Học viện Công Nghệ Bưu Chính Viễn Thông

Phát hiện biên ảnh và ứng dụng

Canny, Sobel, và Laplacian

Nguyễn Tuấn Nam B22DCCN561

Học viện Công Nghệ Bưu Chính Viễn Thông

Tổng quan

Đề tài: Phát hiện biên ảnh và ứng dụng

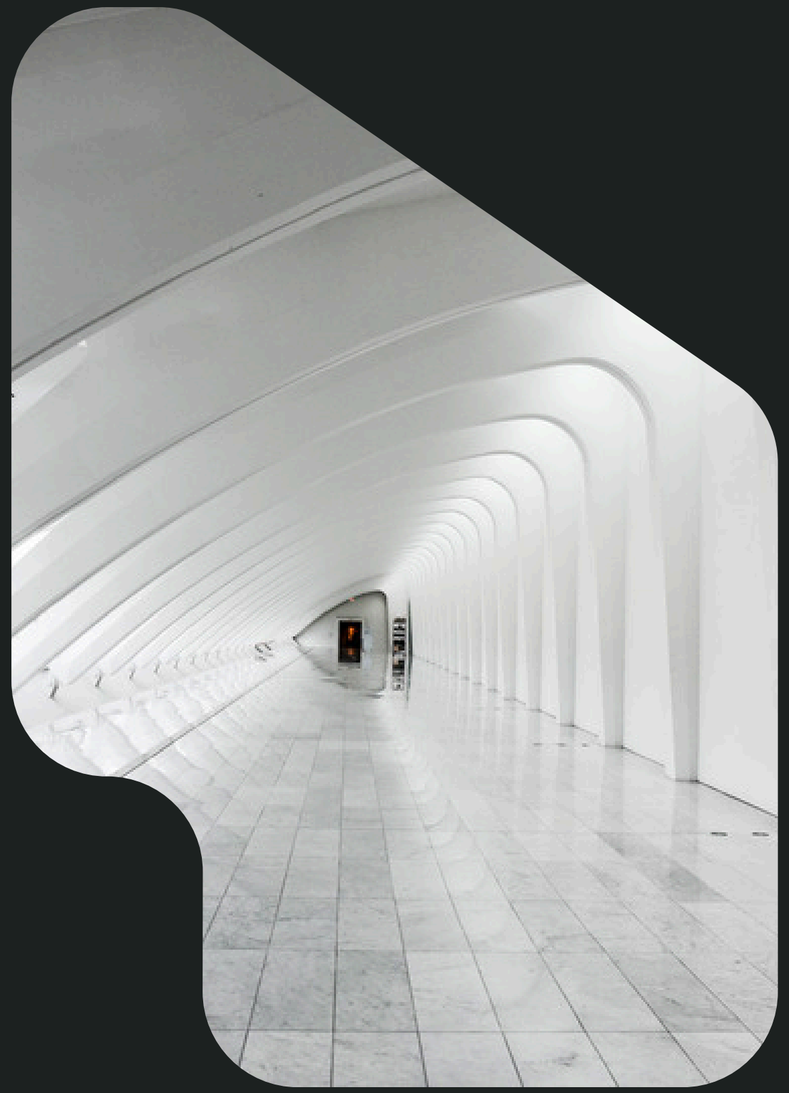
Giới thiệu

- **Nghiên cứu lý thuyết:** Tìm hiểu và phân tích 3 thuật toán phát hiện biên kinh điển: Sobel, Laplacian, và Canny.
- **Xây dựng ứng dụng:**

1. Cài đặt 3 thuật toán trên.
2. Phát triển ứng dụng "Đếm vật thể" dựa trên Canny.
3. (Nâng cao) Phát triển ứng dụng "Phát hiện làn đường" video.

- **Sản phẩm:** Một ứng dụng GUI (Giao diện đồ họa) hoàn chỉnh cho phép tương tác và trực quan hóa kết quả.

Nguồn: Thêm nguồn tham khảo ở đây.



Quay lại Trang Tổng quan.

Công nghệ sử dụng

Công nghệ:

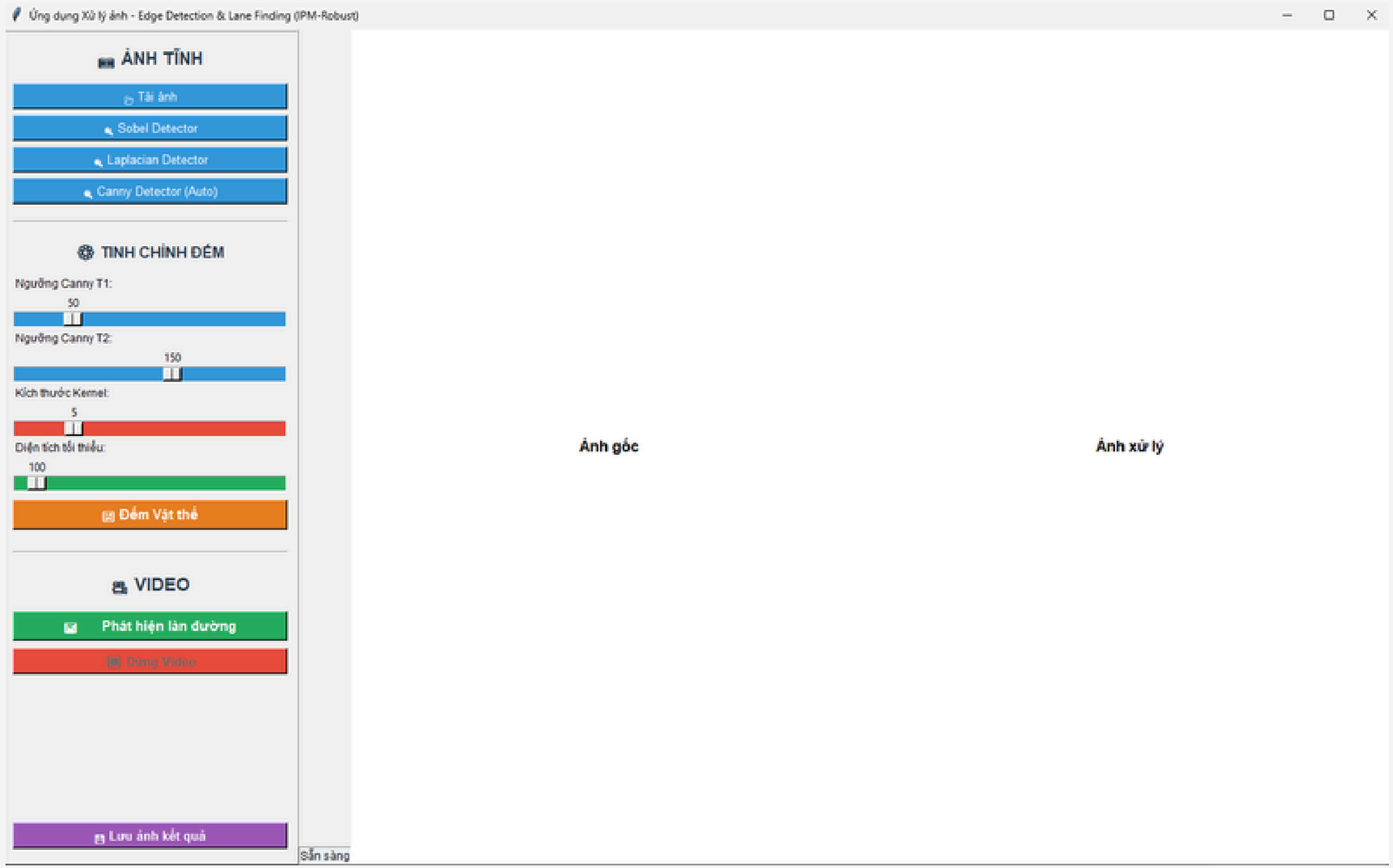
- Ngôn ngữ: Python 3
- Giao diện (GUI): Tkinter
- Xử lý ảnh: OpenCV
- Tính toán ma trận: NumPy
- Hỗ trợ: PIL, collections.deque

NumPy

Quay lại Trang Tổng quan.

Giao diện phần mềm

Quay lại Trang Tổng quan.



Nguồn: Thêm nguồn tham khảo ở đây.

- Khung điều khiển (Trái):

1. Các nút chức năng (Tải ảnh, Chạy thuật toán).
2. Các slider tinh chỉnh (Canny T1, T2, Kernel, Min Area).
3. Nút chạy Video và Lưu ảnh.

- Khung hiển thị (Phải):

1. Panel Ảnh gốc: Hiển thị input.
2. Panel Ảnh xử lý: Hiển thị output (kết quả).

- Mục đích: Cho phép người dùng so sánh trực quan và tinh chỉnh tham số.

Thuật toán Sobel

Quay lại Trang Tổng quan.

- **Lý thuyết:**
- Toán tử Sobel được xác định thông qua việc tính toán các đạo hàm thành phần Gx (theo hướng x) và Gy (theo hướng y) trong một cửa sổ 3x3 lân cận pixel (*i*,*j*)
- Độ lớn gradient: |G| = |Gx|+|Gy|
- Nhạy cảm với nhiễu.

```python
def sobel_detector(gray_img):
    """Phát hiện biên bằng Sobel với cải thiện."""
    blurred = apply_gaussian_blur(gray_img, (3, 3))
    grad_x = cv2.Sobel(blurred, cv2.CV_64F, 1, 0, ksize=3)
    grad_y = cv2.Sobel(blurred, cv2.CV_64F, 0, 1, ksize=3)
    magnitude = np.sqrt(grad_x**2 + grad_y**2)
    magnitude = np.uint8(np.clip(magnitude, 0, 255))
    return magnitude
```

Ảnh gốc

Kết quả Sobel

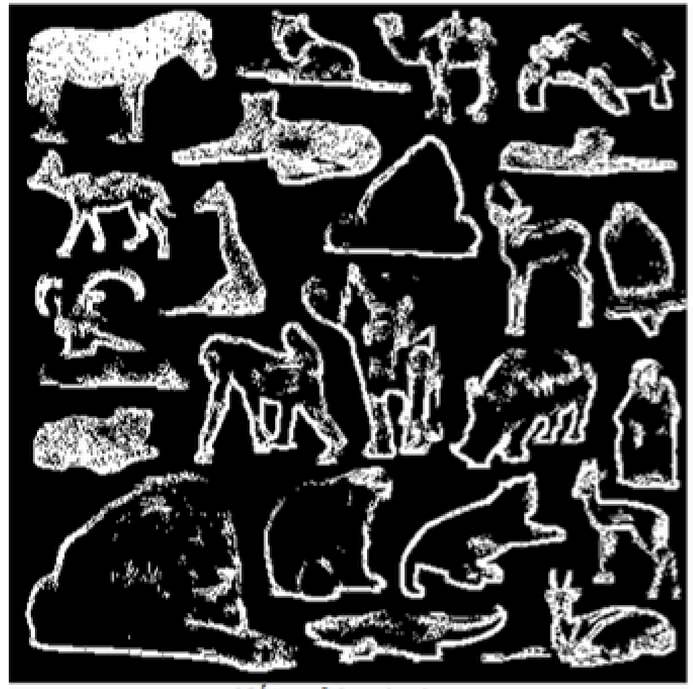
Thuật toán Laplacian

Quay lại Trang Tổng quan.

- *Lý thuyết:*

1. *Sử dụng đạo hàm bậc 2.*
2. *Tìm các điểm "zero-crossing" (nơi đạo hàm đổi dấu) để xác định cạnh.*
3. *Rất nhạy cảm với nhiễu (cần lọc nhiễu tốt trước khi chạy).*

Ảnh gốc



Kết quả Laplacian

```python
def laplacian_detector(gray_img):      Pin selection to current chat prompt (Ctrl+
    """Phát hiện biên bằng Laplacian với cải thiện."""
    blurred_img = apply_bilateral_filter(gray_img)
    lap = cv2.Laplacian(blurred_img, cv2.CV_64F, ksize=3)
    laplacian_result = cv2.convertScaleAbs(lap)
    _, thresholded = cv2.threshold(laplacian_result, 30, 255, cv2.THRESH_BINARY)
    return thresholded
```

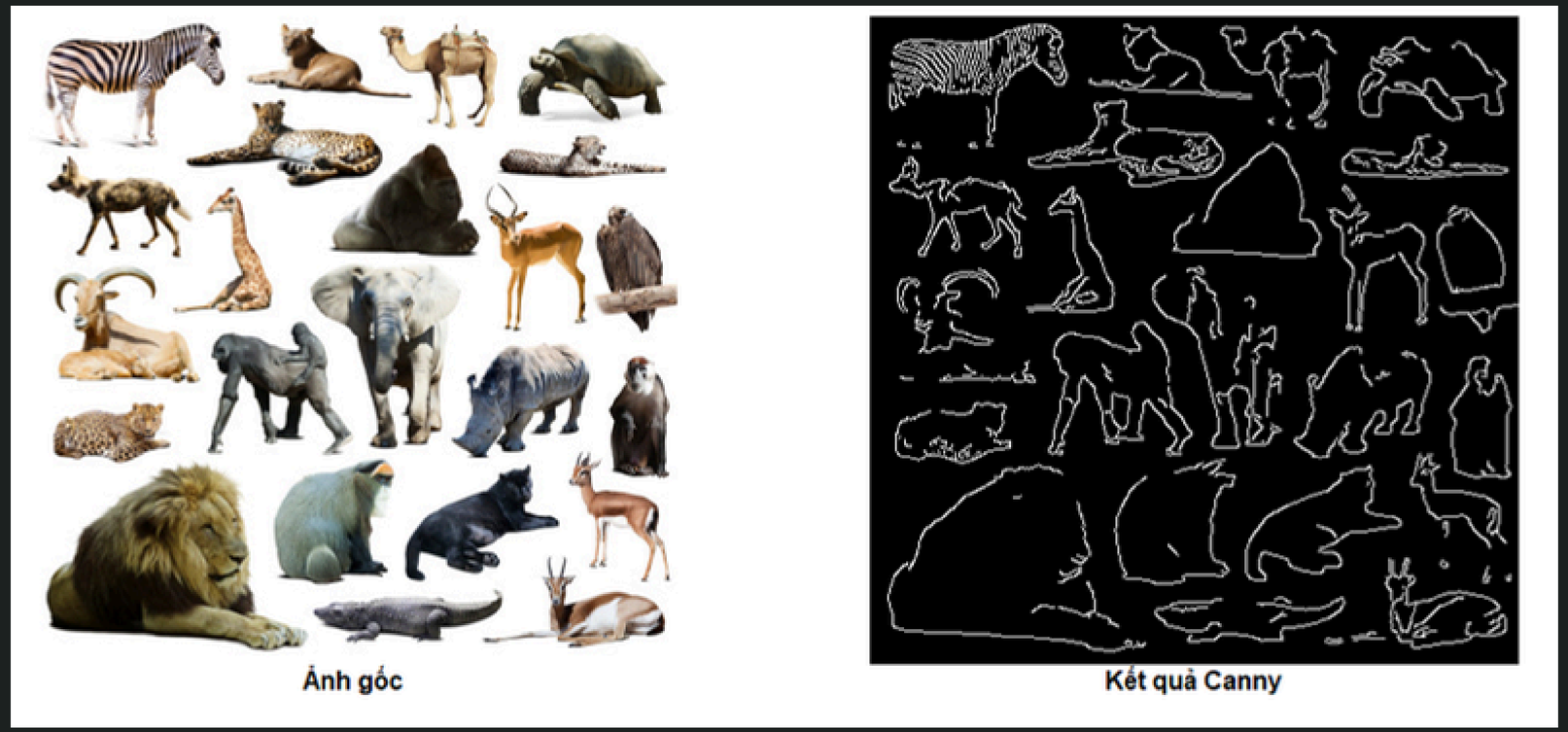
Thuật toán Canny

Quay lại Trang Tổng quan.

- *Lý thuyết (5 bước):*

1. *Gaussian Blur: Giảm nhiễu.*
2. *Sobel: Tính độ lớn và hướng gradient.*
3. *Non-maximum Suppression: Làm mỏng cạnh (chỉ giữ điểm cực đại).*
4. *Ngưỡng kép (T1, T2): Phân loại cạnh (Mạnh, Yếu, Không).*
5. *Hysteresis Thresholding: Liên kết cạnh (biến cạnh Yếu thành Mạnh nếu nối với cạnh Mạnh).*

```python
def canny_detector(gray_img, auto_threshold=True, t_lower=50, t_upper=150):
    """Phát hiện biên bằng Canny với auto threshold cải thiện."""
    blurred_img = apply_bilateral_filter(gray_img)
    if auto_threshold:
        v = np.median(blurred_img)
        sigma = 0.33
        t_lower = int(max(0, (1.0 - sigma) * v))
        t_upper = int(min(255, (1.0 + sigma) * v))
    edges = cv2.Canny(blurred_img, t_lower, t_upper)
```



Ảnh gốc

Kết quả Canny

Ứng dụng 1: Đếm vật thể

Quay lại Trang Tổng quan.

Mục tiêu	Đếm số lượng vật thể (ví dụ: đồng xu, ốc vít) trong ảnh.
Pipeline xử lý	count_objects
Input:	Ảnh gốc + Tham số từ slider (T1, T2, Kernel, Min Area).
Canny	Tạo ảnh biên.
Morphology	Dùng Dilate (giãn) và Close (đóng) để "vá" các đường viền bị đứt, tạo thành các vùng khép kín.
Find Contours	Tìm tất cả các đường viền khép kín
Lọc	Loại bỏ các contour nhiễu.
Output:	Vẽ và đếm các contour hợp lệ.

Chi tiết bước lọc

Quay lại Trang Tổng quan.

- **Tại sao phải lọc?** findContours tìm thấy *tất cả* các viền, kể cả nhiễu li ti hoặc các đường thẳng.
- **Logic Lọc trong Code:**

1. **Lọc theo diện tích:**

- area = cv2.contourArea(cnt)
- if area > min_area: (Loại bỏ các chấm nhiễu nhỏ, min_area từ slider).

2. **Lọc theo "Độ tròn" (Circularity):**

- circularity = 4 * np.pi * area / (perimeter * perimeter)
- if circularity > 0.1: (Giá trị 1.0 là hình tròn hoàn hảo. Giá trị 0.1 loại bỏ các hình quá méo hoặc đường thẳng, vốn không phải là vật thể).

Ảnh gốc

Kết quả: 23 vật thể

Ứng dụng 2: Phát hiện làn đường

- **Thách thức: Xử lý video thời gian thực, chống chịu với bóng râm và thay đổi ánh sáng (làm vạch kẻ mất màu).**
    - Giải pháp Pipeline (Robust Pipeline):

1. Kênh 1 (Màu sắc): Lọc màu Vàng/Trắng (Không gian màu HSV).
2. Kênh 2 (Cấu trúc): Lọc cạnh Canny (Ảnh xám).
3. Kết quả: cv2.bitwise_or(Kenh_1, Kenh_2)

- **Tại sao? Khi xe vào bóng râm (Kênh 1 thất bại), Kênh 2 vẫn hoạt động.**

Quay lại Trang Tổng quan.

Ứng dụng 2: Kỹ thuật IPM (Bird's-Eye View)

- **Vấn đề: Ảnh phối cảnh (Perspective) làm các làn song song bị hội tụ, khó tính toán.**
    - Giải pháp: Inverse Perspective Mapping (IPM)

1. Warp: "Làm phẳng" con đường thành ảnh nhìn từ trên xuống (Bird's-eye view).
2. Xử lý: Trong ảnh này, làn đường song song thực sự, giúp Canny và Hough chạy cực kỳ chính xác.
3. Un-warp: Vẽ vùng làn đường (màu xanh lá) lên ảnh bird's-eye.
4. Dùng ma trận nghịch đảo Minv để "vẽ" vùng này ngược lại ảnh gốc.

Quay lại Trang Tổng quan.

Nghiệm thu sản phẩm

(DEMO ỨNG DỤNG)

Trực quan hóa: Giao diện hiển thị song song Ảnh Gốc và Ảnh Xử lý.

- Chức năng "Lưu ảnh kết quả" (save_image).
- Sử dụng filedialog.asksaveasfilename để chọn nơi lưu.
- Sử dụng cv2.imwrite để lưu ảnh kết quả ra file .png hoặc .jpg.

Quay lại Trang Tổng quan.

Kết luận

Quay lại Trang Tổng quan.

- Hoàn thành cài đặt 3 thuật toán Sobel, Laplacian, Canny.
- Xây dựng thành công ứng dụng Đếm vật thể và Phát hiện làn đường.
- Ứng dụng GUI (Tkinter) hoạt động ổn định, đáp ứng yêu cầu nghiệm thu.

- Hướng phát triển:

1. Tự động cân chỉnh IPM thay vì "chỉnh tay".
2. Nâng cấp lên polyfit bậc 2 để nhận diện đường cong.

Cảm ơn thầy và các bạn đã lắng nghe!

Quay lại Trang Tổng quan.