

# Đề tài 4: Chuyển ảnh thành tranh vẽ

Sinh viên:

**Nguyễn Việt Quang**

**Phạm Công Minh**

# Mục tiêu đề tài

- Xây dựng phần mềm có khả năng chuyển ảnh chụp thông thường thành ảnh mang phong cách tranh vẽ (sketch).
- Ứng dụng các kỹ thuật xử lý ảnh như phát hiện biên, làm mờ, khử nhiễu và giữ biên để tạo hiệu ứng vẽ tay tự nhiên.
- Xây dựng giao diện đơn giản cho phép người dùng tải ảnh lên, xử lý và xem kết quả

# LÝ THUYẾT ÁP DỤNG

# 1. Chuyển ảnh RGB → Grayscale

- Ảnh màu RGB có 3 kênh: Red (R), Green (G), Blue (B).
- Ảnh xám (grayscale) chỉ có 1 kênh độ sáng (luminance).
- Mục đích: tóm tắt thông tin màu thành cường độ sáng, giảm dữ liệu, tiện xử lý ảnh.

Phép chuyển đổi sử dụng công thức chuẩn của hệ NTSC:

$$\text{Gray} = 0.299R + 0.587G + 0.114B$$

# Gaussian Blur – Làm mờ ảnh

Gaussian blur là bộ lọc làm mờ dựa trên hàm phân phối chuẩn.

Trong dự án, Gaussian Blur được dùng trong **thuật toán Pencil Sketch** để làm mượt phần sáng tối trước khi kết hợp với ảnh gốc.

Hàm Gaussian:

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

→ Gaussian giúp ảnh mịn, giảm nhiễu, tránh tạo răng cưa khi tạo sketch.

# Bilateral Filter – Lọc mịn giữ biên

## Nguyên lý hoạt động:

- Mỗi pixel được thay bằng **trung bình có trọng số của các pixel lân cận**.
- Trọng số phụ thuộc vào:
  - a. **Khoảng cách**: pixel càng gần → ảnh hưởng càng mạnh
  - b. **Độ sáng**: pixel càng giống về cường độ → ảnh hưởng càng mạnh
- Kết quả: **vùng phẳng mịn, biên sắc nét không bị mờ**

## Ưu điểm:

- Giữ biên sắc nét
- Mịn vùng phẳng → giảm nhiễu
- Cải thiện kết quả **edge detection** và **sketch effect**

## Ứng dụng:

- Tiền xử lý cho Sobel, Prewitt, Canny
- Tạo **sketch / pencil effect**

# Thuật toán Sobel

Sobel dựa trên **đạo hàm bậc nhất**, dùng 2 kernel để tính gradient theo X và Y.

Kernel Sobel X:

$$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Kernel Sobel Y:

$$\begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Gradient:

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

**Đặc điểm:**

- Bất biến tốt
- Ít nhiễu
- Thích hợp cho hầu hết ảnh tự nhiên

# Thuật toán Prewitt

Cũng là đạo hàm bậc nhất nhưng kernel đơn giản hơn Sobel → tính nhanh hơn.

Kernel Prewitt X:

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Kernel Prewitt Y:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$$

**Đặc điểm:**

- Nhanh, nhẹ
- Độ sắc nét thấp hơn Sobel
- Nhạy nhiễu hơn



# Thuật toán Laplacian

Laplacian dựa trên **đạo hàm bậc hai**, phát hiện biên theo mọi hướng.

Kernel thường dùng:

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

**Đặc điểm:**

- Bắt biên rất mạnh
- Không phụ thuộc hướng
- Rất nhạy nhiễu → cần threshold

# Tách ngưỡng (Thresholding)

Sau khi tính gradient, cần lọc để giữ lại biên quan trọng.

Ví dụ:

- $\text{Pixel} < \text{threshold} \rightarrow \text{loại bỏ}$
- $\text{Pixel} \geq \text{threshold} \rightarrow \text{giữ lại (biên)}$

# Pipeline Tạo Ảnh Tranh Vẽ bằng Phát Hiện Biên



# Chuyển ảnh sang xám (Grayscale)

1

Mỗi pixel được tính theo công thức:

$$\text{Gray} = 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$



# Làm mịn ảnh (Bilateral Filter)

- Giữ biên đồng thời giảm nhiễu.
- Công thức kết hợp Gaussian không gian và Gaussian màu sắc.



# Phát hiện biên (Edge Detection)

1

**Sobel**



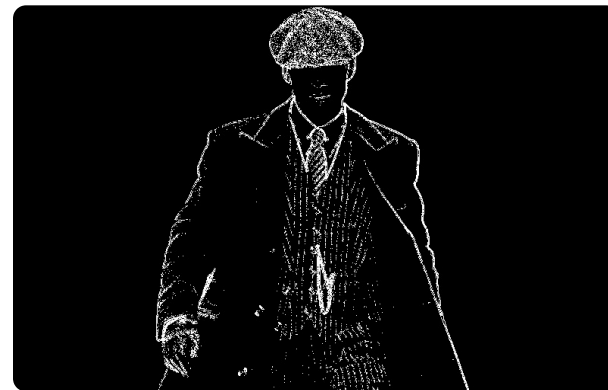
2

**Prewitt**



3

**Laplace**



# Tạo hiệu ứng sketch

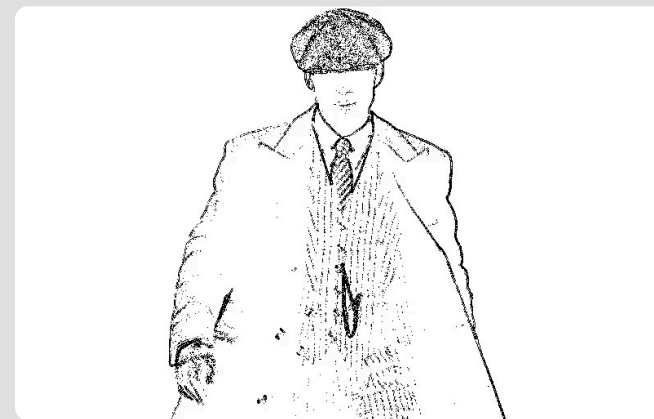
- Đảo màu biên:  $\text{Sketch} = 255 - \text{Edge}$
- Cho ra ảnh giống tranh vẽ tay.



Sobel

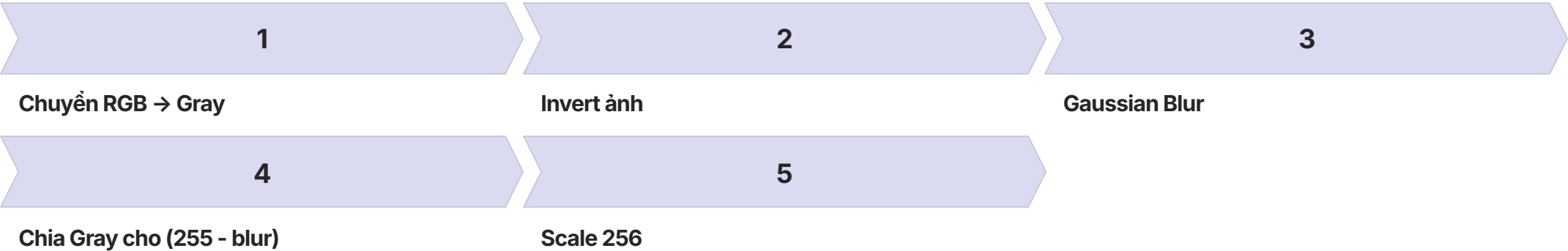


Prewitt



Laplacian

# Pencil Sketch Algorithm





# DEMO