Chương 1 — BIỂU DIỄN ẢNH (chuyển sang Python)

1.1 Đọc, hiển thị và truy xuất pixel; phép cộng/trừ trên pixel

Yêu cầu:

- Đọc ảnh `cell.tif`, in giá trị pixel tại vị trí (i=100, j=20).
- Thực hiện `I[i,j] = I[i,j] + 25` và `I[i,j] = I[i,j] - 25`, quan sát ảnh xám (với clipping 0..255).

Mẫu code (vectorized & single-pixel):

```
import cv2
import numpy as np
import matplotlib.pyplot as plt

I = cv2.imread('cell.tif', cv2.IMREAD_GRAYSCALE)
i, j = 100, 20
print("Original:", I[i,j])

# single pixel add/sub
I_plus = I.copy()
I_plus[i,j] = np.clip(int(I_plus[i,j]) + 25, 0, 255)

I_minus = I.copy()
I_minus[i,j] = np.clip(int(I_minus[i,j]) - 25, 0, 255)

plt.figure(figsize=(8,3))
plt.subplot(1,3,1); plt.imshow(I, cmap='gray'); plt.title('Original')
plt.subplot(1,3,2); plt.imshow(I_plus, cmap='gray'); plt.title('+25 at
(100,20)')
plt.subplot(1,3,3); plt.imshow(I_minus, cmap='gray'); plt.title('-25 at
(100,20)')
plt.show()
```

1.2 Lặp cho tất cả pixel (tạo I1, I2) — chú ý clipping

Yêu cầu: tạo `I1 = I + 25` và `I2 = I - 25` cho toàn ảnh (không cho vượt 0..255).
Gợi ý (vectorized, không dùng vòng for):

```
I1 = np.clip(I.astype(np.int16) + 25, 0, 255).astype(np.uint8)
I2 = np.clip(I.astype(np.int16) - 25, 0, 255).astype(np.uint8)
```

(hoặc dùng `cv2.add(I, 25)` / `cv2.subtract(I, 25)` — cv2 tự clip.)

1.3 Ghi ảnh dưới JPEG và PNG, so sánh

Đọc ảnh “cameraman.tif” và dùng hàm imwrite để ghi lại một ảnh có định dang JPEG (chẳng hạn: Ijpg.jpg) và một ảnh có định dạng PNG (chẳng hạn: Ipng.png). Sau đó đọc hai ảnh Ijpg.jpg và Ipng.png vào hai biến tương ứng Ijpg và Ipng.

- Điều chúng ta mong đợi là hai ảnh trên giống nhau hoàn toàn? Nếu chúng ta so sánh chúng bằng cách trừ tương ứng từng pixel của hai ảnh cho nhau thì liệu này có đúng không?

-Chúng ta thấy rằng sự khác nhau giữa chúng không phải là zero như ta mong đợi. Vì, định dạng JPEG là nén mất thông tin (lossy compression) còn định dạng png là nén không mất thông tin (lossless compression).

Mẫu code:

```
A = cv2.imread('cameraman.tif', cv2.IMREAD_GRAYSCALE)
cv2.imwrite('Ijpg.jpg', A)   # default lossy JPEG
cv2.imwrite('Ipng.png', A)   # lossless

Ijpg = cv2.imread('Ijpg.jpg', cv2.IMREAD_GRAYSCALE)
Ipng = cv2.imread('Ipng.png', cv2.IMREAD_GRAYSCALE)
X = cv2.absdiff(Ijpg, Ipng)
plt.imshow(X, cmap='gray'); plt.title('absdiff'); plt.show()
```

Chương 2 — NÂNG CAO CHẤT LƯỢNG ẢNH

2.1 Xử lý ảnh đa mức xám (`breast_digital_Xray`)

Cho một ảnh đa mức xám r=’ Fig0304(a)(breast_digital_Xray).tif’.

Yêu cầu:
a. Vẽ histogram (phân bố mức xám).
b. Viết code biến đổi âm bản `Sa = L-1 - r` (với L=256).
c. So sánh histogram của `Sa` và `r`.
d. Ngưỡng `t=127` để tạo ảnh nhị phân.

Mẫu code:

```
from skimage import io, exposure
r = io.imread('Fig0304(a)(breast_digital_Xray).tif', as_gray=False)
# histogram
plt.figure(); plt.hist(r.ravel(), bins=256); plt.title('hist'); plt.show()

# negative
Sa = 255 - r
# threshold
_, bw = cv2.threshold(r, 127, 255, cv2.THRESH_BINARY)
```

2.2 Bit-plane slicing (`fractal-iris`)

Cho ảnh đa mức xám với I = 'Fig0122(a)(fractal-iris).tif'

Yêu cầu: tạo ảnh từ bit plane 3, 6; tạo ảnh chứa bit 7 & 8.
Mẫu code:

```
i = cv2.imread('Fig0122(a)(fractal-iris).tif', cv2.IMREAD_GRAYSCALE)
bit3 = ((i >> 2) & 1) * 255   # bit index: 1..8 -> shift by (k-1)
bit6 = ((i >> 5) & 1) * 255
# bit7 + bit8 combined
i78 = (((i >> 6) & 1) << 6) | (((i >> 7) & 1) << 7)
# or set bits into zeros-image
i67 = np.zeros_like(i)
i67 = np.bitwise_or(i67, ((i >> 5) & 1) << 5)
i67 = np.bitwise_or(i67, ((i >> 6) & 1) << 6)
```

2.3 Histogram equalization (`Fig0316(4)(bottom_left).tif`)

Yêu cầu: xem ảnh, histogram; cân bằng histogram bằng `histeq`.
Mẫu code:

```
from skimage import exposure
i = io.imread('Fig0316(4)(bottom_left).tif', as_gray=False)
eq = exposure.equalize_hist(i)        # trả về float 0..1
# nếu muốn uint8:
eq_uint8 = (eq * 255).astype(np.uint8)
```

2.4 Phép toán số học trên ảnh (thực hành)

Các bài nhỏ: `imadd`, `imsubtract`, `imabsdiff`, nhân / chia ảnh, XOR cho ảnh nhị phân, im2bw...
Gợi ý thư viện: `cv2.add`, `cv2.subtract`, `cv2.absdiff`, `cv2.multiply` / NumPy `*`, `cv2.threshold` cho nhị phân, `np.logical_xor`.

2.5 Biến đổi logarit

Yêu cầu: áp `s = c * log(1 + Id)` với các hệ số c khác nhau; hiển thị kết quả.
Mẫu code:

```
I = cv2.imread('cameraman.tif',
cv2.IMREAD_GRAYSCALE).astype(np.float32)/255.0
out1 = 2 * np.log1p(I)
out2 = 3 * np.log1p(I)
out3 = 5 * np.log1p(I)
# scale back hoặc hiển thị trực tiếp (matplotlib sẽ tự chuẩn hóa nếu muốn)
```

2.6 — 2.7 Phân tích & chọn ngưỡng

Yêu cầu: vẽ histogram `imhist` tương đương, chọn ngưỡng cho `pillsetc.png`, `tape.png`, `coins.png`, `eight.tif`.
Gợi ý: dùng `plt.hist` + `cv2.threshold` (global), hoặc `skimage.filters.threshold_otsu` để tự động.

2.8 Phép trừ ảnh với zeroing bit planes

Yêu cầu:
a. Tạo `h` bằng cách đặt 4 bit thấp của `f` bằng 0.
b. `g = f - h`.
c. Cân bằng histogram `g`.

Mẫu code:

```
f = cv2.imread('Fig0122(a)(fractal-iris).tif', cv2.IMREAD_GRAYSCALE)
mask = 0xF0   # giữ 4 bit cao, đặt 4 bit thấp = 0
h = f & mask
g = cv2.subtract(f, h)
g_eq = exposure.equalize_hist(g)
```

2.9 Lọc tuyến tính (mean filter)

Yêu cầu: lọc ảnh `Fig0333(a)(test_pattern_blurring_orig).tif` với mặt nạ 3x3 all-ones /9.
Mẫu code:

```
w = np.ones((3,3), dtype=np.float32) / 9.0
f = cv2.imread('Fig0333(a)(test_pattern_blurring_orig).tif',
cv2.IMREAD_GRAYSCALE)
g = cv2.filter2D(f, -1, w, borderType=cv2.BORDER_DEFAULT)
```

2.10 Lọc phi tuyến — median filter

Yêu cầu:
a. Nạp `coins.png`, thêm nhiễu salt-pepper 3% (`skimage.util.random_noise`), hiển thị.
b. Lọc median (`cv2.medianBlur` hoặc `scipy.signal.medfilt2d`).
c. Gây nhiễu Gaussian 2% và lọc.

Mẫu code:

```
from skimage.util import random_noise
I = io.imread('coins.png', as_gray=True)
noisy_sp = random_noise(I, mode='s&p', amount=0.03)
noisy_sp_uint8 = (noisy_sp * 255).astype(np.uint8)
denoised = cv2.medianBlur(noisy_sp_uint8, ksize=3)
# gaussian noise
noisy_gauss = random_noise(I, mode='gaussian', var=0.02)
```

2.11 So sánh lọc tuyến tính & phi tuyến

Yêu cầu: áp 2 loại lọc trên `Fig0335(a)(ckt_board_saltpep_prob_pt05).tif` và so sánh chất lượng.

Chương 3 & 4 — PHÁT HIỆN BIÊN VÀ PHÂN ĐOẠN

3.1 Canny, LoG với Gaussian tiền lọc

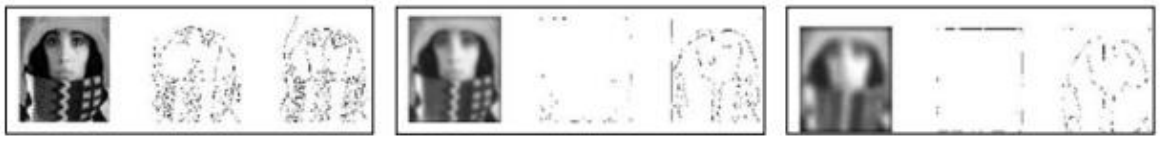
Yêu cầu: nạp ảnh `trui.png`, lọc Gaussian với sigma=6 và 12 (dùng `cv2.GaussianBlur`), sau đó phát hiện cạnh bằng LoG hoặc Canny. So sánh kết quả không lọc / lọc sigma khác nhau.
Mẫu code:

```
A = cv2.imread('trui.png', 0)
h1 = cv2.GaussianBlur(A, (15,15), 6)
h2 = cv2.GaussianBlur(A, (31,31), 12)

edges_canny = cv2.Canny(A, 100, 200)
edges_canny_h1 = cv2.Canny(h1, 100, 200)
edges_canny_h2 = cv2.Canny(h2, 100, 200)

# LoG: có thể dùng skimage.filters.laplace + gaussian
from skimage.filters import laplace
from scipy.ndimage import gaussian_filter
log = laplace(gaussian_filter(A, sigma=6))
```

*Kết quả



không lọc | logarit | Canny | σ=6 | logarit | Canny | σ=12 | logarit | Canny

3.2 Phát hiện biên với mặt nạ tuỳ chỉnh

- Các mặt nạ phát hiện biên

-1	-1	-1
2	2	2
-1	-1	-1

ngang

-1	-1	2
-1	2	-1
2	-1	-1

$+45^0$

2	-1	-1
-1	2	-1
-1	-1	2

dọc

2	-1	-1
-1	2	-1
-1	-1	2

-45^0

Yêu cầu: áp các mặt nạ (ma trận 3×3 như đề) bằng `cv2.filter2D`, quan sát kết quả, so sánh chiều ngang/dọc/±45°.

Mẫu code:

```
w = np.array([[-1,-1,-1],[2,2,2],[-1,-1,-1]], dtype=np.float32)
g = cv2.filter2D(f, -1, w)
```

3.3 So sánh Sobel / Prewitt / Roberts

Yêu cầu: dùng `cv2.Sobel`, `skimage.filters.prewitt`, `skimage.filters.roberts` để phát hiện biên; hiển thị và so sánh.
Mẫu code:

```
import cv2

import numpy as np

import matplotlib.pyplot as plt

from skimage.filters import prewitt, roberts

# ==========================

# 1. Đọc ảnh mức xám

# ==========================

image_path = "Fig0316(4)(bottom_left).tif "

f = cv2.imread(image_path, cv2.IMREAD_GRAYSCALE)
```

```
if f is None:
    raise Exception("Không tìm thấy ảnh, hãy kiểm tra lại đường dẫn !")

# ==========================
# 2. Sobel (tính magnitude)
# ==========================
sobelx = cv2.Sobel(f, cv2.CV_64F, 1, 0, ksize=3)
sobely = cv2.Sobel(f, cv2.CV_64F, 0, 1, ksize=3)
sobel_mag = np.sqrt(sobelx**2 + sobely**2)
sobel_mag = (sobel_mag / sobel_mag.max() * 255).astype(np.uint8)
# ==========================
# 3. Prewitt
# ==========================
pre = prewitt(f)
pre = (pre / pre.max() * 255).astype(np.uint8)
# ==========================
# 4. Roberts
# ==========================
rob = roberts(f)
rob = (rob / rob.max() * 255).astype(np.uint8)
# ==========================
# 5. Hiển thị kết quả
# ==========================
plt.figure(figsize=(12,8))

plt.subplot(2,2,1)
plt.imshow(f, cmap='gray')
plt.title("Ảnh gốc")
plt.axis("off")

plt.subplot(2,2,2)
```

```
plt.imshow(sobel_mag, cmap='gray')
plt.title("Biên Sobel (magnitude)")
plt.axis("off")

plt.subplot(2,2,3)
plt.imshow(pre, cmap='gray')
plt.title("Biên Prewitt")
plt.axis("off")

plt.subplot(2,2,4)
plt.imshow(rob, cmap='gray')
plt.title("Biên Roberts")
plt.axis("off")

plt.tight_layout()
plt.show()
```