PHÁT HIỆN BIÊN ẢNH & ỨNG DỤNG

Vũ Mạnh Hùng – B22DCCN372

NỘI DUNG

1

Thuật toán

Thực hiện các thuật toán phát hiện biên: Sobel, Laplacian, Canny,

Ứng dụng

cho các bài toán phát hiện vật thể và đếm vật

2

3

Xây dựng giao diện

Cho phép người dùng tải/lưu ảnh (file CSV/ảnh thật), xem kết quả trực quan.

THƯ VIỆN SỬ DỤNG

Streamlit

UI, upload ảnh,
hiển thị real-time

PIL / Image

Xử lý ảnh RGB,
đọc/ghi file

Pandas

Xử lí dữ liệu vào/ra

NumPy

Ma trận ảnh,
tính toán pixel

Thuật toán Sobel

Ý tưởng:
Phát hiện biên bằng đạo hàm bậc 1, dựa vào sự thay đổi mức xám mạnh theo 2 hướng ngang (x) và dọc (y).

Đặc điểm:

- Đơn giản, tốc độ nhanh
- Tìm biên theo 2 hướng chính
- Nhạy nhiễu hơn Canny

Mặt nạ Sobel:

$$H_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix}$$

$$H_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix}$$

Thuật toán Laplacian

Ý tưởng:
Dựa vào đạo hàm bậc 2, nhấn mạnh các vùng có sự “uốn cong” lớn trong cường độ sáng.

Đặc điểm:

- Tạo ra điểm zero-crossing → vị trí biên
- Đẳng hướng (không phụ thuộc hướng biên)
- Nhạy nhiễu, thường cần làm mờ trước

Ứng dụng:

- Phát hiện đường mảnh
- Phát hiện cấu trúc hình học

Mặt nạ Laplacian:

$$H_1 = \begin{pmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{pmatrix}$$

Thuật toán CANNY

Ý tưởng:

Canny đáp ứng 3 tiêu chí “biên tối ưu”:

✓ ít nhiễu

✓ vị trí chính xác

✓ biên mỏng và liên tục

Đặc điểm:

- Kết quả mịn và đẹp nhất
- Chính xác hơn Sobel & Laplacian
- Chi phí tính toán lớn nhất

Các bước:

1. Gaussian smoothing: giảm nhiễu
2. Sobel gradient: tính độ lớn & hướng biên
3. Non-Maximum Suppression (NMS): làm mỏng biên
4. Hysteresis thresholding:
    - T1: ngưỡng cao → biên mạnh
    - T2: ngưỡng thấp → biên yếu

Biên yếu được giữ nếu nối được tới biên mạnh

Quy trình xử lí ảnh

Tải ảnh vào hệ thống

→ dashboard_view.py gọi:

```
original_img, input_type = handle_upload(uploaded_file)
```

→ handle_upload() (trong edge_controller.py) gọi:

```
load_image(uploaded_file)
```

→ load_image() (trong utils.py) xử lý ảnh/csv

Chọn thuật toán xử lý ảnh

→ Người dùng chọn 1 trong các phương pháp (Sobel/Canny/Laplacian/).

→ Streamlit truyền lựa chọn sang controller:

```
processed_img, count = run_edge_detection(original_img, method, **params)
```

Controller gọi thuật toán và xử lý model

Mỗi lựa chọn thuật toán xử lí ảnh sẽ gọi 1 hàm tương ứng trong model.

Xuất ảnh ra View và tải về

→ dashboard_view.py

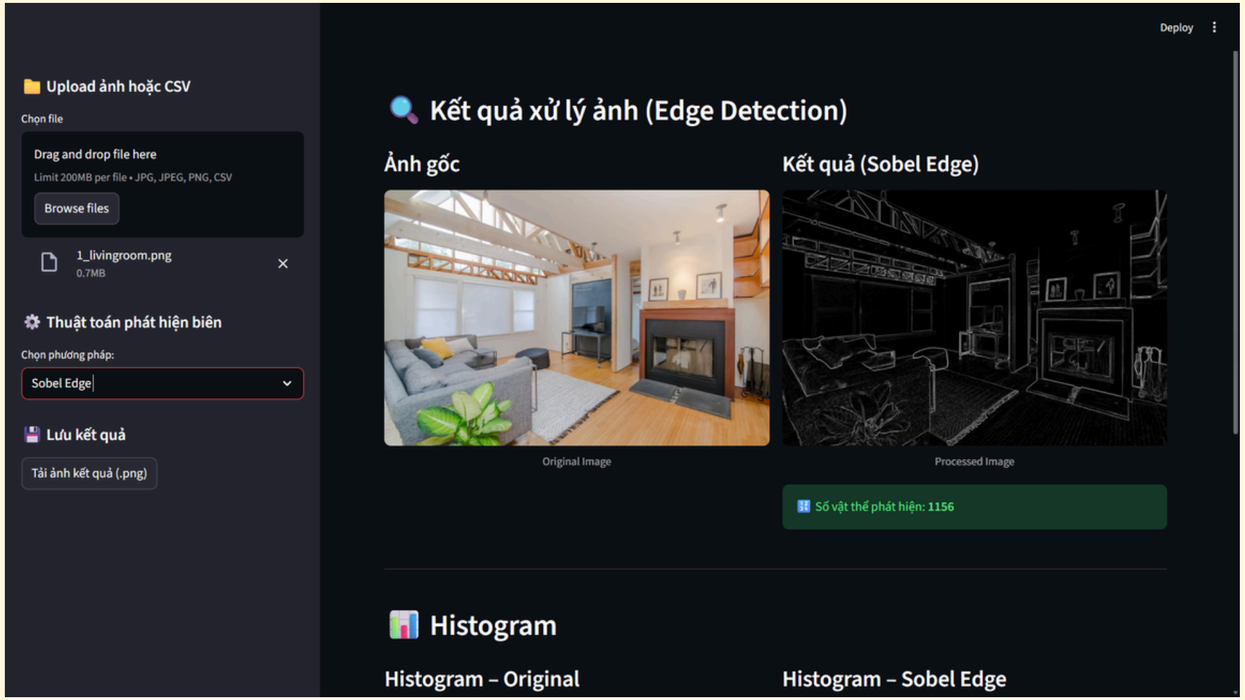
- Hiển thị ảnh gốc và histogram
- Hiển thị ảnh kết quả và histogram

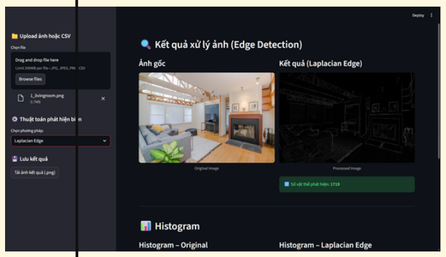
→ Người dùng tải xuống bằng

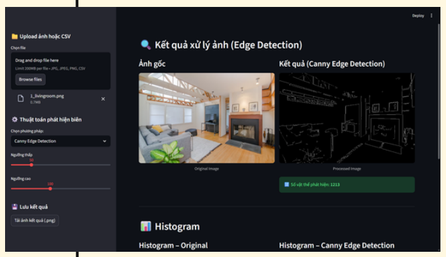
```
download_button
```

```python
def sobel_edge_with_count(img):
    if img.ndim == 3:   # RGB → Grayscale
        img = np.dot(img[...,:3], [0.2989, 0.5870, 0.1140]).astype(np.uint8)
    Gx = np.array([[-1, 0, 1],
                   [-2, 0, 2],
                   [-1, 0, 1]])

    Gy = np.array([[-1,-2,-1],
                   [ 0, 0, 0],
                   [ 1, 2, 1]])

    sx = conv2d(img, Gx)
    sy = conv2d(img, Gy)

    mag = np.sqrt(sx**2 + sy**2)
    mag = (mag / np.max(mag)) * 255
    edge = mag.astype(np.uint8)

    binary = (edge > 50).astype(np.uint8) * 255
    count = count_components(binary)
    return edge, count

# ================================
def canny_edge_with_count(img, t_low=50, t_high=100):
    if img.ndim == 3:   # RGB → Grayscale
        img = np.dot(img[...,:3], [0.2989, 0.5870, 0.1140]).astype(np.uint8)
    blur = gaussian_blur(img)

    # Sobel
    Gx = np.array([[-1,0,1],[-2,0,2],[-1,0,1]])
    Gy = np.array([[-1,-2,-1],[0,0,0],[1,2,1]])

    sx = conv2d(blur, Gx)
    sy = conv2d(blur, Gy)

    mag = np.sqrt(sx**2 + sy**2)
    mag = (mag / np.max(mag)) * 255
    theta = np.arctan2(sy, sx)

    # NMS
    nms_img = nms(mag, theta)

    # Hysteresis (đầy đủ, không cắt)
    edges = hysteresis(nms_img, t_low, t_high)

    # Count
    count = count_components(edges)

    return edges, count

def laplacian_edge_with_count(img):
    if img.ndim == 3:   # RGB → Grayscale
        img = np.dot(img[...,:3], [0.2989, 0.5870, 0.1140]).astype(np.uint8)
    L = np.array([[0,1,0],
                  [1,-4,1],
                  [0,1,0]])

    lap = conv2d(img, L)
    lap = np.abs(lap)
    lap = (lap / np.max(lap)) * 255
    edge = lap.astype(np.uint8)

    binary = (edge > 50).astype(np.uint8) * 255
    count = count_components(binary)
    return edge, count
```

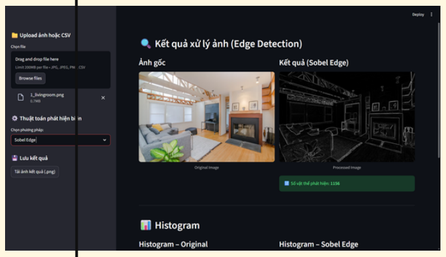
Demo giao diện

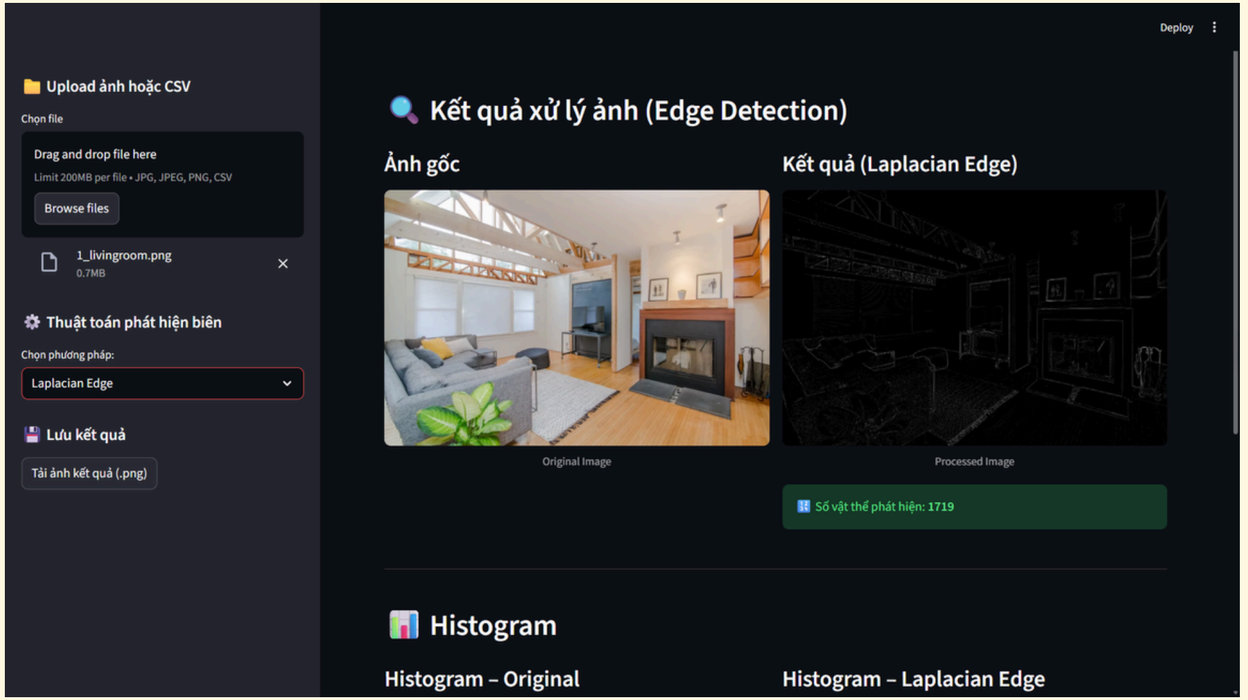


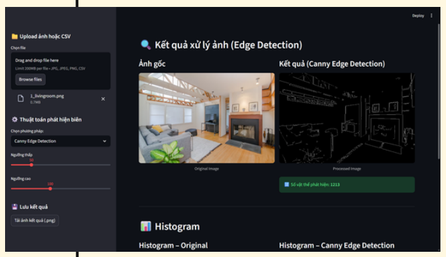




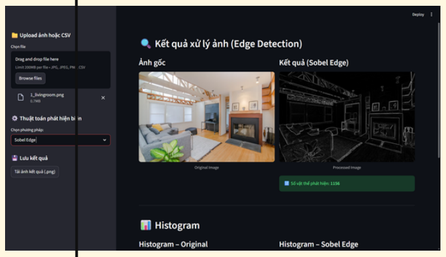
Demo giao diện

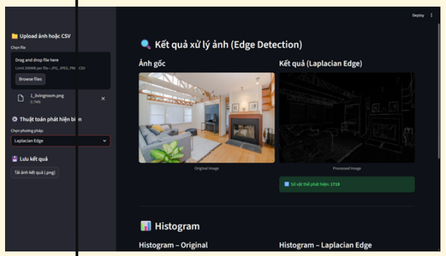


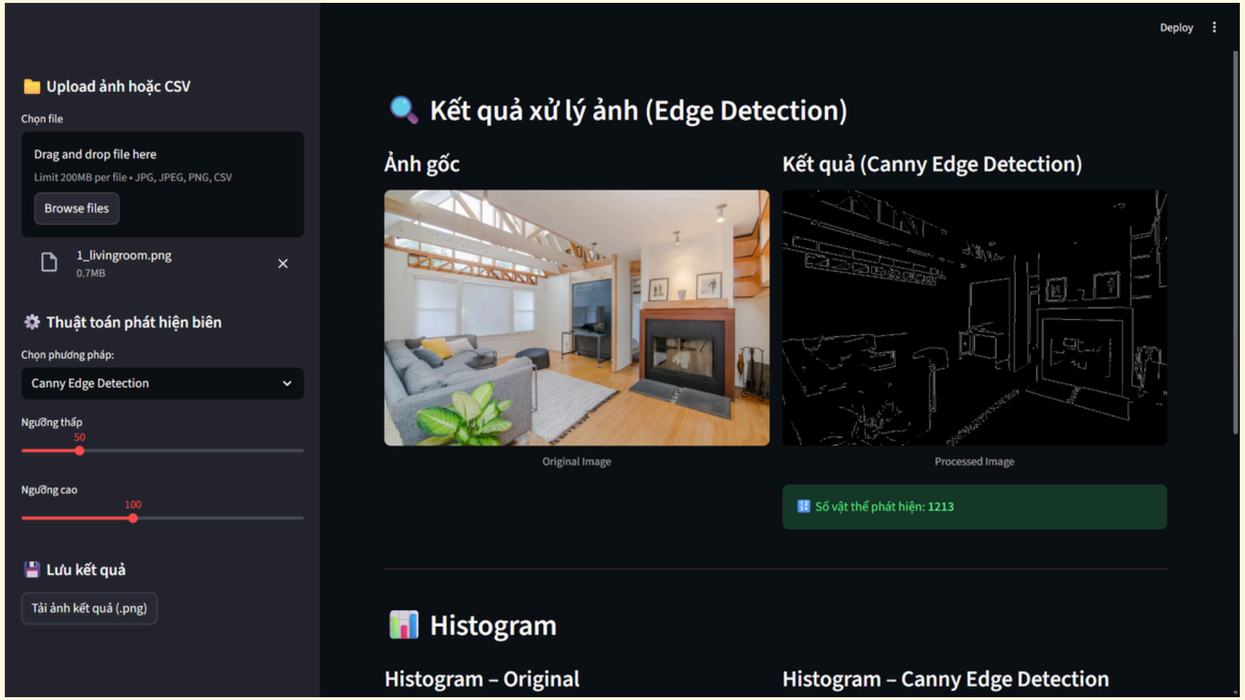




Demo giao diện







Kết luận

Ứng dụng vận hành ổn định, realtime cho mọi phương pháp xử lý ảnh.

Đã tích hợp các thuật toán phát hiện biên cổ điển: Sobel, Laplacian, Canny.

Chức năng đếm vật thể hỗ trợ phân tích ảnh nhị phân.

THANKS FOR WATCHING