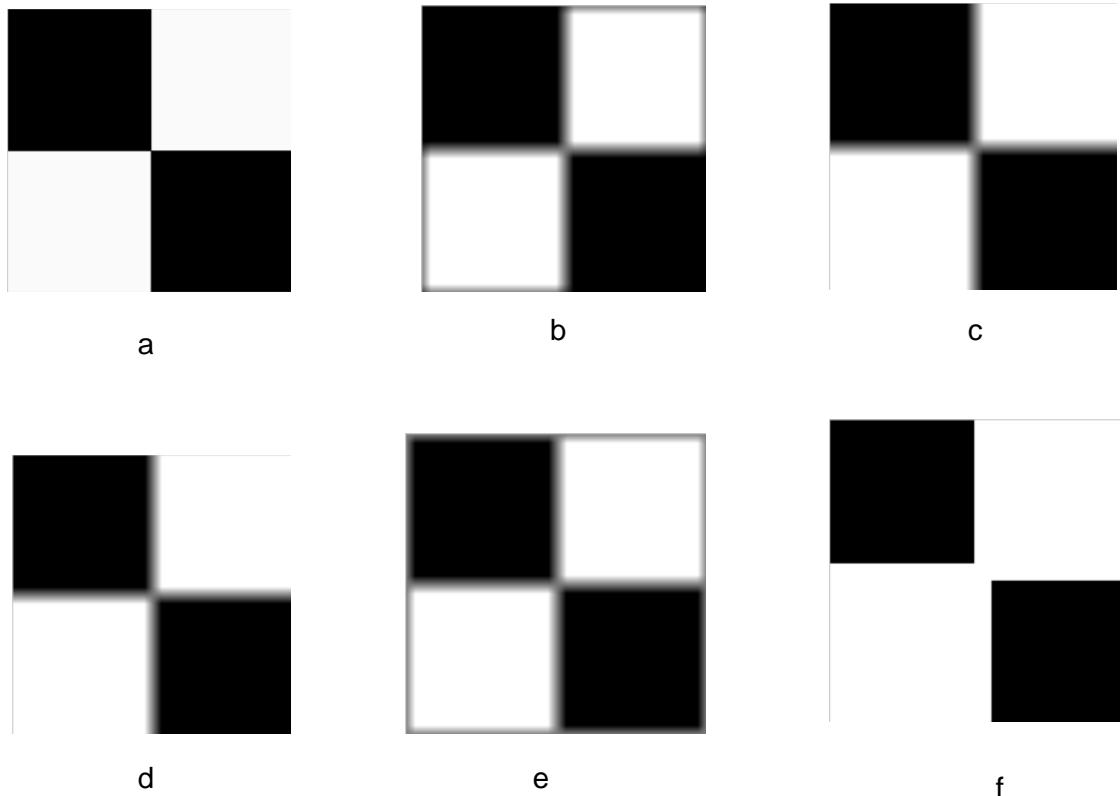
'symmetric': các mức xám bên ngoài đối xứng gương với các điểm ảnh bên trong qua biên.

‘circular’: ảnh được thêm padding trên cơ sở giả thiết ảnh đầu vào là một hàm tuần hoàn.

Ta xét một ví dụ cụ thể sử dụng hàm imfilter: Giả sử ta có một ảnh f ở class double, kích thước 512×512

```
>>w=ones(31);
```

cho ta mặt nạ lọc là ma trận vuông 31×31 . Do đây là ma trận đối xứng nên toán tương quan và chập là như nhau.



Hình a là ảnh ban đầu. Hình b là kết quả của quá trình lọc ảnh dùng padding là các mức xám giá trị 0(màu đen), ta thấy ảnh bị làm mờ đi ở cạnh giữa 2 vùng trắng và đen, cũng như giữa phần biên ảnh với vùng trắng. Điều này có thể giải thích như sau: Do mức xám tại một điểm là tổng của các tích mức xám các điểm vùng lân cận với hệ số của bộ lọc, ở đây các hệ số bộ lọc là 1, do đó mỗi điểm ảnh xem như là giá trị trung bình của các điểm ảnh xung quanh, dẫn đến kết quả như trên. Ta có thể loại bỏ phần mờ ở vùng biên bằng cách dùng thông số ‘replicate’ hoặc ‘symmetric’ như ở kết quả c và d. Với hình e, ta sử dụng thông số ‘circular’. Do sự lặp lại có tính chu kỳ làm cho vùng sáng và tối nằm cạnh nhau, dẫn đến kết quả là toàn bộ biên ảnh cũng như phần cạnh giữa 2 vùng sáng và tối bên trong bức ảnh bị mờ.

Nếu ta sử dụng ảnh ban đầu là class uint8 và sử dụng bộ lọc w như trên, ta nhận được kết quả là hình f với một phần dữ liệu ảnh gốc bị mất. Lý do là các giá trị lớn hơn 255 đều bị gán giá trị 255. Để giải quyết vấn đề đó, cửa sổ lọc cần được chuẩn hóa trước khi tiến hành lọc:

```
>>w=w/(sum(w(:)));
```

Ta có công thức tính mức xám của ảnh sau xử lý qua bộ lọc chuẩn hóa:

$$g(x, y) = \frac{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t) f(x + s, y + t)}{\sum_{s=-a}^a \sum_{t=-b}^b w(s, t)}$$

3.2)Lọc phi tuyến:

Cũng như lọc tuyến tính, lọc phi tuyến sử dụng một cửa sổ lọc và trượt qua các pixels của ảnh gốc. Tuy nhiên nếu lọc tuyến tính dựa theo việc lấy tổng có trọng số các pixels lân cận thì lọc phi tuyến sẽ thực hiện một phép toán phi tuyến với các pixels đó. Ví dụ, gán giá trị tại mỗi pixel bằng giá trị lớn nhất của các pixel lân cận là một phép toán phi tuyến.

Matlab cung cấp cho ta 2 hàm nlfilter và colfilt để thực hiện lọc phi tuyến một cách tổng quát. Hàm nlfilter thực hiện trực tiếp trên ma trận 2 chiều, trong khi hàm colfilt lọc theo từng cột. Hàm colfilt đòi hỏi nhiều bộ nhớ hơn nlfilter, nhưng tốc độ thực thi lại nhanh hơn đáng kể. Các ứng dụng thường đòi hỏi tốc độ cao nên hàm colfilt được sử dụng nhiều hơn. Ta nói rõ hơn về cách dùng hàm colfilt.

Giả sử ta có một ảnh f kích thước $M \times N$, và một cửa sổ lọc kích thước $m \times n$, colfilt sẽ tạo ra một ma trận, giả sử tên là A, với kích thước lớn nhất có thể là $mn \times MN$, trong đó mỗi cột sẽ tương ứng với các phần tử điểm ảnh lân cận điểm ảnh cần lọc. Ví dụ như cột đầu tiên sẽ tương ứng với các pixels lân cận điểm ảnh ở vị trí đầu tiên của ảnh. Đối với các điểm ảnh gần biên thì cột tương ứng của ma trận A sẽ có thêm các thành phần padding, colfilt sử dụng padding là 0. Thông thường A có kích thước các cột nhỏ hơn MN vì hàm colfilt thường chia ảnh f ra làm nhiều ảnh nhỏ để tiết kiệm bộ nhớ.

```
>>g=colfilt(f,[m n], 'sliding', @fun);
```

Trong đó f là ảnh gốc, g là ảnh sau xử lý, cửa sổ lọc có kích thước $m \times n$, 'sliding' là thông số sử dụng trong lọc phi tuyến, chỉ ra quá trình xử lý là trượt cửa sổ lọc qua các pixels của ảnh f, fun là một hàm phi tuyến đã được định nghĩa từ trước.

Do cách sắp xếp của ma trận A như trên, hàm fun phải tiến hành trên mỗi cột của A, tạo ra một vector hàng v, trong đó mỗi phần tử là kết quả của phép toán trên mỗi cột của A. Ta có thể suy ra là vector v có kích thước lớn nhất có thể là $1 \times MN$.

Do colfilt tự tạo padding cho ảnh có giá trị là 0 và không thể thay đổi được. Do đó, không như lọc tuyến tính dùng hàm imfilter có thể tùy biến padding, đối với lọc phi tuyến ta phải tiến hành thêm padding ngay lúc đầu cho ảnh trước khi tiến hành lọc. Matlab cung cấp cho ta hàm padarray để thực hiện việc này.

```
>> fp=padarray(f, [r c], method, direction);
```

Trong đó f là ảnh ban đầu, fp là ảnh sau khi thêm padding, [r c] là số hàng và cột mà ta muốn thêm vào ảnh, thông số method có các lựa chọn P(giá trị mức xám), ‘symmetric’, ‘replicate’ và ‘circular’ với cách thức thực hiện đã được đề cập ở trên, giá trị mặc định là 0 , direction có thể là ‘pre’, ‘post’, ’both’(mặc định) cho phép thêm padding vào trước phần tử đầu tiên, sau phần tử đầu tiên của mỗi chiều, hoặc cả hai.

Xét ví dụ sau:

```
>>f= [ 1 2; 3 4];
>>fp=padarray(f, [3 2], 'replicate', 'post')
fp =
```

```
1   2   2   2
3   4   4   4
3   4   4   4
3   4   4   4
3   4   4   4
```

Bây giờ chúng ta sẽ định nghĩa một hàm lọc phi tuyến và lấy hàm này làm tham số cho hàm colfilt

```
function v=gmean(A)
mn= size(A,1);
v=prod(A,1).^(1/mn);
```

Hàm trên dùng để tính trung bình nhân(geometric mean) của các giá trị mức xám lân cận điểm ảnh xử lý. Công thức tính trung bình nhân tổng quát là:

$$x = \sqrt[n]{x_1 \cdot x_2 \dots x_n} = \left(\sum_{i=1}^n x_i \right)^{1/n}$$

Tiếp theo ta tiến hành thêm padding cho ảnh gốc:

```
>>f=padarray(f,[m n], 'replicate');
```

Cuối cùng ta dùng hàm colfilt để tạo ảnh mới:

```
>>g=colfilt(f, [m n], 'sliding', @gmean);
```

Hàm colfilt sẽ lấy kết quả thực hiện từ hàm gmean, tạo ra một vector hàng chứa kết quả phép toán trung bình nhân thực hiện với mỗi cột của A, sau đó sắp xếp lại thành một ma trận 2 chiều là ảnh ra của bộ lọc.

Phép toán lấy trung bình nhân ta sẽ gặp trong phần phục hồi ảnh.

3.3) Ứng dụng lọc ảnh không gian:

3.3.1) Các bộ lọc làm mịn ảnh:

Các bộ lọc làm mịn ảnh được sử dụng để làm mờ và giảm nhiễu. Làm mờ ảnh được sử dụng trong quá trình tiền xử lý ảnh, nhằm mục đích loại bỏ các chi tiết nhỏ ra khỏi ảnh trước khi tiến hành tách các thành phần lớn hơn khỏi ảnh, làm mờ còn được sử dụng để làm liền lại những đứt quãng nhỏ của đường thẳng hoặc đường cong. Chúng ta cũng có thể giảm nhiễu bằng cách làm mờ ảnh bằng các bộ lọc tuyến tính cũng như phi tuyến.

a) Lọc tuyến tính:

Như ta đã biết, lọc tuyến tính là một quá trình trong đó mỗi điểm ảnh có giá trị bằng trung bình của các điểm ảnh lân cận xác định bởi bộ lọc. Do đó ảnh sau xử lý trở nên mượt hơn, giảm độ sắc nét so với ảnh gốc. Kết quả là các thành phần nhiễu ngẫu nhiên, thường có mức xám khác biệt với các vùng lân cận sẽ được loại bỏ. Tuy nhiên một hạn chế dễ thấy khi tiến hành làm mượt ảnh là tại các vị trí biên (chi tiết được sử dụng nhiều trong xử lý ảnh), nơi có sự thay đổi nhanh chóng của các mức xám, lại bị làm mờ đi ảnh hưởng đến các bước tiếp theo trong xử lý ảnh. Tuy nhiên nếu sử dụng cửa sổ lọc thích hợp, ta có thể giảm nhiễu mà chỉ ít ảnh hưởng đến biên ảnh.

Một cửa sổ lọc chuẩn hóa thường thấy và đã được đề cập là cửa sổ có các hệ số giống nhau:

$w = 1/9$

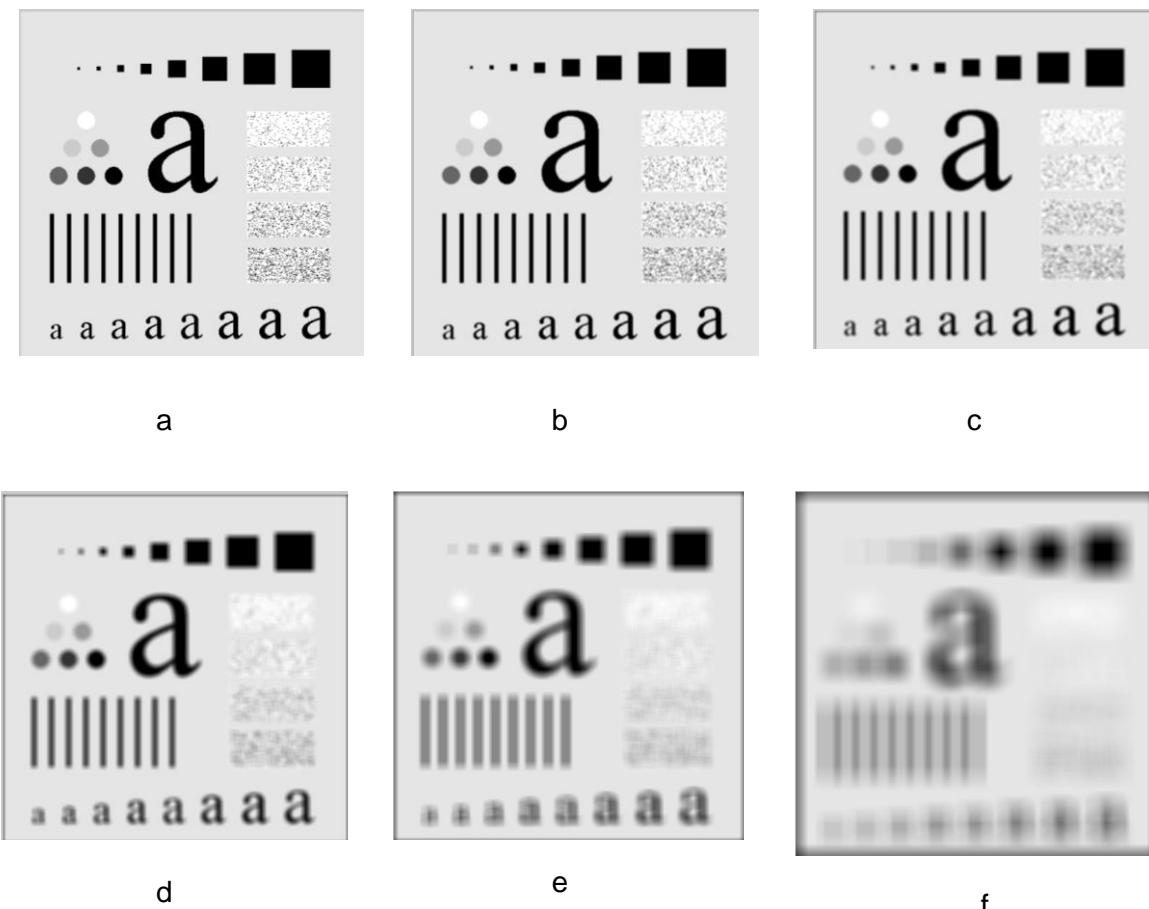
1	1	1
1	1	1
1	1	1

Một loại cửa sổ lọc chuẩn hóa khác có các hệ số khác nhau:

$w = 1/16$

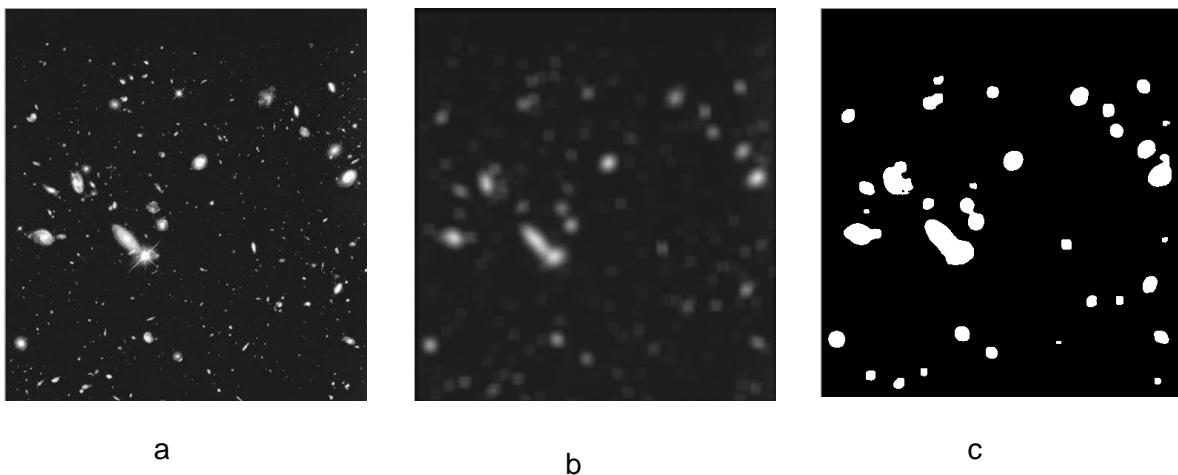
1	2	1
2	4	2
1	2	1

Bộ lọc trên tiến hành lấy trung bình có trọng số đối với các điểm ảnh lân cận, tức là mỗi điểm ảnh được nhân với một hệ số khác nhau, hệ số càng lớn thì điểm ảnh đó có trọng số càng lớn. Ở bộ lọc trên, điểm ảnh trung tâm có vai trò quan trọng nhất trong phép toán tính trung bình, càng ra xa điểm ảnh trung tâm trọng số của các điểm ảnh cũng giảm dần. Mục đích của việc làm này là hạn chế ảnh bị mờ khi tiến hành làm mượt. Tuy nhiên ta rất khó thấy sự khác biệt giữa hai bộ lọc trên do cửa sổ lọc có kích thước nhỏ hơn nhiều so với kích thước ảnh



Ta xét ví dụ trên với việc sử dụng các bộ lọc trung bình có kích cỡ khác nhau, ở đây ta dùng padding là các giá trị 0. Hình a là ảnh gốc. Hình b sử dụng bộ lọc 3×3 , hình bị mờ đi chút ít, với các chi tiết nhỏ như chữ a nhỏ và thành phần nhiễu bị mờ nhiều hơn so với các chi tiết khác. Kết quả tương tự cũng xảy ra với hình c sử dụng bộ lọc kích thước 5×5 . Ta thấy các chi tiết nhỏ như nhiễu đã giảm dần, các mép hình răng cưa cũng đã được làm mượt hơn, nhưng vẫn đảm bảo cách thành phần kích thước lớn không bị ảnh hưởng nhiều. Ảnh d dùng cửa sổ 9×9 , ảnh mờ hơn, các chi tiết nhiễu đã được giảm khá nhiều. Ảnh e và ảnh f dùng các bộ lọc tương ứng 16×16 và 35×35 , các chi tiết nhỏ gần như đã bị loại khỏi ảnh, do đó có thể dễ dàng lấy được các thành phần có kích thước lớn.

Xét một ví dụ khác:



Hình a là ảnh gốc. Hình b là ảnh sau lọc với bộ lọc kích thước 15×15 , cho ta thấy các chi tiết nhỏ gần như bị loại bỏ. Để thu các thành phần kích thước lớn của ảnh ta có thể biến đổi ảnh b thành ảnh nhị phân. Ở đây ta cho mức ngưỡng là 25% giá trị mức xám lớn nhất của ảnh b. Kết quả ta được ở ảnh c, tất cả các chi tiết nhỏ bị loại bỏ, chỉ còn các thành phần kích thước lớn.

b) Bộ lọc hạng(Order-Statistics filters):

Lọc hạng là phương pháp lọc trong đó mức xám tại một điểm được tính dựa trên sự xếp hạng các điểm ảnh lân cận. Trong Matlab, hàm `ordfilt2` cung cấp cho ta bộ lọc dạng này.

```
>>g=ordfilt2(f, order, domain)
```

Hàm `ordfilt` sắp xếp thứ tự từ nhỏ đến lớn các điểm nằm trong một miền xác định `domain`, `domain` là một ma trận có kích thước của cửa sổ lọc, gồm có các phần tử có giá trị 0 hoặc 1 dùng để xác định các điểm ảnh lân cận được sử dụng, các điểm ảnh ứng với 0 sẽ không được xếp hạng. Pixel của ảnh gốc sẽ được thay thế bằng giá trị mức xám thứ `order`. Giả sử muốn lấy phần tử nhỏ nhất trong lân cận $m \times n$, ta dùng lệnh:

```
>>g=ordfilt2(f,1,ones(m,n));
```

hoặc muốn lấy phần tử lớn nhất:

```
>>g=ordfilt2(f,m*n,ones(m,n));
```

Một trường hợp đặc biệt của lọc hạng là bộ lọc trung vị, trong đó phần tử được chọn là phần tử xếp hạng chính giữa. Bộ lọc này thường được sử dụng trong thực tế.

```
>>g=ordfilt2(f,median(1:m*n),ones(m,n));
```

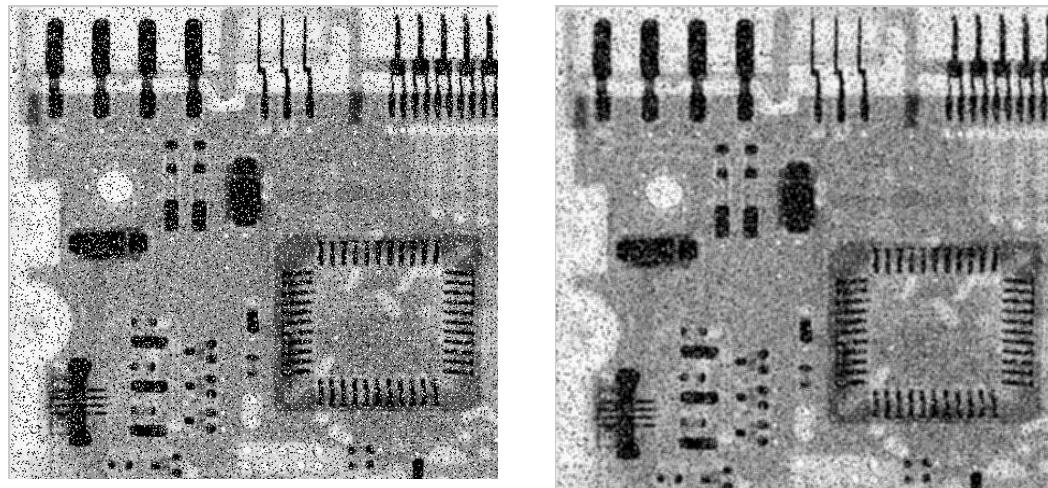
Matlab cũng cung cấp hàm `medfilt2` để thực hiện lọc trung vị.

```
>>g=medfilt2(f, [m n], padopt);
```

Trong đó $[m \times n]$ cho ta các thành phần nằm lân cận có kích thước $m \times n$, padopt có thể là ‘zeros’, ‘symmetric’ và ‘index’, khi đó các pixels thêm vào là 1 nếu ảnh thuộc kiểu double và 0 nếu thuộc các kiểu khác.

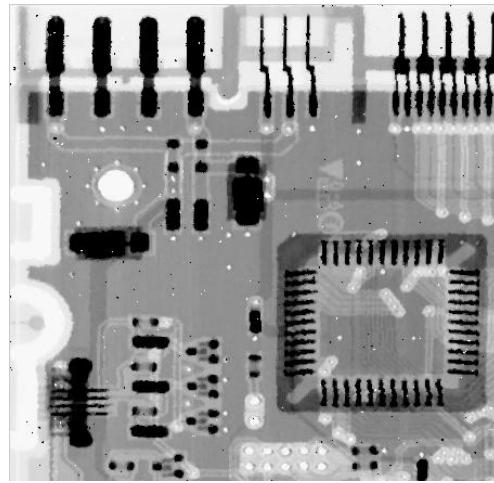
Bộ lọc trung vị sử dụng hiệu quả trong những trường hợp có các giá trị pixel lớn hơn hoặc nhỏ hơn hẳn các giá trị lân cận, ví dụ như nhiều “salt and pepper”, khi đó các thành phần nhiễu này do có mức xám khác biệt với các điểm lân cận sẽ được thay thế bằng mức xám gần bằng các điểm xung quanh.

Xét ví dụ:



a

b



c

Hình a là ảnh chụp của một board mạch bị nhiễu salt and pepper. Sử dụng bộ lọc trung bình kích thước 3×3 cho ta kết quả hình b, ảnh đã bớt nhiễu nhưng bị mờ đi. Hình c là kết quả của phép lọc trung vị dùng cửa sổ lọc 3×3 cho ta hình ảnh được cải thiện một cách rõ rệt.

3.3.2) Các bộ lọc làm sắc nét ảnh:

Mục đích của việc làm sắc ảnh là nổi bật các chi tiết trong ảnh hoặc làm sắc các chi tiết bị mờ bởi quá trình làm mượt ảnh. Ta đã biết quá trình làm mượt ảnh là thực hiện phép lấy trung bình các giá trị lân cận điểm ảnh cần xử lý, tương tự như phép toán tích phân, trong khi đó quá trình làm sắc nét ảnh tập trung vào sự sai khác giữa các chi tiết trong ảnh, giống như phép toán vi phân. Kết quả là biến ảnh và các chi tiết nhiễu, nơi có sự khác biệt về mức xám với các điểm ảnh xung quanh, được làm nổi bật lên.

Trong phần này, ta sẽ thực hiện làm sắc nét ảnh dựa trên đạo hàm bậc một và bậc hai của hàm rời rạc.

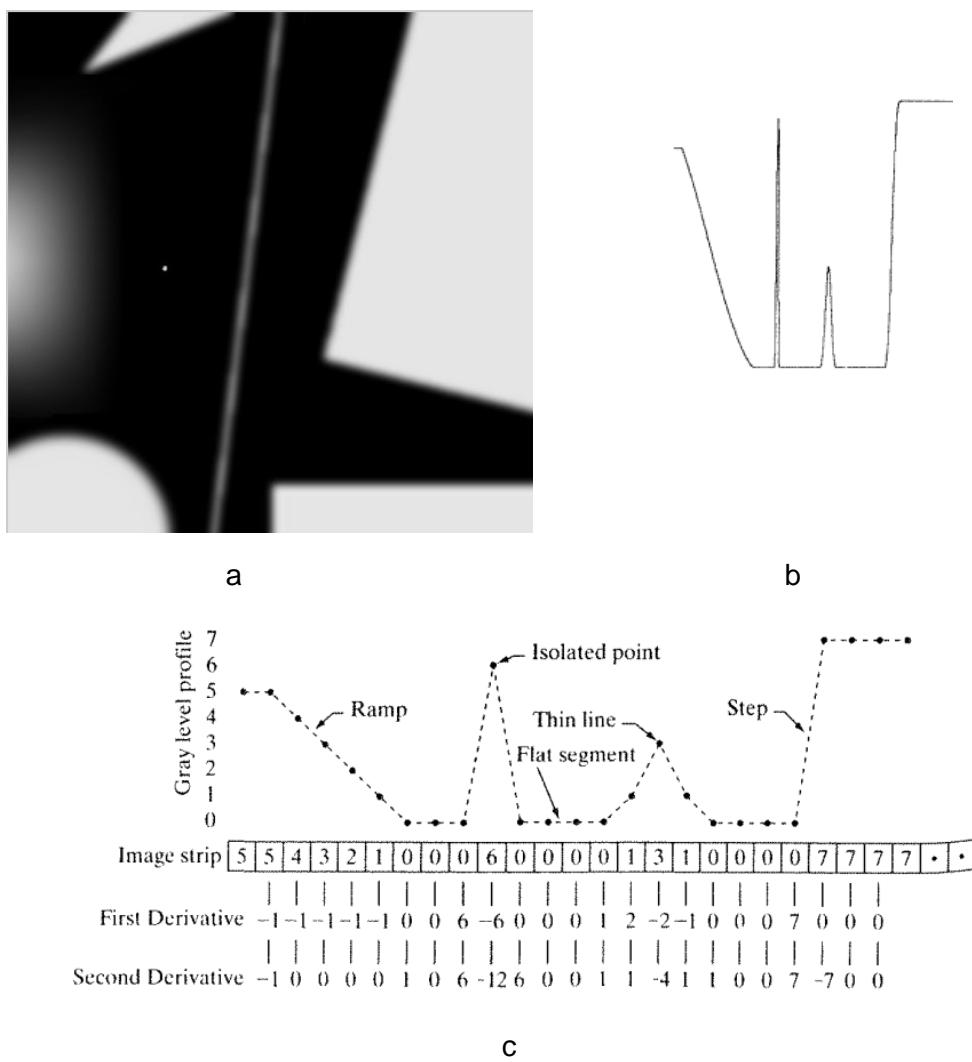
Đạo hàm bậc 1 của hàm rời rạc một chiều:

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

Và đạo hàm bậc 2 :

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$

Để thấy được sự giống cũng như khác nhau về cơ bản giữa hai phương pháp sử dụng đạo hàm bậc 1 và bậc 2, ta xét một ví dụ cụ thể sau:



Ta để ý thấy ảnh trên hình a có một đường chéo và một điểm nhiễu. Hình b biểu diễn mức xám của các pixels nằm trên đường ngang qua điểm giữa của ảnh, bao gồm cả điểm nhiễu. Hình c đơn giản hóa các giá trị mức xám của hình b, chỉ gồm 8 mức xám khác nhau. Từ hình c ta có thể phân tích ảnh hưởng của phương pháp đạo hàm bậc 1 và bậc 2 đối với điểm nhiễu, với đường chéo và cạnh biên giữa đối tượng và nền.

Các đoạn có mức xám không đổi thì đạo hàm bậc 1 và 2 đều cho đáp ứng là 0. Với đoạn dốc thoái(ramp), đạo hàm bậc 1 cho các mức khác 0 trên toàn đoạn, đạo hàm bậc 2 chỉ cho các giá trị khác 0 ở đầu và cuối đoạn, điều đó chứng tỏ với các đoạn chuyển tiếp như thi đạo hàm bậc 1 tạo ra cạnh dày hơn và đạo hàm bậc 2 tạo ra cạnh sắc hơn . Với điểm nhiễu(isolated point), đáp ứng xung quanh và tại điểm nhiễu đạo hàm bậc 2 lớn hơn so với đạo hàm bậc 1, do đó đạo hàm bậc 2 tạo ra chi tiết sắc hơn đối với nhiễu và các giá trị mức xám thay đổi nhanh. Đường chéo(thin line) cũng tương tự với điểm nhiễu. Còn lại với bước nhảy(step) thì đáp ứng của đạo hàm bậc 1 và 2 là tương tự nhau.

Ta có thể kết luận: (1) Đạo hàm bậc 1 tạo ra các cạnh dày hơn so với đạo hàm bậc 2, (2)Đạo hàm bậc 2 ảnh hưởng nhiều hơn đến các chi tiết tinh.

Đạo hàm bậc 2 được sử dụng nhiều hơn trong xử lý ảnh do nó làm nổi bật các chi tiết sắc, do đó chúng ta sẽ tập trung vào phương pháp này để làm sắc nét ảnh.

Do ảnh là một hàm rời rạc hai chiều nên ta cần có đạo hàm bậc hai của hàm 2 chiều.

Toán tử Laplace của hàm 2 biến là:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Ta cũng có công thức tính đạo hàm bậc 2 cho hàm rời rạc hai chiều thường sử dụng là:

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

Và:

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

Toán tử Laplace trong không gian rời rạc 2 chiều là:

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

Biểu thức trên có thể thực hiện đôi với các phân tử của ảnh bằng cách nhân chập ảnh với ma trận:

0	1	0
1	-4	1
0	1	0

hoặc

0	-1	0
-1	4	-1
0	-1	0

Một định nghĩa khác về đạo hàm bậc 2 có sử dụng thêm các phân tử đường chéo:

1	1	1
1	-8	1
1	1	1

hoặc

-1	-1	-1
-1	8	-1
-1	-1	-1

Do Laplace là toán tử đạo hàm, nó làm sắc bén các chi tiết tinh hay các thành phần mức xám biến đổi nhanh, nhưng lại dẫn đến giá trị 0 cho những vùng có cùng một mức xám cũng như giảm giá trị của các thành phần mức xám ít biến đổi. Một cách đơn giản phục hồi lại các vùng này nhưng vẫn giữ cho các chi tiết sắc bén là cộng ảnh gốc với ảnh thực hiện phép lọc bằng toán tử Laplace.

$$g(x,y) = f(x,y) + c[\nabla^2 f(x,y)]$$

Trong đó $c=1$ nếu hệ số trung tâm của mặt nạ lọc là dương, $c=-1$ nếu ngược lại.

Xét ví dụ:



a



b



c



d

Hình a là ảnh gốc. Hình b là ảnh sau xử lý dùng mặt nạ Laplace, ở đây ta thấy vùng có thành phần mức xám biến đổi chậm sau khi lọc sẽ bị giảm giá trị. Trong khi đó phần cạnh biên và các chi tiết có mức xám biến đổi nhanh được thể hiện rõ nét hơn. Hình c là kết quả của việc cộng ảnh gốc với ảnh Laplace, phục hồi lại phần bị giảm mức xám nhưng vẫn giữ được sự sắc nét của các chi tiết. Hình d sử dụng mặt nạ có thêm các giá trị đường chéo, cho hình ảnh sắc nét hơn c.

Mặt nạ Laplace có thể tạo ra trong Matlab nhờ hàm fspecial

`>>fspecial('laplacian', alpha)`

$\frac{\alpha}{1+\alpha}$	$\frac{1-\alpha}{1+\alpha}$	$\frac{\alpha}{1+\alpha}$
$\frac{1-\alpha}{1+\alpha}$	$\frac{-4}{1+\alpha}$	$\frac{1-\alpha}{1+\alpha}$
$\frac{\alpha}{1+\alpha}$	$\frac{1-\alpha}{1+\alpha}$	$\frac{\alpha}{1+\alpha}$

Trong đó hệ số α cho phép chỉnh mức độ sắc nét của hình ảnh.

Ta có thể dùng mặt nạ lọc trực tiếp như sau:

0	-1	0
-1	5	-1
0	-1	0

-1	-1	-1
-1	9	-1
-1	-1	-1

- **Bộ lọc tăng cường (high-boost filter):**

Bộ lọc tăng cường cũng là một ứng dụng của toán tử Laplace:

0	-1	0
-1	A+4	-1
0	-1	0

-1	-1	-1
-1	A+8	-1
-1	-1	-1

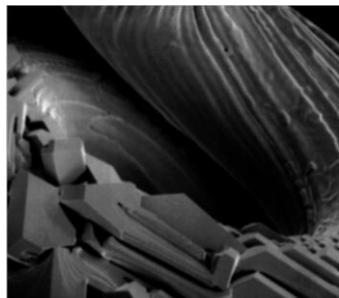
Với $A \geq 1$

Với A bằng 1, bộ lọc tăng cường giống như mặt nạ lọc trực tiếp ở trên. $A > 1$ thì khả năng làm sắc nét ảnh giảm dần, nếu A đủ lớn thì ảnh sau xử lý gần như giống ảnh ban đầu nhân với 1 hằng số.

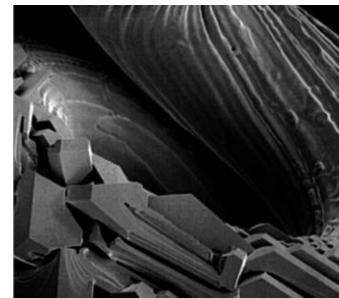
Một ứng dụng của bộ lọc tăng cường là làm sáng ảnh vì nó nâng mức xám trung bình của ảnh ban đầu mà vẫn giữ được tính chất làm nét ảnh.

Ví dụ:

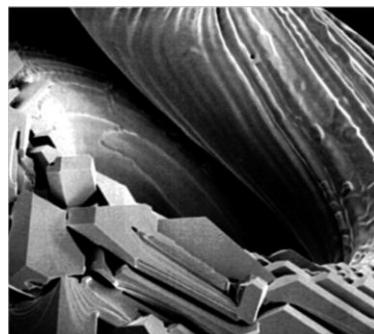
Ảnh gốc



$A=1$



$A=1.7$



4) Lọc ảnh trong miền tần số:

Trong phần trước chúng ta đã đề cập đến các bộ lọc không gian để nâng cao chất lượng ảnh số. Trong phần này, chúng ta sẽ thực hiện các bộ lọc ảnh trong miền tần số thông qua biến đổi Fourier. Biến đổi Fourier đóng vai trò quan trọng trong xử lý ảnh, có khả năng linh hoạt cao trong thiết kế và tiến hành các phương pháp lọc trong việc nâng cao chất lượng ảnh, phục hồi ảnh, nén ảnh... Trong phần này ta sẽ tập trung vào các bộ lọc để nâng cao chất lượng ảnh.

4.1) Biến đổi Fourier rời rạc 2 chiều(2-D Discrete Fourier Transform(DFT)):

Giả sử ta có một ảnh kích thước $M \times N$ được mô tả bởi hàm 2 chiều $f(x,y)$, DFT của f là $F(u,v)$ được cho bởi biểu thức:

$$F(u,v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

Với $u=0,1,2,\dots,M-1$ và $v=0,1,2,\dots,N-1$. Kết quả ta được hệ trực hai chiều trong miền tần số với hai biến u,v . Các giá trị $F(u,v)$ tạo thành hình chữ nhật kích thước $M \times N$, cùng kích thước với ảnh gốc.

Biến đổi Fourier ngược:

$$f(x,y) = \frac{1}{MN} \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

Trong Matlab bắt đầu với giá trị 1 trong ma trận, $F(1,1)$ và $f(1,1)$ sẽ tương ứng với $F(0,0)$ và $f(0,0)$ trong biểu thức trên. $F(0,0)$ gọi là thành phần hằng số hoặc thành phần 1 chiều(DC) của biến đổi Fourier, $F(0,0)$ bằng MN lần tổng giá trị $f(x,y)$.

Ta nhận thấy $f(x,y)$ là số thực, còn $F(u,v)$ lại là số phức.

Phổ biên độ: $|F(u,v)| = |R(u,v) + jI(u,v)| = [R^2(u,v) + I^2(u,v)]^{1/2}$

Và pha: $\varphi(u,v) = \tan^{-1} \left[\frac{I(u,v)}{R(u,v)} \right]$

Mật độ phổ công suất: $P(u,v) = |F(u,v)|^2$

Trong miền tần số ta sẽ quan tâm đến $|F(u,v)|$ và $P(u,v)$.

Với $f(x,y)$ là thực, ta được: $|F(u,v)| = |F(-u,-v)|$

$F(u,v)$ tuân hoán nên ta có: $F(u,v) = F(u+M, v) = F(u, v+N) = F(u+M, v+N)$

Biến đổi ngược cũng cho ta $f(x,y) = f(x+M,y) = f(x,y+N) = f(x+M,y+N)$ tuân hoàn:

Do tính chất đối xứng qua điểm $(0,0)$ và tuân hoàn của $|F(u,v)|$, ta có thể dịch điểm $(0,0)$ về vị trí trung tâm tức là vị trí $(M/2, N/2)$ của phô. Ta có

$$f(x,y) e^{j2\pi(u_0x/M+v_0y/N)} = F(u-u_0, v-v_0)$$

$$\text{Với } u_0=M/2 \text{ và } v_0=N/2 : \quad e^{j2\pi(u_0x/M+v_0y/N)} = (-1)^{(x+y)}$$

$$F(u-M/2, v-N/2) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) (-1)^{(x+y)} e^{-j2\pi(\frac{ux}{M} + \frac{vy}{N})}$$

Do đó nhân $f(x,y)$ với $(-1)^{(x+y)}$ thì $F(0,0)$ sẽ dịch đến vị trí trung tâm.

Việc dịch vị trí như vậy cho ta quan sát phô một cách dễ dàng hơn và thực hiện lọc ảnh một cách trực quan. Từ đây khi nói đến giá trị DC, ta xem điểm đó ở vị trí trung tâm của phô.

Trong Matlab, hàm fft2 cho ta biến đổi Fourier của ma trận không gian 2 chiều:

```
>> F=fft2(f);
```

Phép dịch phô để đưa giá trị $F(0,0)$ về trung tâm:

```
>> F2=fftshift(f);
```

Để quan sát phô ta sử dụng hàm imshow, với lưu ý là phô biên độ:

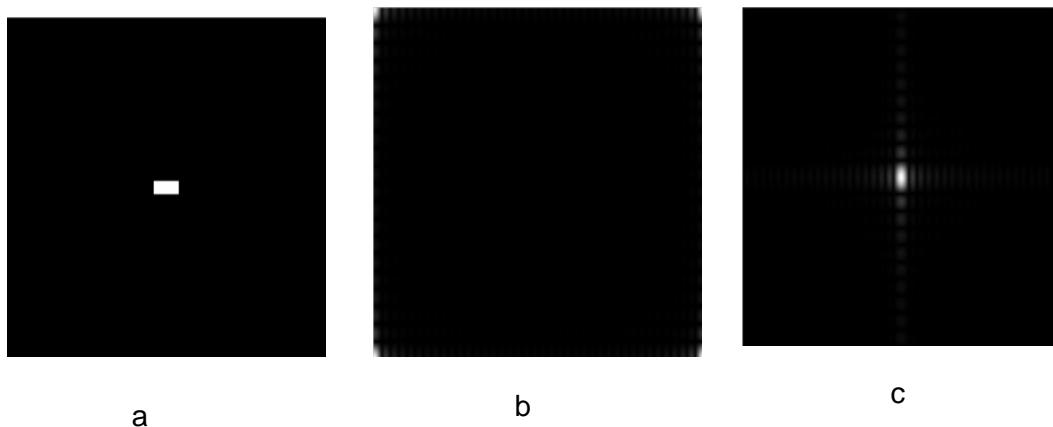
```
>> imshow(abs(F2), [ ]);
```

Ta còn có hàm ifft2 là phép biến đổi Fourier ngược:

```
>> f=ifft2(F);
```

Và hàm ifftshift đưa điểm vị trí trung tâm về góc trái trên cùng của phô:

```
>> F=ifftshift(F2);
```



Hình b là phô của ảnh trong hình a, c là kết quả của quá trình dịch phô trong hình b.

4.2) Lọc ảnh trong miền tần số:

Mỗi giá trị $F(u,v)$ chứa tất cả các thành phần $f(x,y)$ nhân với thành phần mũ, do đó phô Fourier có liên quan đến sự thay đổi các giá trị mức xám của ảnh. Tần số thấp ứng với các thành phần có sự thay đổi chậm về mức xám của ảnh, trong khi đó tần số cao ứng với sự thay đổi mức xám nhanh hơn, ví dụ như nhiễu và cạnh biên. Như vậy nếu lọc đi các thành phần tần số cao và lấy thành phần tần số thấp thì ảnh thu được sẽ mượt và giảm nhiễu, trong khi đó nếu ta chỉ lấy các thành phần tần số cao thì ảnh sau lọc sẽ sắc nhọn và các chi tiết như nhiễu sẽ nổi bật hơn. Ta có các bộ lọc tương ứng là bộ lọc thông thấp và bộ lọc thông cao.

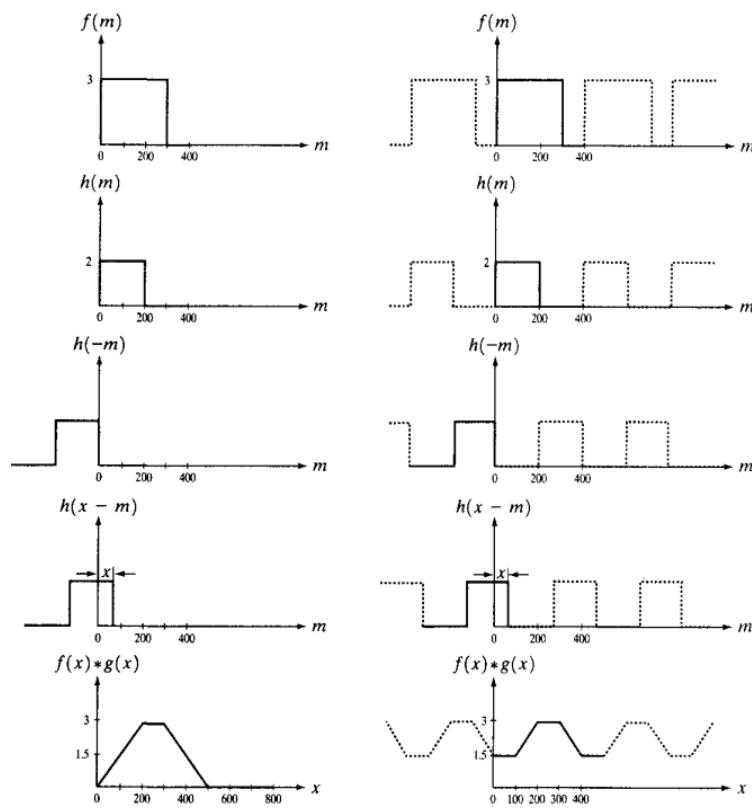
4.2.1) Các khái niệm cơ bản:

Ta có $f(x,y) * h(x,y) \Leftrightarrow H(u,v)F(u,v)$

Ta đã biết lọc ảnh trong không gian là phép chập giữa ảnh $f(x,y)$ và mặt nạ $w(x,y)$. Nó tương đương với phép nhân $F(u,v)$ và $H(u,v)$ trong miền tần số. Ta có thể tiến hành lọc trong miền tần số rồi dùng IDFT để có ảnh sau lọc

Một lưu ý là ảnh và biến đổi của nó ở miền tần số sẽ có tính chu kỳ như đã đề cập khi dùng DFT và IDFT, dẫn đến sự tác động lẫn nhau giữa các thành phần khác 0 của 2 chu kỳ gần nhau làm ảnh sau xử lý bị biến dạng.

Xét một ví dụ sau:



Bên trái là phép chập giữa hai tín hiệu không có tính chu kỳ, tương ứng với phép lọc trong miền tần số. Bên phải là tích chập của hai tín hiệu tương tự nhưng có tính tuần hoàn. Ta thấy kết quả tích chập cũng là một hàm mang tính chu kỳ nhưng đã bị biến dạng với tín hiệu bên trái.

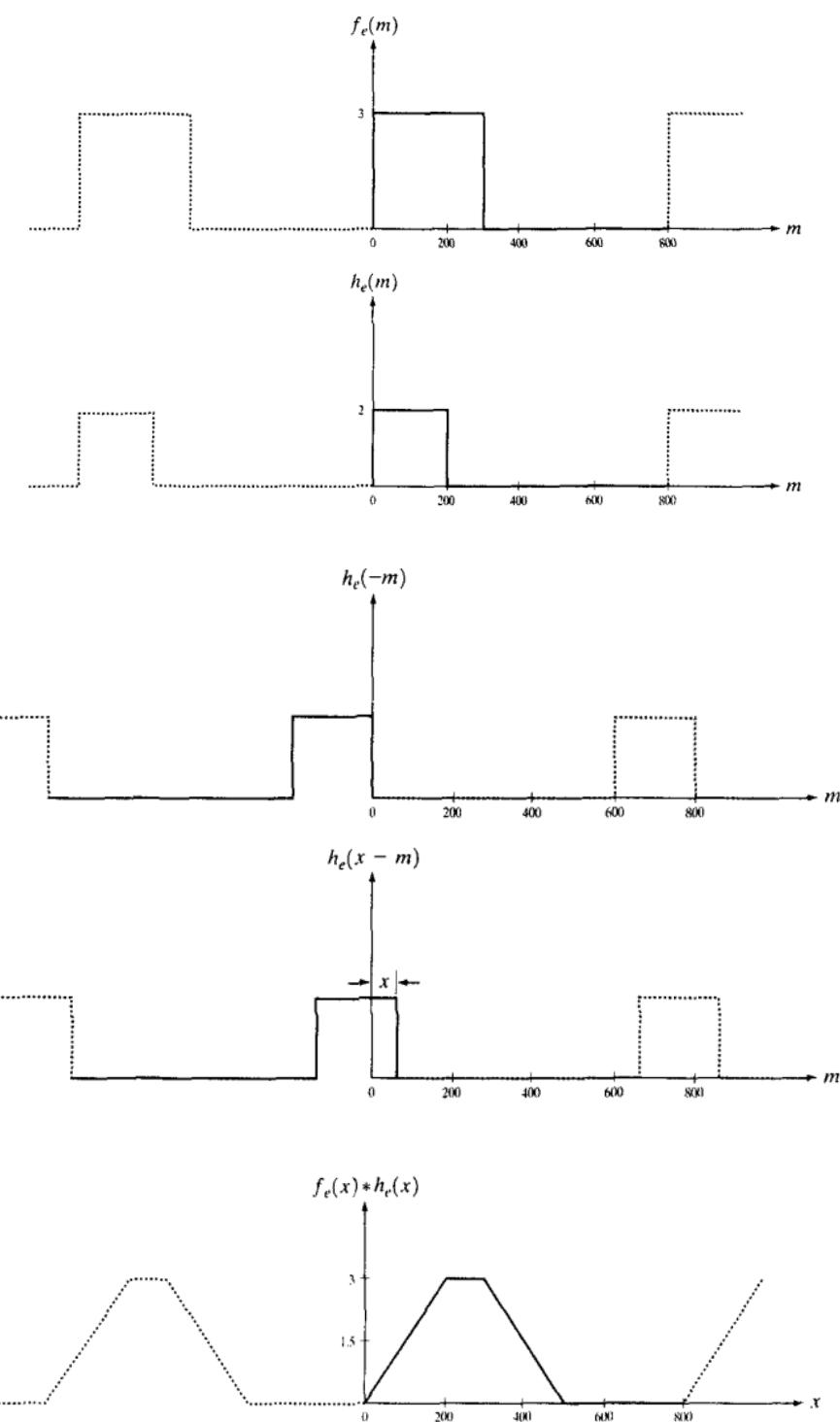
Ta có thể giải quyết vấn đề này bằng cách thêm vào các giá trị 0 cho cả hai hàm f và h . Giả sử f và h bao gồm A và B điểm, hàm sau khi thêm vào các giá trị 0:

$$f_e(x) = \begin{cases} f(x) & 0 \leq x \leq A - 1 \\ 0 & A \leq x \leq P \end{cases}$$

Và

$$h_e(x) = \begin{cases} h(x) & 0 \leq x \leq B - 1 \\ 0 & B \leq x \leq P \end{cases}$$

Với P phải thỏa điều kiện: $P \geq A + B - 1$ để các thành phần khác 0 của 2 chu kỳ kế nhau không tác động lẫn nhau:



Ta được kết quả của phép chập sau khi thêm các giá trị 0 không còn bị biến dạng.

Xét ảnh $f(x,y)$ và bộ lọc $h(x,y)$ có kích thước lần lượt là $A \times B$ và $C \times D$, thực hiện tương tự như trên bằng cách thêm padding là các giá trị 0 cho cả hai hàm. Hai hàm sau khi thêm có cùng kích thước, giả sử là $P \times Q$. Điều kiện ảnh sau lọc không bị biến dạng là:

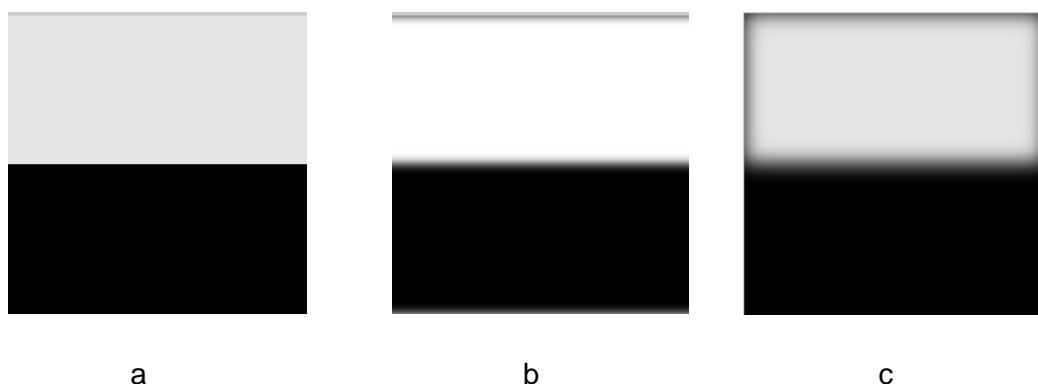
$$P \geq A + C - 1$$

Và

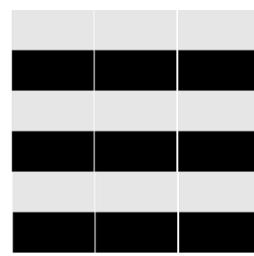
$$Q \geq B + D - 1$$

Do ta sử dụng các bộ lọc cùng kích thước với ảnh trong miền tần số nên $A=C$ và $B=D$

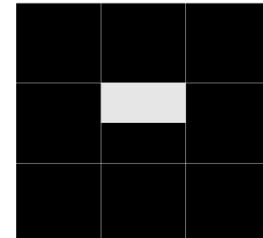
Ví dụ:



Hình a là ảnh gốc. Các hình b và c là ảnh sau lọc dùng bộ lọc thông thấp, như đã đề cập là bộ lọc làm mịn ảnh. Hình b sử dụng bộ lọc nhưng không có padding, ta thấy tác động của các thành phần hai chu kỳ cạnh nhau tác động lẫn nhau. Còn ở hình c, do có padding là các giá trị 0 nên ta có thể thấy kết quả là với vùng sáng bị làm mờ ở các phần cạnh và biên. Ảnh dưới cho ta thấy sự lặp lại của ảnh theo chu kỳ để giải thích rõ hơn cho hình b và c.



Không dùng padding



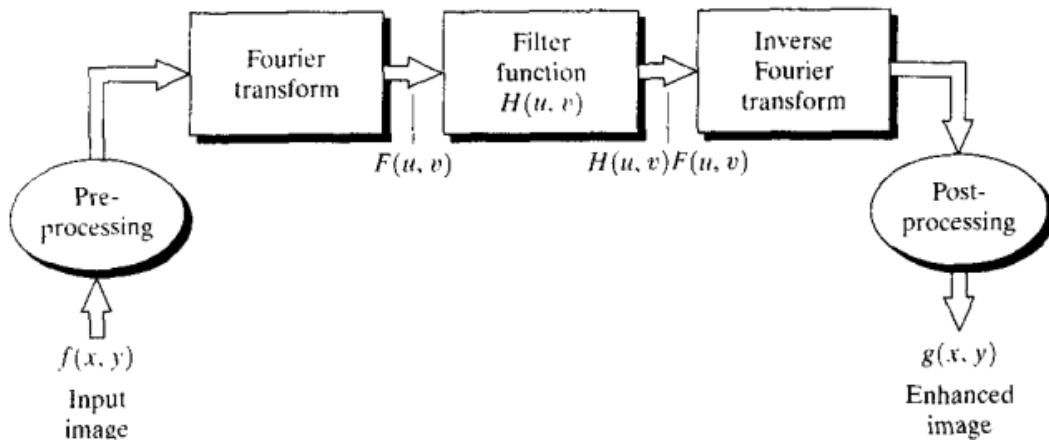
Sử dụng padding

4.2.2) Lọc thông thấp:

Quá trình lọc ảnh trong miền tần số có thể tóm tắt như sau:

- a) Nhân $f(x,y)$ với $(-1)^{x+y}$
- b) Biến đổi Fourier để xử lý ảnh ở miền tần số
- c) Nhân $F(u,v)$ với bộ lọc $H(u,v)$
- d) Tiến hành lấy Fourier ngược của kết quả (c)
- e) Lấy phần thực của kết quả (d)
- f) Nhân kết quả từ (e) với $(-1)^{x+y}$ cho ta ảnh sau lọc

Sơ đồ quá trình lọc ảnh trong miền tần số:



Ta đã biết bộ lọc thông thấp giúp làm mịn ảnh, tương đương với bộ lọc trung bình trong miền không gian.

Ta sẽ xét 3 loại bộ lọc thông thấp là bộ lọc lý tưởng, bộ lọc Butterworth và bộ lọc Gauss.

Bộ lọc thông thấp lý tưởng có hàm truyền đạt:

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

Với D_0 là một giá trị khác 0, gọi là ngưỡng cắt và $D(u, v)$ là khoảng cách từ điểm (u, v) đến tâm. Bộ lọc này không có trong thực tế, nhưng có thể mô phỏng bằng Matlab.

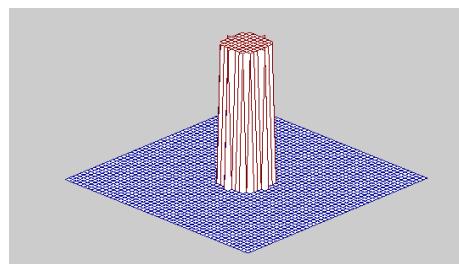
Bộ lọc Butterworth bậc n, với ngưỡng cắt D_0 , có dạng:

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

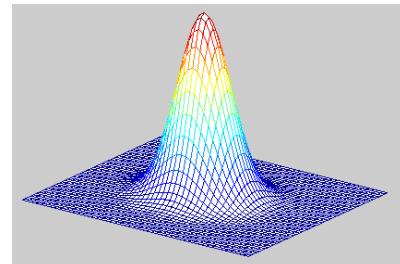
Bộ lọc Gauss có dạng:

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

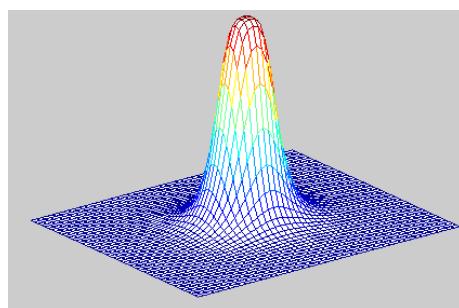
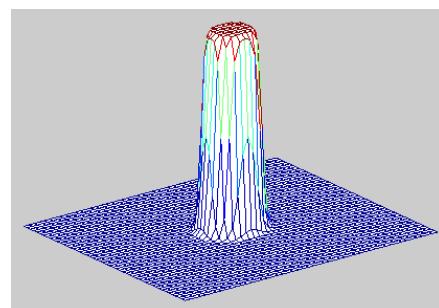
σ gọi là độ lệch chuẩn, nếu thay σ bằng D_o , ta có giá trị ngưỡng cắt là D_o .



Bộ lọc thông thấp lý tưởng



Bộ lọc thông thấp Gauss

Bộ lọc thông thấp Butterworth
bậc 2Bộ lọc thông thấp Butterworth
bậc 3

Trên là hình các bộ lọc có cùng kích thước 500×500 , ngưỡng cắt là $D_o=50$. Ta có một nhận xét là độ dốc của bộ lọc Gauss thấp nhất, tại ví trí ngưỡng cắt D_o biên độ giảm còn 60,7% so với giá trị lớn nhất là 1, bộ lọc Butterworth có thể xem là sự chuyển tiếp giữa bộ lọc lý tưởng và bộ lọc Gauss, với bậc thấp bộ lọc Butterworth có độ dốc gần giống bộ lọc Gauss, nhưng bậc càng cao thì lại càng dốc. Giá trị tại ngưỡng cắt bằng 50% giá trị lớn nhất.



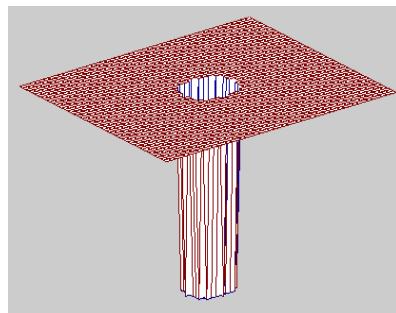
Ví dụ trên sử dụng bộ lọc Butterworth bậc 2 đối với các ảnh ở trên và bộ lọc Gauss với các hình ở dưới. Nhận xét:

- Mức cắt D₀ càng nhỏ, ảnh càng bị mờ, do bộ lọc thông thấp lọc các thành phần tần số thấp, tương ứng với các giá trị mức xám thay đổi chậm.
- Cùng một mức cắt D₀, bộ lọc Butterworth cho ảnh mờ hơn so với bộ lọc Gauss, lý do là bộ lọc Butterworth dốc hơn, chọn lọc tần số tốt hơn.

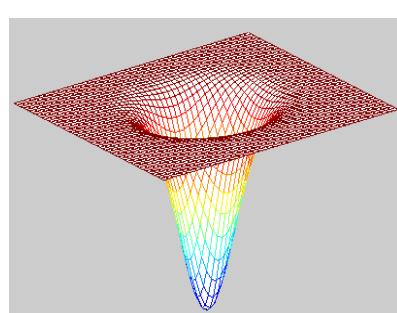
4.2.3) Lọc thông cao:

Bộ lọc thông cao có thể suy ra từ bộ lọc thông thấp qua biểu thức:

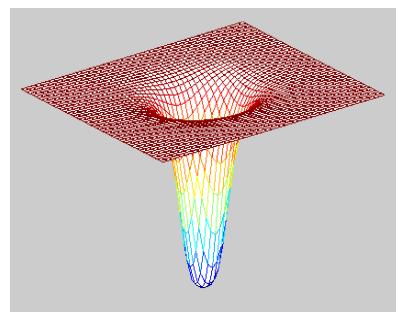
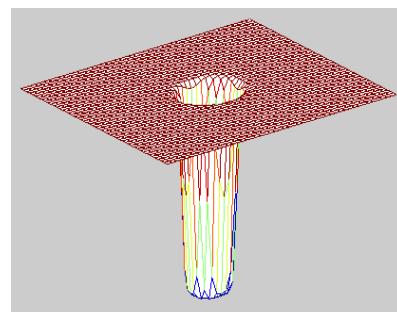
$$H_{hp}(u, v) = 1 - H_{lp}(u, v)$$



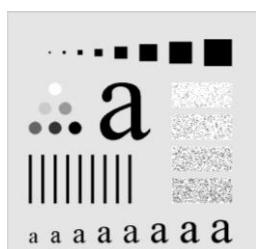
Bộ lọc thông cao lý



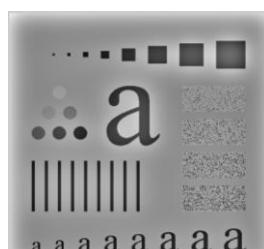
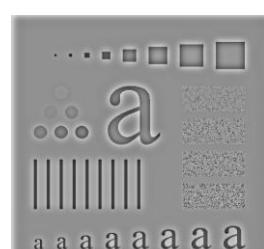
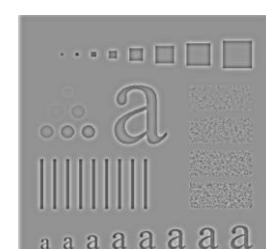
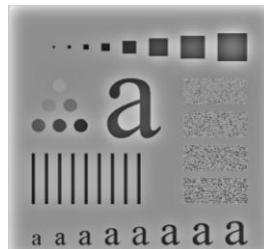
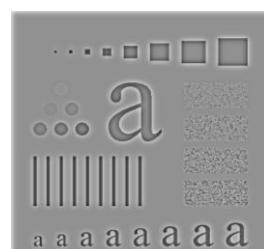
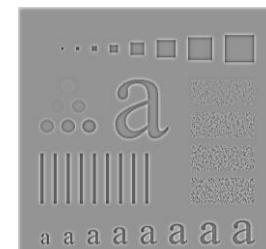
Bộ lọc thông cao

Bộ lọc thông cao
Butterworth bậc 2Bộ lọc thông cao
Butterworth bậc 9

Bộ lọc thông cao làm sắc nét ảnh và nổi bật các chi tiết như nhiễu và cạnh biên.



Ảnh

 $D_o=10$  $D_o=30$  $D_o=50$  $D_o=10$  $D_o=30$  $D_o=50$

Những hình ở trên sử dụng bộ lọc Butterworth bậc 2, những hình ở dưới sử dụng bộ lọc Gauss

Cũng tương tự như bộ lọc thông thấp, ta cũng có nhận xét sau:

- D_o càng lớn, ảnh sau xử lý càng sắc nét hơn, các chi tiết như cạnh biên và nhiễu càng được thể hiện rõ.
- Cùng một giá trị D_o , bộ lọc Butterworth tạo ảnh sắc nét hơn bộ lọc Gauss.
- Giá trị $F(0,0) = 0$ làm ảnh sau xử lý giảm cường độ mức xám, vấn đề này sẽ được khắc phục với bộ lọc High-Frequency Emphasis.

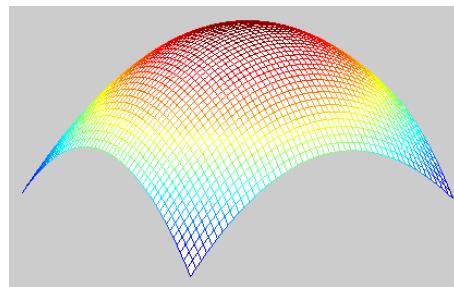
Biến đổi Laplace trong miền tần số:

$$\mathcal{L}\{\nabla^2[f(x,y)]\} = -(u^2 + v^2)F(u,v)$$

Với $\mathcal{L}\{\nabla^2[f(x,y)]\}$ là biến đổi Fourier của toán tử Laplace.

Do đó ta có bộ lọc Laplace trong miền tần số $H(u,v) = -(u^2 + v^2)$

Đáp ứng của bộ lọc có dạng:



Ta cũng có bộ lọc trực tiếp áp dụng toán tử Laplace

$$g(x) = f(x) - \nabla^2[f(x,y)]$$

Do H mang các giá trị âm nên ta thực hiện phép trừ trong miền không gian.

Ta sẽ được kết quả bộ lọc $H(u,v)$:

$$H(u,v) = 1 + (u^2 + v^2)$$

Kết quả của phép lọc Laplace trong miền tần số cũng tương tự trong miền không gian

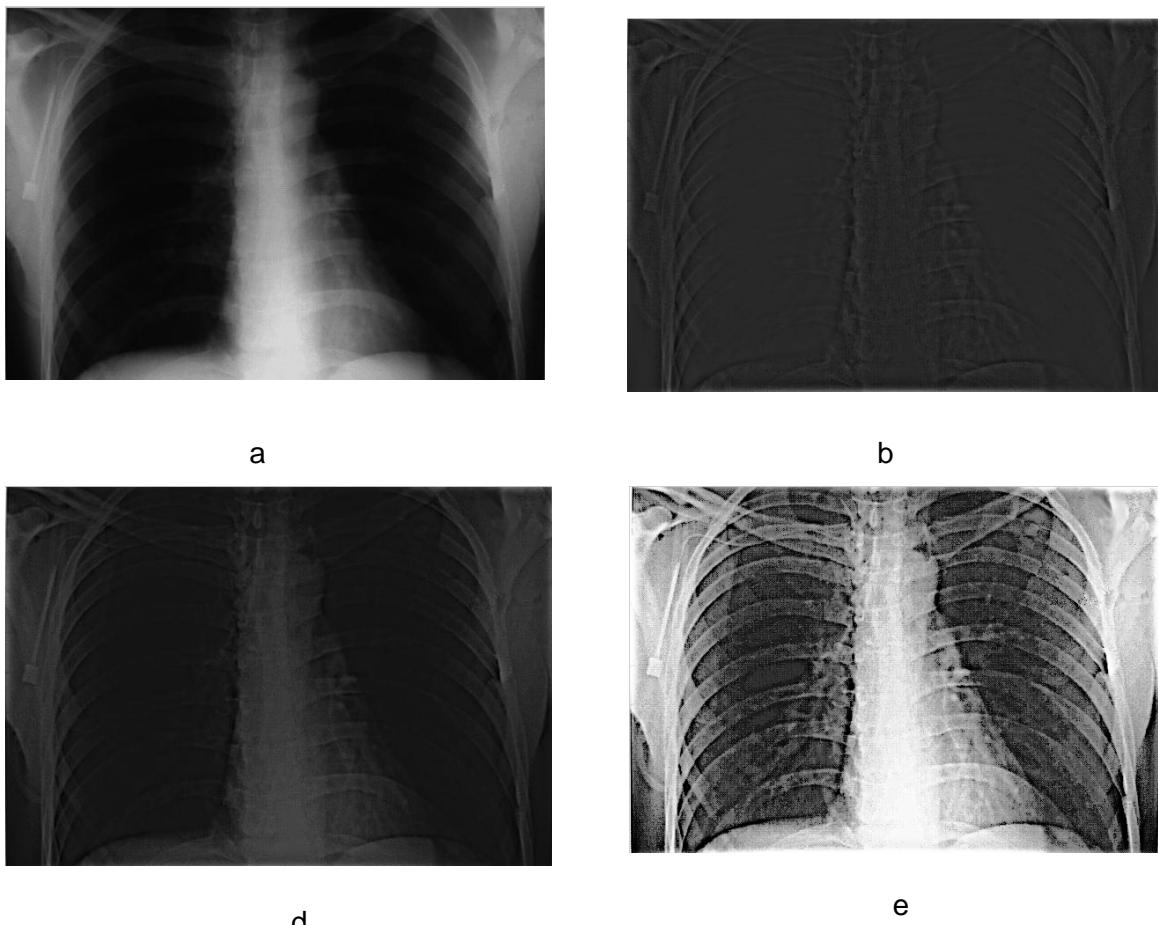
Bộ lọc High-frequency emphasis:

Các bộ lọc thông cao vừa được trình bày có một đặc điểm là giá trị $F(0,0)=0$, do đó làm cho ảnh sau xử lý có tổng các giá trị mức xám bằng 0. Một giải pháp là thêm vào bộ lọc thông cao mức offset lớn hơn 0. Nếu giá trị offset này kết hợp cùng với việc nhân các hệ số bộ lọc thông

cao với một giá trị >1 thì ta có bộ lọc High-frequency emphasis. Hệ số nhân này sẽ làm tăng nhanh biên độ của các giá trị tần số cao, trong khi các giá trị tần số thấp thay đổi rất ít. Do đó mức xám ảnh sau xử lý được tăng cường, và ảnh vẫn sắc nét.

$$H_{hf\theta}(u, v) = a + bH_{hp}(u, v)$$

Với a là mức offset, b là hệ số nhân, thường $0,25 \leq a \leq 0,5$ và $1,5 \leq b \leq 2,5$.



Hình a là ảnh chụp X-quang lồng ngực, ảnh bị mờ với thành phần mức xám tập trung gần giá trị 0(tối). Hình b là kết quả sau khi lọc bằng bộ lọc thông cao Butterworth bậc 2, có D_o nhỏ, ta thấy các chi tiết cạnh biên được làm nổi bật nhưng mức xám ảnh bị giảm xuống đáng kể. Ảnh c sử dụng bộ lọc High-Frequency Emphasis có $a=0,5$ và $b=2$, các chi tiết cạnh biên vẫn nổi bật và mức xám cũng tăng lên. Nhưng các giá trị mức xám vẫn tập trung chủ yếu ở vùng tối, ta thực hiện cân bằng histogram để cho ảnh có độ tương phản cao hơn, kết quả của cân bằng histogram là hình d. Đây là ứng dụng kết hợp nâng cao chất lượng ảnh trong miền tần số(lọc) và miền không gian(cân bằng histogram).

CHƯƠNG IV:

KHÔI PHỤC ẢNH

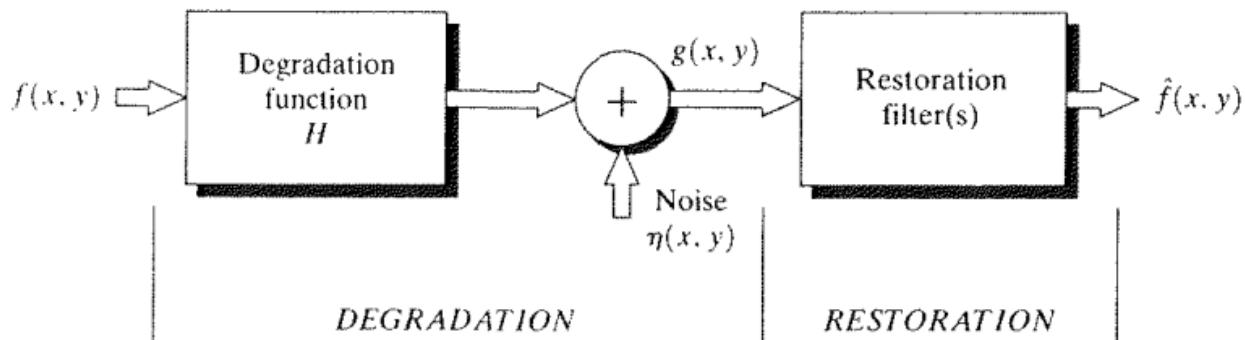
I) Giới thiệu:

Khôi phục ảnh tập trung vào việc loại bỏ hay giảm thiểu sự biến dạng xảy ra trong quá trình thu nhận ảnh. Sự biến dạng ảnh có thể bao gồm : Nhiều-là những sai khác trong giá trị của pixel, ảnh hưởng quang học : sự mờ do việc chuyển động của cameran... Ta có dạng tổng quát của ảnh bị biến dạng:

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

tương ứng trong miền tần số ta có :

$$G(i, j) = F(i, j).H(i, j) + N(i, j)$$



II) Nhiễu:

1) Nhiễu Salt and Pepper:

Còn gọi là nhiễu xung, nhiễu nhị phân.

$$p(z) = \begin{cases} P_a & \text{for } z = a \\ P_b & \text{for } z = b \\ 0 & \text{otherwise} \end{cases}$$

Nếu $b > a$, mức xám b sẽ xuất hiện tương ứng là điểm sáng trên ảnh còn mức xám a sẽ tương ứng với điểm đen xuất hiện trên ảnh.

Để cộng nhiễu “Salt and pepper” vào một ảnh ta dùng câu lệnh sau :

`t = imnoise(image,'salt & pepper')`

số lượng nhiễu được cộng vào mặc định là 10%. Ta có thể cung cấp thêm các thông số để thay đổi lượng nhiễu được cộng vào này.



(a) Original image



(b) With added salt & pepper noise

2) Nhiễu Gaussian:

Là một dạng lý tưởng của nhiễu trắng, được gây ra bởi những dao động ngẫu nhiên của tín hiệu. Nhiễu Gaussian là nhiễu trắng có phân bố chuẩn.

$$p(z) = \frac{1}{\sqrt{2\pi}\delta} e^{-(z-u)^2/2\delta^2}$$

Nếu ta có ảnh I, nhiễu Gaussian là N ta sẽ có ảnh nhiễu = I + N.

Để tạo ra ảnh với nhiễu Gaussian ta dùng câu lệnh sau :

t = imnoise(image, 'gaussian')

giá trị mặc định của kỳ vọng và phương sai của nhiễu là 0 và 0.01.



(a) Original image



(a) Gaussian noise

3) Nhiễu Speckle:

Có thể được mô hình bằng cách nhân các giá trị ngẫu nhiên với giá trị của các pixel. Nhiễu Speckle là vấn đề quan tâm chủ yếu trong các ứng dụng radar

Trong Matlab ảnh với nhiễu Speckle được tính toán : $I^*(1 + N)$

$$t = \text{imnoise}(t, 'speckle')$$

Nhiễu N có phân phối chuẩn với giá trị trung bình =0. Có thể cung cấp thêm thông số để xác định giá trị kỳ vọng của N, giá trị mặc định của nó là 0.04.



(a) Original image



(b) Speckle noise

1. Nhiễu tuần hoàn (Periodic noise)

Nếu tín hiệu hình ảnh là tín hiệu tuần hoàn, chúng ta có thể có ảnh bị ảnh hưởng bởi nhiễu tuần hoàn.



(a) Original image

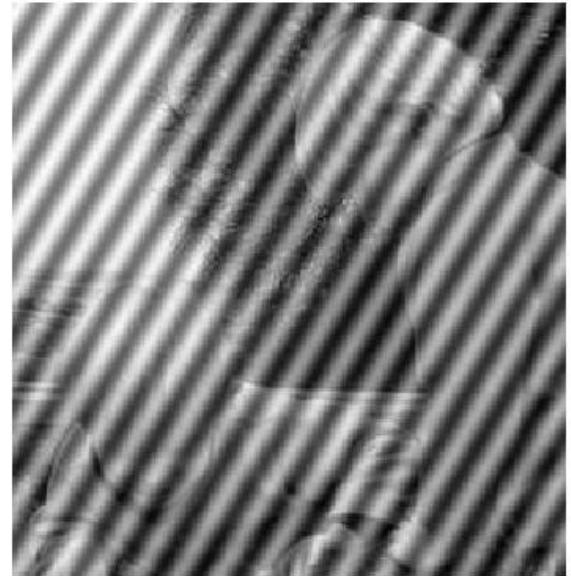


Figure 5.3: The twins image corrupted by periodic noise

Hàm **imnoise** không có tùy chọn để tạo ra nhiễu tuần hoàn. Ta có thể tạo ra một dạng đơn giản của nhiễu tuần hoàn bằng cách cộng vào ảnh một ma trận tuần hoàn.

II) Khôi phục ảnh với các bộ lọc trong miền không gian:

1) Bộ lọc trung bình số học (Arithmetic Mean filter):

$$\hat{f} = \frac{1}{mn} \sum_{(s,t) \in S_{xy}} g(s,t)$$

Giá trị của ảnh được khôi phục tại tọa độ (x,y) đơn giản là trung bình số học của những pixel trong miền S_{xy} .

Bộ lọc trên được thực hiện trong IPT như sau :

```
w = fspecial('average',[m,n])
f = imfilter(g,w)
```

2) Bộ lọc trung bình hình học (Geometric Mean filter):

$$\hat{f} = \left[\prod_{(s,t) \in S_{xy}} g(s,t) \right]^{\frac{1}{mn}}$$

Mỗi giá trị pixel của ảnh phục hồi : là tích của những pixel trong miền S_{xy} , sau đó lấy lũy thừa $1/mn$.

IPT không hỗ trợ hàm để tính toán trực tiếp bộ lọc này.

3) Bộ lọc trị số trung bình (Median filter):

$$\hat{f}(x, y) = \underset{(s,t) \in S_{xy}}{\text{median}} g(s, t)$$

Bộ lọc thay thế giá trị của một pixel bởi trị số trung bình của những giá trị mức xám trong miền lân cận của pixel này được xác định bởi S_{xy} .

Trong IPT bộ lọc được thực hiện bởi hàm **medfilt2** :

$f = \text{medfilt2}(g, [m, n])$

4) Bộ lọc MIN & MAX:

✓ Bộ lọc Max:

Bộ lọc này hữu dụng trong việc xác định điểm sáng nhất trong ảnh. Vì nhiều pepper có giá trị rất thấp nên nhiễu này sẽ bị loại trừ như là kết quả của quá trình lựa chọn trị max trong miền xác định bởi S_{xy} .

Được thực hiện thông qua hàm **ordfilt2**:

$f = \text{ordfilt2}(g, m*n, \text{ones}(m, n))$

✓ Bộ lọc Min:

Bộ lọc này hữu dụng trong việc xác định điểm tối nhât trong ảnh. Do đó nó sẽ loại trừ nhiễu Salt như là kết quả của quá trình lựa chọn mức tối trong miền xác định bởi S_{xy} .

$f = \text{ordfilt2}(g, 1, \text{ones}(m, n))$.

5) Bộ lọc trung bình hài (Harmonic Mean filter):

$$\hat{f}(x, y) = \frac{mn}{\sum_{(s,t) \in S_{xy}} \frac{1}{g(s, t)}}.$$

Bộ lọc này làm việc tốt với nhiễu Salt, nhưng lại không hiệu quả với nhiễu Pepper.

6) Bộ lọc điểm giữa (Midpoint filter):

$$\hat{f}(x, y) = \frac{1}{2} \left[\max_{(s,t) \in S_{xy}} \{g(s, t)\} + \min_{(s,t) \in S_{xy}} \{g(s, t)\} \right].$$

Bộ lọc tính điểm trung bình giữa giá trị Max và giá trị Min trong vùng bao quanh bởi S_{xy} . Bộ lọc làm việc tốt với những nhiễu có phân phối ngẫu nhiên như nhiễu Gaussian.

III) Giảm nhiễu tuần hoàn với các bộ lọc trong miền tần số:

1) Bộ lọc chấn dải:

Bộ lọc chấn dải loại bỏ hay làm suy hao một dải băng tần trong biến đổi Fourier ban đầu.

✓ Bộ lọc chấn dải lý tưởng được biểu diễn :

$$H(u, v) = \begin{cases} 1 & \text{if } D(u, v) < D_0 - \frac{W}{2} \\ 0 & \text{if } D_0 - \frac{W}{2} \leq D(u, v) \leq D_0 + \frac{W}{2} \\ 1 & \text{if } D(u, v) > D_0 + \frac{W}{2} \end{cases}$$

✓ Bộ lọc chấn dải Butterworth

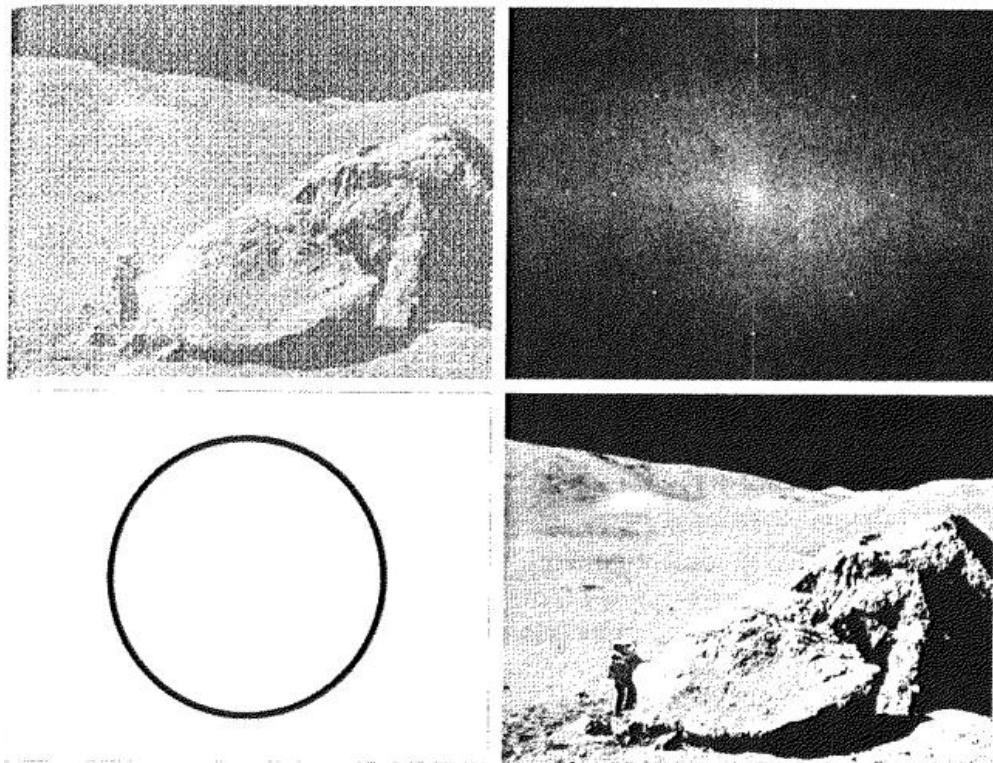
Bộ lọc chấn dải butterworth bậc n được biểu diễn như sau :

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

✓ Bộ lọc chấn dải Gaussian

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

Bộ lọc chấn dải loại bỏ nhiễu trong những ứng dụng mà ta đã biết trước khoảng tần số của những thành phần nhiễu. Ví dụ như một ảnh bị ảnh hưởng của nhiễu tuần hoàn – xem tương đương như là hàm sine của hàm 2 biến.



a
b
c
d

FIGURE 5.16
(a) Image corrupted by sinusoidal noise.
(b) Spectrum of (a).
(c) Butterworth bandreject filter (white represents 1). (d) Result of filtering. (Original image courtesy of NASA.)

Ta thấy ở hình b-phổ Fourier của ảnh nhiễu, các thành phần nhiễu xấp xỉ nằm trên một đường tròn. Do đó một bộ lọc chấn dải đối xứng xuyên tâm là lựa chọn tối ưu. Hình c biểu diễn bộ lọc butterworth bậc 4, với bán kính và độ dày thích hợp để có thể bao quanh hoàn toàn các thành phần nhiễu.

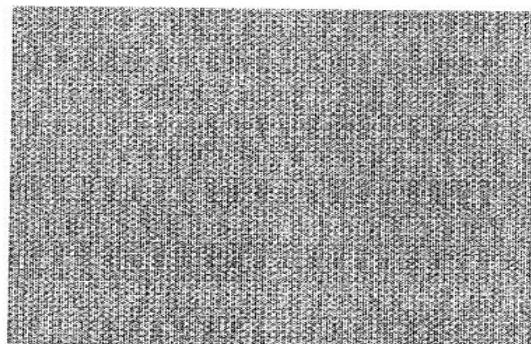
2) Bộ lọc thông dài:

Bộ lọc thông dải có hoạt động ngược lại với bộ lọc chấn dải. Hàm truyền của bộ lọc thông dải có thể suy ra từ bộ lọc chấn dải :

$$H_{bp}(u, v) = 1 - H_{br}(u, v)$$

Bộ lọc thông dải loại bỏ nhiều chi tiết của ảnh. Tuy nhiên bộ lọc thông dải khá hữu dụng trong việc tách ảnh hưởng của một dải tần số lên ảnh.

FIGURE 5.17
Noise pattern of
the image in
Fig. 5.16(a)
obtained by
bandpass filtering.



ảnh trên được tạo ra như sau :

- ✓ Tính đáp ứng của bộ lọc thông dải từ bộ lọc chấn dải
- ✓ Biến đổi ngược của biến đổi bộ lọc thông dải

Ta thấy hầu hết chi tiết của ảnh đã bị mất, nhưng những thông tin còn lại rất hữu dụng. Đó chính là mô hình nhiễu-giống với nhiều tác động trong ảnh ở hình a.

3) Bộ lọc Notch:

Bộ lọc Notch loại bỏ hay cho qua những tần số lân cận xác định trước quanh tần số trung tâm.

- ✓ Bộ lọc Notch chấn dải lý tưởng

$$H(u, v) = \begin{cases} 0 & \text{if } D_1(u, v) \leq D_0 \quad \text{or} \quad D_2(u, v) \leq D_0 \\ 1 & \text{otherwise} \end{cases}$$

Trong đó :

$$D_1(u, v) = [(u - M/2 - u_0)^2 + (v - N/2 - v_0)^2]^{1/2}$$

$$D_2(u, v) = [(u - M/2 + u_0)^2 + (v - N/2 + v_0)^2]^{1/2}$$

- ✓ Bộ lọc Notch chấn dải Butterworth

$$H(u, v) = \frac{1}{1 + \left[\frac{D_0^2}{D_1(u, v)D_2(u, v)} \right]^n}$$

- ✓ Bộ lọc Notch chấn dải Gaussian

$$H(u, v) = 1 - e^{-\frac{1}{2} \left[\frac{D_1(u, v)D_2(u, v)}{D_0^2} \right]}$$

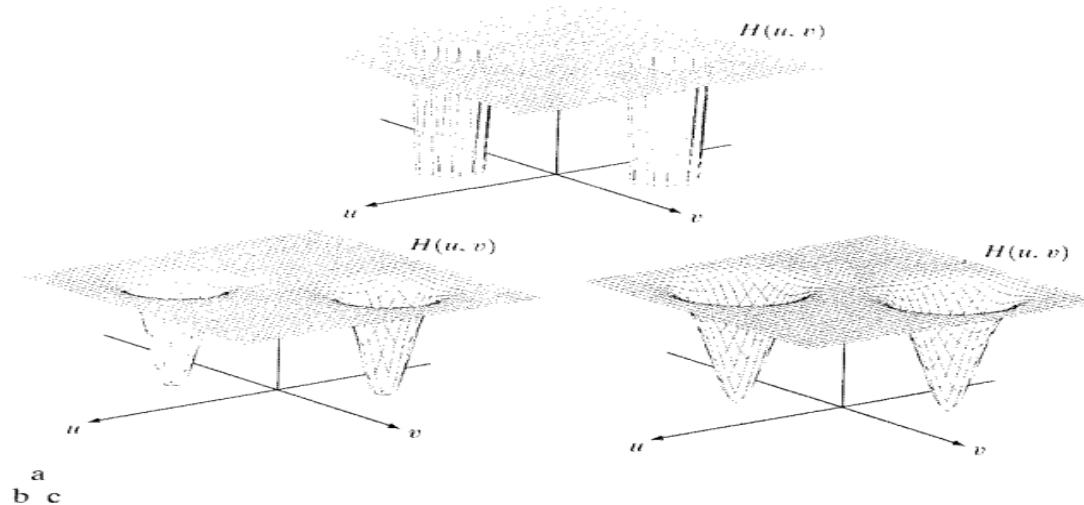
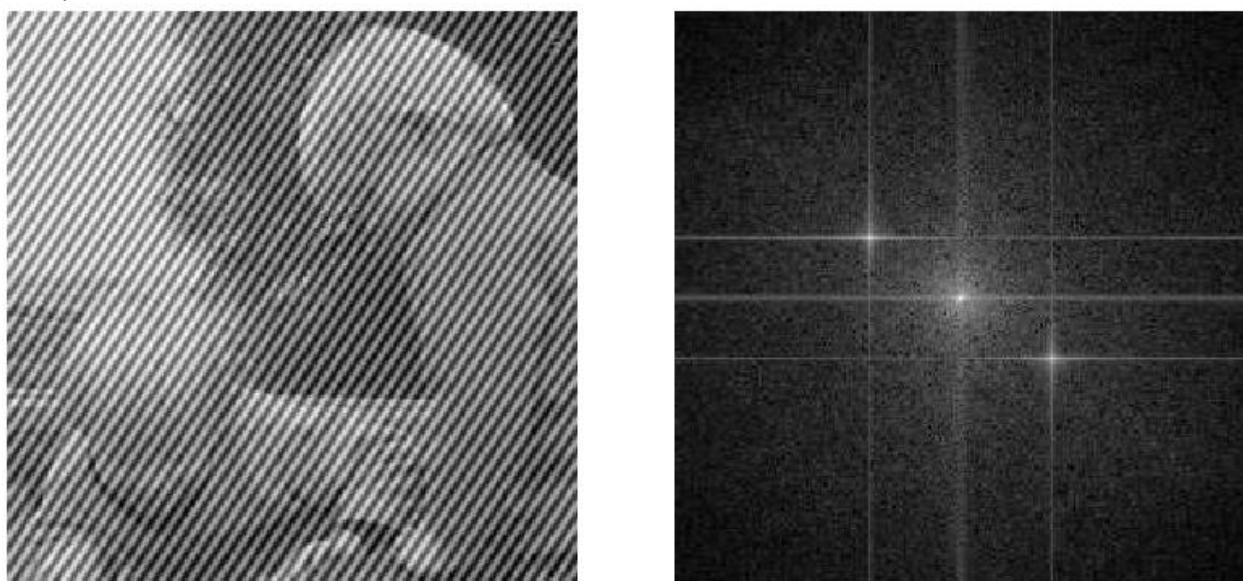


FIGURE 5.18 Perspective plots of (a) ideal, (b) Butterworth (of order 2), and (c) Gaussian notch (reject) filters.

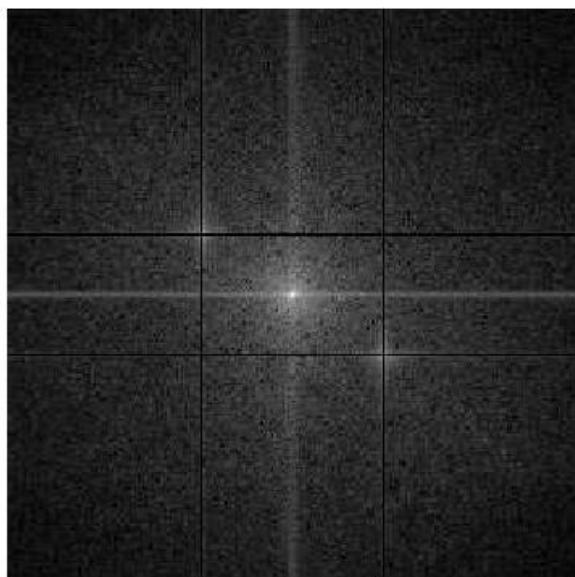
Ví dụ:



Đơn giản là cho hàng và cột của thành phần phẳng nhiễu bằng 0. Giả sử tọa độ của các thành phần nhiễu này lần lượt là (156,170), (102,88).

```
>> tf(156,:)=0;
>> tf(102,:)=0;
>> tf(:,170)=0;
>> tf(:,88)=0;
```

Kết quả :



(a) A notch filter



(b) After inversion

Nhiều nhiễu ở trung tâm đã bị loại bỏ. Tạo nhiều hàng và nhiều cột dịch chuyển về zero sẽ loại bỏ được nhiều nhiễu hơn.

- ✓ Bộ lọc Notch thông dải

Có hoạt động ngược lại với hoạt động của bộ lọc Notch chấn dải. Ta dễ dàng suy ra hàm truyền của bộ lọc Notch thông dải :

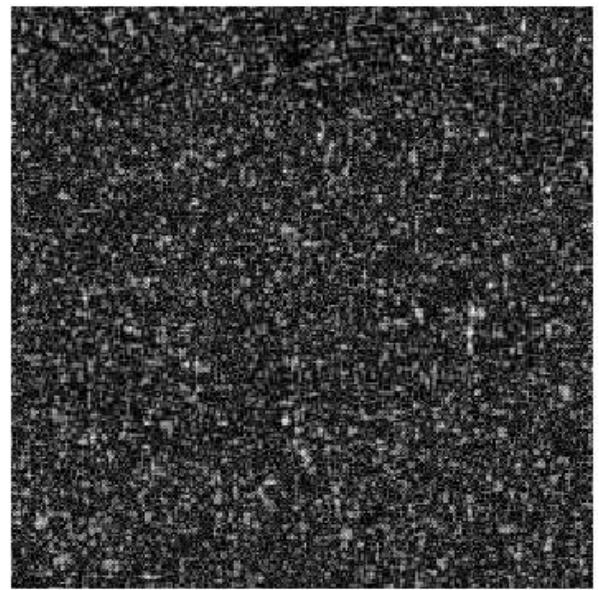
$$H_{np}(u, v) = 1 - H_{nr}(u, v)$$

IV) Bộ lọc ngược:

Ta có : $\mathbf{Y(i,j)} = \mathbf{X(i,j)} * \mathbf{F(i,j)}$

Từ đó ta có thể khôi phục DFT của ảnh ban đầu : $\mathbf{X(i,j)} = \mathbf{Y(i,j)} / \mathbf{F(i,j)}$

Tuy nhiên một số thành phần của bộ lọc rất nhỏ, nên phép chia sẽ tạo ra giá trị rất lớn-lần át, quyết định giá trị ngõ ra. Nên ta sẽ khó thu được kết quả ảnh gốc chấp nhận được.



Có thể giải quyết vấn đề trên như sau:

- ✓ Áp một bộ lọc thông thấp vào phép chia :

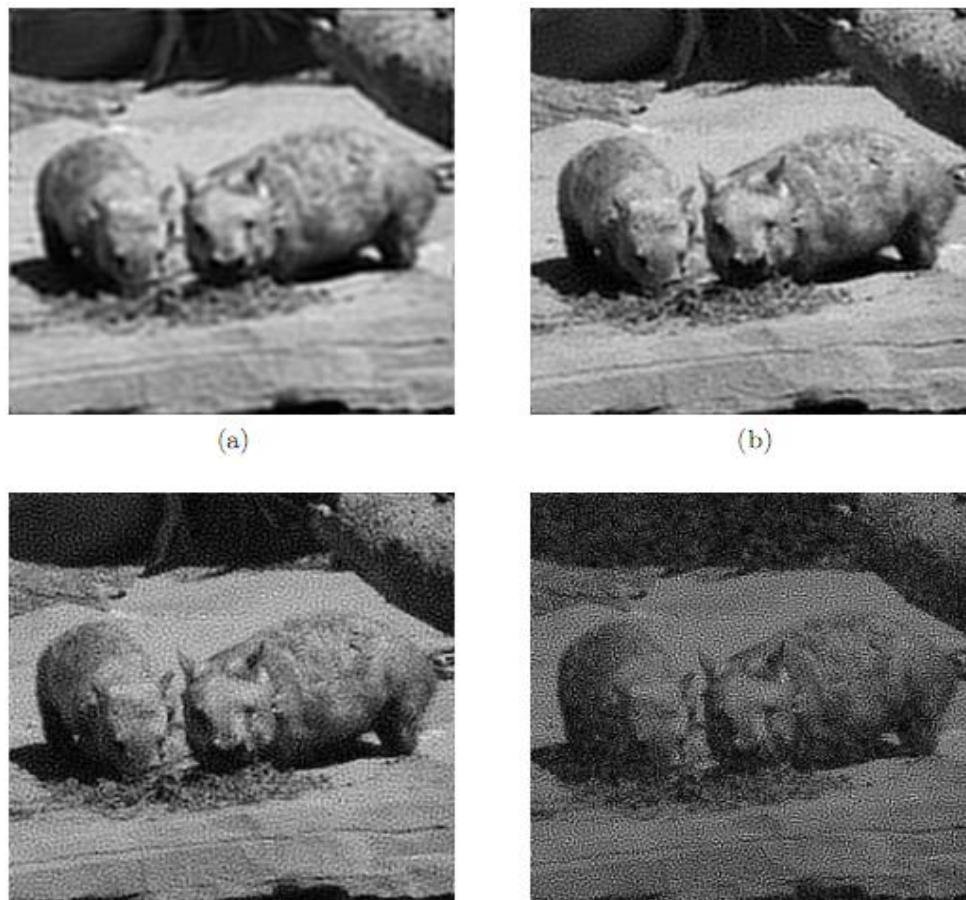
$$X(i, j) = \frac{Y(i, j)}{F(i, j)} L(i, j).$$

Sẽ loại bỏ những giá trị rất nhỏ của bộ lọc $F(i, j)$

- ✓ Chọn một ngưỡng giá trị d , nếu $|F(i, j)| < d$ chúng ta sẽ không thực hiện phép chia mà giữ giá trị ban đầu.

$$X(i, j) = \begin{cases} \frac{Y(i, j)}{F(i, j)} & \text{if } |F(i, j)| \geq d, \\ Y(i, j) & \text{if } |F(i, j)| < d. \end{cases}$$

Ví dụ :

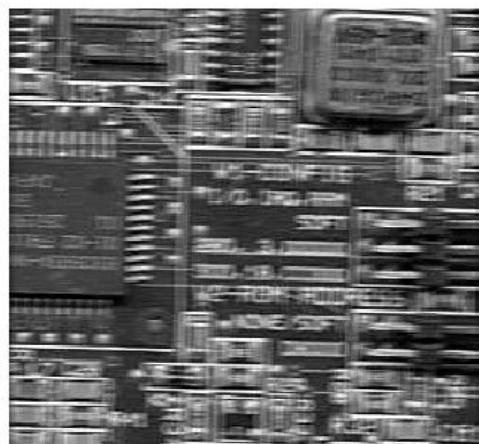
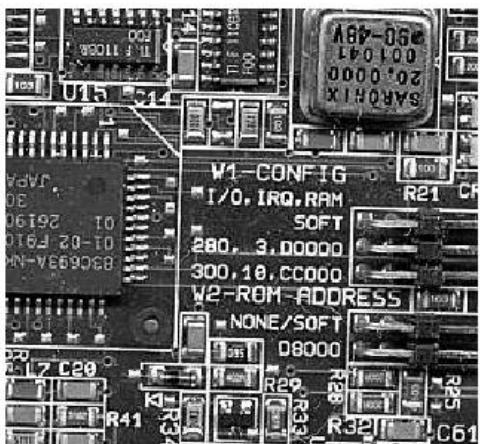


Một ứng dụng khác của bộ lọc ngược : làm rõ ảnh bị mờ do chuyển động.

Ví dụ :

```
>> bc=imread('board.tif');
>> bg=im2uint8(rgb2gray(bc));
>> b=bg(100:355,50:305);
>> imshow(b)
>> m=fspecial('motion',7,0);
>> bm=imfilter(b,m);
>> imshow(bm)
```

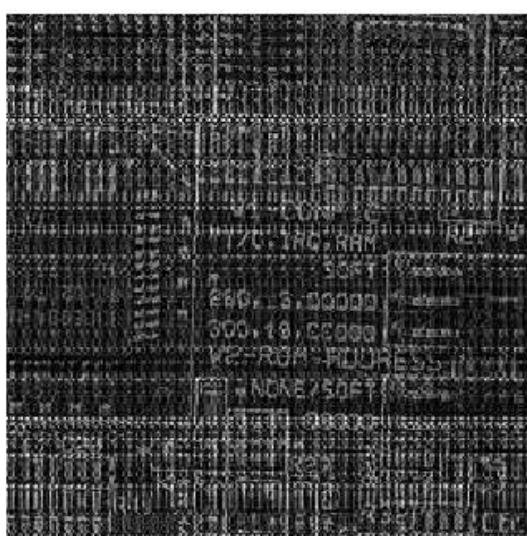
Cho ta kết quả sau :



Để làm rõ ảnh ta cần thực hiện phép chia giữa DFT của ảnh bị làm mờ cho DFT của bộ lọc làm mờ ảnh. Có nghĩa là trước tiên ta cần tạo ma trận tương đương với sự biến đổi làm mờ ảnh.

```
>> m2=zeros(256,256);
>> m2(1,1:7)=m;
>> mf=fft2(m2);
>> bmi=ifft2(fft2(bm)./mf);
>> fftshow(bmi,'abs')
```

Kết quả như sau:

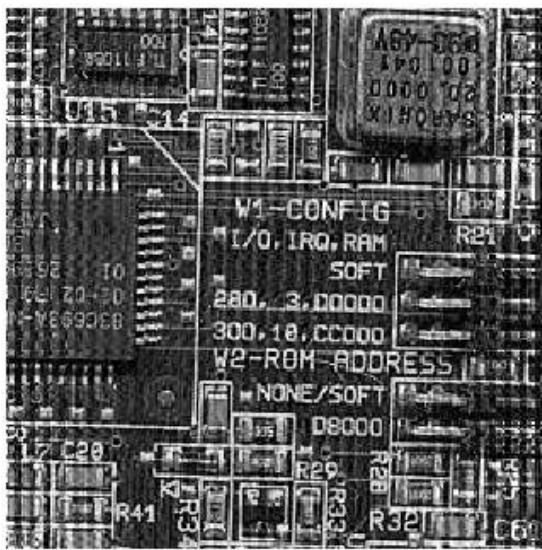


Kết quả không được tốt do đặc tính của bộ lọc ngược.

Theo phương pháp thứ 2 để khắc phục đặc tính của bộ lọc ngược ta định nghĩa một giá trị ngưỡng **d=0.02**

```
>> mf=fft2(m2);mf(find(abs(mf)<d))=1;
>> bmi=ifft2(fft2(bm)./mf);
>> imshow(mat2gray(abs(bmi))*2)
```

Ta có kết quả :



V) Bộ lọc Wiener:

Như trên, ta thấy bộ lọc ngược cho kết quả không được tốt. Kết quả sẽ tệ hơn khi ảnh ban đầu bị tác động bởi nhiễu.

$$Y(i, j) = X(i, j)F(i, j) + N(i, j)$$

Nên :

$$X(i, j) = \frac{Y(i, j) - N(i, j)}{F(i, j)}$$

Do đó không những có vấn đề trong phép chia mà còn vấn đề với nhiễu : nhiễu có thể lẩn át, quyết định giá trị ngõ ra, làm cho việc sử dụng bộ lọc ngược trực tiếp là không thể.

Gọi **M** : ảnh ban đầu, **R** : ảnh khôi phục.

Điều mong muốn là **R** càng gần với **M** càng tốt. Để xét sự chênh lệch gần nhất giữa **R**, **M** ta sét hàm :

$$\sum (m_{i,j} - r_{i,j})^2$$

Nếu ta có thể tối thiểu hóa giá trị của tổng trên, ta có thể chắc chắn rằng ta sẽ thu được kết quả tốt nhất có thể.

Bộ lọc có đặc điểm của hàm tính “bình phương tối thiểu” gọi là bộ lọc Weiner.

$$X(i, j) \approx \left[\frac{1}{F(i, j)} \frac{|F(i, j)|^2}{|F(i, j)|^2 + K} \right] Y(i, j)$$

Trong đó K là hằng số. K được dùng để xấp xỉ nhiễu. Nếu phương sai của nhiễu được biết trước thì $K = 2\delta^2$.

```
>> K=0.01;
>> wbf=fftshift(fft2(wba));
>> w1=wbf.*(abs(b).^2./abs(b).^2+K)./b
>> w1a=abs(ifft2(w1));
>> imshow(mat2gray(w1a))
```

$$K = 0.001$$



(a)



(b)



$$K = 0.0001$$



$$K = 0.00001$$

CHƯƠNG V: TÁCH BIÊN ẢNH

I) Cơ sở lý thuyết tách biên:

Tách biên là phương pháp thông dụng nhất để tách theo nghĩa gián đoạn trong các giá trị cường độ. Sự gián đoạn được tách sử dụng đạo hàm bậc nhất và bậc hai. Đạo hàm bậc nhất lựa chọn trong xử lý ảnh I gradient (độ dốc).

$$\text{Gradient của hm 2-D } f(x, y) \text{ được định nghĩa dưới dạng vector: } \nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

Biên độ của vectơ này:

$$\nabla f = mag(\nabla f) = \sqrt{G_x^2 + G_y^2}$$

Để tính toán đơn giản, con số này được xấp xỉ bằng cách sử dụng giá trị tuyệt đối : $\nabla f \approx |G_x| + |G_y|$ chúng bằng 0 trong các vùng có cường độ không đổi, và giá trị chúng tỷ lệ với bậc của sự thay đổi cường độ trong vùng có các giá trị pixel biến thiên. Nó được xem là biên độ của gradient hoặc xấp xỉ đơn giản của nó dưới dạng “gradient”.

Đặc tính cơ bản của vectơ gradient là tồn tại hướng có tỷ lệ thay đổi hàm f tại tọa độ (x, y) lớn nhất. Góc xảy ra tỷ lệ thay đổi lớn nhất là: $\alpha(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$

Đạo hàm bậc hai trong xử lý ảnh được tính sử dụng toán tử Laplace :

$$\nabla^2 f(x, y) = \frac{\partial^2 f(x, y)}{\partial x^2} + \frac{\partial^2 f(x, y)}{\partial y^2}$$

Toán tử Laplace ít khi được dùng để tách biên vì, vi

phân bậc hai, dễ bị ảnh hưởng bởi nhiễu, biên độ của nó sinh ra các biên kép, và không thể tách hướng biên.

Ý tưởng cơ bản đầu tiên sau tách biên là tìm các nơi trong ảnh có cường độ thay đổi nhanh, sử dụng một trong hai tiêu chuẩn tổng quát sau:

- Tìm các nơi đạo hàm bậc nhất của cường độ sáng có biên độ hơn một ngưỡng.

- Tìm các nơi đạo hàm bậc hai của cường độ sáng có sự thay đổi qua mức 0

II) Tách biên ảnh trong Matlab:

Cấu trúc tổng quát của hàm này là

$$[g, t] = \text{edge}(f, 'method', \text{parameters})$$

Trong đó ‘method’ gồm : Sobel, Prewitt, Roberts, LoG, Zero Crossing, Canny.

1) Bộ tách biên Sobel:

Bộ tách biên Sobel sử dụng các mặt nạ trong hình dưới để xấp xỉ đạo hàm bậc nhất G_x và G_y . Nói cách khác, gradient tại điểm tâm trong một lân cận được tính theo bộ tách Sobel:

$$g = \left[G_x^2 + G_y^2 \right]^{1/2} = \{ [(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)]^2 + [(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)]^2 \}^{1/2}$$

Khi đó, ta nói rằng vị trí (x,y) là pixel biên nếu $g \geq T$ tại vị trí đó, trong đó T là một ngưỡng được chỉ định.

Mặt nạ của bộ lọc Sobel :

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

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

Sobel

$$G_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$G_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

Cú pháp gọi bộ tách Sobel tổng quát là

$$[g, t] = \text{edge}(f, 'sobel', T, \text{dir})$$

$$g = \text{edge}(f), \text{ hoặc là } [g, t] = \text{edge}(f).$$

2) Bộ tách biên Prewitt:

Bộ tách biên Prewitt sử dụng mặt nạ:

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Prewitt

$$G_x = (z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)$$

$$G_y = (z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)$$

để xấp xỉ theo phương pháp số đạo hàm bậc nhất G_x và G_y

Cú pháp gọi tổng quát là:

[g, t] = edge(f, 'prewitt', T, dir)

Bộ tách Prewitt hơi đơn giản hơn để hiện thực bằng máy tính so với bộ tách Sobel, nhưng nó có khuynh hướng sinh ra một chút nhiễu. (Nó có thể được thể hiện qua hệ số 2 trong bộ tách biên Sobel).

3) Bộ tách biên Roberts:

Bộ tách biên Roberts sử dụng mặt nạ :

-1	0
0	1

0	-1
1	0

Roberts

$$G_x = z_9 - z_5$$

$$G_y = z_8 - z_6$$

để xấp xỉ theo phương pháp số đạo hàm bậc nhất G_x và G_y .

Cú pháp gọi tổng quát là:

[g , t] = edge(f, 'roberts', T, dir)

Bộ tách Roberts là một trong những bộ tách biên xưa nhất trong xử lý ảnh số v nó cũng đơn giản nhất. Bộ tách biên này được dùng ít hơn đáng kể các bộ tách khác do chức năng giới hạn của nó (nó không đối xứng và không thể được tổng quát hóa để tách biên là thừa số của 45^0). Tuy nhiên, nó vẫn được dùng thường xuyên trong hiện thực phần cứng khi tính đơn giản và tốc độ là các yếu tố chi phối.

4) Bộ tách biên Laplace của hàm Gauss (LoG):

Xét hàm Gauss

$$h(r) = -e^{-\frac{r^2}{2\sigma^2}}$$

Trong đó $r^2 = x^2 + y^2$ và σ là độ lệch chuẩn. Đây là hàm tròn, nếu nó chập với một ảnh, sẽ làm mờ ảnh. Độ mờ được xác định bởi giá trị σ .

Toán tử Laplace của hàm này (đạo hàm bậc 2 theo r):

$$\nabla^2 h(r) = - \left[\frac{r^2 - \sigma^2}{\sigma^4} \right] e^{-\frac{r^2}{2\sigma^2}} \quad (\text{Được gọi là hm Laplace của Gaussian LoG})$$

Vì đạo hàm bậc hai là toán tử tuyến tính, chập (lọc) với một ảnh bằng $\nabla^2 h(r)$ giống như đầu tiên chập ảnh với hàm tròn và sau đó tính kết quả của toán tử Laplace. Chúng ta chập ảnh bằng $\nabla^2 h(r)$ biết nó có 2 tác động: nó làm mịn ảnh (do đó giảm nhiễu) và nó tính toán tử Laplace, làm cong một ảnh biên kép. Định vị các biên sau đó tìm các điểm giao zero giữa các biên kép. Cú pháp gọi tổng quát là:

[g , t] = edge(f, 'log', T, sigma)

Trong đó sigma là độ lệch chuẩn, giá trị mặc định của sigma là 2.

5) Bộ tách biên điểm giao zero:

Bộ tách biên này dựa trên khái niệm giống phương pháp LoG, nhưng phép chập được thực hiện sử dụng hàm lọc được chỉ định H. Cú pháp gọi hàm :

[g , t] = edge(f, 'zerocross', T, H)

6) Bộ tách biên Canny:

Là bộ tách biên mạnh nhất cung cấp bởi hàm **edge**. Có thể tóm tắt phương pháp này như sau:

1. Ảnh được làm tròn sử dụng một bộ lọc Gauss với độ lệch chuẩn σ , để giảm nhiễu
2. Gradient cục bộ, $g(x, y) = \sqrt{G_x^2 + G_y^2}$ và hướng biên $\alpha(x, y) = \tan^{-1}(\frac{G_y}{G_x})$ được tính toán tại mỗi điểm. Một điểm biên được định nghĩa là điểm có độ dài là cực đại địa phương theo hướng của gradient.
3. Điểm biên được xác định tăng lên đến các đỉnh trong gradient biên độ ảnh. Sau đó thuật toán tìm đỉnh của các đỉnh này và đặt giá trị 0 vào tất cả pixel không thật sự nằm trên đỉnh vì vậy tạo ra một đường mỏng ở ngõ ra, một quá trình được biết là *sobel lại không cực đại*. Các pixel đỉnh được đặt ngưỡng dùng hai ngưỡng, T_1 và T_2 . Các pixel đỉnh lớn hơn T_2 được gọi là các pixel biên “mạnh”. Các pixel đỉnh nằm giữa T_1 và T_2 được gọi là các pixel biên “yếu”.
4. Cuối cùng, thuật toán thực hiện biên kết nối bằng cách kết hợp các pixel yếu mà có dạng kết nối-8 với các pixel mạnh.

Cú pháp bộ tách biên Canny là:

`[g, t] = edge(f, 'canny', T, sigma)`

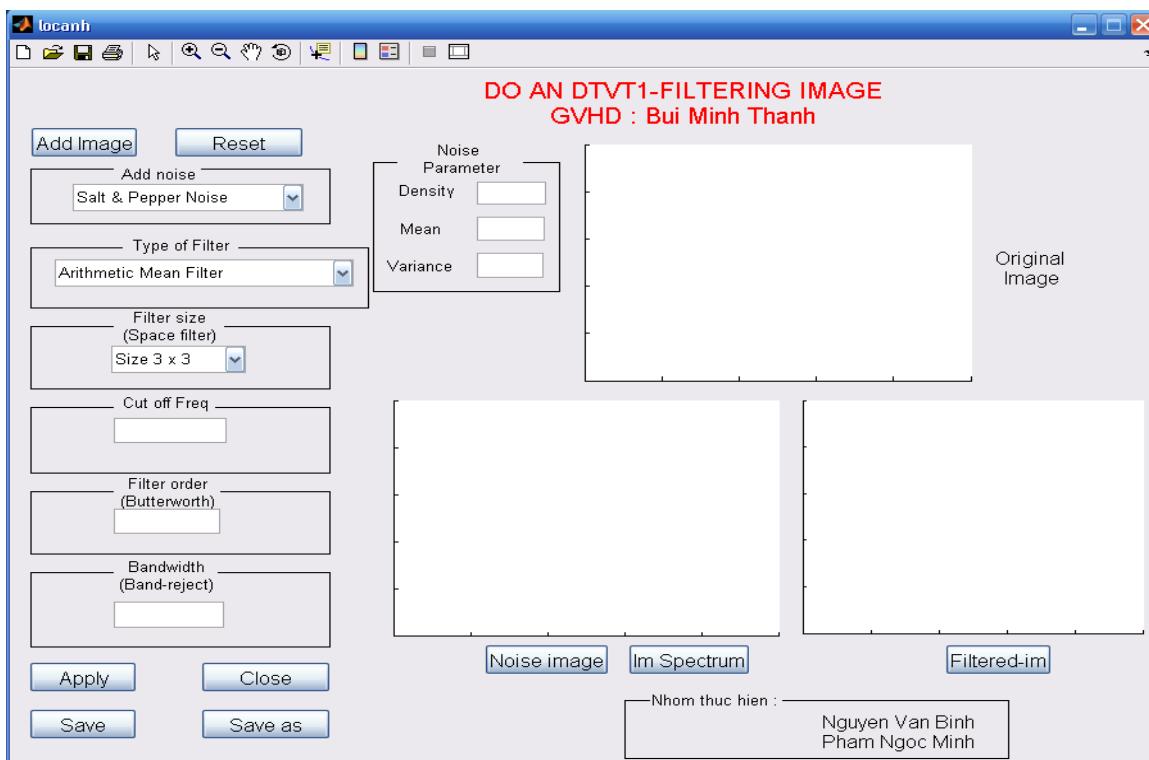
Trong đó T là một vector, $T = [T_1 \quad T_2]$ là 2 ngưỡng được giải thích trong bước 3 của thủ tục trước và sigma là độ lệch chuẩn của bộ lọc làm tròn. Giá trị mặc định của sigma là 1.

CHƯƠNG VI:

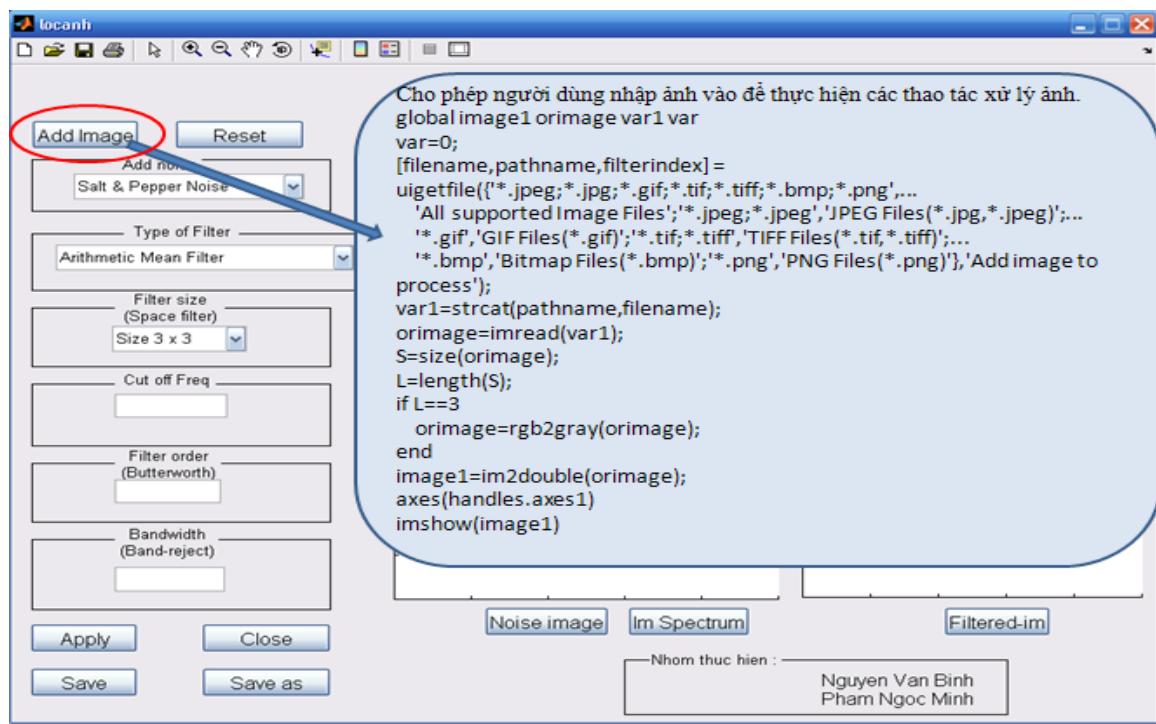
CHƯƠNG TRÌNH LỌC ẢNH KẾT HỢP GUI

I) Giới thiệu về giao diện của chương trình và chức năng của các thành phần:

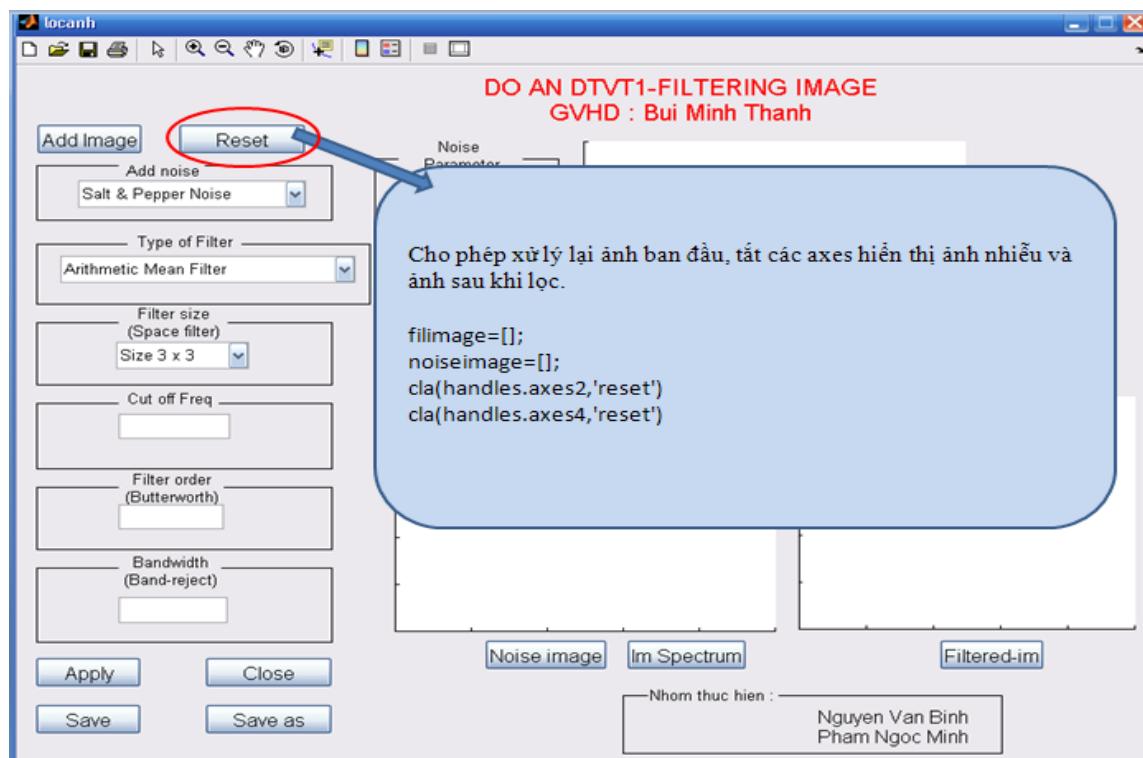
1) Giao diện tổng quát của chương trình:

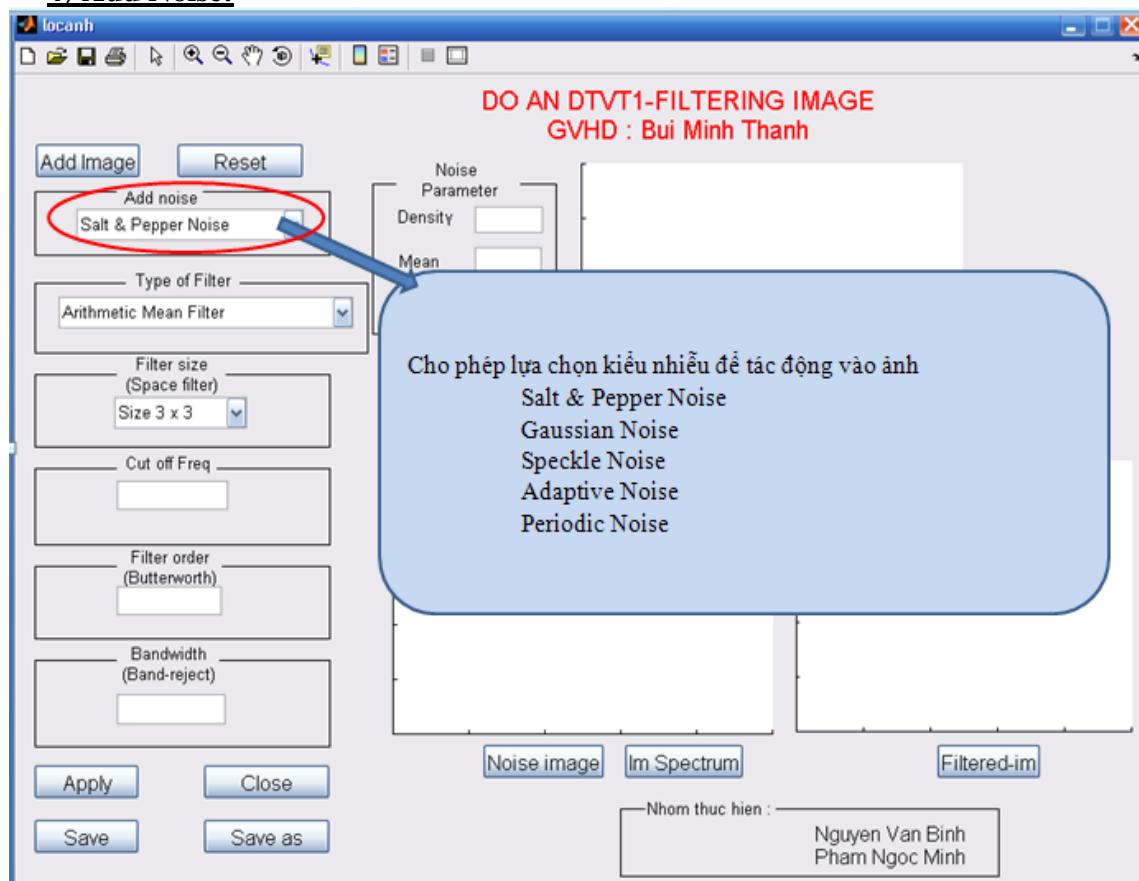


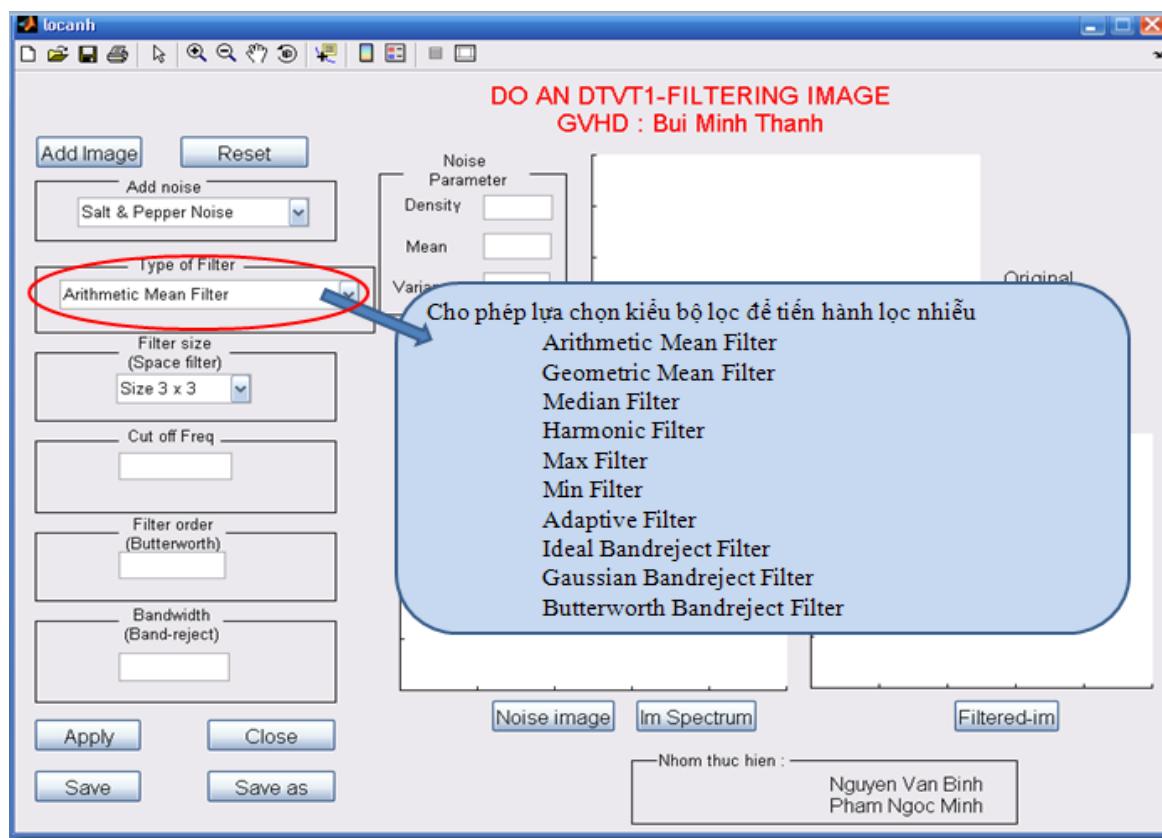
2) Nút “Add image”:



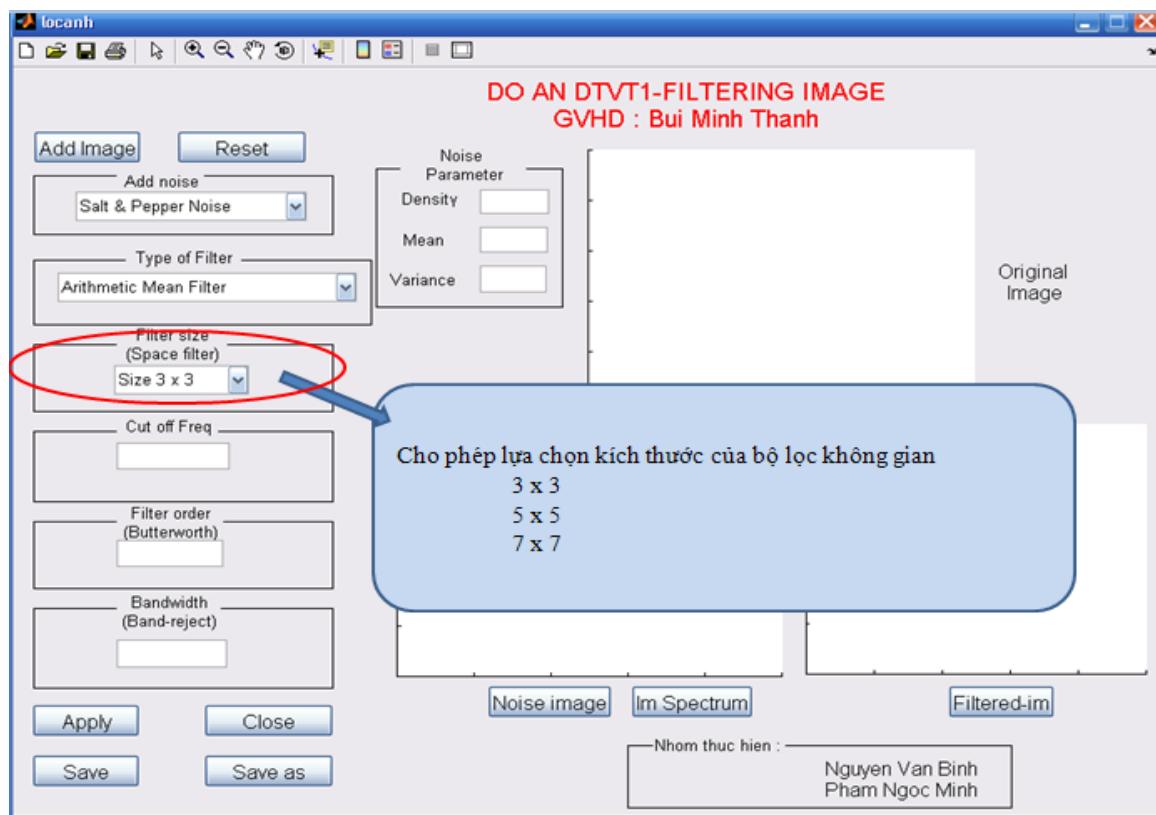
3) Nút “Reset”:



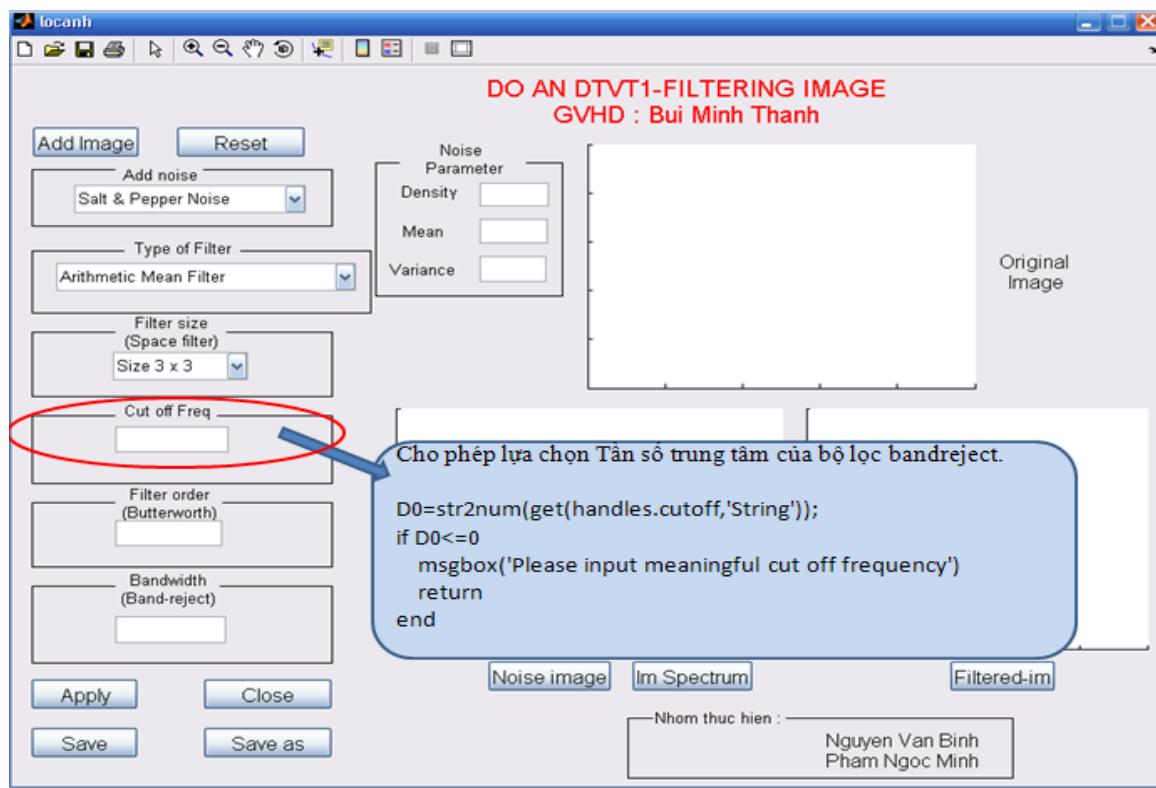
4) Add Noise:**5) Lựa chọn kiểu bộ lọc:**



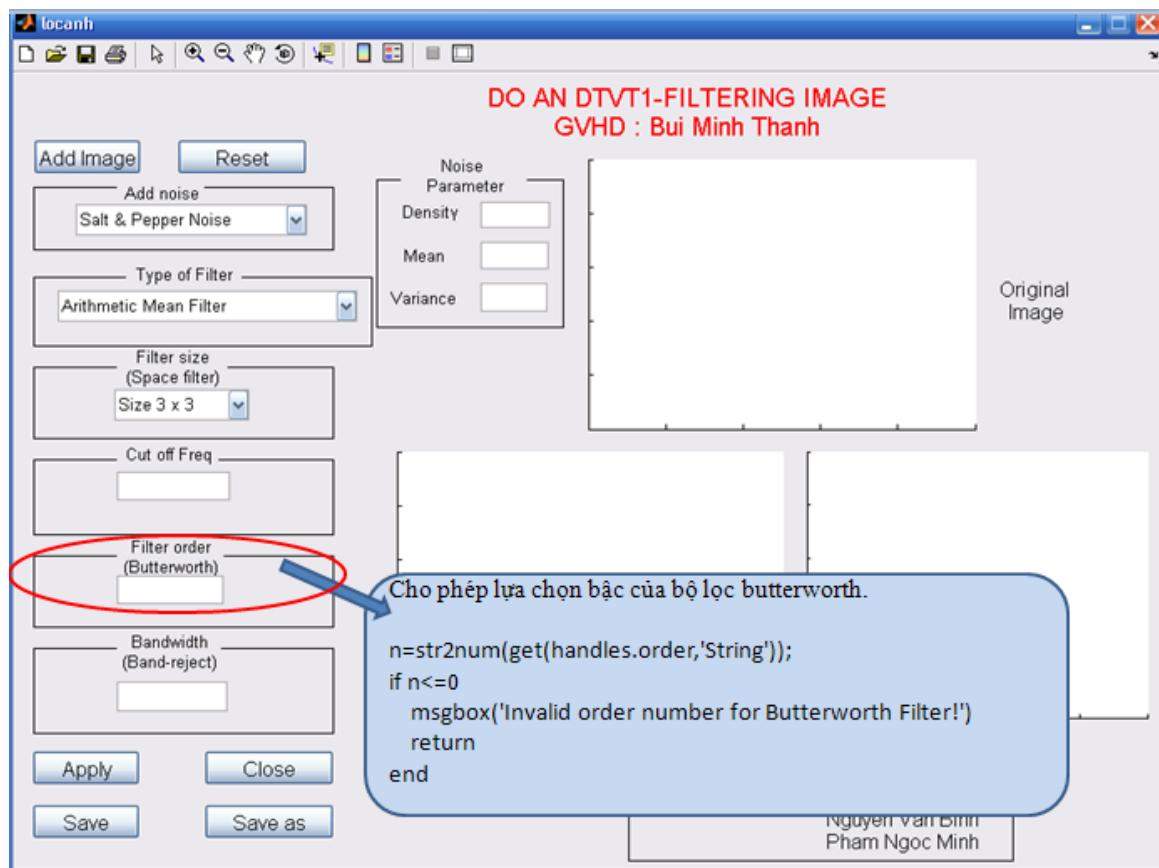
6) Kích thước bộ lọc:



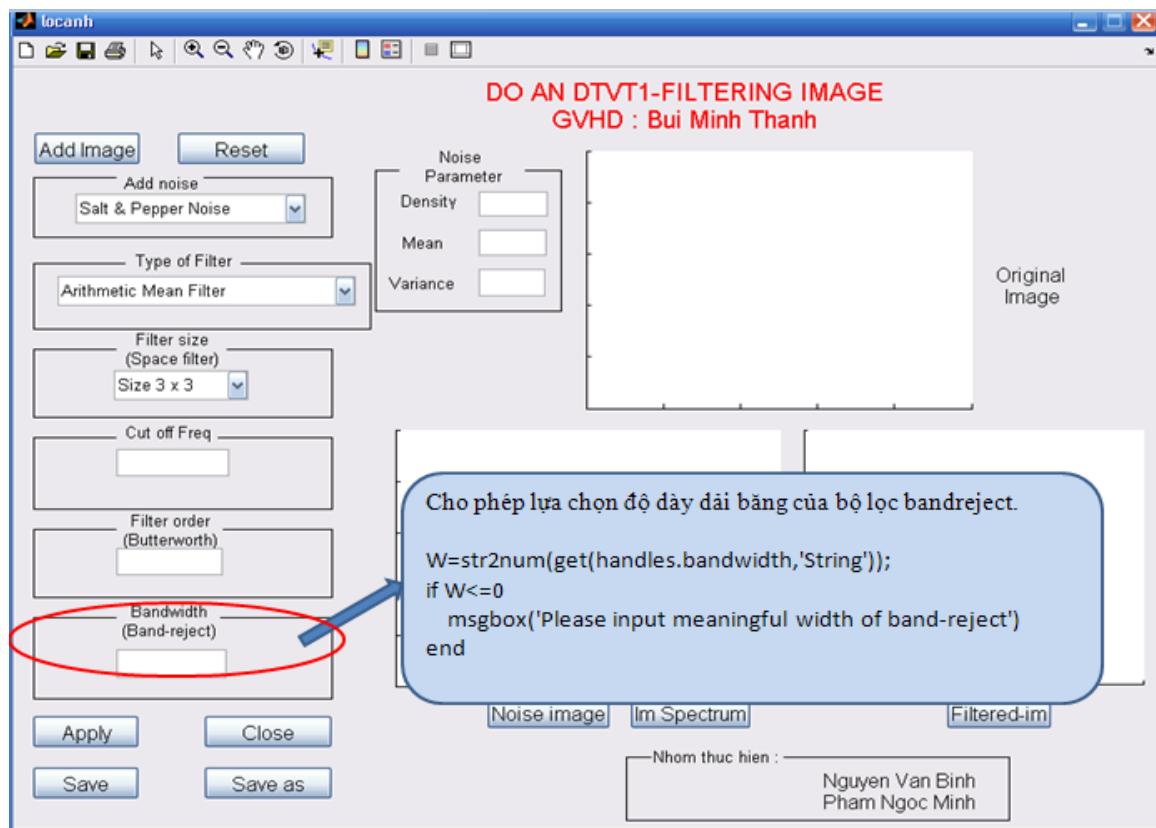
7) Tần số trung tâm của bộ lọc Bandreject:



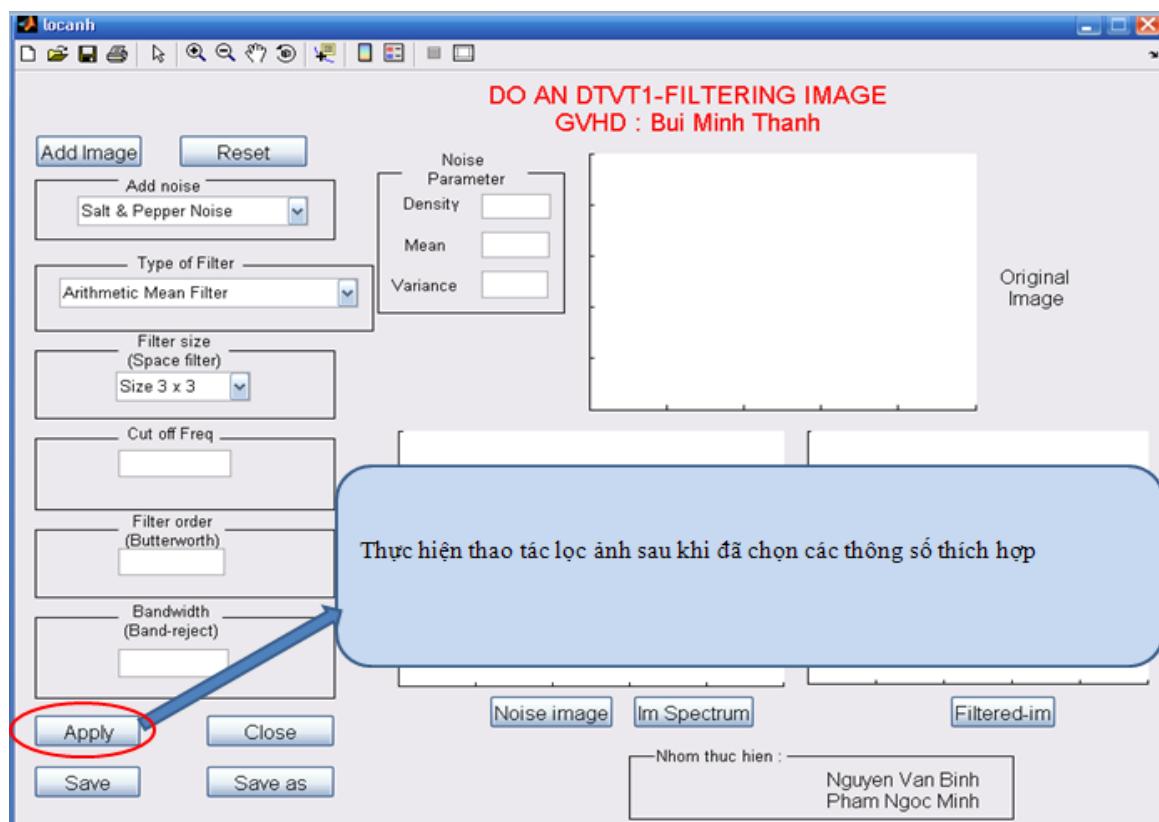
8) Bậc của bộ lọc Butterworth bandreject:

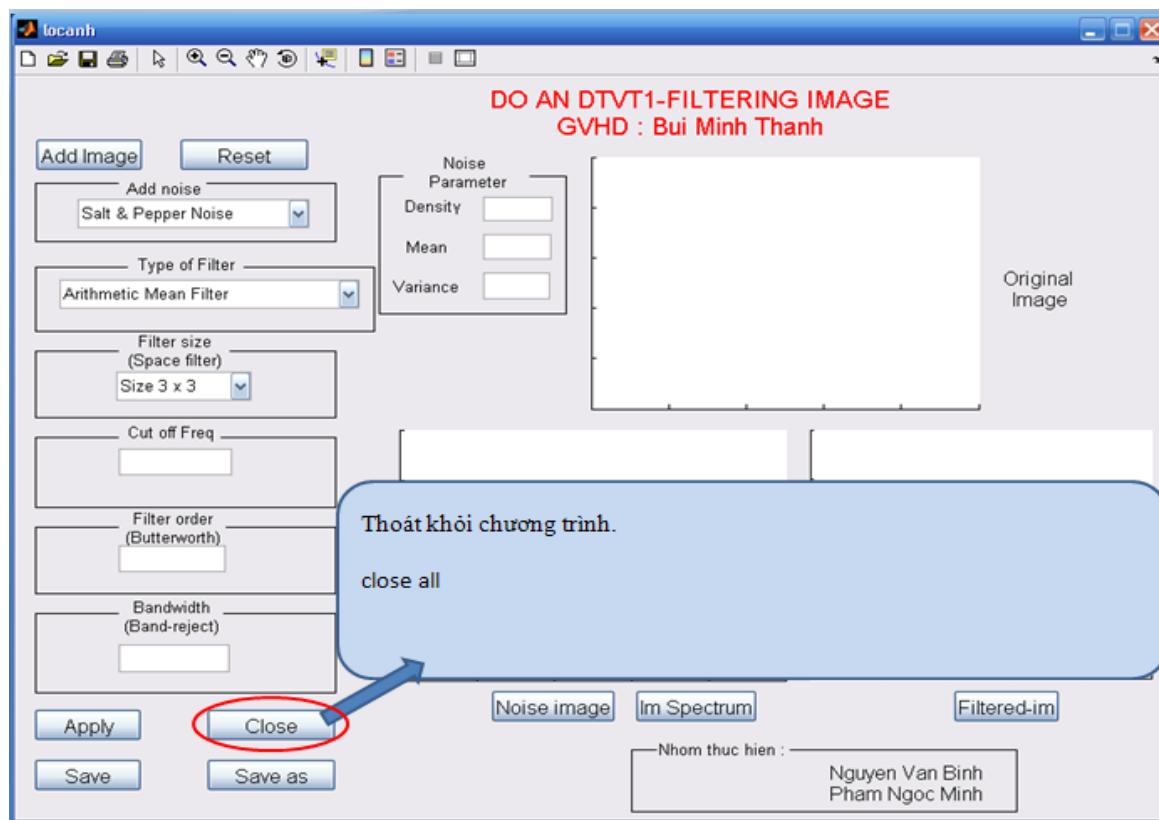


9) Độ rỗng dải băng của bộ lọc Bandreject:

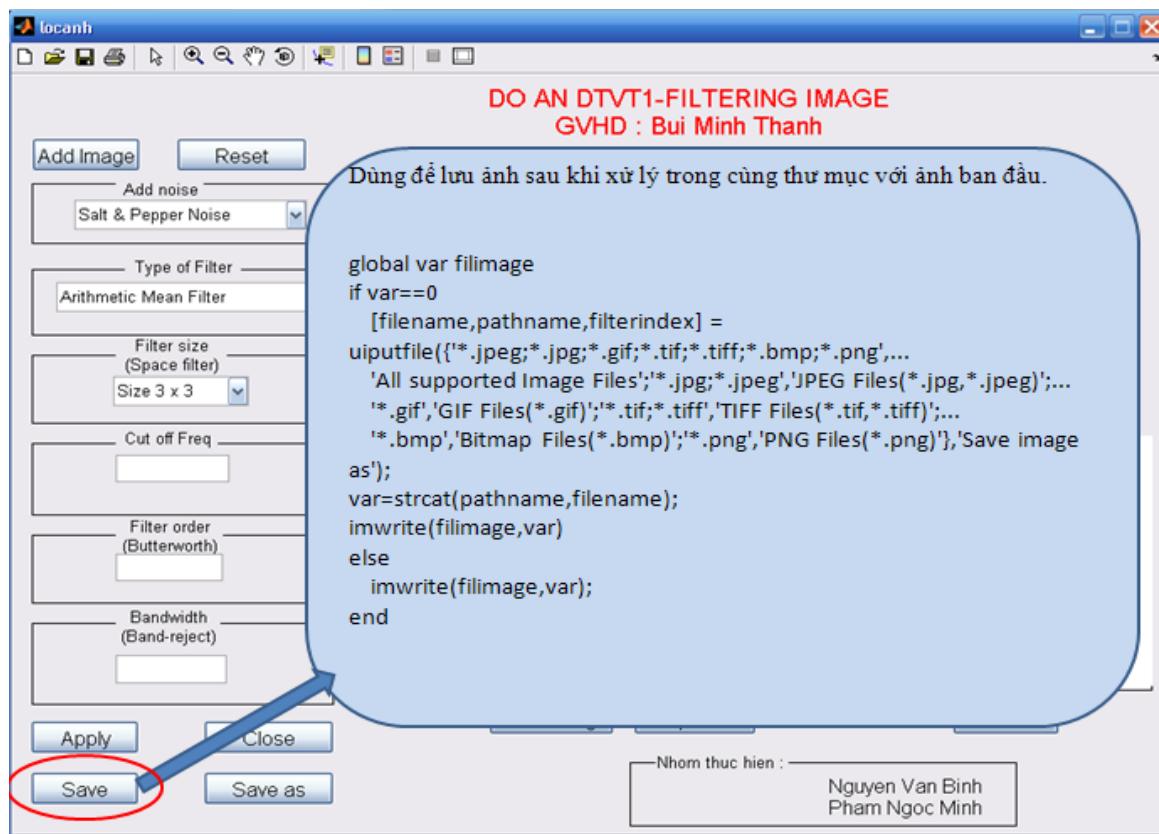


10) Nút “Apply”:

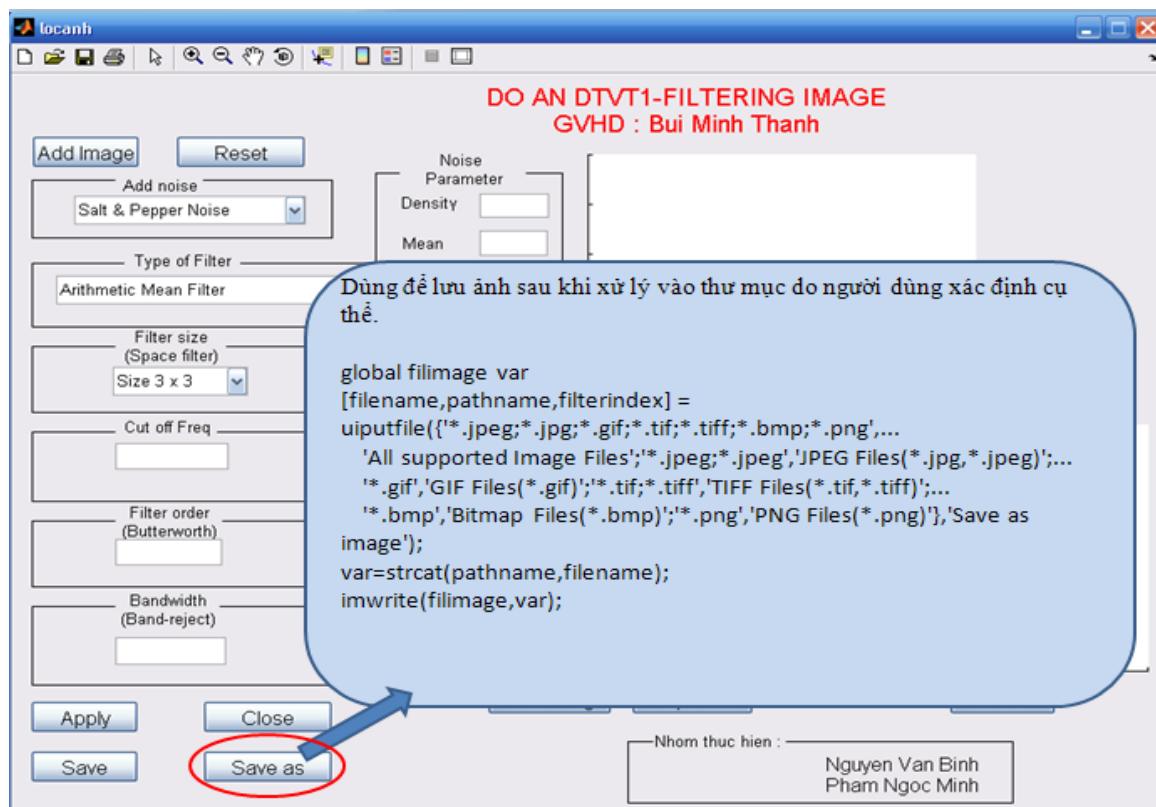
**11) Nút “Close”:**



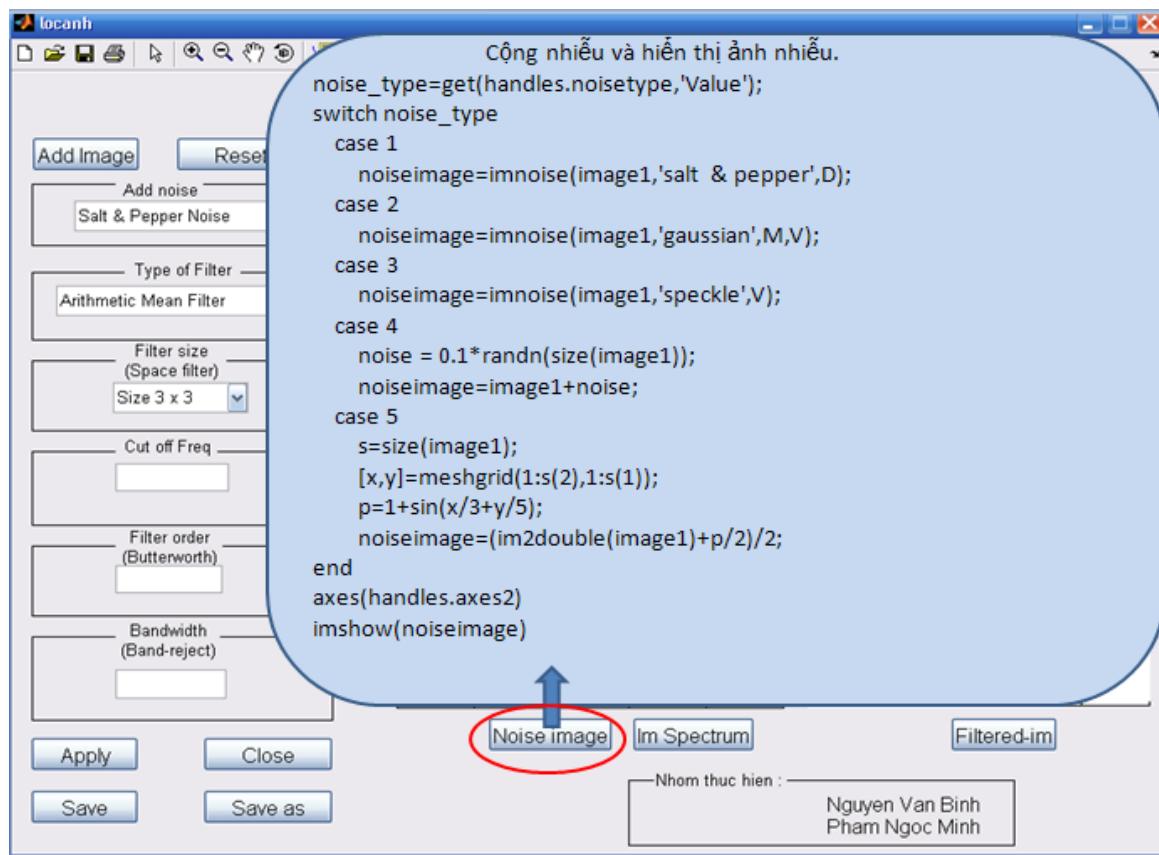
12) Nút “Save”:



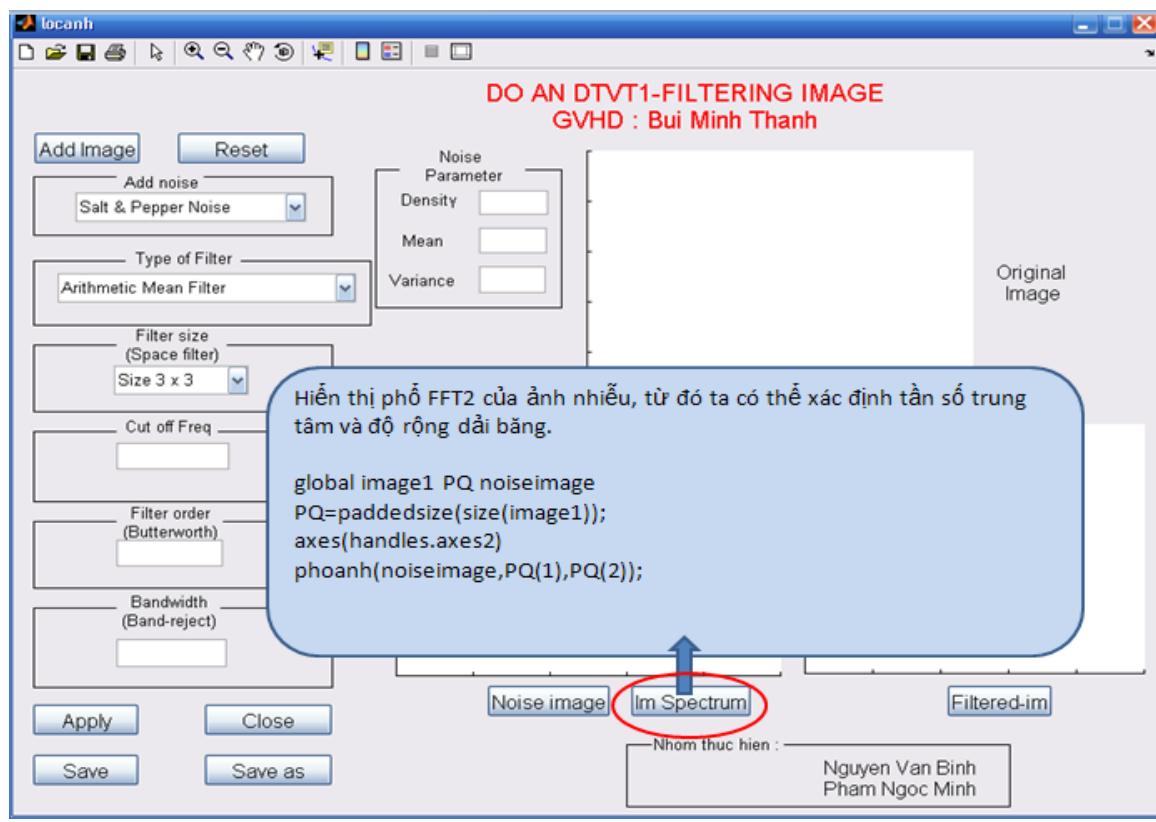
13) Nút “Save as”:



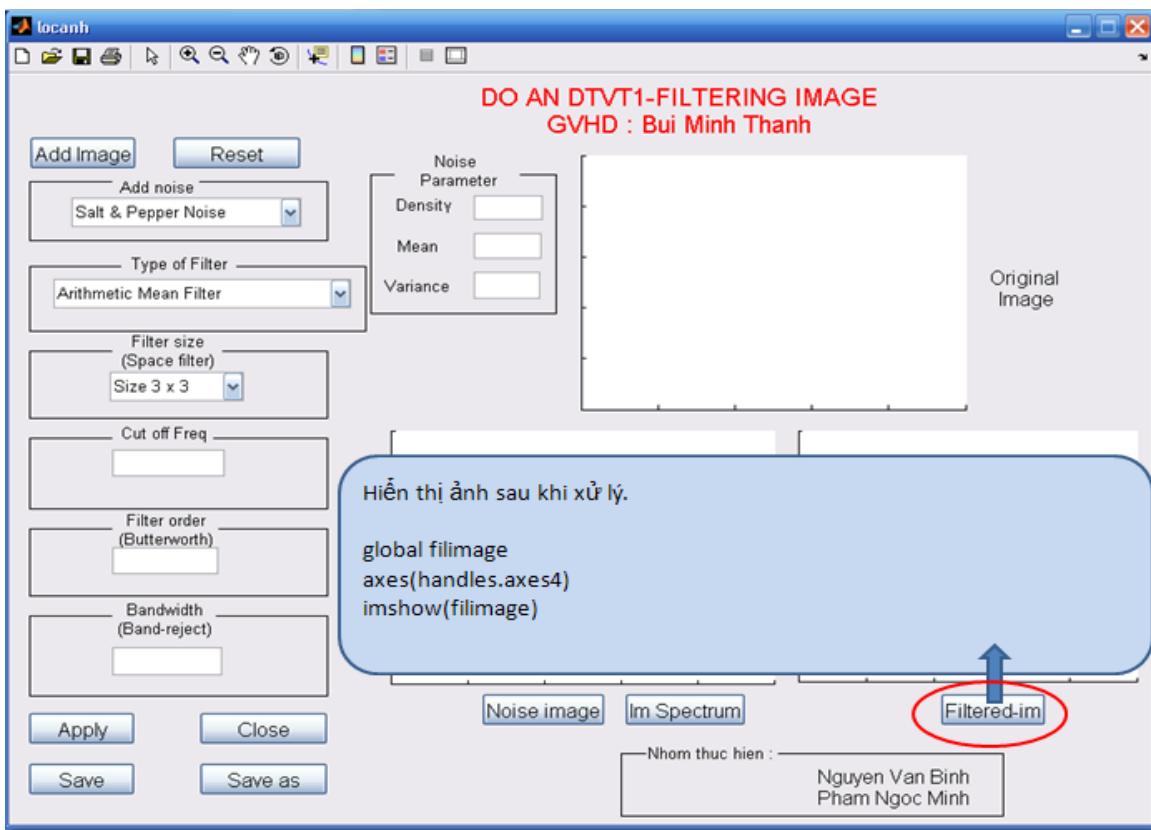
14) Nút “Noise image”:



15) Nút “Im spectrum”:



16) Nút “Filtered Im”:



II) Giải thuật của các hàm M-file trong chương trình:

1) Hàm “padaddedsize”:

Mục đích: Ảnh sau biến đổi Fourier có tính tuần hoàn, do đó để tránh nhiễu giữa các chu kỳ kế cận nhau, ta cần mở rộng ảnh với các giá trị 0 để loại bỏ nhiễu này.

Giả sử với $f(x,y)$ và $h(x,y)$ có kích thước $A \times B$ và $C \times D$, ta sẽ thêm các giá trị 0 vào $f(x,y)$ và $h(x,y)$ để có kích thước như nhau là:

$$P \geq A+C-1$$

$$Q \geq B+D-1$$

Thông thường ta xét ảnh và hàm lọc có cùng kích thước, do đó $P \geq 2M - 1$, $Q \geq 2N - 1$, với M và N là kích thước của ảnh và hàm lọc.

Xét hàm sau với AB, CD, PQ là các vector lần lượt gồm các thành phần $[A B], [C D], [P Q]$

```
function PQ=padaddedsize(AB,CD)
```

```
if nargin==1
```

```
    PQ=2*AB; % Neu chi anh va bo loc co cung kich thuoc
```

```
elseif nargin==2
```

```
    PQ=AB+CD-1; % Neu anh va bo loc khac kich thuoc
```

```
    PQ=2*ceil(PQ/2); % PQ co cac thanh phan la cac so chan
```

```
else
```

```

    error('Wrong number of inputs');
end

```

2) Hàm “changeclass”:

Tạo ảnh sau lọc có cùng kiểu với ảnh đầu vào.

```

switch class
case 'uint8'
    image = im2uint8(image);
case 'uint16'
    image = im2uint16(image);
case 'double'
    image = im2double(image);
otherwise
    error('Unsupported IPT data class.');
end

```

3) Hàm “gmean”:

Thực hiện giải thuật của bộ lọc Geometric Mean Filter.

```

function f=gmean(image,m,n)
inclass=class(image);
image=im2double(image);
warning off;
f=exp(imfilter(log(image),ones(m,n),'replica')).^(1/m/n);
warning on;
f=changeclass(inclass,f);

```

4) Hàm “harmean”:

Thực hiện giải thuật của bộ lọc Harmonic Filter.

```

function f=harmean(image,m,n)
inclass=class(image);
image=im2double(image);
f=m*n./imfilter(1./(image+eps),ones(m,n),'replicate');
f=changeclass(inclass,f);

```

5) Hàm “phoanh”:

```

function phoanh(I,P,Q)
I=im2double(I);
J=fftshift(fft2(I,P,Q));
fl=log(1+abs(J));
fm=max(fl(:));
imshow(im2double(fm));
h=impixelinfo; → Hiển thị pixel nơi con trỏ trỏ tới.
k=imdhistline → Cho phép đo khoảng cách từ tâm ảnh đến tọa độ nhiễu, từ đó xác định được tần số cắt cũng như độ rộng băng cần thiết.

```

6) Hàm “bandreject”:

```

function H=bandreject(PQ,D0,W,type,n)
M=PQ(1);
N=PQ(2); → Xác định kích thước của bộ lọc sau khi mờ rộng ảnh
H=ones(M,N);
for i=1:M
    for j=1:N
        D=sqrt((i-M/2).^2+(j-N/2).^2); → Tính kích thước từ một điểm bất kỳ đến trung tâm ảnh
        switch type
            case 'ideal'
                if((D<D0-W/2)|(D>D0+W/2)) → Đáp ứng của bộ lọc bandreject lý tưởng
                    H(i,j)=1;
                else
                    H(i,j)=0;
                end
            case 'butter'
                if nargin==4
                    n=1;
                end
                H(i,j)=1./(1+(D.^2/(D.^2-D0.^2)).^(2*n)); → Đáp ứng của bộ lọc bandreject Butterworth
            case 'Gaussian'
                H(i,j)=1-exp(-1/2*((D.^2-D0.^2)/(D.^2)).^2); → Đáp ứng của bộ lọc band_Gaussian
            otherwise
                disp('Unknown bandreject filter')
            end
        end
    end
end

```

III) Các bước tính toán trong nút “Apply”:

global image1 H filimage PQ value noiseimage

```

PQ=paddedsize(size(image1));
D0=str2num(get(handles.cutoff,'String'));
if D0<=0
    msgbox('Please input meaningful cut off frequency')
    return
end
n=str2num(get(handles.order,'String'));
if n<=0
    msgbox('Invalid order number for Butterworth Filter!')
    return
end
W=str2num(get(handles.bandwidth,'String'));
if W<=0
    msgbox('Please input meaningful width of band-reject')
end
fysize=get(handles.size,'Value');
switch fysize
    case 1
        m=3;
        n=3;
    case 2
        m=5;
        n=5;
    case 3
        m=7;
        n=7;
    end
value=get(handles.filtertype,'Value');
switch value
    case 1 %Arithmetic Mean Filter
        H=fspecial('average',[m n]);
        filimage=imfilter(noiseimage,H);
    case 2 %Geometric Mean Filter
        filimage=gmean(noiseimage,m,n);
    case 3 %Median Filter
        filimage=medfilt2(noiseimage,[m n],'symmetric');
    case 4 %Harmonic Filter
        filimage=harmean(noiseimage,m,n);
    case 5 %Max Filter
        filimage=ordfilt2(noiseimage,m*n,ones(m,n),'symmetric');
    case 6 %Min Filter
        filimage=ordfilt2(noiseimage,1,ones(m,n),'symmetric');
    case 7 %Adaptive Filter (Weiner)
        filimage=wiener2(noiseimage,[m n]);

```

```
case 8          %Ideal Bandreject Filter
    type='ideal';
    H=bandreject(PQ,D0,W,type,n);
    F=fftshift(fft2(image1,PQ(1),PQ(2)));
    J=F.*H;
    filimage=real(ifft2(J));
    filimage=filimage(1:size(image1,1),1:size(image1,2));
case 9          %Gaussian Bandreject Filter
    type='butter';
    H=bandreject(PQ,D0,W,type,n);
    F=fftshift(fft2(image1,PQ(1),PQ(2)));
    J=F.*H;
    filimage=real(ifft2(J));
    filimage=filimage(1:size(image1,1),1:size(image1,2));
case 10         %Butterworth Bandreject Filter
    type='Gaussian';
    H=bandreject(PQ,D0,W,type,n);
    F=fftshift(fft2(image1,PQ(1),PQ(2)));
    J=F.*H;
    filimage=real(ifft2(J));
    filimage=filimage(1:size(image1,1),1:size(image1,2));
end
```