BÁO CÁO BÀI TẬP GIỮA KÌ MÔN HỌC XỬ LÍ ẢNH

(AIT3002#_2)

Link github

Mã số sinh viên:22022584

Sinh viên thực hiện: Nguyễn Huy Hoàng

Ngày: 01/04/2025

Mục lục:

How to run code :

Install requirements

pip install -r requirements.txt

Running main code
python object_counter.py

1. Giới thiệu

Triển khai một ứng dụng nhỏ có khả năng đếm số lượng đối tượng/vật thể trong một ảnh dựa trên một mẫu do người dùng định nghĩa, cho phép người dùng lựa chọn ảnh, xác định mẫu, và tinh chỉnh các tham số xử lý ảnh để đạt được kết quả đếm chính xác nhất có thể trong các điều kiện ảnh khác nhau. Chương trình được viết bằng ngôn ngữ Python sử dụng thư viện OpenCV (cv2) cho các tác vụ xử lý ảnh chính.

2. Giải thích phương pháp

2.1. Tổng quan quy trình

1. **Tải ảnh:** Người dùng chọn (hoặc chương trình tải mặc định) ảnh đầu vào cần xử lý.
2. **Định nghĩa Template:** Người dùng vẽ thủ công đường viền xung quanh một đối tượng mẫu trong ảnh.
3. **Tiền xử lý ảnh:** Ảnh gốc được xử lý để giảm nhiễu và tăng cường các đặc trưng quan trọng (như biên cạnh).
4. **Phát hiện biên:** Áp dụng thuật toán phát hiện biên để tạo ra một ảnh nhị phân làm nổi bật đường viền của các đối tượng.
5. **Tìm đường viền:** Tìm tất cả các đường viền (contours) khép kín trong ảnh biên.
6. **So khớp và Lọc:** So sánh từng đường viền tìm được với đường viền của template dựa trên các tiêu chí về hình dạng, kích thước, và tùy chọn màu sắc. Các đường viền phù hợp được giữ lại.
7. **Hiển thị kết quả:** Vẽ các đường viền khớp lên ảnh gốc và hiển thị số lượng đối tượng đếm được. Cung cấp thanh công cụ điều chỉnh tham số để tối ưu hóa kết quả.

2.2. Định nghĩa đối tượng mẫu (Template Definition)

- **Mục đích:** Cung cấp một tham chiếu chuẩn về hình dạng (và tùy chọn màu sắc) của đối tượng cần đếm.
- **Cách dùng:** Người dùng nhấp và kéo chuột trái để vẽ một đa giác tự do bao quanh đối tượng mẫu. Khi nhả chuột, điểm cuối được nối với điểm đầu để tạo thành một vùng kín.
- **Lưu trữ:** Vùng được vẽ được chuyển đổi thành một đường viền (template_contour) bằng cv2.findContours trên một mặt nạ (mask) được tạo từ các điểm vẽ. Đường viền này và các đặc trưng của nó (diện tích, moment, histogram màu) được lưu lại để so sánh sau này.

2.3. Tiền xử lý ảnh (Image Preprocessing)

- **Mục đích:** Giảm nhiễu, chuẩn hóa độ sáng/tương phản, và làm nổi bật các cấu trúc cần thiết cho việc phát hiện biên, giúp thuật toán phát hiện biên hoạt động hiệu quả hơn.
- **Các bước và tùy chọn:**
    - **Chuyển đổi ảnh xám (cv2.cvtColor):** Hầu hết các thuật toán phát hiện biên hoạt động trên ảnh đơn kênh.
    - **Làm mờ (Blurring):**
        - *Gaussian Blur (cv2.GaussianBlur):* Giảm nhiễu tần số cao (chi tiết nhỏ, nhiễu Gaussian) trước khi phát hiện biên. Kích thước kernel (blur_kernel) có thể điều chỉnh.
    - **Cải thiện tương phản (Tùy chọn preprocess_method = 1):**
        - *CLAHE (cv2.createCLAHE):* Cân bằng histogram thích ứng cục bộ, hữu ích cho ảnh có điều kiện ánh sáng không đồng đều.
    - **Xử lý hình thái học (Tùy chọn preprocess_method = 2):**
        - *Closing (cv2.morphologyEx với cv2.MORPH_CLOSE):* Lấp các lỗ nhỏ bên trong đối tượng hoặc các khoảng trống nhỏ trên đường viền.
        - *Opening (cv2.morphologyEx với cv2.MORPH_OPEN):* Loại bỏ các đốm nhiễu nhỏ bên ngoài đối tượng.
        - Sử dụng kernel (morph_kernel) có thể điều chỉnh.

2.4. Phát hiện biên (Edge Detection)

- **Mục đích:** Tạo ra một ảnh nhị phân chỉ chứa các pixel thuộc về biên cạnh của các đối tượng trong ảnh.
- **Các phương pháp (Tùy chọn detection_method):**
    - **Canny (cv2.Canny):** Phổ biến nhất, dùng đạo hàm bậc nhất (Sobel), non-maximum suppression và hai ngưỡng (hysteresis thresholding - canny_threshold1, canny_threshold2) để phát hiện biên mạnh và nối các đoạn biên yếu liên kết với biên mạnh.
    - **Sobel (cv2.Sobel):** Tính toán gradient theo hướng x và y. Độ lớn gradient được dùng để xác định pixel biên. Ít phức tạp hơn Canny nhưng có thể tạo biên dày hơn. Ngưỡng hóa (cv2.threshold) được áp dụng trên độ lớn gradient.
    - **Laplacian (cv2.Laplacian):** Sử dụng đạo hàm bậc hai. Phát hiện tốt các thay đổi cường độ nhanh nhưng rất nhạy cảm với nhiễu. Ngưỡng hóa được áp dụng trên giá trị tuyệt đối của Laplacian.
    - **Difference of Gaussians (DoG):** Tính hiệu của hai ảnh đã được làm mờ bằng Gaussian blur với hai độ lệch chuẩn (hoặc kernel size) khác nhau. Đây là một cách xấp xỉ bộ lọc Laplacian of Gaussian (LoG), giúp phát hiện biên ở các tỷ lệ khác nhau. Ngưỡng hóa được áp dụng trên ảnh hiệu.
- **Ngưỡng thích ứng (Tùy chọn use_adaptive_threshold):**
    - *cv2.adaptiveThreshold*: Tính ngưỡng cục bộ cho từng vùng ảnh nhỏ, hữu ích khi ảnh có độ sáng thay đổi đáng kể. Kết quả có thể được kết hợp (OR) với kết quả của phương pháp phát hiện biên chính.
- **Giãn nở (Dilation - cv2.dilate):**
    - Làm dày hoặc nối liền các đoạn biên bị đứt gãy do nhiễu hoặc phát hiện biên không hoàn hảo. Số lần lặp (dilation_iterations) có thể điều chỉnh. Có thể kết hợp với Erosion để lọc nhiễu nhỏ phát sinh sau dilation.

2.5. Tìm và lọc đường viền (Contour Finding and Filtering)

- **Tìm đường viền (cv2.findContours):** Áp dụng trên ảnh biên nhị phân đã qua xử lý. Thuật toán này tìm tất cả các đường cong liên tục bao quanh các vùng pixel có cùng cường độ (trong trường hợp này là các vùng trắng trên nền đen của ảnh biên). cv2.RETR_EXTERNAL chỉ lấy các đường viền ngoài cùng.

- **Lọc nhiễu ban đầu:** Loại bỏ các đường viền quá nhỏ (min_contour_area) vì chúng thường là nhiễu.

2.6. So khớp đường viền (Contour Matching)

- **Mục đích:** Xác định xem một đường viền tìm được có tương ứng với đối tượng mẫu hay không.
- **Các tiêu chí so khớp:**
    - **Tỷ lệ diện tích (Area Ratio):** Diện tích của đường viền đang xét (cv2.contourArea) phải nằm trong một khoảng tỷ lệ (area_ratio_min, area_ratio_max) so với diện tích của template. Điều này giúp loại bỏ các đối tượng quá lớn hoặc quá nhỏ so với mẫu.
    - **Độ tương đồng hình dạng (Shape Similarity):**
        - *cv2.matchShapes*: So sánh hai đường viền dựa trên Hu Moments (bất biến với phép dịch chuyển, xoay và tỷ lệ). Giá trị trả về càng nhỏ thì độ tương đồng càng cao. Metric cv2.CONTOURS_MATCH_I3 được sử dụng. Kết quả so sánh với similarity_threshold.
    - **Độ tương đồng màu sắc (Tùy chọn use_color_matching):**
        - *Histogram màu HSV*: Tính toán histogram 3D trong không gian màu HSV cho vùng ảnh bên trong template và bên trong đường viền đang xét. HSV thường ổn định hơn BGR dưới sự thay đổi ánh sáng.
        - *cv2.calcHist, cv2.normalize, cv2.compareHist*: dùng phương pháp so sánh tương quan (cv2.HISTCMP_CORREL) để đo độ tương đồng giữa hai histogram. Giá trị càng gần 1 càng tương đồng.
        - *Kết hợp điểm*: Điểm tương đồng hình dạng và màu sắc được kết hợp bằng trọng số (color_weight). Nếu độ tương đồng màu thấp hơn color_similarity_threshold, đối tượng bị loại bỏ ngay lập tức.
    - **Đặc trưng hình học bổ sung (Tùy chọn use_shape_context):**
        - *Độ tròn (Circularity)*: Tính từ chu vi (cv2.arcLength) và diện tích. Đo mức độ "tròn" của hình dạng.
        - *Tỷ lệ khung hình (Aspect Ratio)*: Tính từ chiều rộng và chiều cao của hình chữ nhật xoay nhỏ nhất bao quanh đường viền (cv2.minAreaRect).

    - *Kết hợp điểm*: Sự khác biệt về độ tròn và tỷ lệ khung hình giữa template và contour được tính toán và kết hợp với điểm matchShapes để tạo ra một điểm đánh giá tổng hợp, giúp phân biệt các hình dạng phức tạp hơn mà chỉ Hu Moments có thể không đủ.

2.7. Điều chỉnh tham số tương tác (Interactive Parameter Tuning)

- Sử dụng các thanh trượt (trackbars) của OpenCV (cv2.createTrackbar) cho phép người dùng thay đổi các tham số quan trọng (ngưỡng Canny, ngưỡng tương đồng, tỷ lệ diện tích, kernel size, phương pháp, trọng số màu, v.v.) và xem kết quả cập nhật ngay lập tức. Điều này rất quan trọng vì không có bộ tham số nào tối ưu cho mọi ảnh và mọi đối tượng.

3. Chi tiết triển khai

3.1. Cấu trúc chương trình (Class ObjectCounter)

- Toàn bộ logic được đóng gói trong lớp ObjectCounter.

3.2. Triển khai lựa chọn Template

- Sử dụng hàm mouse_callback với cửa sổ hiển thị (cv2.setMouseCallback).
- Tạo một ảnh mask đen (np.zeros) cùng kích thước với ảnh gốc.
- Dùng cv2.fillPoly để vẽ đa giác được tạo từ template_points lên mask với màu trắng, cv2.findContours trên mask để trích xuất đường viền duy nhất (template_contour).

Cho phép **vẽ lại (r key)** nếu người dùng không hài lòng.

3.3. Triển khai các phương pháp tiền xử lý và phát hiện biên

- Hàm preprocess_image thực hiện các bước tiền xử lý và phát hiện biên.
- Cấu trúc if/elif dùng các giá trị của self.preprocess_method và self.detection_method để lựa chọn và áp dụng các hàm (cv2.GaussianBlur, cv2.createCLAHE, cv2.morphologyEx, cv2.Canny, cv2.Sobel, cv2.Laplacian, tính toán DoG thủ công).
- **Vấn đề gặp phải:** Các đối tượng khác nhau và điều kiện ảnh khác nhau đòi hỏi các kỹ thuật tiền xử lý và phát hiện biên khác nhau để tối ưu.
- **Giải pháp:** Cung cấp nhiều tùy chọn phổ biến (Standard, CLAHE, Morphological cho tiền xử lý; Canny, Sobel, Laplacian, DoG cho phát hiện biên) để người dùng

có thể thử nghiệm và chọn phương pháp phù hợp nhất thông qua trackbar. Các tham số kernel và ngưỡng cũng có thể điều chỉnh.

3.4. Triển khai cơ chế so khớp và lọc

- Hàm match_contours nhận danh sách các đường viền tìm được và đường viền template.
- Lặp qua từng đường viền, tính toán diện tích (cv2.contourArea).
- Áp dụng bộ lọc tỷ lệ diện tích (area_ratio_min, area_ratio_max).
- Tính toán độ tương đồng hình dạng cv2.matchShapes và so sánh với similarity_threshold.
- Các logic điều kiện (if self.use_color_matching, if self.use_shape_context) được thêm vào để tích hợp các tiêu chí so khớp nâng cao.
- **Vấn đề gặp phải:** Chỉ dựa vào một tiêu chí duy nhất (ví dụ: chỉ matchShapes) thường không đủ mạnh mẽ, dẫn đến nhận diện sai (false positives) hoặc bỏ sót (false negatives).
- **Giải pháp:** Kết hợp nhiều tiêu chí (diện tích, hình dạng cơ bản, tùy chọn màu sắc, tùy chọn đặc trưng hình học bổ sung) để tăng độ chính xác và độ tin cậy của việc so khớp.

3.5. Vấn đề gặp phải và giải pháp: Độ tương đồng hình dạng đơn thuần không đủ -> Triển khai So khớp màu sắc

- **Vấn đề:** Trong nhiều trường hợp, có thể có các đối tượng nhiễu hoặc đối tượng không mong muốn có hình dạng khá giống với template nhưng lại có màu sắc hoàn toàn khác. Chỉ dùng cv2.matchShapes sẽ dễ bị nhầm lẫn trong các tình huống này (ví dụ: đếm chuột xám nhưng ảnh có cả cục tẩy cùng hình dạng nhưng màu trắng).
- **Triển khai Giải pháp:**
    1. Thêm tùy chọn use_color_matching (điều khiển bằng trackbar).
    2. Khi định nghĩa template, tính và lưu histogram màu 3D HSV (self.template_color_histogram) của vùng ảnh bên trong template_contour bằng cv2.calcHist. Dùng không gian màu HSV vì nó tách biệt kênh màu (Hue) khỏi kênh độ sáng (Value), giúp ổn định hơn với thay đổi ánh sáng.
    3. Trong hàm match_contours, nếu use_color_matching được bật:

        - Tính histogram màu HSV cho đường viền đang xét.
        - So sánh hai histogram bằng cv2.compareHist (dùng cv2.HISTCMP_CORREL).
        - Loại bỏ ngay nếu độ tương đồng màu dưới ngưỡng color_similarity_threshold.
        - Kết hợp điểm tương đồng màu và điểm tương đồng hình dạng (matchShapes) bằng trọng số color_weight. Điểm tương đồng hình dạng được chuẩn hóa ngược (1 - matchShapes) để cùng thang đo với điểm màu (cao hơn là tốt hơn), sau đó kết hợp lại và chuyển về thang đo khoảng cách (thấp hơn là tốt hơn) để phù hợp với logic similarity_threshold.
    - **Kết quả:** Giúp loại bỏ hiệu quả các đối tượng có hình dạng giống nhưng màu sắc khác biệt, tăng độ chính xác.

3.6. Vấn đề gặp phải và giải pháp: matchShapes nhạy cảm với biến dạng nhỏ -> Triển khai Đặc trưng hình học bổ sung (Shape Context)

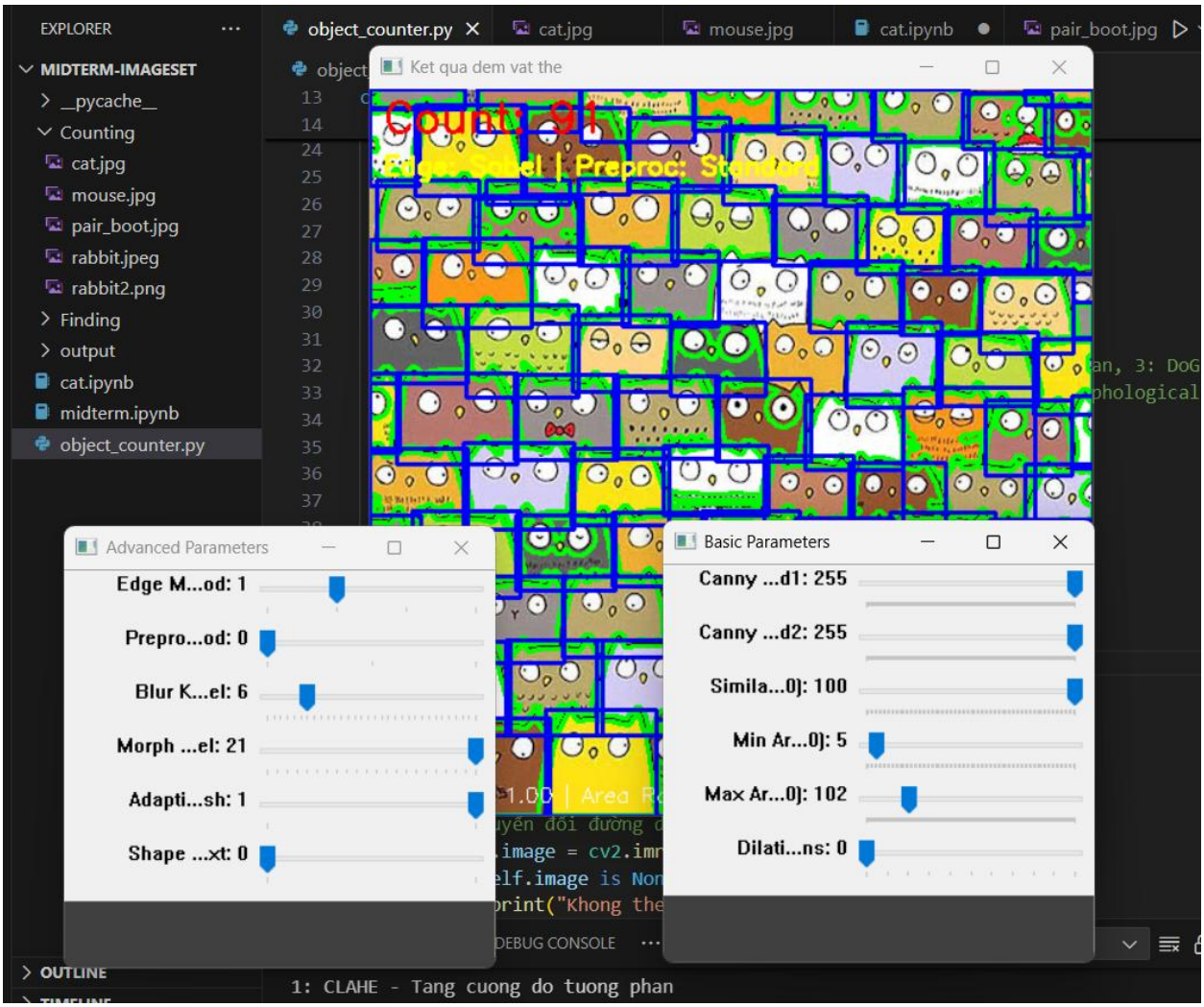
- **Vấn đề:** Hu Moments (cơ sở của matchShapes) tuy bất biến với xoay, tỷ lệ, dịch chuyển nhưng đôi khi vẫn nhạy cảm với nhiễu trên đường viền, biến dạng nhỏ hoặc tắc nghẽn (occlusion) nhẹ. Hai hình dạng trông khá giống nhau đối với mắt người có thể có điểm matchShapes khác biệt đáng kể.
- **Triển khai Giải pháp:**
    1. Thêm tùy chọn use_shape_context (điều khiển bằng trackbar).
    2. Trong match_contours, nếu use_shape_context được bật:
        - Tính toán các đặc trưng hình học bổ sung cho cả template và đường viền đang xét:
            - Độ tròn (Circularity): 4 * pi * area / perimeter^2.
            - Tỷ lệ khung hình (Aspect Ratio): Từ cv2.minAreaRect.
        - Tính toán sự khác biệt tuyệt đối về độ tròn và tỷ lệ khung hình giữa template và đường viền.
        - Kết hợp các điểm khác biệt này với điểm matchShapes gốc bằng cách dùng trọng số (ví dụ: 50% matchShapes, 25% khác biệt độ tròn, 25% khác biệt tỷ lệ khung hình) để tạo ra combined_score.

        - Nếu use_color_matching cũng đang bật, điểm màu cũng được tích hợp vào điểm tổng hợp này.
        - Sử dụng combined_score này (thay vì chỉ matchShapes) để so sánh với ngưỡng tương đồng (có thể cần điều chỉnh ngưỡng một chút, ví dụ similarity_threshold * 1.5).
    - **Kết quả:** Việc xem xét thêm các đặc trưng như độ tròn và tỷ lệ khung hình giúp mô tả hình dạng một cách toàn diện hơn, làm cho việc so khớp trở nên mạnh mẽ hơn trước các biến dạng nhỏ hoặc các đặc điểm mà Hu Moments không nắm bắt tốt.

3.7. Giao diện điều chỉnh tham số

Hàm update_parameter cập nhật giá trị thuộc tính tương ứng trong lớp (setattr(self, param_name, value)) và sau đó gọi lại hàm detect_and_display để người dùng thấy ngay ảnh hưởng của việc thay đổi tham số.

4. Kết quả


EXPLORER
object_counter.py
cat.jpg
mouse.jpg
cat.ipynb
pair_boot.jpg
MIDTERM-IMAGESET
__pycache__
Counting
cat.jpg
mouse.jpg
pair_boot.jpg
rabbit.jpeg
rabbit2.png
Finding
output
cat.ipynb
midterm.ipynb
object_counter.py
Ket qua dem vat the
Count: 91
Advanced Parameters
Edge M...od: 1
Prepro...od: 0
Blur K...el: 6
Morph ...el: 21
Adapti...sh: 1
Shape ...xt: 0
Basic Parameters
Canny ...d1: 255
Canny ...d2: 255
Simila...0]: 100
Min Ar...0]: 5
Max Ar...0]: 102
Dilati...ns: 0
DEBUG CONSOLE
OUTLINE
1: CLAHE - Tang cuong do tuong phan


Ket qua dem vat the
Count: 29
Edge: Sobel | Preproc: Standard
Preprocessed I
Advanced Parameters
Edge M...od: 1
Prepro...od: 0
Blur K...el: 8
Morph ...el: 21
Adapti...sh: 1
Shape ...xt: 1
Basic Parameters
Canny ...d1: 255
Canny ...d2: 255
Simila...0]: 50
Min Ar...0]: 35
Max Ar...0]: 88
Dilati...ns: 0

5. Kết luận

Chương trình object_counter.py đã triển khai thành công một hệ thống đếm đối tượng dựa trên template với nhiều tùy chọn linh hoạt. Nhưng vẫn cần cải thiện thêm ,thêm các phương pháp xử lí mới để xử lí các hình ảnh phức tạp như mouse,pair_boot