LAB-05: Image Filtering

Bài 1 – So sánh các bộ lọc làm mịn ảnh (Average, Gaussian, Median)

Mục tiêu

- Hiểu và áp dụng các bộ lọc làm mịn ảnh cơ bản.
- So sánh sự khác biệt giữa Average filter, Gaussian filter, Median filter.
- Nhận xét ưu, nhược điểm của từng loại trong xử lý ảnh nhiễu.

Yêu cầu:

1. Áp dụng Average filter (mean filter) với kernel 3×3, 5×5.

```
# Áp dụng Average filter với kernel 3x3 và 5x5
avg3 = cv2.blur(img, (3,3))
avg5 = cv2.blur(img, (5,5))
```

2. Áp dụng Gaussian filter với σ = 1, kernel size = 3, 5, 9.

```
# Áp dụng Gaussian filter với σ = 1 và các kernel size khác nhau
gauss3 = cv2.GaussianBlur(img, (3,3), sigmaX=1)
gauss5 = cv2.GaussianBlur(img, (5,5), sigmaX=1)
gauss9 = cv2.GaussianBlur(img, (9,9), sigmaX=1)
```

3. Áp dụng Median filter với kernel 3×3, 5×5.

```
# Áp dụng Median filter với kernel 3x3 và 5x5
med3 = cv2.medianBlur(img, 3)
med5 = cv2.medianBlur(img, 5)
```

4. Hiển thị ảnh kết quả và so sánh.

Bài 2 – Ảnh hưởng của kích thước kernel trong Gaussian Blur

Mục tiêu

- Hiểu rõ vai trò của kích thước kernel trong Gaussian Blur.
- Quan sát sự thay đổi mức độ mờ khi thay đổi kích thước kernel với cùng giá trị σ.

Yêu cầu

1. Áp dụng Gaussian Blur cho một ảnh bất kỳ với:

- σ = 1
- Kernel size = 3, 5, 9, 15

2. Hiển thị ảnh gốc và ảnh sau khi làm mờ.
3. So sánh kết quả và viết nhận xét về ảnh hưởng của kích thước kernel đến độ mờ.

Bài 3 – Khử nhiễu ảnh bằng bộ lọc (Noise Removal)

Mục tiêu

- Làm quen với các loại nhiễu ảnh thường gặp: Gaussian noise, Salt & Pepper noise.
- So sánh hiệu quả của các bộ lọc khác nhau trong khử nhiễu.

Yêu cầu

1. Thêm **Gaussian noise** và **Salt & Pepper noise** vào ảnh gốc.
   - **Gaussian noise**: ảnh có hạt nhỏ trắng/xám phân bố đều trên toàn ảnh.
   - **Salt & Pepper noise**: ảnh có nhiều điểm **đen hoàn toàn** hoặc **trắng hoàn toàn** rải rác.

2. Khử nhiễu bằng:
   - Gaussian Blur
   - Median Filter
3. Hiển thị kết quả so sánh.
4. Đánh giá bằng PSNR, SSIM.

Gợi ý code:

```
# ========================
# 1. Gaussian Noise
# ========================
mean = 0
sigma = 25  # điều chỉnh mức độ nhiễu
gaussian_noise = np.random.normal(mean, sigma, img_gray.shape).astype(np.float32)

gaussian_noisy_img = img_gray.astype(np.float32) + gaussian_noise
gaussian_noisy_img = np.clip(gaussian_noisy_img, 0, 255).astype(np.uint8)
```

```
# ========================
# 2. Salt & Pepper Noise
# ========================
prob = 0.02  # tỉ lệ nhiễu
sp_noisy_img = img.copy()

# Mặt nạ ngẫu nhiên
rnd = np.random.rand(*img.shape)

# Pixel = 0 (muối)
sp_noisy_img[rnd < (prob / 2)] = 0
# Pixel = 255 (tiêu)
sp_noisy_img[rnd > 1 - (prob / 2)] = 255
```

Bài 4 – Làm mịn và làm sắc nét ảnh (Smoothing & Sharpening)

Mục tiêu

- Hiểu sự khác biệt giữa bộ lọc làm mịn (smoothing) và làm sắc nét (sharpening).
- Thực hành áp dụng Gaussian Blur, Average Blur để giảm nhiễu.
- Tạo bộ lọc sharpening bằng công thức Unsharp Masking.
- Quan sát ảnh trước/sau để thấy rõ tác động.

Yêu cầu:

1. Thực hiện smoothing:
    - Gaussian Blur với kernel 3×3, 5×5.
    - Average Blur với kernel 3×3, 5×5.
2. Tạo bộ lọc sharpening bằng công thức Unsharp Masking:

$$I_{\text{sharp}} = I + \alpha(I - I_{\text{blur}})$$

trong đó:

- I: ảnh gốc
- I_{blur}: ảnh đã làm mịn
- α: hệ số tăng cường chi tiết (thường 0.5 → 2).

3. Thử với nhiều giá trị α (ví dụ: 0.5, 1.0, 2.0).
4. So sánh ảnh trước và sau xử lý

Bài 5 – Làm nổi bật chi tiết (Detail Enhancement)

Mục tiêu

- Hiểu cách tách chi tiết (high-frequency) và kết hợp lại với ảnh gốc.
- Ứng dụng trong tăng cường chi tiết ảnh.

Yêu cầu:

1. Làm mờ ảnh bằng Gaussian Blur.
2. Lấy phần chi tiết:

$$Detail = I - I_{blur}$$

3. Tạo ảnh tăng cường chi tiết:

$$I_{enhanced} = I + \alpha \cdot Detail$$

4. So sánh kết quả với α = 0.5, 1.0, 2.0.

Bài 6 – Ảnh hưởng của Padding trong Convolution

Mục tiêu

- Hiểu rõ vai trò của **padding** khi áp dụng convolution.
- Quan sát ảnh hưởng của các chế độ padding khác nhau lên kết quả lọc ảnh.

Yêu cầu:

1. Cài đặt các chế độ padding:
    - **Zero-padding (constant = 0)**
    - **Replicate (sao chép biên)**
    - **Reflect (phản chiếu biên)**
    - **Wrap (xoay vòng/circular)**
2. Áp dụng cùng một kernel Laplacian:

$$L = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

3. So sánh kết quả thu được
4. Nhận xét ảnh hưởng của padding đến biên ảnh?

Bài 7 – Phát hiện biên (Edge Detection)

Mục tiêu

- Hiểu cách phát hiện biên ảnh bằng convolution với các kernel Sobel, Prewitt.
- Biết cách kết hợp gradient theo công thức vector.
- So sánh kết quả với thuật toán Canny trong OpenCV.
- Rút ra ưu/nhược điểm của từng phương pháp.

Yêu cầu:

1. Áp dụng **convolution** với các kernel Sobel, Prewitt theo cả hai hướng x và y.

   Sobel kernels:

   $$S_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad S_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

   Prewitt kernels:

   $$P_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}, \quad P_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

2. Kết hợp gradient theo công thức:

   $$G = \sqrt{G_x^2 + G_y^2}$$

3. So sánh với **Canny** (`cv2.Canny`).
4. Viết nhận xét: ưu điểm/nhược điểm giữa convolution thủ công, Sobel, và Canny.

Câu hỏi: Tại sao Canny thường cho biên "mảnh" và ít nhiễu hơn Sobel?

Bài 8 – Phát hiện biên bằng Laplacian of Gaussian (LoG)

Mục tiêu

- Kết hợp Gaussian blur và Laplacian để phát hiện biên.
- So sánh với Sobel và Canny.

Yêu cầu:

1. Làm mờ ảnh bằng Gaussian (σ = 1.4).
2. Áp dụng Laplacian kernel.
3. Ngưỡng hóa kết quả để tạo biên nhị phân.
4. So sánh với Sobel và Canny.

Bài tập ứng dụng thực tế

Bài 9 – Phát hiện biên bản số xe

1. Lấy một ảnh xe có biển số (ảnh màu hoặc ảnh xám).
2. Chuyển ảnh về grayscale để dễ xử lý.
3. Dùng Sobel theo cả hai hướng **x, y**, kết hợp gradient:

$$G = \sqrt{G_x^2 + G_y^2}$$

4. Áp dụng Laplacian để phát hiện biên.
5. Áp dụng Canny (`cv2.Canny`).
6. So sánh kết quả 3 phương pháp trong việc làm nổi bật khung biển số.
7. Viết nhận xét.

Bài 10 – Phát hiện đường biên trong ảnh y tế (ví dụ ảnh X-ray, MRI)

1. Lấy một ảnh y tế (X-ray hoặc MRI).
2. Chuyển ảnh về **grayscale** nếu là ảnh màu.
3. Áp dụng **Gaussian blur** để giảm nhiễu (σ=1, kernel size = 5).
4. Áp dụng các phương pháp phát hiện biên:
   - **Sobel** (theo X, Y, kết hợp gradient).
   - **Laplacian**.
   - **Canny**.
5. Hiển thị kết quả và so sánh.
6. Viết nhận xét về hiệu quả từng phương pháp khi áp dụng cho ảnh y tế.

Bài 11 – Tăng cường ảnh cũ (Image Restoration)

1. **Chuẩn bị dữ liệu:**
    - Lấy một ảnh cũ (ảnh chân dung hoặc ảnh scan tài liệu).
    - Chuyển ảnh về **grayscale** để xử lý dễ dàng.
2. **Mô phỏng ảnh bị suy giảm:**
    - Làm mờ ảnh bằng Gaussian Blur (kernel = 9×9, σ = 5).
    - Thêm nhiễu muối tiêu (salt & pepper noise).
3. **Phục hồi ảnh:**
    - Áp dụng **Median filter** để loại bỏ nhiễu muối tiêu.
    - Áp dụng **Sharpening (Unsharp Masking / Laplacian)** để khôi phục chi tiết.
4. **Đánh giá:**
    - Hiển thị ảnh gốc, ảnh bị suy giảm, ảnh phục hồi.
    - So sánh bằng trực quan.
    - Tính các chỉ số định lượng:
        - **PSNR (Peak Signal-to-Noise Ratio)**
        - **SSIM (Structural Similarity Index)**