

**BỘ THÔNG TIN VÀ TRUYỀN THÔNG**  
**HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG**

---



**BÁO CÁO BÀI TẬP LỚN**  
**HỌC PHẦN: XỬ LÝ ẢNH**  
**ĐỀ TÀI: BỘ LỌC ẢNH NÂNG CAO CHẤT LƯỢNG SỐ**

<b>Giảng viên hướng dẫn</b>	<b>: Phạm Hoàng Việt</b>
<b>Họ và tên sinh viên</b>	<b>: Nguyễn Huy Hoàng - B22DCCN336</b>
<b>Lớp</b>	<b>: D22HTTT06</b>
<b>Nhóm</b>	<b>: 11</b>

*Hà Nội – 2025*

## MỤC LỤC

<b>Chương 1: Giới thiệu chung</b>	<b>1</b>
1. Mục tiêu đề tài	1
2. Phạm vi nghiên cứu	1
<b>Chương 2: Cơ sở lý thuyết</b>	<b>1</b>
1. Lý thuyết về Tích chập (Convolution)	1
2. Các bộ lọc làm mịn ảnh (Smoothing Filters)	1
3. Các bộ lọc phát hiện biên (Edge Detection)	2
<b>Chương 3: Cài đặt và thử nghiệm</b>	<b>3</b>
1. Cấu trúc chương trình:	3
2. Cài đặt thuật toán tích chập:	3
3. Cài đặt các bộ lọc làm mịn (Smoothing)	4
3.1. Bộ lọc Trung bình (Mean Filter) và Gaussian	4
3.2. Bộ lọc Trung vị (Median Filter) - Kỹ thuật Sliding Window	5
4. Cài đặt các bộ lọc phát hiện biên (Edge Detection)	5
<b>Chương 4: Kết quả thực nghiệm và đánh giá</b>	<b>6</b>
1. Môi trường thử nghiệm	6
2. Kết quả thực nghiệm	6
3. Đánh giá khả năng làm mịn và khử nhiễu	9
4. Đánh giá khả năng phát hiện biên	10
5. Kết luận chung	10
<b>TÀI LIỆU THAM KHẢO</b>	<b>11</b>

## **Lời cảm ơn**

Lời đầu tiên, em xin gửi lời cảm ơn chân thành đến Thầy **Phạm Hoàng Việt**, người đã trực tiếp giảng dạy và truyền đạt những kiến thức quý báu trong môn học Xử lý ảnh. Những bài giảng tâm huyết và sự hướng dẫn của Thầy chính là nền tảng quan trọng giúp em tiếp cận và hiểu rõ bản chất các thuật toán trong đề tài này.

Mặc dù thực hiện đề tài dưới hình thức cá nhân, gặp không ít khó khăn trong việc tự nghiên cứu và cài đặt các thuật toán phức tạp, nhưng đây cũng là cơ hội để em rèn luyện tính chủ động và tư duy lập trình. Quá trình xây dựng công cụ này đã giúp em củng cố vững chắc kiến thức.

Dù đã rất nỗ lực hoàn thiện, nhưng do giới hạn về thời gian và kiến thức, bài báo cáo khó tránh khỏi những thiếu sót. Em rất mong nhận được những ý kiến đóng góp và nhận xét từ Thầy để em có thể rút kinh nghiệm và hoàn thiện hơn trong những dự án tiếp theo.

Em xin chân thành cảm ơn!

Sinh viên thực hiện

Nguyễn Huy Hoàng

## Chương 1: Giới thiệu chung

### 1. Mục tiêu đề tài

- Tìm hiểu và cài đặt các thuật toán xử lý ảnh cơ bản trong miền không gian (Spatial Domain).
- Xây dựng công cụ (Tool) có giao diện đồ họa (GUI) cho phép người dùng tải ảnh, lựa chọn bộ lọc và quan sát kết quả trực quan.
- So sánh hiệu quả của các bộ lọc làm mịn (Mean, Gaussian, Median) và phát hiện biên (Sobel, Prewitt, Laplacian).

### 2. Phạm vi nghiên cứu

- **Ngôn ngữ lập trình:** python
- **Thư viện sử dụng:**
  - + **NumPy:** Xử lý ma trận và tính toán tích chập thủ công.
  - + **OpenCV:** Đọc/Ghi ảnh và xử lý giao diện màu sắc.
  - + **Tkinter:** Xây dựng giao diện người dùng.
- **Các thuật toán cài đặt:**
  - + **Nhóm làm mịn:** Mean Filter, Gaussian Filter, Median Filter.
  - + **Nhóm phát hiện biên:** Sobel, Prewitt, Laplacian.

## Chương 2: Cơ sở lý thuyết

### 1. Lý thuyết về Tích chập (Convolution)

Hầu hết các bộ lọc trong miền không gian đều dựa trên phép toán tích chập giữa ảnh đầu vào  $f(x,y)$  và một mặt nạ (kernel)  $w$ . Giá trị của điểm ảnh mới  $g(x,y)$  được tính bằng tổng trọng số của các pixel trong vùng lân cận:

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

Trong đó, kích thước kernel thường là số lẻ (3x3, 5x5) để xác định được tâm xử lý.

### 2. Các bộ lọc làm mịn ảnh (Smoothing Filters)

Mục đích: Giảm nhiễu và làm mờ các chi tiết thừa.

- Bộ lọc Trung bình (Mean Filter):
  - + Là bộ lọc tuyến tính. Thay thế giá trị pixel bằng trung bình cộng các pixel lân cận.
  - + Kernel 3x3: Ma trận toàn số 1 nhân với hệ số 1/9.

- + *Nhược điểm*: Làm nhòe ảnh và mờ biên vật thể.
- **Bộ lọc Gaussian**:
  - + Là bộ lọc tuyến tính nhưng có trọng số. Các pixel gần tâm có trọng số cao hơn, giảm dần ra xa theo hình quả chuông (Bell curve).
  - + *Ưu điểm*: Làm trơn ảnh tự nhiên hơn và giữ chi tiết tốt hơn Mean Filter.
- **Bộ lọc Trung vị (Median Filter)**:
  - + Là bộ lọc phi tuyến (Non-linear). Sắp xếp giá trị các pixel trong cửa sổ và chọn giá trị nằm giữa (Median).
  - + *Ưu điểm đặc biệt*: Loại bỏ hoàn toàn nhiễu muối tiêu (Salt-and-pepper noise) mà không làm mờ cạnh biên.

### 3. Các bộ lọc phát hiện biên (Edge Detection)

Mục đích: Tìm nơi có sự thay đổi độ sáng đột ngột (Gradient).

- Bộ lọc Sobel & Prewitt (Đạo hàm bậc 1): Sử dụng hai kernel để tính đạo hàm theo hướng ngang ( $G_x$ ) và dọc ( $G_y$ ).
  - + Prewitt: Hệ số  $[1, 1, 1]$  (chia đều). Dễ bị nhiễu.
  - + Sobel: Hệ số  $[1, 2, 1]$  (Gaussian).
  - + Nhấn mạnh điểm giữa giúp biên mượt và liền mạch hơn.
  - + Độ lớn Gradient:

$$\sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|$$

- Bộ lọc Laplacian (Đạo hàm bậc 2): Tìm biên dựa trên điểm Zero-crossing (đạo hàm bậc 2 đổi dấu).
  - + Là bộ lọc vô hướng (Isotropic), bắt biên theo mọi hướng cùng lúc.
  - + *Nhược điểm*: Rất nhạy cảm với nhiễu, thường tạo ra biên đôi.

## Chương 3: Cài đặt và thử nghiệm

### 1. Cấu trúc chương trình:

- main.py: Điểm khởi chạy ứng dụng, chịu trách nhiệm khởi tạo cửa sổ chính.
- gui.py: Xây dựng bằng thư viện Tkinter. Chịu trách nhiệm hiển thị hình ảnh, các nút điều khiển, thanh trượt tham số và xử lý sự kiện người dùng (Load/Save, Click).
- processors.py: chứa logic thuật toán xử lý ảnh

### 2. Cài đặt thuật toán tích chập:

Thay vì sử dụng hàm có sẵn `cv2.filter2D` của OpenCV em thực hiện cài đặt thủ công thuật toán tích chập để nắm vững bản chất toán học. Tuy nhiên, thay vì sử dụng 4 vòng lặp lồng nhau (duyệt qua từng pixel  $H \times W$  rồi duyệt qua kernel  $k \times k$ ) gây chậm chương trình, nhóm đã áp dụng kỹ thuật Vector hóa (Vectorization) trên thư viện NumPy.

Nguyên lý cài đặt ("Shift & Accumulate"):

- Padding: Tạo viền giả cho ảnh bằng phương pháp nhân bản biên (`mode='edge'`) để xử lý các pixel ở mép ảnh mà không làm giảm kích thước ảnh.
- Broadcasting: Thay vì đứng tại một pixel và nhìn ra xung quanh, thuật toán thực hiện dịch chuyển (shift) toàn bộ bức ảnh tương ứng với từng vị trí của kernel.
- Accumulate: Nhân toàn bộ ảnh đã dịch chuyển với trọng số tương ứng trong kernel và cộng dồn vào kết quả.

```

def _manual_convolution(self, image, kernel):
    h_img, w_img = image.shape[:2]
    k_height, k_width = kernel.shape

    # Tính lề (padding)
    pad_h = k_height // 2
    pad_w = k_width // 2

    # 2. Tạo ảnh có viền (Padding)
    if len(image.shape) == 3: # Ảnh màu
        image_padded = np.pad(image, ((pad_h, pad_h), (pad_w, pad_w), (0, 0)), mode='edge')
        output = np.zeros_like(image, dtype=np.float32)
    else: # Ảnh xám
        image_padded = np.pad(image, ((pad_h, pad_h), (pad_w, pad_w)), mode='edge')
        output = np.zeros_like(image, dtype=np.float32)

    # 3. THUẬT TOÁN SHIFT & ACCUMULATE (Dịch chuyển và Cộng dồn)
    kernel = np.flip(kernel)

    for i in range(k_height):
        for j in range(k_width):
            # Lấy trọng số tại vị trí i, j của kernel
            weight = kernel[i, j]

            # Cắt vùng ảnh tương ứng (Slicing) - Đây là bước Vector hóa
            # Vùng này có kích thước đúng bằng kích thước ảnh gốc (h_img, w_img)
            # Nó trượt đi theo i và j
            if len(image.shape) == 3:
                region = image_padded[i : i + h_img, j : j + w_img, :]
            else:
                region = image_padded[i : i + h_img, j : j + w_img]

            # Cộng dồn vào kết quả: Output += Ảnh_dịch * Trọng_số
            output += region * weight

    # Cắt giá trị về [0, 255]
    return np.clip(output, 0, 255).astype(np.uint8)

```

Ưu điểm: Tốc độ xử lý nhanh gấp nhiều lần so với vòng lặp Python thông thường, tiệm cận tốc độ của thư viện C++.

### 3. Cài đặt các bộ lọc làm mịn (Smoothing)

#### 3.1. Bộ lọc Trung bình (Mean Filter) và Gaussian

- **Mean Filter:** Tạo kernel kích thước  $k \times k$  với tất cả giá trị bằng  $1/(k^2)$
- **Gaussian Filter:** Tạo kernel động dựa trên kích thước  $k$  người dùng nhập vào. Các hệ số được tính theo hàm phân phối chuẩn Gaussian 2D:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Sau khi tạo Kernel, gọi hàm `_manual_convolution` để xử lý.

### 3.2. Bộ lọc Trung vị (Median Filter) - Kỹ thuật Sliding Window

Bộ lọc trung vị không dùng tích chập. Để tối ưu hóa việc lấy "cửa sổ trượt" mà không dùng vòng lặp, nhóm sử dụng kỹ thuật `sliding_window_view` của NumPy.

- Kỹ thuật này tạo ra một "khung nhìn" (view) lên bộ nhớ, cho phép truy cập tất cả các vùng lân cận  $k \times k$  của mọi pixel cùng lúc dưới dạng mảng nhiều chiều.
- Sau đó áp dụng `np.median` trên trục cuối cùng để tìm giá trị giữa.

### 4. Cài đặt các bộ lọc phát hiện biên (Edge Detection)

Các bộ lọc biên được cài đặt bằng cách định nghĩa thủ công các Ma trận Kernel (Mask) theo lý thuyết, sau đó đưa vào hàm tích chập chung.

- Sobel: Sử dụng 2 kernel  $G_x$  (dọc) và  $G_y$  (ngang) với hệ số làm tròn  $[1, 2, 1]$ . Kết quả cuối cùng được tổng hợp:  $G \approx |G_x| + |G_y|$ .
- Prewitt: Tương tự Sobel nhưng dùng hệ số làm tròn  $[1, 1, 1]$ .
- Laplacian: Sử dụng 1 kernel duy nhất (vô hướng) với tổng các hệ số bằng 0 (Tâm -4, xung quanh 1).

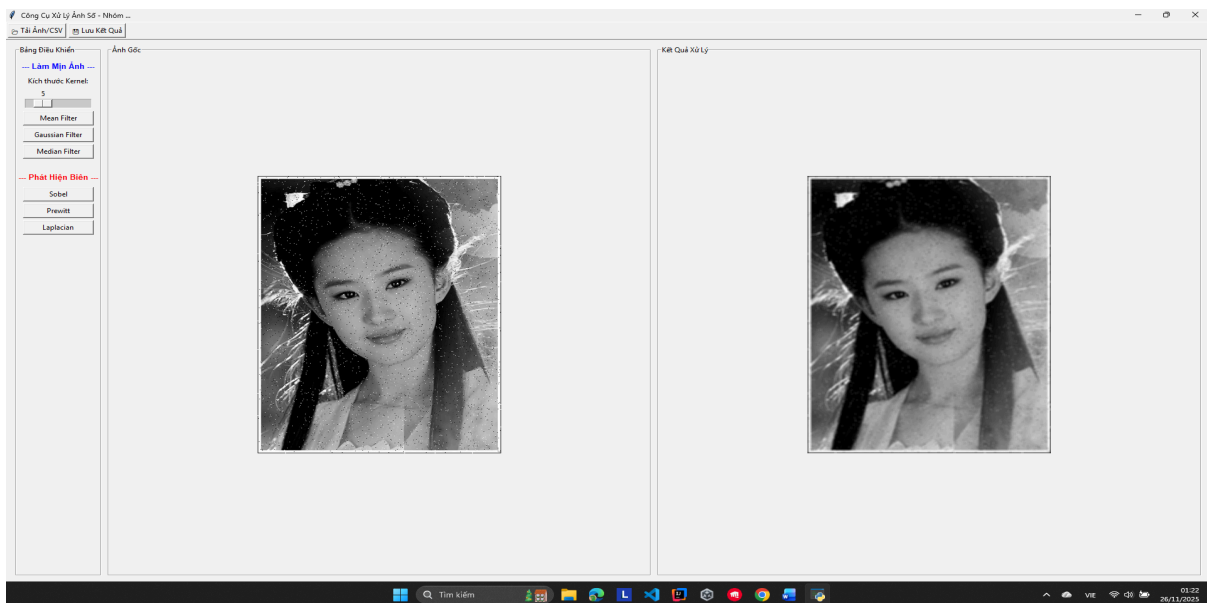
## Chương 4: Kết quả thực nghiệm và đánh giá

### 1. Môi trường thử nghiệm

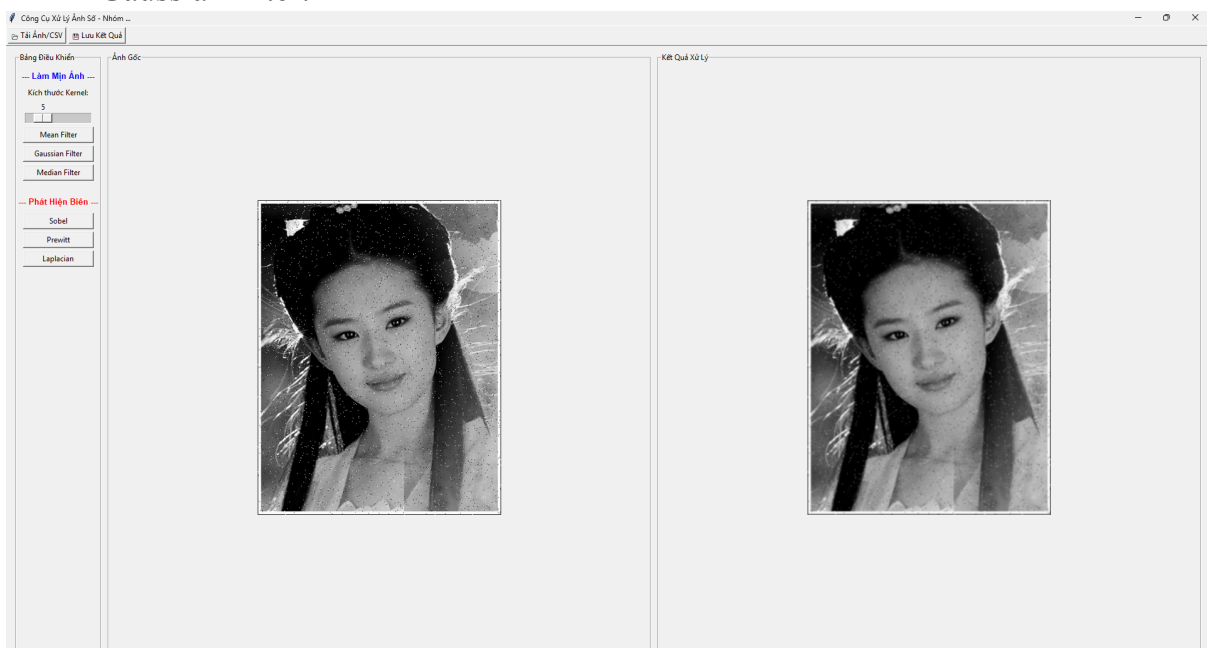
- Công cụ: Tool tự xây dựng (Python/Tkinter).
- Dữ liệu:
  - + Bộ ảnh chuẩn (Lenna, Cameraman) để kiểm tra độ sắc nét.
  - + Bộ ảnh nhiễu nhân tạo (Nhiều muối tiêu - Salt & Pepper) để kiểm tra khả năng lọc nhiễu.

### 2. Kết quả thực nghiệm

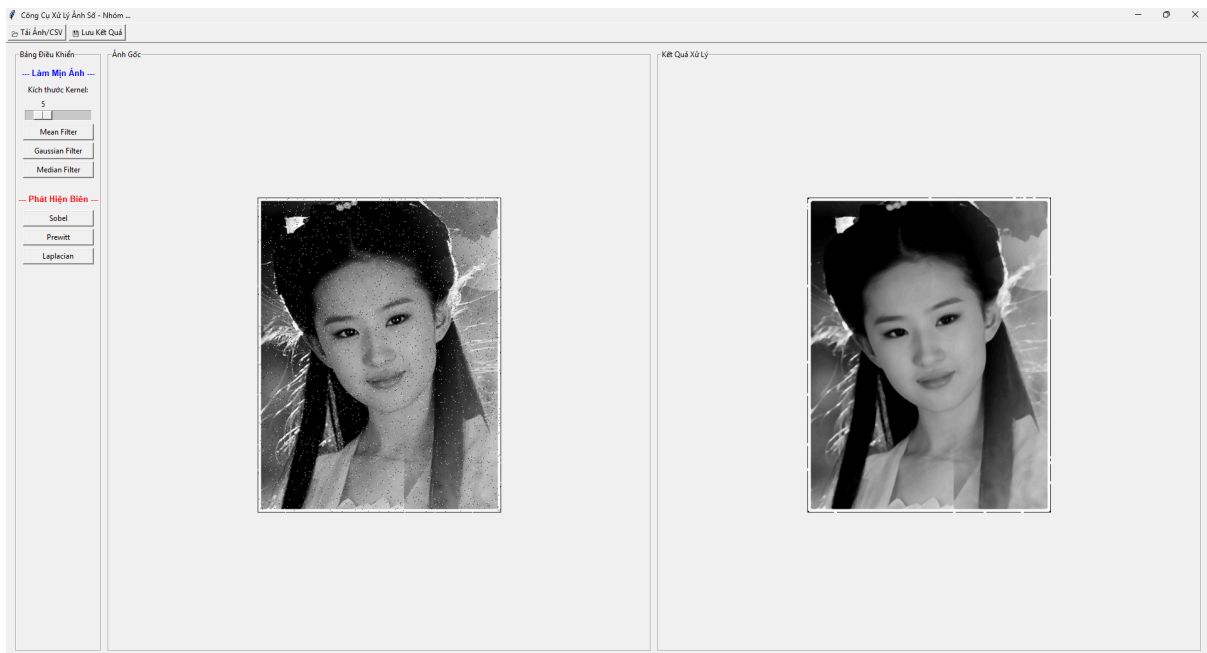
- Mean filter:



- Gaussian filter:

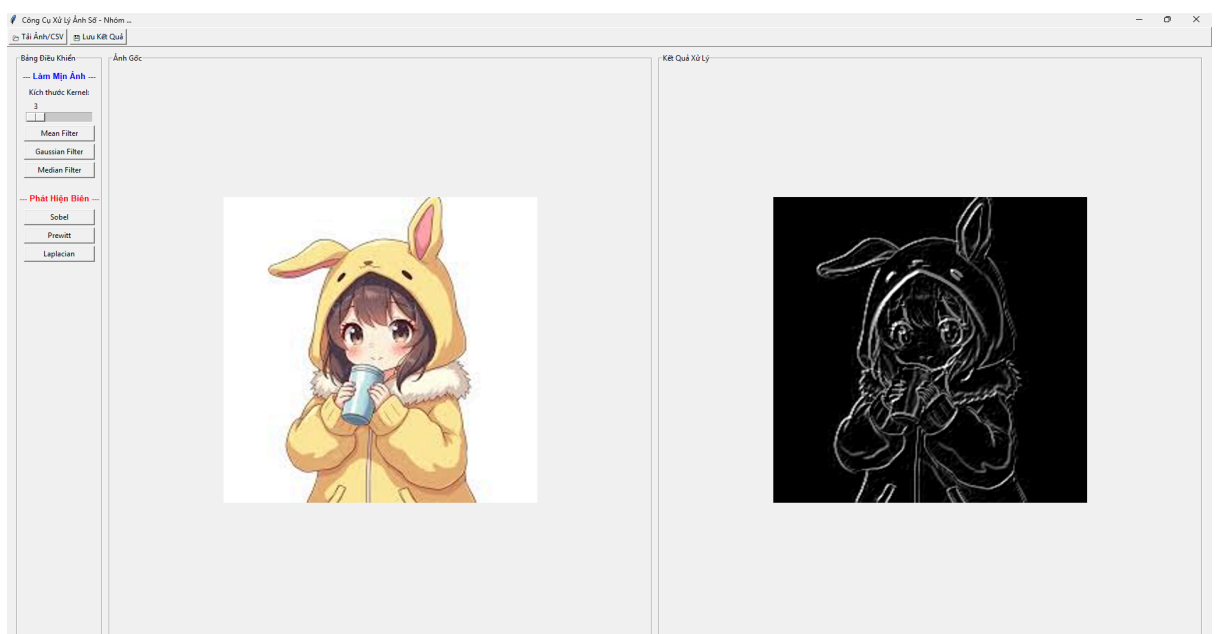


- Median filter:

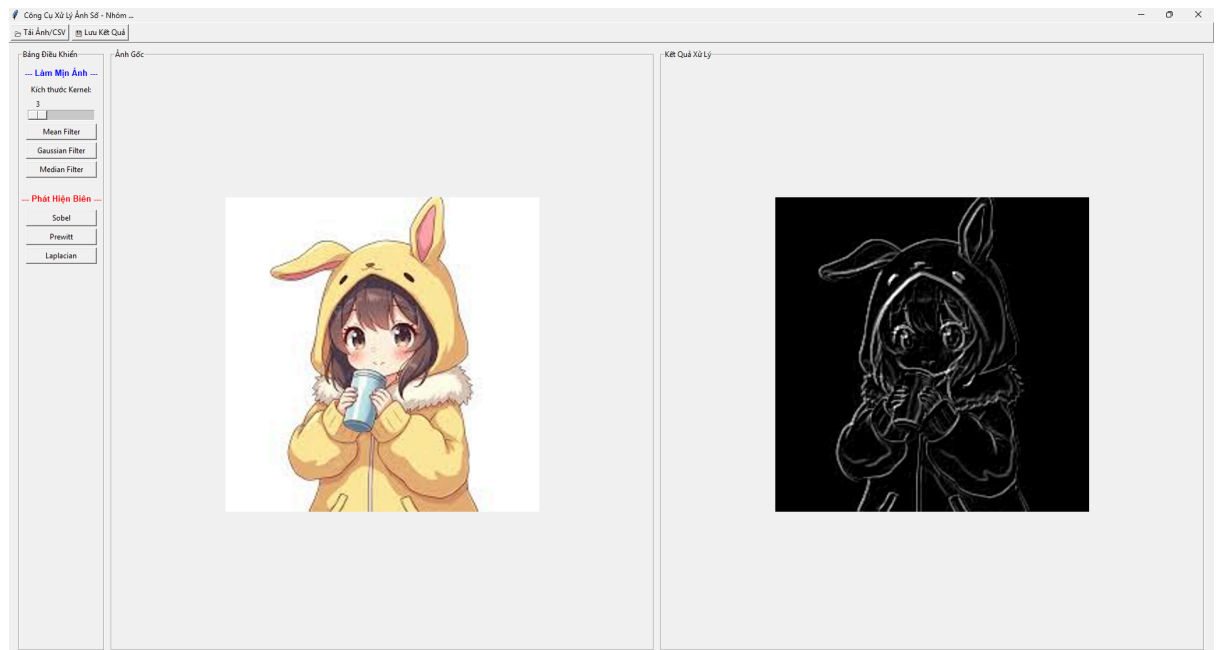


**\*\*Nhận xét:**

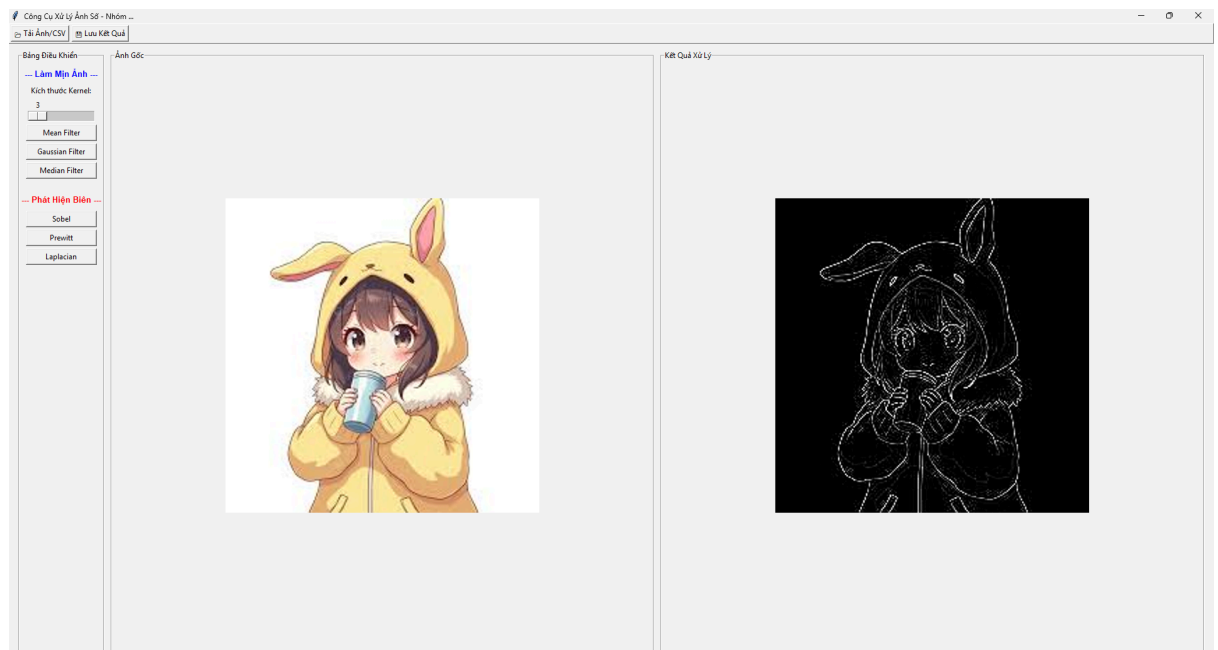
- **Đầu vào:** ảnh bị nhiễu hạt
- **Đầu ra:** Sau khi áp dụng bộ lọc Median với kích thước kernel 5x5, các hạt nhiễu đã bị loại bỏ hoàn toàn. Đáng chú ý là các đường biên của vật thể vẫn giữ được độ sắc nét, không bị nhòe như khi dùng bộ lọc Mean và Gaussian. Điều này chứng minh tính hiệu quả của bộ lọc phi tuyến trong việc xử lý nhiễu xung.
- Sobel



- Prewitt



- Laplacian



**\*\* Nhận xét:**

- **Đầu vào:** Ảnh vật thể có độ tương phản rõ ràng.
- **Đầu ra:** Thuật toán Sobel đã tách được các đường viền bao quanh vật thể một cách liên mạch. Các đường biên có độ dày vừa phải và rõ nét nhờ hệ số làm trơn Gaussian tích hợp trong hạt nhân Sobel.

Nền ảnh tương đối sạch, ít xuất hiện nhiễu vụn vặt, cho thấy thuật toán hoạt động ổn định trên ảnh thực tế.

### 3. Đánh giá khả năng làm mịn và khử nhiễu

Thuật toán	Hình ảnh kết quả (Mô tả)	Nhận xét & Đánh giá
Mean Filter (Kernel 5x5)	Ảnh bị mờ nhòe (blur). Các hạt nhiễu trắng/đen không mất đi mà bị loang ra thành các vết xám mờ.	Kém hiệu quả với nhiễu xung. Do cơ chế tính trung bình cộng nên nhiễu bị hòa vào nền chứ không bị loại bỏ.
Gaussian Filter (Kernel 5x5)	Ảnh mờ dịu hơn Mean, giữ được cấu trúc khối tốt hơn nhưng nhiễu vẫn còn tồn tại dưới dạng các đốm mờ.	Trung bình. Phù hợp để làm mịn da hoặc xóa nhiễu hạt mịn (Gaussian noise) hơn là nhiễu muối tiêu.
Median Filter (Kernel 5x5)	Kết quả ấn tượng. Các hạt nhiễu biến mất hoàn toàn. Các cạnh biên của vật thể vẫn sắc nét, không bị nhòe.	Xuất sắc. Do cơ chế chọn giá trị trung vị (phi tuyến), thuật toán loại bỏ triệt để các giá trị đột biến (outliers) là nhiễu.

#### 4. Đánh giá khả năng phát hiện biên

Thuật toán	Hình ảnh kết quả (Mô tả)	Nhận xét & Đánh giá
Prewitt	Đường biên tìm được khá thô. Nền ảnh xuất hiện nhiều chấm nhỏ li ti (nhiều).	Khả năng chống nhiễu kém. Do trọng số làm mịn đều nhau (1,1,1) nên dễ bắt nhầm nhiễu thành biên.
Sobel	Đường biên dày, rõ nét và liền mạch. Nền ảnh sạch hơn so với Prewitt.	Tốt nhất cho ảnh thường. Nhờ hệ số trọng tâm (2) theo phân phối Gaussian, Sobel vừa tìm biên vừa khử nhiễu nhẹ, giúp biên mượt mà.
Laplacian	Đường biên rất mảnh (1 pixel), sắc nét. Tuy nhiên, nhiễu xuất hiện dày đặc trên toàn bộ ảnh.	Rất nhạy cảm. Đạo hàm bậc 2 bắt biên cực tốt nhưng bắt cả nhiễu. Cần kết hợp làm trơn trước khi dùng (LoG).

#### 5. Kết luận chung

- Chức năng: Tool hoạt động ổn định, đọc được cả tên file tiếng Việt (nhờ cải tiến np.fromfile).
- Hiệu năng: Tốc độ xử lý ảnh kích thước trung bình (500x500) là tức thời (real-time) nhờ thuật toán Vector hóa, chứng minh tính hiệu quả của giải pháp cài đặt.
- Trực quan: Giao diện cho phép so sánh song song (Side-by-side) giúp dễ dàng nhận diện sự khác biệt giữa các thuật toán.

## TÀI LIỆU THAM KHẢO

1. Giáo trình xử lý ảnh Học viện Công nghệ Bưu chính Viễn thông
2. OpenCV Documentation. Image Filtering and Edge Detection.  
Truy cập tại: <https://docs.opencv.org/>
3. NumPy Documentation. Array manipulation and Mathematical functions. Truy cập tại: <https://numpy.org/doc/>
4. Python Tkinter Documentation. Graphical User Interfaces with Tk.  
Truy cập tại: <https://docs.python.org/3/library/tkinter.html>
5. Github dự án: