Bài tập cuối kì

Xử lý ảnh (05/2025)

Tên sinh viên/Số thứ tự (STT)/Tỉ lệ đóng góp:
Trần Anh Toàn – STT: 31 - Tỉ lệ: 33.33%
Trịnh Minh Việt – STT: 35 - Tỉ lệ: 33.33%
Trần Lê Long Vũ – STT: 36 - Tỉ lệ: 33.33%

Chú ý:

- *Trả lời ngắn gọn, rõ ràng, kèm code. Mỗi nhóm in và nộp một bản.*
- *Báo cáo không quá 15 trang.*

Bài 1. Phân vùng ảnh

Trình bày ngắn gọn (không quá 2 trang) một phương pháp phân vùng ảnh (theo nội dung đã chuẩn bị ở các tuần trước). Tập trung vào các ý sau:

- Phương pháp
- Kết quả mô phỏng (có thể mô tả thêm dữ liệu được sử dụng nếu thấy cần thiết)
- Nhận xét

Bài 2. Laplacian Pyramid

Trong bài này ta sẽ tìm hiểu cách phân tích ảnh sử dụng Laplacian pyramid, trong đó bước đầu tiên liên quan đến Gaussian pyramid. Sinh viên sử dụng ảnh xám phù hợp để minh họa các kết quả mô phỏng theo yêu cầu dưới đây.

A. Gaussian pyramid

Xét bộ lọc 1D kí hiệu là h = [¼ - a/2, ¼, a, ¼, ¼ - a/2] (h là vector hàng). Bộ lọc 2D kí hiệu là W được xác định như sau W = h'* h ('*' là kí hiệu tích chập, h' là chuyển vị của h). Ta sẽ phân tích ảnh I thành Gaussian pyramid G như sau.

Gọi N là số mức (level) của Gaussian pyramid (N cũng là số mức của Laplacian pyramid ở phần sau). **Mức đầu tiên** của Gaussian pyramid chính là ảnh gốc I, tức là G_1 = I. Sau đó, G_1 được lọc bởi bộ lọc W ở trên. Ảnh thu được sau khi lọc, được lấy mẫu xuống hai lần theo mỗi phương ngang/dọc (ví dụ, theo mỗi phương, cứ hai pixel thì giữ lại 1 pixel). Ảnh thu được sau khi lấy mẫu xuống được gọi là **mức 2** của Gaussian pyramid (G_2).

Một cách tương tự, G_2 được lọc bởi W và lấy mẫu xuống để tạo ra G_3, ứng với **mức 3** của Gaussian pyramid. Quá trình này được tiếp tục thực hiện đến **mức N**, và tạo ra Gaussian pyramid N mức của ảnh gốc I.

Yêu cầu*:*

1 *Mô phỏng quá trình xây dựng Gaussian pyramid trên với N = 5, a = 0.4. Hiển thị ảnh tại các mức của Gaussian pyramid.*

2 *W là bộ lọc loại gì? Giải thích vai trò của W trong quá trình trên.*

3 *Trong công thức W = h'* h, nếu thay phép tích chập bằng phép nhân (ma trận) thì kết quả của quá trình mô phỏng trên có thay đổi không? Giải thích.*

4 *Để thực hiện phép lọc, ta nên dùng trực tiếp ma trận W hay dùng lần lượt h và h'? Giải thích.*

B. Laplacian pyramid

Laplacian pyramid được tạo ra từ Gaussian pyramid ở trên. Gọi L_1, L_2,.., L_N là ảnh tại các mức 1,2,..N của Laplacian pyramid. L_i được tính như sau:

$L_i = G_i - UP\{G_{i+1}\}$, với i = 1, 2, .., N-1, và $L_N = G_N$.

Trong đó, *UP* là phép lấy mẫu lên để G_{i+1} có cùng kích thước với G_i. Để đơn giản, ta sử dụng hàm *resize* có sẵn để thực hiện phép toán này.

Yêu cầu:

1 *Hiển thị ảnh tại các mức của Laplacian pyramid.*

2 *Laplacian pyramid thể hiện thông tin gì của ảnh gốc I ? Nhận xét.*

3 *Đề xuất một cách để khôi phục ảnh gốc nếu ta chỉ có Laplacian pyramid. So sánh ảnh khôi phục này với ảnh gốc. Nhận xét.*

Bài 3. Phổ của ảnh tự nhiên

Gọi $|S_I(u,v|$ là phổ biên độ của ảnh xám I(x,y) với I(x,y) là một ảnh tự nhiên. Ta tìm $|S(w)|$ như sau

$$|S(w)| = \sum_{u,v \text{ với } w<\sqrt{u^2+v^2}\le w+dw} |S_I(u,v)|.$$

Hay nói một cách khác, $|S(w)|$ là tổng các giá trị $|S_I(u,v)|$ với (u,v) là các điểm thuộc miền giới hạn giữa hai đường tròn đồng tâm có bán kính lần lượt là w và $w+dw$, với dw là hằng số chọn trước, w bắt đầu từ 0. Để thuận tiện, ta xét u,v là các tần số chuẩn hóa thuộc $[-0.5; 0.5]$.

Yêu cầu:

1 *Mô phỏng đường cong $|S(w)|$ theo w với một ảnh xám nào đó*

2 *Mô phỏng đường cong $|S(w)|$ với một vài ảnh khác nhau. Nhận xét.*

---Hết---

Báo cáo bài tập cuối kì

Bài 1. Phân vùng ảnh

1. Phương pháp: Phân vùng ảnh dùng **K-means Clustering**
   K-means Clustering là phương pháp lượng tử hóa vector dùng để phân các điểm dữ liệu cho trước vào các cụng khác nhau
   Các bước thực hiện:
   - Bước 1: Tạo các điểm trung tâm ngẫu nhiên
     Ở bước khởi tạo này, cần chọn số lượng cụm (k) mà chúng ta muốn phân chia dữ liệu. Sau đó, chúng ta sẽ chọn ngẫu nhiên k điểm làm trung tâm cụm (centroid) ban đầu.
     Tập hợp các điểm centroid được biểu diễn như sau:
     $$\mathbb{C}^{(0)} = \left\{ m_1^{(0)},\ m_2^{(0)},\ \ldots,\ m_k^{(0)} \right\}$$
   - Bước 2: Gán các điểm dữ liệu vào các cụm
     Với mỗi điểm dữ liệu, ta sẽ tính khoảng cách của nó tới các trung tâm bằng khoảng cách Euclid. Ta sẽ gán chúng vào trung tâm gần nhất. Tập hợp các điểm được gán vào cùng 1 trung tâm sẽ tạo thành một cụm được biểu diễn như sau:
     $$\mathbb{S}_i^{(t)} = \left\{ x_p : \| x_p - m_i^{(t)} \|^2 \leq \| x_p - m_j^{(t)} \|^2 \right\},\ \forall j,\ 1 \leq j \leq k$$
   - Bước 3: Cập nhật trung tâm
     Với mỗi cụm đã tìm được ở bước 2, trung tâm mới sẽ là trung bình cộng của các điểm dữ liệu trong cụm đó.
     $$m_i^{(t+1)} = \frac{1}{\left| \mathbb{S}_i^{(t)} \right|} \sum_{x \in \mathbb{S}_i^{(t)}} x_j$$
   - Bước 4: Lặp lại bước 2 và 3 đến khi kết thúc
     Các bước 2 và 3 được lặp lại cho đến khi một trong các điều kiện hội tụ sau được thỏa mãn:
     - Các trung tâm cụm không thay đổi đáng kể
     - Các điểm dữ liệu không thay đổi cụm
     - Đạt đến số lần lặp tối đa

     Khi một trong các điều kiện trên được đáp ứng, thuật toán K-means sẽ dừng lại, và chúng ta sẽ có một tập hợp các cụm với các trung tâm cố định. Các cụm này đại diện cho các nhóm tự nhiên trong dữ liệu dựa trên độ tương đồng giữa các điểm.
2. Kết quả mô phỏng
   [Code]:

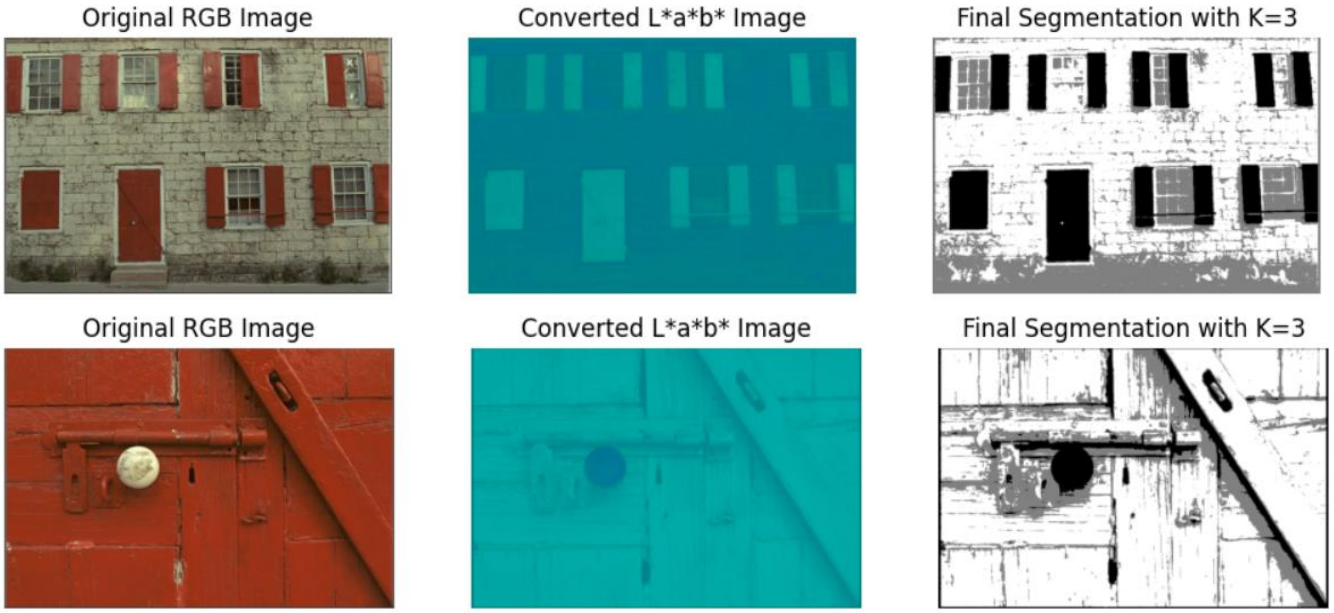
```
import cv2
import matplotlib.pyplot as plt
import numpy as np
from scipy.signal import convolve2d
import math
```

```
image_path = r'D:\py\Digital_Image_Processing\Image\6.png'
I = cv2.imread(image_path, cv2.IMREAD_COLOR_RGB)

r, g, b = I[:,:,0] / 255.0, I[:,:,1] / 255.0, I[:,:,2] / 255.0

def gamma_correct(c):
    return np.where(c > 0.04045, ((c + 0.055) / 1.055) ** 2.4, c / 12.92)

R = gamma_correct(r)
G = gamma_correct(g)
B = gamma_correct(b)

X = 0.4124 * R + 0.3567 * G + 0.1805 * B
Y = 0.2126 * R + 0.7152 * G + 0.0722 * B
Z = 0.0193 * R + 0.1192 * G + 0.9505 * B


Xn, Yn, Zn = 0.95047, 1.00000, 1.08883
x = X / Xn
y = Y / Yn
z = Z / Zn

def f(t):
    delta = 6/29
    return np.where(t > delta**3, t ** (1/3), (t / (3 * delta**2)) + 4/29)

fx = f(x)
fy = f(y)
fz = f(z)

L = 116 * fy - 16
a = 500 * (fx - fy)
b = 200 * (fy - fz)

L = np.zeros_like(fy)

lab_image = np.stack([L, a, b], axis=-1)

L_vis = np.clip(L * (255 / 100), 0, 255)
a_vis = np.clip(a + 128, 0, 255)
b_vis = np.clip(b + 128, 0, 255)
lab_vis = np.stack([L_vis, a_vis, b_vis], axis=-1).astype(np.uint8)

def kmeans_manual (data, K,  max_iters = 100, tol = 1e-4):
```

```
    N, D = data.shape

    np.random.seed(42)
    initial_indices = np.random.choice(N, K, replace = False)
    centeroids = data[initial_indices]

    for iter in range(max_iters):
        distance = np.linalg.norm(data[:, np.newaxis] - centeroids, axis=2)
        labels = np.argmin(distance, axis=1)

        new_centeroid = np.zeros((K, D))
        for k in range(K):
            clustered_point = data[labels == k]
            if len(clustered_point) > 0:
                new_centeroid[k] = np.mean(clustered_point, axis = 0)
            else:
                new_centeroid[k] = data[np.random.choice(N)]

        if np.linalg.norm(new_centeroid - centeroids) < tol:
            break

        centeroids = new_centeroid

    return labels, centeroids

ab = np.stack([a, b], axis=-1)
h, w, _ = ab.shape
ab_flat = ab.reshape((-1, 2))

labels, centroids = kmeans_manual(ab_flat, K=3)

segmentation = labels.reshape((h, w))

plt.figure(figsize=(12, 6))
plt.subplot(1, 3, 1)
plt.imshow(I)
plt.title("Original RGB Image")
plt.axis('off')

plt.subplot(1, 3, 2)
plt.imshow(lab_vis, cmap="gray")
plt.title("Converted L*a*b* Image")
plt.axis('off')

plt.subplot(1, 3, 3)
```

```
plt.imshow(segmentation, cmap='gray')
plt.axis('off')
plt.title(f"Final Segmentation with K={K}")
```

[Kết quả]:



[Mô tả thêm]:

Ở đây trước khi thực hiện thuật toán K-means, nhóm đã thực hiện chuyển không gian màu từ RGB sang Lab với mục đích:

- Tránh sự ảnh hưởng của độ sáng, sẽ làm cho việc áp dụng K-means Clustering đúng hơn
- Không gian màu L*a*b có tính tuyến thị giác cao

Chuẩn bị dữ liệu đầu vào:

- Lấy kênh a và b biểu diễn đặc trưng màu, loại bỏ kênh L
- Biến đổi thành vector phẳng (N, 2)
- Mục đích: Chuyển ảnh từ dạng không gian 2D sang dạng tập dữ liệu 2 chiều phù hợp để thuật toán K-means hoạt động.

3. Nhận xét

Ưu điểm:

- Đơn giản và dễ hiểu
- Khả năng mở rộng tốt
- Tạo ra các cụm riêng biệt, không chồng chéo

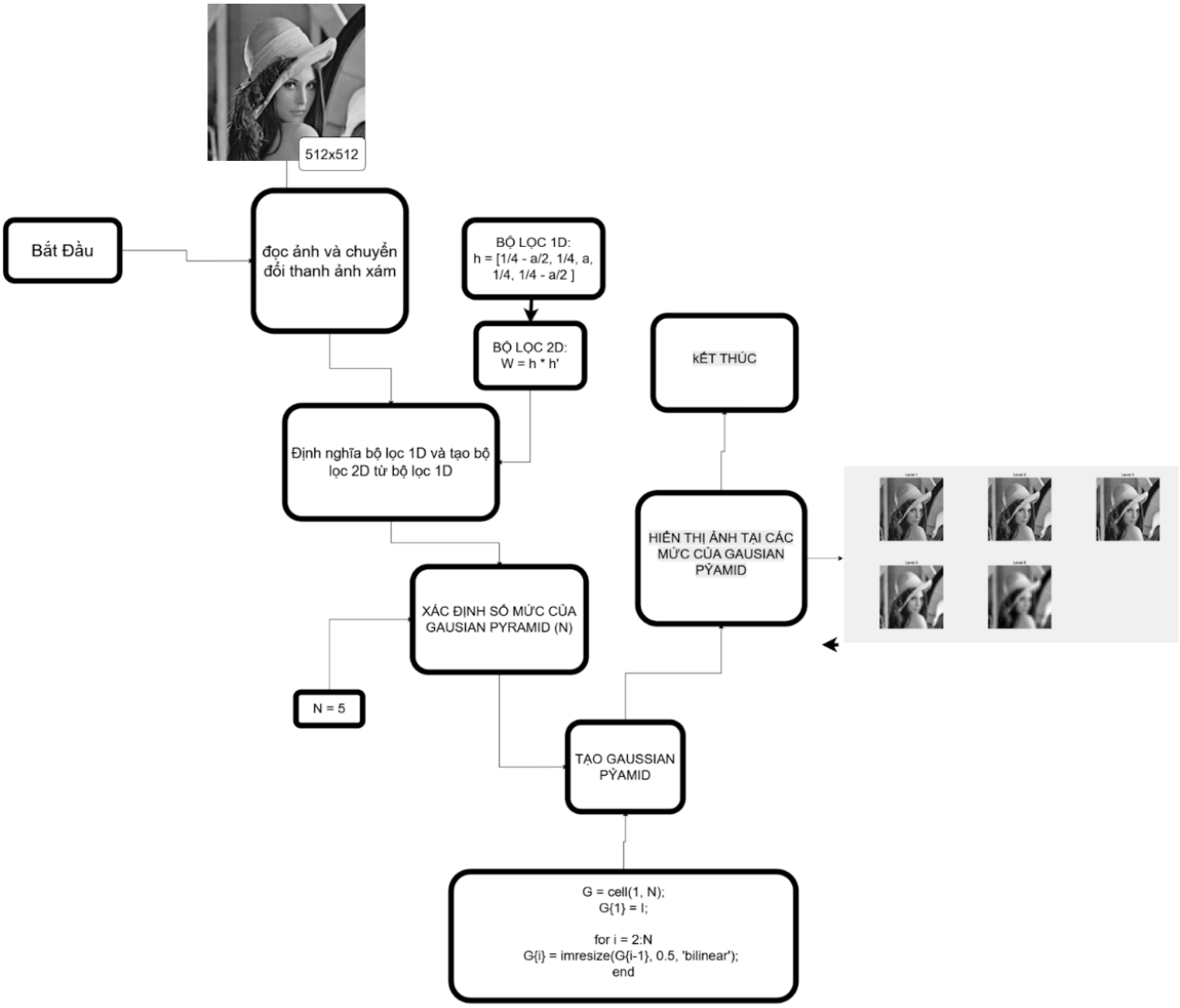
Nhược điểm:

- Phụ thuộc vào số cụm (k)
- Nhạy cảm với việc khởi tạo trung tâm ngẫu nhiên
- Nhạy cảm với các điểm ngoại lai (outliers)

Bài 2. Laplacian Pyramid

A. Gaussian pyramid

1. *Mô phỏng quá trình xây dựng Gaussian pyramid trên với N = 5, a = 0.4. Hiển thị ảnh tại các mức của Gaussian pyramid.*



[Code]:

```
clc; clear; close all;
I = imread('lena_gray.bmp');
I = im2double(I);
a = 0.4;
N = 5;
h = [0.25 - a/2, 0.25, a, 0.25, 0.25 - a/2];
W = h' * h;
G{1} = I;
for k = 2:N
```

```
    G{k} = imfilter(G{k-1}, W, 'symmetric', 'same');
    G{k} = G{k}(1:2:end, 1:2:end);
end
figure('Name','Gaussian Pyramid Levels');
for k = 1:N
    subplot(2,3,k);
    imshow(G{k}, []);
    title(['Level ', num2str(k)]);
end
```

[Hiển thị ảnh tại các mức của Gaussian pyramid với N=5 và a=0.4]:

2. *W là bộ lọc loại gì? Giải thích vai trò của W trong quá trình trên.*

- W là bộ lọc Gaussian 2D và là bộ lọc thông thấp (low-pass filter).
- **Vai trò của W:**
    - Làm mờ, lọc các thành phần tần số cao, làm mượt ảnh giúp loại bỏ nhiễu và làm mịn các chi tiết nhỏ trong ảnh.
    - Việc làm mờ mỗi mức trước khi lấy mẫu xuống. Điều này giúp bảo toàn các đặc trưng quan trọng của ảnh ở mỗi mức thấp hơn, tránh mất mát thông tin quá lớn khi giảm kích thước.

3. *Trong công thức W = h'* h, nếu thay phép tích chập bằng phép nhân (ma trận) thì kết quả của quá trình mô phỏng trên có thay đổi không? Giải thích*

- Vì phép nhân của một vector hàng 1D và một vector cột 1D sẽ tương đồng với phép tích chập 2D của vector hàng và vector cột đó hay h'*h=h'x h. Nên kết quả sẽ không thay đổi

4. *Để thực hiện phép lọc, ta nên dùng trực tiếp ma trận W hay dùng lần lượt h và h'? Giải thích.*

- Ta nên dùng lần lượt h và h' thay vì sử dụng trực tiếp ma trận W.

- Bởi vì bộ lọc W=h′∗h là có thể tách được, nên ta có thể thực hiện phép lọc bằng cách lọc theo hàng với h, sau đó lọc theo cột với h’ mà vẫn giữ nguyên kết quả.
- Nếu sử dụng lần lượt h và h’, tốc độ xử lý sẽ nhanh hơn. Lý do là:
    - Bộ lọc W có kích thước 5×5 ,tức là cần 25 phép tính cho mỗi pixel.
    - Trong khi đó, lọc lần lượt theo h và h’ chỉ cần 5 + 5 = 10 phép tính cho mỗi pixel.
- Do đó, hiệu suất xử lý được cải thiện, trong khi kết quả lọc vẫn chính xác như dùng W trực tiếp.

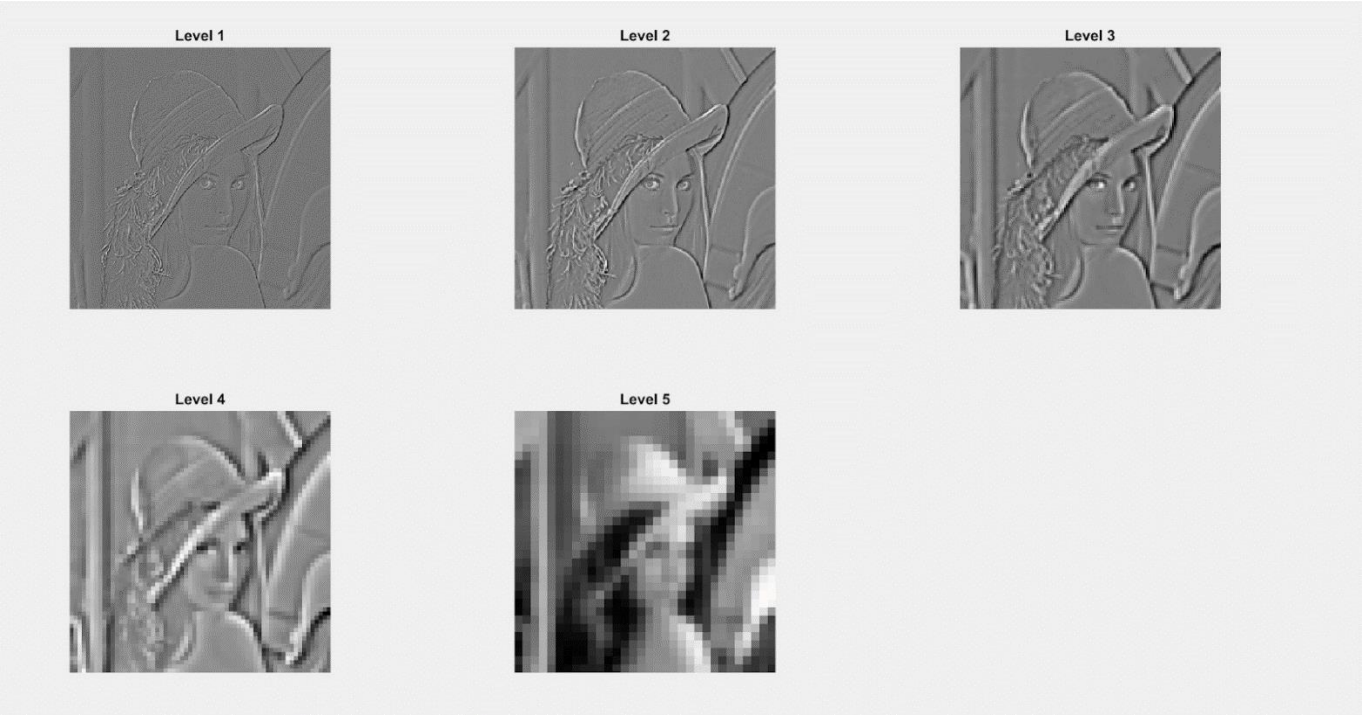
B. Laplacian pyramid

1. *Hiển thị ảnh tại các mức của Laplacian pyramid*

[Code]:

```
for k = 1:N-1
    Gi_up = imresize(G{k+1}, size(G{k}), 'bilinear');
    L{k} = G{k} - Gi_up;
end
L{N} = G{N};
figure('Name','Laplacian Pyramid Levels');
for k = 1:N
    subplot(2,3,k);
    imshow(L{k}, []);
    title(['Level ', num2str(k)]);
end
```

[Hiển thị ảnh tại các mức của Gaussian pyramid với N=5 và a=0.4]:



2. *Laplacian pyramid thể hiện thông tin gì? Nhận xét*
    - Mỗi tầng Laplacian biểu diễn sự khác biệt giữa ảnh Gaussian hiện tại và ảnh ở tầng tiếp theo đã được nội suy.
    - Các tầng thấp chứa thông tin tần số cao như cạnh sắc, viền và chi tiết nhỏ.
    - Các tầng giữa giữ lại kết cấu chi tiết, mẫu lặp lại và biến thiên nhẹ.
    - Tầng cao nhất chính là ảnh Gaussian cuối cùng, chứa thông tin tần số thấp và biểu diễn cấu trúc tổng quát của ảnh.

    [Nhận xét]:

    - L1: Cạnh và viền rõ nhất, chứa nhiều chi tiết sắc nét.
    - L2: Cạnh mềm hơn, bắt đầu thấy kết cấu và đặc trưng lớn hơn.
    - L3: Chi tiết mờ hơn, xuất hiện vùng lớn và răng cưa.
    - L4: Chủ yếu còn lại cấu trúc lớn, hình ảnh rất mịn.
    - L5: Chỉ còn biến đổi tổng quát về sáng tối và bố cục ảnh.

    → Laplacian pyramid phân tách ảnh theo mức độ chi tiết từ cao đến thấp. Tầng càng cao thì độ phân giải và chi tiết càng ít, chỉ giữ lại thông tin tổng quát của ảnh gốc.
3. *Đề xuất một cách để khôi ảnh gốc nếu ta chỉ có Laplacian pyramid. So sánh ảnh khôi phục này với ảnh gốc. Nhận xét*

    [Cách khôi phục]:
    - Lấy tầng trên cùng của Laplacian pyramid (tầng có độ phân giải thấp nhất) làm ảnh gốc khôi phục đầu tiên.
    - Từ tầng cuối cùng đến tầng đầu tiên của Laplacian pyramid, thực hiện các bước sau cho mỗi tầng:
        - Điều chỉnh kích thước ảnh tầng hiện tại lên đúng kích thước của ảnh tầng trước đó đã được giảm độ phân giải.
        - Cộng ảnh tầng hiện tại với ảnh tầng trước đó đã được giảm độ phân giải, nhằm khôi phục các chi tiết đã bị mất trong quá trình giảm độ phân giải.
    - Sau khi đã tái tạo tới tầng đầu tiên của Laplacian pyramid, ta sẽ thu được ảnh gốc ban đầu.

    [Code]:

```
N = length(L);
R = cell(1, N);
R{N} = L{N};
for k = N-1:-1:1
    upsampled = imresize(R{k+1}, size(L{k}), 'bilinear');
    R{k} = upsampled + L{k};
end
figure('Name','So sánh ảnh gốc và ảnh khôi phục');
subplot(1,2,1);
imshow(I, []); title('Ảnh gốc');
subplot(1,2,2);
```

```
imshow(R{1}, []); title('Ảnh khôi phục');

if isequal(size(I), size(R{1}))
    I_ref = im2double(I);
    I_rec = im2double(R{1});

    % Tính PSNR
    mse = mean((I_ref(:) - I_rec(:)).^2);
    if mse == 0
        psnr_value = Inf;
    else
        psnr_value = 20 * log10(1 / sqrt(mse));
    end

    % Tính SSIM
    if exist('ssim', 'file')
        ssim_value = ssim(I_rec, I_ref);
    else
        ssim_value = NaN;
    end
    fprintf('PSNR: %.2f dB\n', psnr_value);
    fprintf('SSIM: %.4f\n', ssim_value);
```

[Ảnh sau khi khôi phục]:

- Giá trị PSNR: 367.84
- Giá trị SSIM: 1.000

[Nhận xét]:

- Dựa vào hình ảnh thu được sau khi khôi phục thì hình ảnh thu được có nét tương đồng so với ảnh gốc về mặc cấu trúc, màu sắc
- Với PSNR = 367.84 là một giá trị khá cao, cho thấy rằng ảnh phục hồi có chất lượng gần như tuyệt đối
- Với SSIM = 1 cho thấy ảnh phục hồi gần như tương đồng tuyệt đối với ảnh gốc về mặt cấu trúc, độ sáng và độ tương phản.

→ Ảnh khôi phục và ảnh gốc có độ tương đồ rất cao cho thấy quá trình khôi phục ảnh được thực hiện chính xác.

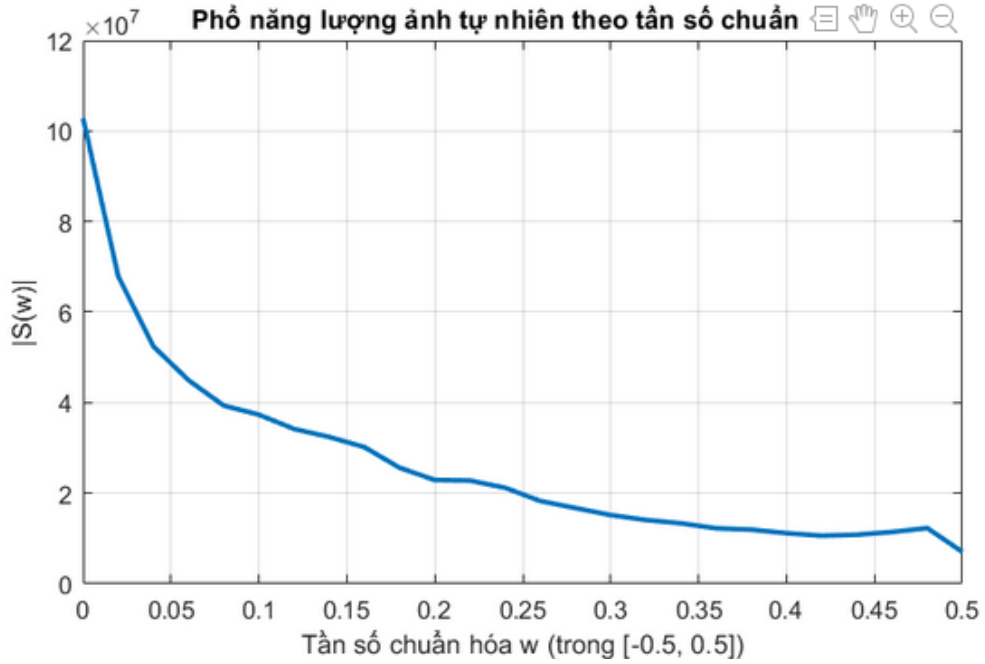
Bài 3. Phổ của ảnh tự nhiên

1. *Mô phỏng đường cong $|S(w)|$ theo w với một ảnh xám nào đó*

[Code]:

```
I = imread('XLA.jpeg');
I = rgb2gray(I);
I = im2double(I);

F = fftshift(fft2(I));
S = abs(F);

[M, N] = size(I);
[u, v] = meshgrid(linspace(-0.5, 0.5, N), linspace(-0.5, 0.5, M));
radius = sqrt(u.^2 + v.^2);
dw = 0.02;
w_range = 0:dw:0.5;
Sw = zeros(size(w_range));

for i = 1:length(w_range)
    w = w_range(i);
    mask = (radius > w) & (radius <= w + dw);
    Sw(i) = sum(S(mask));
end

plot(w_range, Sw, 'LineWidth', 2);
xlabel('f');
ylabel('|S(w)|');
title('Pho Nang Luong');
grid on;
```

[Kết quả]:



2. *Mô phỏng đường cong $|S(w)|$ với một vài ảnh khác nhau. Nhận xét*


1.png


13.png

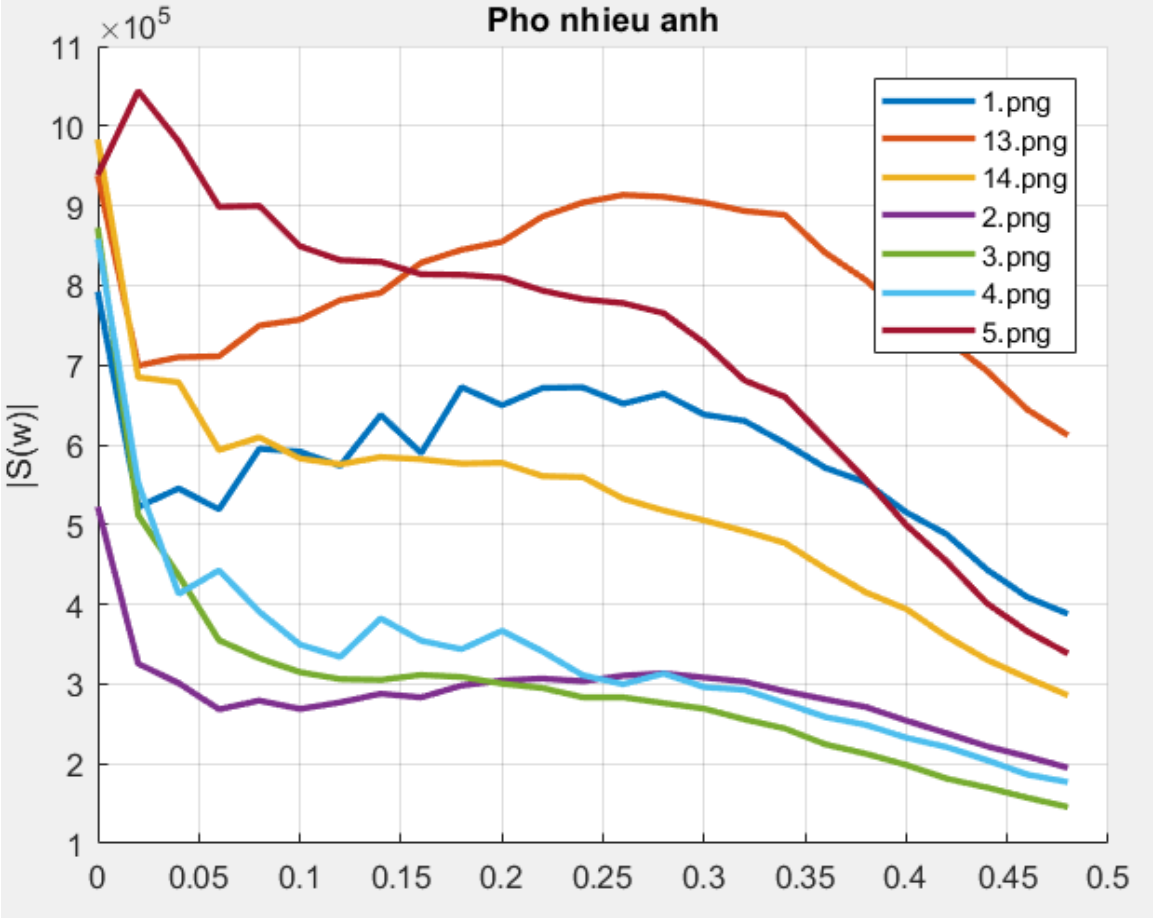

14.png


2.png


3.png


4.png


5.png


Pho nhieu anh
×10^5
11
10
9
8
7
6
5
4
3
2
1
|S(w)|
0
0.05
0.1
0.15
0.2
0.25
0.3
0.35
0.4
0.45
0.5
1.png
13.png
14.png
2.png
3.png
4.png
5.png

[Nhận xét]:

- **Lưu ý: ảnh trên được mô phỏng trên tập ảnh kodims**
- Trong miền tần số của ảnh, mỗi mức tần số mang một ý nghĩa khác nhau đối với đặc trưng thị giác. Tần số thấp đại diện cho những vùng ảnh thay đổi chậm về cường độ, chẳng hạn như các vùng đều màu, các mảng sáng tối lớn hoặc bố cục tổng thể của ảnh. Đây là nơi chứa phần lớn năng lượng trong các ảnh tự nhiên. Ngược lại, tần số trung bình thường thể hiện các chi tiết vừa phải như đường nét, hoa văn nhẹ hoặc biên mềm. Đây là vùng chứa nhiều thông tin đặc trưng của vật thể mà không quá sắc nét. Trong khi đó, tần số cao phản ánh những thay đổi nhanh về cường độ sáng – điển hình là các cạnh sắc, chi tiết nhỏ như chữ viết, đường gãy, hoặc thậm chí là nhiễu. Chẳng hạn như ảnh kodim13 có nhiều chi tiết hoa văn, đường nét nhiều nên tần số trung bình khoảng từ 0.2 tới 0.4 ngược lại kodim3 có nền background trùng màu lớn dẫn tới tần số thấp có giá trị lớn .Việc phân biệt rõ các dải tần số giúp ta hiểu được cấu trúc ảnh, từ đó áp dụng hiệu quả vào các bài toán như nén ảnh, lọc nhiễu hay trích xuất đặc trưng.