Chương 10 XỬ LÝ ẢNH

Mục đích của bài thực hành là:

- Hiểu biết về ảnh, cách biểu diễn ảnh và chuyển đổi ảnh trong các kênh mầu khác nhau
- Phân tích và xử lý ảnh cơ bản trong miền thời gian
- Phân tích và xử lý ảnh cơ bản trong miền tần số.

10.1 Hiển thị ảnh và chuyển đổi giữa các kênh mầu khác nhau

Ảnh được coi là tín hiệu 2 chiều, mô tả thông tin cường độ sáng. Ảnh thường được biểu diễn dạng ma trận 2 chiều MxN hoặc 3 chiều MxNx3 (ví dụ gồm 3 ma trận MxN tương ứng với 3 kênh mầu R, G, B). Mỗi giá trị trong ma trận tương ứng với cường độ sáng/tối tại 1 điểm ảnh (pixel). Có nhiều cách biểu diễn ảnh. Nói chung 3 cách cơ bản nhất là

- Ảnh xám (gray scale): ma trận ảnh thường được biểu diễn dạng MxN trong đó mỗi pixel nhận giá trị trong khoảng [0, 255] hoặc [0,1]
- Ảnh mầu: Có nhiều định dạng kênh mầu, tương ứng với cách biểu diễn ảnh, RGB hoặc HSV hoặc YCrCb, LAB... Ảnh mầu được biểu diễn dạng MxNx3 tương ứng với các kênh mầu.
- Ảnh nhị phân (BW image): ảnh chỉ có 2 mức đen và trắng.

Trong bài thực hành này, chúng ta sẽ làm quen với ảnh và xử dụng một số hàm Matlab để phân tích và xử lý ảnh.

10.1.1 Đọc và hiển thị ảnh

Để đọc ảnh vào Matlab, ta sử dụng lệnh $imread$, kết quả được trả về là ma trận ảnh tương ứng với 3 kênh mầu RGB. Ảnh có thể được hiển thị lại, sử dụng $imshow$.

Matlab code 10.1.1

```
RGB = imread('peppers.png');
imshow(RGB)% display
```

Hình 10.1: Ví dụ về đọc và hiển thị ảnh trên Matlab

10.1.2 Chuyển đổi ảnh trong các không gian mầu (colorspace)

Ảnh xám:
Để chuyển đổi ảnh mầu RGB sang dạng ảnh xám (gray) có một vài cách.

- Phương pháp độ đậm nhạt, tính trung bình các màu nổi bật nhất và ít nổi bật nhất: (max (R, G, B) + min (R, G, B)) / 2
- Phương pháp trung bình: (R + G + B) / 3.
- Phương pháp trung bình có trọng số: Đây là phương pháp phức tạp hơn phương pháp trung bình, với các thành phần được nhân thêm trọng số, để tính đến cảm quan mầu sắc và nhận thức của con người. Chúng ta nhạy cảm với màu xanh lá cây hơn các màu khác, vì vậy màu xanh lá cây có trọng lượng

lớn nhất. Công thức của độ sáng là: $Gray = 0,21R + 0,71G + 0,07B$. Một công thức khác được sử dụng rộng rãi hơn là $Gray = 0,299R + 0,587G + 0,114B$ (công thức được dùng trong Matlab tương ứng với lệnh $rgb2gray$).

Ảnh HSV, YCrCb, Lab:

Bên cạnh ảnh xám, ảnh mầu RBG còn có thể được biểu diễn thông qua các kênh mầu khác. Ý nghĩa của các phép chuyển đổi này là giảm sự tương quan giữa 3 kênh R, G, B mà tập trung thông tin ảnh như cường độ sáng, độ chói,... vào các kênh riêng biệt. Hình 10.2 mô tả các lệnh cơ bản để chuyển đổi ảnh giữa các không gian mầu khác nhau.

rgb2hsv	Convert RGB colors to HSV
rgb2lab	Convert RGB to CIE 1976 L*a*b*
rgb2ntsc	Convert RGB color values to NTSC color space
rgb2xyz	Convert RGB to CIE 1931 XYZ
rgb2ycbcr	Convert RGB color values to YCbCr color space
rgbwide2ycbcr	Convert wide-gamut RGB color values to YCbCr color values
rgbwide2xyz	Convert wide-gamut RGB color values to CIE 1931 XYZ color values
hsv2rgb	Convert HSV colors to RGB
lab2rgb	Convert CIE 1976 L*a*b* to RGB
lab2xyz	Convert CIE 1976 L*a*b* to CIE 1931 XYZ
ntsc2rgb	Convert NTSC values to RGB color space
xyz2lab	Convert CIE 1931 XYZ to CIE 1976 L*a*b*
xyz2rgb	Convert CIE 1931 XYZ to RGB
xyz2rgbwide	Convert CIE 1931 XYZ color values to wide-gamut RGB color values
ycbcr2rgb	Convert YCbCr color values to RGB color space
ycbcr2rgbwide	Convert YCbCr color values to wide-gamut RGB color values
colorcloud	Display 3-D color gamut as point cloud in specified color space

Hình 10.2: Các lệnh Matlab thường dùng để chuyển đổi giữa các không gian mầu

Ảnh nhị phân:

Sử dụng ngưỡng chuyển đổi ảnh xám sang dạng ảnh nhị phân. Các giá trị pixel > ngưỡng nhận giá trị 1. Ngược lại, nhận giá trị 0. Khi đó, ta sẽ thu được ảnh dạng nhị phân. Các ảnh nhị phân có ý nghĩa quan trọng trong việc định vị các vùng xử lý (ví dụ như vị trí object, đường biên...).

Bài tập

1. Ảnh trông như thế nào, nếu ta thay đổi thứ tự các kênh mầu trong ảnh? Kiểm thử lại bằng cách viết lệnh để thực hiện hiển thị ảnh theo thứ tự B-R-G, G-B-R
2. Viết lệnh để chuyển 1 ảnh RGB sang kiểu ảnh xám, theo các cách nêu trên. Chúng ta có thể chuyển ngược từ ảnh xám sang ảnh mầu được không?
3. Thử chuyển 1 ảnh thành kiểu HSV hoặc YCrCb. Nhận xét kết quả nếu 1 trong các ma trận mầu (H hoặc S hoặc V ...) bằng 0.
4. Viết lệnh chuyển 1 ảnh RGB sang ảnh nhị phân với các giá trị ngưỡng khác nhau và hiển thị ảnh đó.

10.2 Độ phân giải ảnh

Quá trình tạo ảnh số có thể được mô tả đơn giản như hình **??**, bao gồm việc lấy mẫu và lượng tử hoá. Tín hiệu hình ảnh liên tục từ vật thể được lấy mẫu bởi ma trận cảm biến từ máy chụp ảnh. Tuỳ vào cấu hình phần cứng, mà hình ảnh chụp có độ phân giải khác nhau. Sau đó, thông tin từ cảm biến được số hoá. Tuỳ vào số bít sử dụng và cách ánh xạ (mapping) trên không gian mầu (colorspace) mà các máy ảnh khác nhau sẽ hiển thị ảnh mầu RGB có đôi chút khác nhau. Tuy độ phân giải ảnh phụ thuộc vào phần cứng của máy chụp ảnh; nhưng hiện nay cũng có rất nhiều thuật toán (hay phần mềm) cho phép tăng hoặc giảm độ phân giải ảnh.

10.2.1 Tăng độ phân giải sử dụng thuật toán nội suy-interpolation

Có rất nhiều thuật toán nội suy. Matlab cung cấp một số thuật toán nội suy thông dụng như interp2 với một số lựa chọn thông dụng như trong hình 10.3.

10.2.2 Giảm độ phân giải ảnh sử dụng Lấy mẫu-Resampling

Có thể giảm độ phân giải ảnh một cách đơn giản bằng lấy mẫu đều. Lệnh Matlab thực hiện đơn giản như sau. $Im1 = Im(1:k:end, 1:k:end, :)$; thực hiện cứ k mẫu lấy 1 mẫu; k là một số nguyên cho trước.

Bài tập

5. Viết chương trình giảm độ phân giải ảnh lối vào 3 lần, sau đó thực hiện nội suy để tạo lại ảnh có độ phân giải cũ. So sánh ảnh thu được với ảnh gốc và xác

Method	Description	Continuity	Comments
`'linear'`	The interpolated value at a query point is based on linear interpolation of the values at neighboring grid points in each respective dimension. This is the default interpolation method.	C^0	• Requires at least two grid points in each dimension • Requires more memory than `'nearest'`
`'nearest'`	The interpolated value at a query point is the value at the nearest sample grid point.	Discontinuous	• Requires two grid points in each dimension. • Fastest computation with modest memory requirements
`'cubic'`	The interpolated value at a query point is based on a cubic interpolation of the values at neighboring grid points in each respective dimension. The interpolation is based on a cubic convolution.	C^1	• Grid must have uniform spacing in each dimension, but the spacing does not have to be the same for all dimensions • Requires at least four points in each dimension • Requires more memory and computation time than `'linear'`
`'spline'`	The interpolated value at a query point is based on a cubic interpolation of the values at neighboring grid points in each respective dimension. The interpolation is based on a cubic spline using not-a-knot end conditions.	C^2	• Requires four points in each dimension • Requires more memory and computation time than `'cubic'`

Hình 10.3: Lệnh interp2 của Matlab với 1 số lựa chọn để nội suy

định xem phương pháp nội suy nào tốt nhất? Thử với các loại ảnh khác nhau (ảnh ít/nhiều chi tiết, ảnh nhiều đường/ít đường....) và cho nhận xét.

10.3 Toán tử trong ảnh- Image operator

Khi cho tín hiệu ảnh qua một hệ thống xử lý, tín hiệu ảnh lối ra sẽ là tích chập giữa ảnh và hệ thống. Gọi h(x,y) là ma trận đặc trưng cho hệ thống; tích chập giữa ảnh I(x,y) và hệ thống h được định nghĩa như sau: $Iout\left[i,j\right] = \sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty} h\left[m,n\right] \cdot I\left[i-m, j-n\right]$

Quá trình tính tích chập (2D) trên ảnh được mô tả như hình vẽ, trong đó giá trị tại 1 pixel được tính dựa trên nhân và cộng tất cả các pixel xung quanh. Tuỳ vào hệ thống h, ảnh lối ra thu được sẽ khác nhau. Dưới đây là ví dụ thực hiện toán tử trung bình cộng, có tác dụng làm mượt (smooth) ảnh:

Matlab code 10.3.1

```
I=imread('trees.tif');
subplot(2,2,1); imshow(J); title('original image');
h=ones(3,3)/9;
J=imfilter(I,f,'circular');
subplot(2,2,2); imshow(J); title('averaged image');
```

Bài tập

6. Áp dụng với các đặc trưng của hệ thống như sau và hiển thị kết quả ảnh thu được? Tác dụng của các hệ thống trên như thế nào?
   - $h1 = [1, 0, -1; 1, 0, -1; 1, 0, -1]$
   - $h2 = [1, 0, -1; 1, 0, -1; 1, 0, -1]^T$.
   - $h3 = ([121; 000; -1 - 2 - 1])/2$;
   - $h4 = ones(3)/9$

Bên cạnh đó, Matlab cũng cung cấp một số toán tử đặc biệt để thực hiện biến đổi trên ảnh; có thể tham khảo qua keyword edge detection.

10.4 Miền tần số

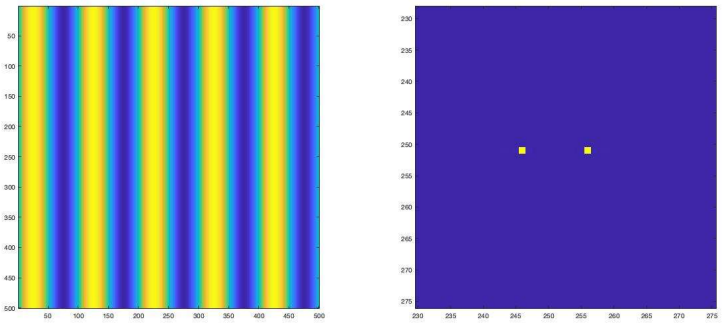
Ảnh có thể coi là tín hiệu 2 chiều. Do đó, có thể áp dụng biến đổi biến đổi Frequency 2D cho ảnh. Công thức của Fourier Transform cho tín hiệu 2 chiều có dạng:

$$F(u,v) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)e^{-j\,2\pi\left(\frac{um}{M}+\frac{vn}{N}\right)}$$

Trong đó (m,n) là không gian ảnh, (u,v) là miền tần số. F(u,v) là ma trận hệ số Fourier của ảnh. Dựa vào F(u,v) ta có thể vẽ phổ biên độ của ảnh. Nó có dạng 2 chiều; vùng tần số thấp là vùng xung quanh gốc; vùng bên ngoài tương ứng với các giá trị tần số cao. Trong Matlab, ta sử dụng fft2 và ifft2 để chuyển đổi từ miền ảnh sang miền tần số và ngược lại. Để có thể cảm nhận về tần số và phổ của ảnh, ta có thể xem xét các hình vẽ bên dưới. Ở đây, ta tạo ra 1 tín hiệu 2 chiều đơn giản, có sự biến đổi chỉ theo trục x (dạng sin); trục y giữ nguyên (tần số bằng 0). Khi đó đồ thị biểu diễn trong miền tần số có dạng 2 điểm theo trục ngang, tương ứng với +/- tần số của tín hiệu sin.

Matlab code 10.4.1

```
n = [0:1/100:5];
X = meshgrid(n,n);
Im = sin(2*pi*X);
subplot(1,2,1); imagesc(Im);
Y = fft2(Im);
subplot(1,2,2); imagesc(abs(fftshift(Y)))
```



Hình 10.4: Ảnh trong miền không gian và miền tần số, kết quả từ đoạn code Matlab 10.4.1

Tăng giảm giá trị của tần số sin, phổ biên độ cũng có sự thay đổi tương ứng (10.4.2).

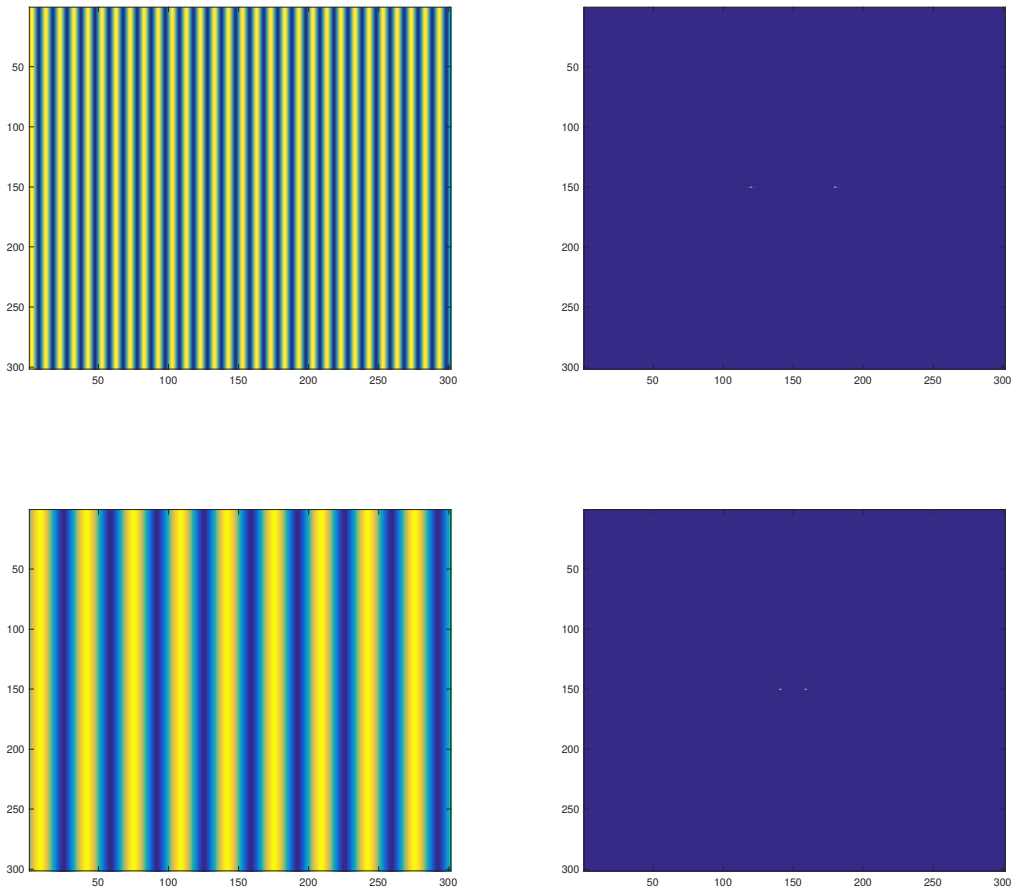
Matlab code 10.4.2

```
Im1= sin(2*pi*X*10);
Y1 = fft2(Im1);

Im2= sin(2*pi*X*3);
Y2 = fft2(Im2);

subplot(2,2,1); imagesc(Im1);
subplot(2,2,2); imagesc(abs(fftshift(Y1)));

subplot(2,2,3); imagesc(Im2);
subplot(2,2,4); imagesc(abs(fftshift(Y2)));
```

Khi thực hiện thêm việc xoay (10.4.3), ta cũng có thể quan sát trên ảnh ở miền tần số:

Matlab code 10.4.3

```
Y = X';
N= X*cos(pi/9) + Y*sin(pi/9); % xoay goc pi/9
Im3 = sin( 2*pi*N);
subplot(1,2,1); imagesc(Im3);
Y = fft2(Im3);
```



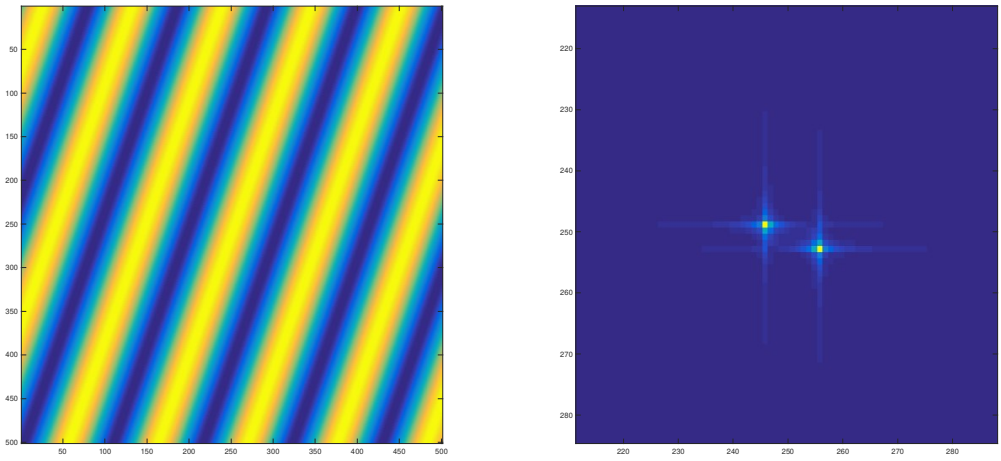
Hình 10.5: Khi thay đổi tần số tín hiệu trong miền ảnh (ảnh phía trái) và biểu diễn tương ứng trên miền tần số (ảnh phía phải)

```
subplot(1,2,2); imagesc(abs(fftshift(Y)))
```

Khi tín hiệu có dạng xung chữ nhật 2 chiều. Phổ cũng có dạng sinc 2 chiều tương ứng. Khi tín hiệu là sự kết hợp từ nhiều tần số, quan sát phổ ta cũng thấy các vạch phổ tương ứng khác nhau (hình 10.6).

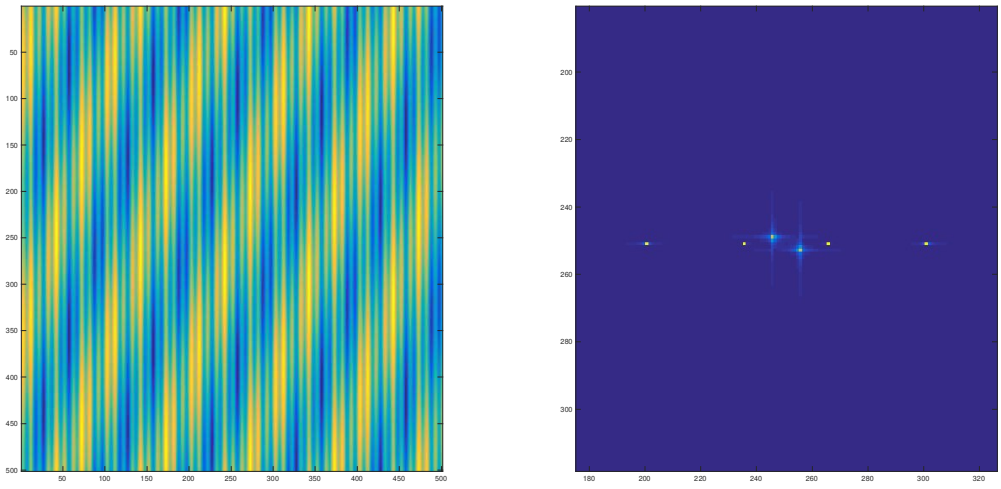
Matlab code 10.4.4

```
Im4 = Im1+Im2+Im3;
```



Hình 10.6: Phổ tần số tương ứng khi ảnh bị xoay trong không gian ảnh

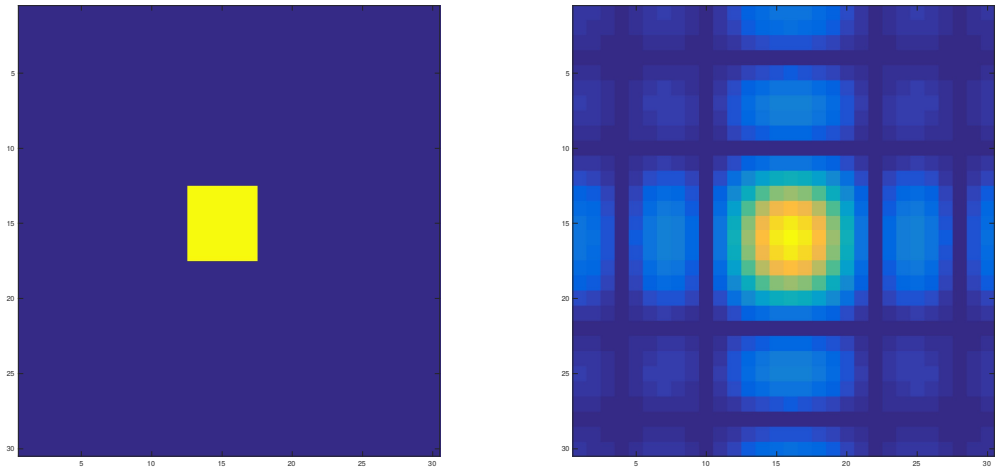
```
subplot(1,2,1); imagesc(Im4);
Y = fft2(Im4);
subplot(1,2,2); imagesc(abs(fftshift(Y)))
```



Hình 10.7: Phổ tần số tương ứng của ảnh gồm nhiều tần số

Matlab code 10.4.5

```
Im5=zeros(30,30);
Im5(13:17,13:17)=1;
Y5 = fft2(Im5);
subplot(1,2,1); imagesc(Im5);
subplot(1,2,2); imagesc(abs(fftshift(Y5)));
```



Hình 10.8: Phổ tần số của tương ứng của ảnh xung chữ nhật

Kiểm thử với lệnh $ifft2$ để tìm lại ảnh ban đầu:

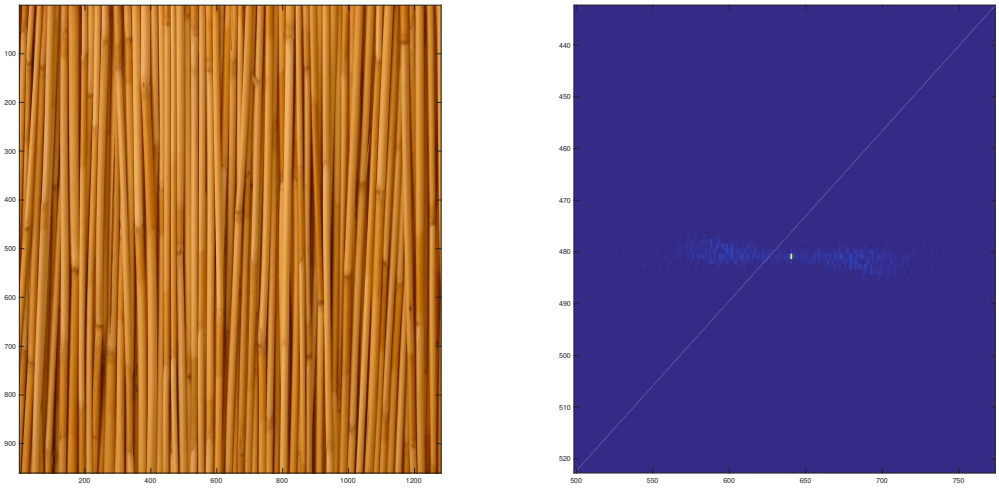
Matlab code 10.4.6

```
Im6 = ifft2(Y5);
subplot(1,2,1); imagesc(Im5);
subplot(1,2,2); imagesc(Im6);
```

Việc áp dụng fft2 trên ảnh cho ta thông tin về tần số của ảnh. Lưu ý, cần chuyển sang ảnh gray khi phân tích thông tin về tần số. Ví dụ bức ảnh sau canes.jpg có đặc trưng (texture) chủ yếu theo chiều dọc; nên biến đổi tần số biến đổi chủ yếu theo chiều ngang. Phổ biên độ của nó cũng tập trung tần số theo chiều ngang.

Matlab code 10.4.7

```
Im= imread('canes.jpg');
Im_gray = rgb2gray(Im)/255;
freq_Im = fft2(Im_gray);
subplot(1,2,1);imagesc(Im);
subplot(1,2,2); imagesc(abs(fftshift(freq_Im)));
```



Hình 10.9: Ảnh ban đầu (canes.jpg) và biểu diễn tần số của nó

Bài tập

7. Trong kỹ thuật tạo ảnh cộng hưởng từ (MRI), ảnh được tổng hợp từ thiết bị thu thập ảnh và được lưu trữ thành ảnh trong một không gian gọi là K-space (không gian K). Chi tiết về phương pháp tạo ảnh MRI các bạn có thể tham khảo trong cuốn sách *"MRI Basic Principles and Applications, by Brian M. Dale, Mark A. Brown and Richard C. Semelka, 2015, John Wiley and Sons.* Tuy nhiên về mặt xử lý tín hiệu thì ảnh trong không gian K thực chất là ảnh biến đổi Fourier của ảnh MRI trong không gian ảnh pixel. Bởi vậy việc tạo ảnh MRI về cơ bản, chỉ việc áp dụng biến đổi Fourier ngược lên ma trận tín hiệu biến đổi tần số từ máy chụp MRI. Trong phần này, chúng ta sẽ thực hiện việc dựng lại ảnh MRI từ dữ liệu MRI brain thu được từ máy chụp. Hãy hiển thị phổ của bức ảnh trên và áp dụng biến đổi Fourier ngược để thu được ảnh MRI.

10.4.1 Lọc trong miền tần số

Một tính chất quan trọng của biến đổi Fourier là phép nhân chập trong miền thời gian (miền ảnh) là phép nhân vô hướng trong miền tần số. Do đó, có thể áp dụng các toán tử lên ảnh thông qua việc thiết kế các bộ lọc ở miền tần số và thực hiện nhân vô hướng giữa 2 ma trận. Điều này, giúp chúng ta thực hiện tăng tốc việc tính toán trên không gian ảnh. Bên cạnh đó, việc tính toán nhân chập có thể dẫn đến hiệu ứng không tốt ở rìa ảnh (artifacts); nên thông thường các kỹ thuật padding (thêm 0 vào xung quanh ảnh) thường được sử dụng kèm theo. Đọc thêm về hàm paddedsize để biết thêm thông tin. Sau đây tóm tắt các bước cơ bản lọc DFT:

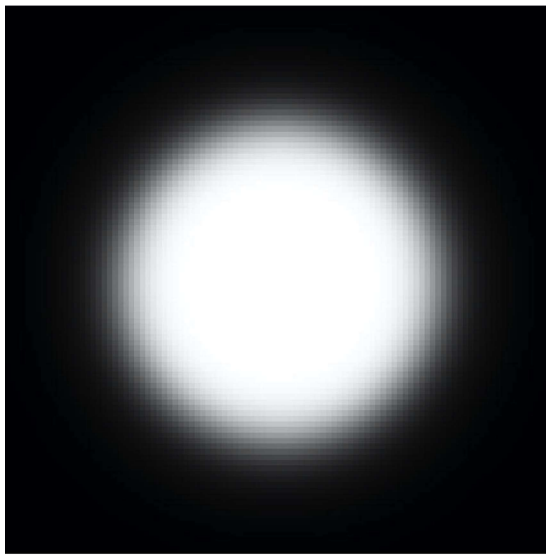
- Lấy thông số đệm bằng hàm paddedsize, dựa theo ảnh lối vào f:
  PQ = paddedsize (size(f));
- Lấy biến đổi Fourier của hình ảnh có kết hợp kỹ thuật padding:
  F = fft2 (f, PQ (1), PQ (2));
- Tạo bộ lọc, H, cùng kích thước với hình ảnh. Có thể thiết kế bộ lọc qua toán tử h rồi dùng fft2 hoặc thiết kế lọc trực tiếp trong miền tần số, dùng các hàm lọc có sẵn của Matlab như lọc Butterword
- Nhân hình ảnh được chuyển đổi với bộ lọc: G = H. * F;(nhân từng thành phần của 2 ma trận với nhau element-wise)
- Lấy phần thực của FFT nghịch đảo của G: $g = real(ifft2(G))$;
- Cắt hình chữ nhật trên cùng, bên trái về kích thước ban đầu. Ảnh g là ảnh kết quả thu được sau khi xử lý.
  $g = g(1 : size(f, 1), 1 : size(f, 2))$;

Bài tập

8. Áp dụng lowpassfilter bên dưới cho phép tạo 1 bộ lọc thông thấp trong miền tần số. Áp dụng nó trên ảnh, và kiểm tra kết quả sau khi lọc.

Matlab code 10.4.8

```
% Loc thong thap ly tuong, co bac n va tan so cat cutoff
% sze: size cua image

function f = lowpassfilter(sze, cutoff, n)
```



Hình 10.10: Đáp ứng biên độ của bộ lọc thông thấp.

```
if cutoff < 0 | cutoff > 0.5
   error('cutoff frequency must be between 0 and 0.5');
end

if rem(n,1) ~= 0 | n < 1
   error('n must be an integer >= 1');
end
rows = sze(1);
cols = sze(2);

% X and Y matrices with ranges normalized to +/- 0.5
x = (ones(rows,1) * [1:cols] - (fix(cols/2)+1))/cols;
y = ([1:rows]' * ones(1,cols) - (fix(rows/2)+1))/rows;

% A matrix with every pixel = radius relative to centre
radius = sqrt(x.^2 + y.^2); .

f = 1 ./ (1.0 + (radius ./ cutoff).^(2*n)); % The filter
% End Function
```

References

Tham khảo: 2D Fast Fourier Transform with Matlab