

**TRƯỜNG ĐẠI HỌC HỌC VĂN LANG KHOA CÔNG NGHỆ THÔNG TIN**

**BÁO CÁO ĐỒ ÁN MÔN HỌC**

**NHẬP MÔN XỬ LÝ ẢNH SỐ**

**NGÀNH: CÔNG NGHỆ THÔNG TIN**

Chủ đề:

**LỌC ẢNH BẰNG PHƯƠNG PHÁP ĐẠO  
HÀM BẬC 2 (LAPLACE)**

**GVHD: TS Vũ Thanh Hiền**

**SVTH: Ngô Quốc Huy – 2174802010265 SVTH: Vũ Lê Minh Khôi – 2174802010690 SVTH: Đinh Thạch Bảo – 207CT27605**

**TP. Hồ Chí Minh – năm 2025**

**MỤC LỤC**

[1. PHÁT BIỂU BÀI TOÁN 3](#_Toc194141003)

[***1.1 Mô tả chi tiết bài toán:*** 3](#_Toc194141004)

[***Toán tử Laplace:*** 4](#_Toc194141005)

[2. HƯỚNG GIẢI QUYẾT 5](#_Toc194141006)

[***2.1 Về toán tử Laplace:*** 5](#_Toc194141007)

[***2.2 Áp dụng toán tử Laplace lên ảnh:*** 5](#_Toc194141008)

[***2.3 Xử lý ảnh kết quả:*** 5](#_Toc194141009)

[***2.4 Tinh chỉnh kết quả:*** 6](#_Toc194141010)

[3. THUẬT TOÁN 7](#_Toc194141011)

[***3.1 Toán tử Laplacian:*** 7](#_Toc194141012)

[***3.2 Tiền xử lý ảnh:*** 7](#_Toc194141013)

[4. KỸ THUẬT CÀI ĐẶT 9](#_Toc194141014)

[***4.1 Giao diện người dùng với Streamlit:*** 9](#_Toc194141015)

[***4.2 Xử lý ảnh với OpenCV:*** 10](#_Toc194141016)

[5. CÁC VẤN ĐỀ KHÁC 12](#_Toc194141017)

[***5.1 Phương pháp Unsharp Masking:*** 12](#_Toc194141018)

[***5.2 Áp dụng trong ứng dụng:*** 12](#_Toc194141019)

[***5.3 Ý nghĩa và lợi ích:*** 13](#_Toc194141020)

# 1. PHÁT BIỂU BÀI TOÁN

Bài toán phát hiện cạnh trong ảnh bằng toán tử Laplace là một kỹ thuật quan trọng trong xử lý ảnh, nhằm xác định các vùng có sự thay đổi đột ngột về cường độ sáng – thường là biên giới hoặc cạnh của các đối tượng trong ảnh.

Mục tiêu của bài toán là xây dựng một thuật toán có khả năng phát hiện các cạnh bằng cách áp dụng toán tử Laplace – một toán tử vi phân bậc hai – để tính đạo hàm bậc hai của cường độ sáng tại mỗi pixel. Kết quả mong muốn là một ảnh đầu ra với các cạnh được làm nổi bật, hỗ trợ các ứng dụng như phân tích ảnh, nhận diện đối tượng, hoặc xử lý ảnh y khoa.

## ***1.1 Mô tả chi tiết bài toán:***

**Đầu vào:** Một ảnh xám (grayscale image) được biểu diễn dưới dạng ma trận trong đó là giá trị cường độ sáng tại pixel

**Mục tiêu:** Xác định các vùng có sự thay đổi cường độ sáng mạnh (các cạnh).

**Phương pháp:**

* Áp dụng toán tử Laplace để tính toán đạo hàm bậc hai của ảnh.
* Phát hiện các điểm nơi đạo hàm bậc hai đổi dấu (zero-crossing), đây là vị trí của cạnh.
* Đầu ra: Một ảnh nhị phân hoặc ảnh đánh dấu các cạnh, trong đó các pixel thuộc cạnh được xác định.

## ***Toán tử Laplace:***

Toán tử Laplace cho một hàm liên tục được định nghĩa là:

Trong không gian rời rạc của ảnh số, toán tử Laplace được xấp xỉ bằng cách sử dụng mặt nạ (mask) tích chập. Một mặt nạ Laplace phổ biến là:

Hoặc phiên bản mở rộng với các đường chéo:

# ****2. HƯỚNG GIẢI QUYẾT****

## ***2.1 Về toán tử Laplace:***

Toán tử Laplace là một công cụ toán học dùng để phát hiện các điểm có sự thay đổi đột ngột trong hàm số, trong trường hợp này là cường độ sáng của ảnh.

Nó hoạt động bằng cách tính **đạo hàm bậc hai**, giúp làm nổi bật các vùng chuyển tiếp mạnh trong ảnh.

Toán tử Laplace được định nghĩa là:

Công thức cho thấy rằng Laplace lấy tổng của hai đạo hàm bậc hai theo trục ***x*** và ***y***

## ***2.2 Áp dụng toán tử Laplace lên ảnh:***

Trong không gian ảnh rời rạc, toán tử Laplace được tính bằng cách tích chập ảnh với một ma trận kernel

* Một kernel Laplace phổ biến có dạng:

Kernel này giúp phát hiện cạnh theo cả hướng ngang và dọc

Một phiên bản mở rộng để phát hiện cạnh theo tất cả hướng (ngang, dọc, chéo) là:

## ***2.3 Xử lý ảnh kết quả:***

Sau khi áp dụng toán tử Laplace, ảnh đầu ra sẽ làm nổi bật các vùng có giá trị pixel cao tại những vị trí có sự thay đổi mạnh về cường độ sáng. Để tinh chỉnh kết quả, ta áp dụng phương pháp **ngưỡng (thresholding)** nhằm phân loại các pixel thành điểm thuộc cạnh hoặc không thuộc cạnh. Ngoài ra, phương pháp **zero-crossing** cũng có thể được sử dụng để xác định vị trí các điểm biên bằng cách phát hiện những điểm mà tại đó đạo hàm bậc hai đổi dấu.

## ***2.4 Tinh chỉnh kết quả:***

Do toán tử Laplace rất nhạy với nhiễu, nên trước khi áp dụng, ta thường làm mịn ảnh bằng **bộ lọc Gaussian**. Bước tiền xử lý này giúp giảm nhiễu và cải thiện độ chính xác của việc phát hiện cạnh. Sự kết hợp giữa làm mịn Gaussian và toán tử Laplace được gọi là **Laplacian of Gaussian (LoG)**, một phương pháp hiệu quả giúp hạn chế nhiễu và nâng cao chất lượng kết quả phát hiện cạnh.

# 3. THUẬT TOÁN

## ***3.1 Toán tử Laplacian:***

Toán tử sẽ phát hiện cạnh dựa trên đạo hàm bậc hai của anh. Hoạt động bằng cách tính toán sự thay đổi đột ngột cường độ màu sắc hoặc độ sáng của các điểm ảnh, từ đó làm nổi bật các đường biên (cạnh) trong ảnh. Trong ứng dụng này có:

* Nếu chọn chế độ "**Áp dụng trên ảnh màu**" (apply\_on\_color=True), Laplacian được áp dụng riêng lẻ trên từng kênh màu **(R, G, B)** của ảnh, sau đó các kênh được ghép lại để tạo ra kết quả. Cách làm này giữ được thông tin màu sắc và làm nổi bật các cạnh trong từng kênh màu riêng biệt.
* Nếu không chọn chế độ này, ảnh được chuyển sang thang độ xám (**grayscale**) trước khi áp dụng Laplacian, giúp đơn giản hóa quá trình xử lý và tập trung vào cấu trúc cạnh tổng quát mà không phụ thuộc vào màu sắc.
* Kết quả của Laplacian được chuẩn hóa bằng hàm **convertScaleAbs** để đảm bảo giá trị đầu ra nằm trong khoảng hiển thị được **(0-255)**, tránh hiện tượng giá trị âm hoặc vượt quá giới hạn.
* Kích thước **kernel (ksize)** là một tham số có thể điều chỉnh **(1, 3, 5, 7)**, ảnh hưởng đến mức độ chi tiết của cạnh được phát hiện**: kernel** càng lớn, càng phát hiện các cạnh lớn hơn và ít nhiễu hơn, trong khi **kernel** nhỏ phù hợp với các chi tiết tinh tế.

## ***3.2 Tiền xử lý ảnh:***

Tiền xử lý ảnh:

Trước khi áp dụng toán tử Laplacian, ảnh được tiền xử lý qua các bước:

* Chuyển đổi từ định dạng **RGBA** (nếu có) sang **BGR** để tương thích với OpenCV.
* Thay đổi kích thước ảnh với hàm **resize\_with\_aspect\_ratio**, giữ nguyên tỷ lệ khung hình, dựa trên chiều rộng tối đa **(max\_width)** do người dùng chọn.
* Chuyển đổi ảnh sang **grayscale** để chuẩn bị cho việc phát hiện cạnh bằng toán tử Laplacian.
* Nếu người dùng bật chế độ làm nét **(Unsharp Masking)**, ảnh gốc sẽ được tăng cường độ sắc nét với mức độ do người dùng điều chỉnh.

# 4. KỸ THUẬT CÀI ĐẶT

Ứng dụng được xây dựng bằng Python, sử dụng các thư viện mạnh mẽ như **Streamlit**, **OpenCV (cv2)**, **NumPy**, và **PIL** để tạo giao diện người dùng tương tác và xử lý ảnh hiệu quả. Dưới đây là các kỹ thuật cài đặt chính:

## ***4.1 Giao diện người dùng với Streamlit:***

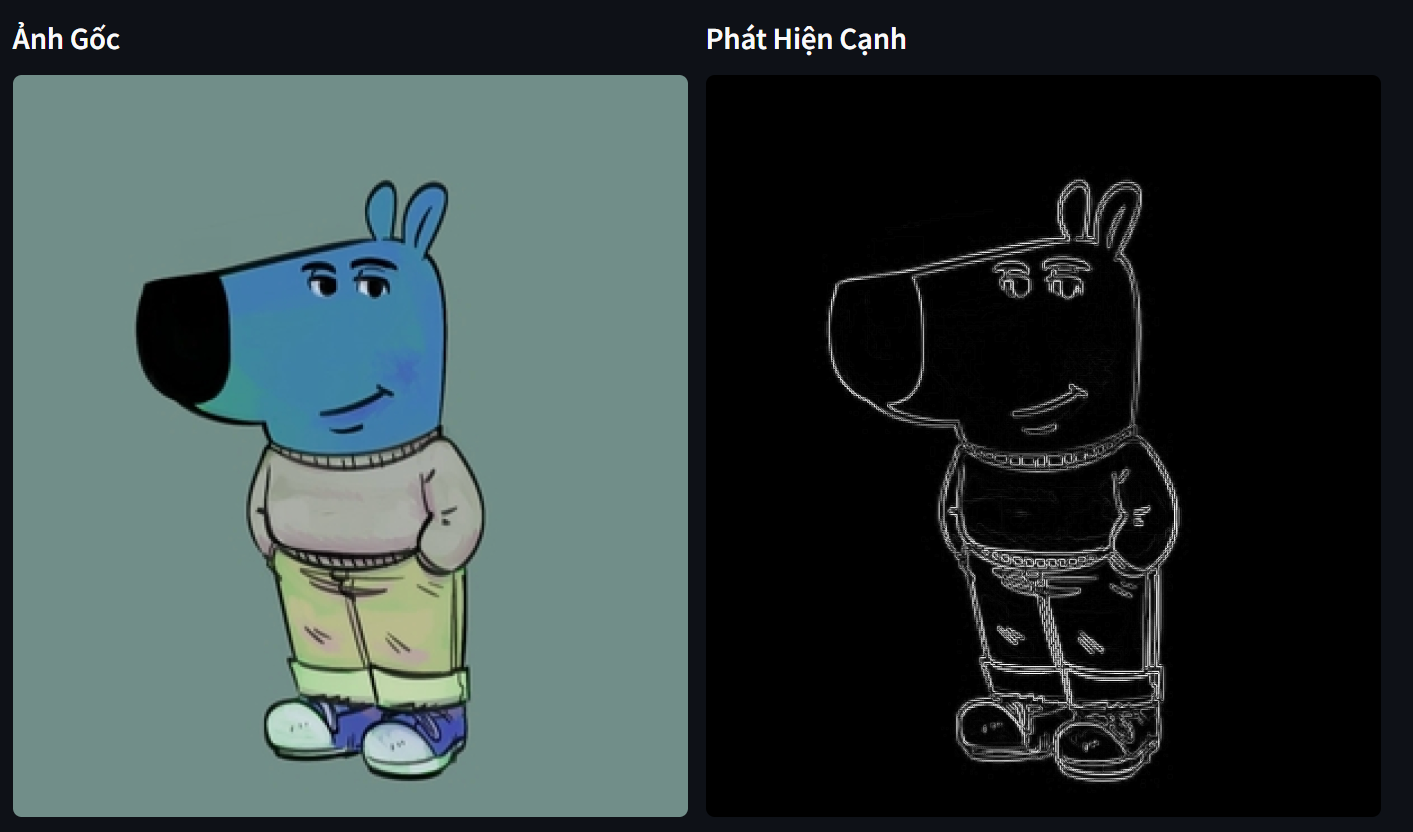
* + **Streamlit** được sử dụng để tạo một ứng dụng web đơn giản, trực quan. Trang chính hiển thị tiêu đề, hướng dẫn, và khu vực tải ảnh lên.
  + Thanh bên (**sidebar**) cung cấp các tùy chọn cấu hình như kích thước kernel, chiều rộng tối đa, chế độ áp dụng Laplacian, và tùy chọn làm nét.
  + Hai cột (**st.columns**) được sử dụng để hiển thị song song ảnh gốc và ảnh phát hiện cạnh, cùng với các nút tải xuống kết quả.



**Hình 4.1** Giao diện web của ứng dụng

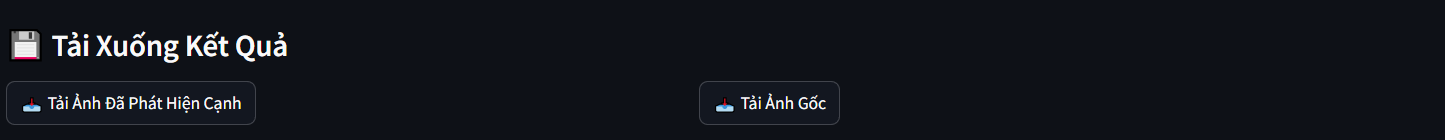
## ***4.2 Xử lý ảnh với OpenCV:***

* Hàm **resize\_with\_aspect\_ratio** đảm bảo ảnh được thay đổi kích thước mà không làm biến dạng tỷ lệ, sử dụng phép nội suy mặc định (**cv.INTER\_AREA**).
* Hàm **apply\_laplacian** triển khai toán tử Laplacian với hai chế độ (màu hoặc grayscale), tận dụng các hàm **cv.Laplacian** và **cv.convertScaleAbs** của OpenCV.
* Hàm tổng quát **process\_image** kết hợp tiền xử lý và thuật toán Laplacian, trả về cả ảnh gốc và ảnh phát hiện cạnh.



**Hình 4.2** Hình ảnh gốc và sau khi được xử lý

 **Quản lý dữ liệu và xuất kết quả**:

* Ảnh tải lên được đọc bằng PIL.Image.open và chuyển thành mảng NumPy để xử lý với OpenCV.
* Kết quả được lưu tạm vào bộ đệm io.BytesIO dưới định dạng PNG, sau đó cung cấp cho người dùng qua nút tải xuống (st.download\_button).
* Ảnh gốc được chuyển từ BGR sang RGB trước khi xuất để đảm bảo màu sắc hiển thị chính xác. 

**Hình 4.3** Cho phép người dùng tải về ảnh gốc hoặc ảnh đã xử lý

 **Xử lý lỗi**:

* Ứng dụng bao gồm cơ chế xử lý ngoại lệ **(try-except)** để thông báo lỗi nếu quá trình xử lý ảnh thất bại, giúp tăng tính ổn định và thân thiện với người dùng.

 **Tối ưu hóa hiệu suất**:

* Các tham số như kích thước **kernel** và chiều rộng tối đa được giới hạn trong khoảng hợp lý để tránh tiêu tốn tài nguyên không cần thiết.
* Việc sử dụng **NumPy** cho các phép toán trên mảng ảnh đảm bảo tốc độ **xử lý nhanh**, ngay cả với ảnh kích thước lớn

# 5. CÁC VẤN ĐỀ KHÁC

## ***5.1 Phương pháp Unsharp Masking:***

Ngoài việc phát hiện cạnh bằng toán tử Laplace, ứng dụng còn tích hợp phương pháp **Unsharp Masking** để tăng độ sắc nét của ảnh gốc, mang lại trải nghiệm linh hoạt hơn cho người dùng. Phương pháp này không trực tiếp liên quan đến phát hiện cạnh, nhưng đóng vai trò quan trọng trong việc cải thiện chất lượng hình ảnh đầu vào trước hoặc sau khi xử lý cạnh, tùy thuộc vào nhu cầu cụ thể.

**Nguyên lý hoạt động:**

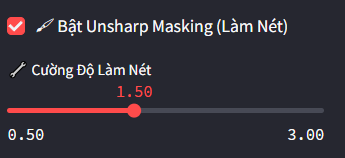
Unsharp Masking là một kỹ thuật làm nét ảnh dựa trên việc tăng cường độ tương phản tại các vùng chuyển tiếp (cạnh) trong ảnh. Thuật toán hoạt động theo ba bước chính:

* Tạo một phiên bản mờ của ảnh gốc bằng cách áp dụng bộ lọc Gaussian với kernel kích thước 5x5 (**cv.GaussianBlur(image, (5, 5), 0)**). Bộ lọc này làm giảm chi tiết nhỏ và nhiễu, tạo ra một ảnh mờ làm cơ sở để so sánh với ảnh gốc.
* Tính toán sự khác biệt giữa ảnh gốc và ảnh mờ, sau đó nhân với với một hệ số cường độ (strength) để khuếch đại các chi tiết bị mất trong quá trình làm mờ.
* Kết hợp ảnh gốc và ảnh mờ bằng hàm **cv.addWeighted** với công thức:

sharpened = (1 + strength) \* ảnh gốc - strength \* ảnh mờ

## ***5.2 Áp dụng trong ứng dụng:***

Unsharp Masking được triển khai thông qua hàm **unsharp\_mask(image, strength=1.5)** trong code. Cho phép người dùng bật/tắt tính năng này thông qua checkbox của app. Có thể điều chỉnh độ làm nét trong khoản từ 0.5 đến 3.0 (mặc định sẽ là 1.5). Kết quả sẽ làm cho ảnh gốc trở nên sắc nét hơn, giúp các chi tiết trở nên rõ ràng cho việc nhận dạng, đặc biệt hữu ích đối với các ảnh bị mờ hoặc thiếu tương phản.

****

**Hình 5.1** Chức năng Unsharp Masking trong ứng dụng

## ***5.3 Ý nghĩa và lợi ích:***

Việc bổ sung **Unsharp Masking** không chỉ cải thiện chất lượng hình ảnh mà còn hỗ trợ quá trình phát hiện cạnh bằng toán tử Laplace. Khi ảnh được làm nét trước, các cạnh trở nên rõ ràng hơn, giúp Laplacian phát hiện chính xác hơn các vùng chuyển tiếp mạnh. Điều này đặc biệt hiệu quả trong các trường hợp ảnh có nhiễu hoặc độ tương phản thấp.