

Xử lý ảnh



CANNY, SOBEL LAPLACIAN



Giảng viên: Phạm Hoàng Việt

Nội dung

THUẬT TOÁN CANNY

THUẬT TOÁN SOBEL

THUẬT TOÁN LAPLACIAN

ỨNG DỤNG ĐÊM VẬT THỂ

NEXT >>>

Canny

Sobel

Laplacian

Thuật toán Canny (Canny Edge Detector),
được John Canny đề xuất năm 1986

Là một trong những phương pháp phát hiện biên
hiệu quả và phổ biến nhất.

Canny

Canny

Sobel

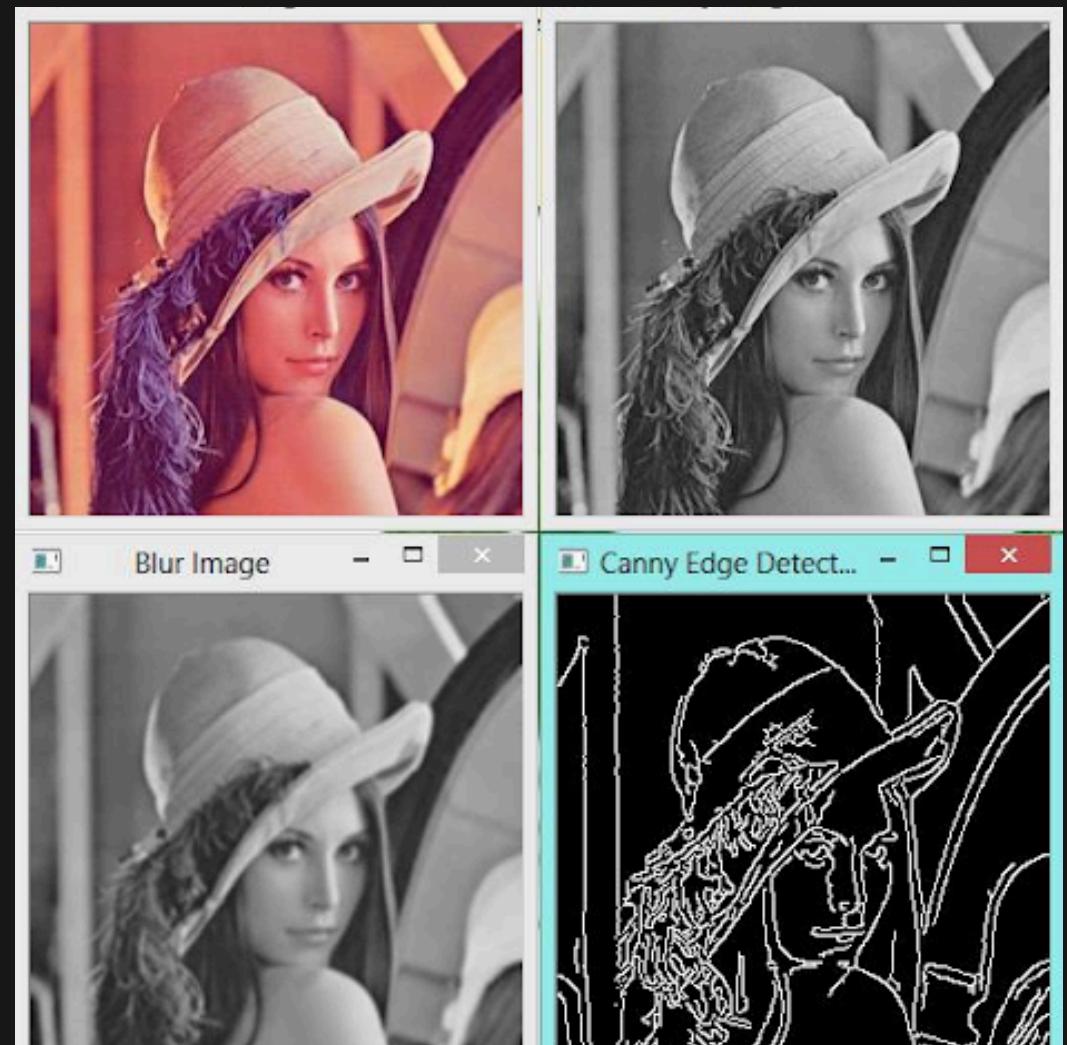
Laplacian

Thuật toán Canny (Canny Edge Detector),
được John Canny đề xuất năm 1986

Là một trong những phương pháp phát hiện biên
hiệu quả và phổ biến nhất.

Gồm nhiều giai đoạn:

- 1. Làm mượt ảnh để giảm nhiễu
- 2. Tính gradient để xác định vùng biến đổi cường độ mạnh
- 3. Làm mảnh biên (non-maximum suppression)
- 4. Kết nối biên (hysteresis thresholding).

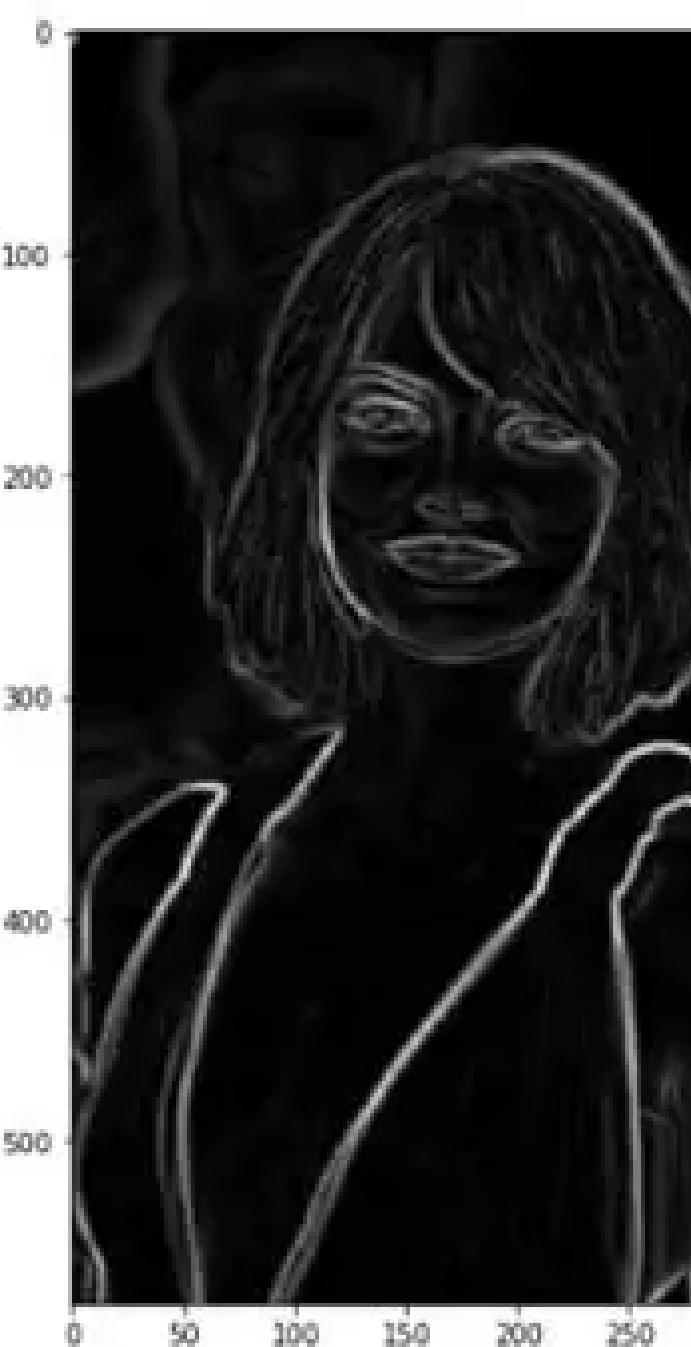


Canny

Canny

Sobel

Laplacian



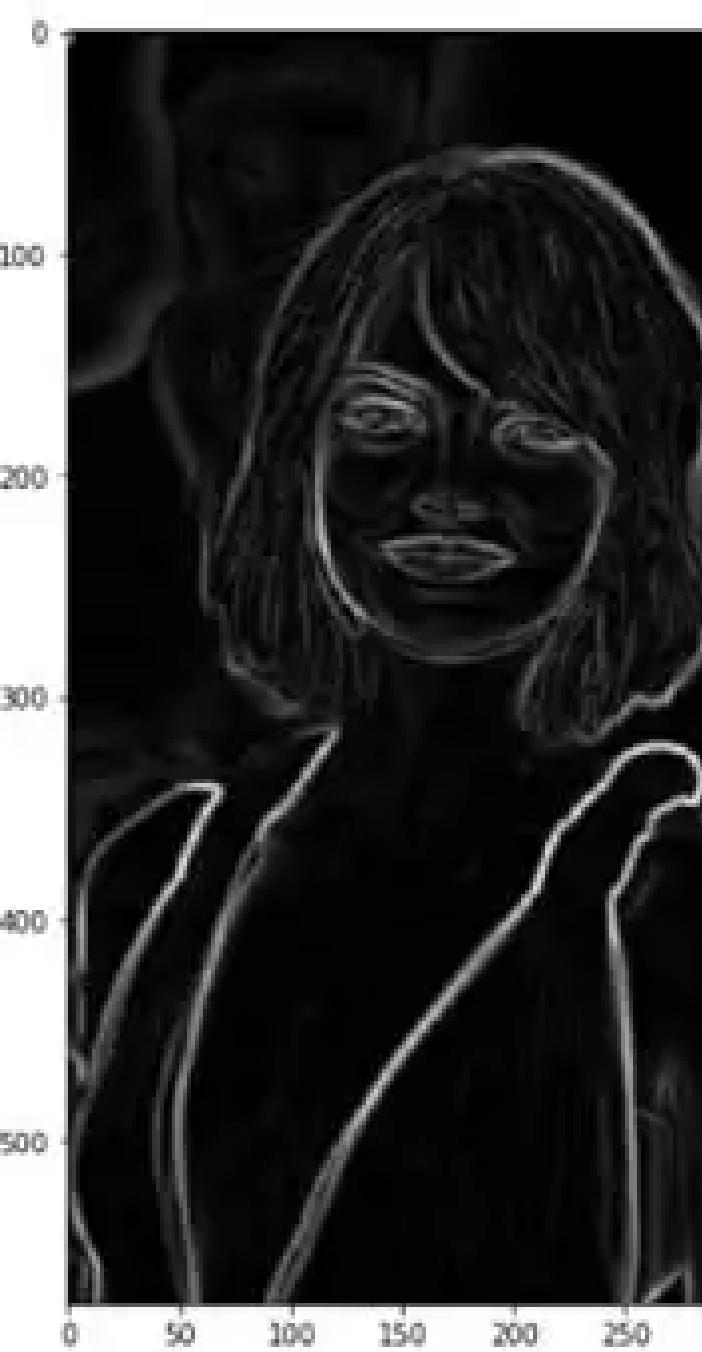
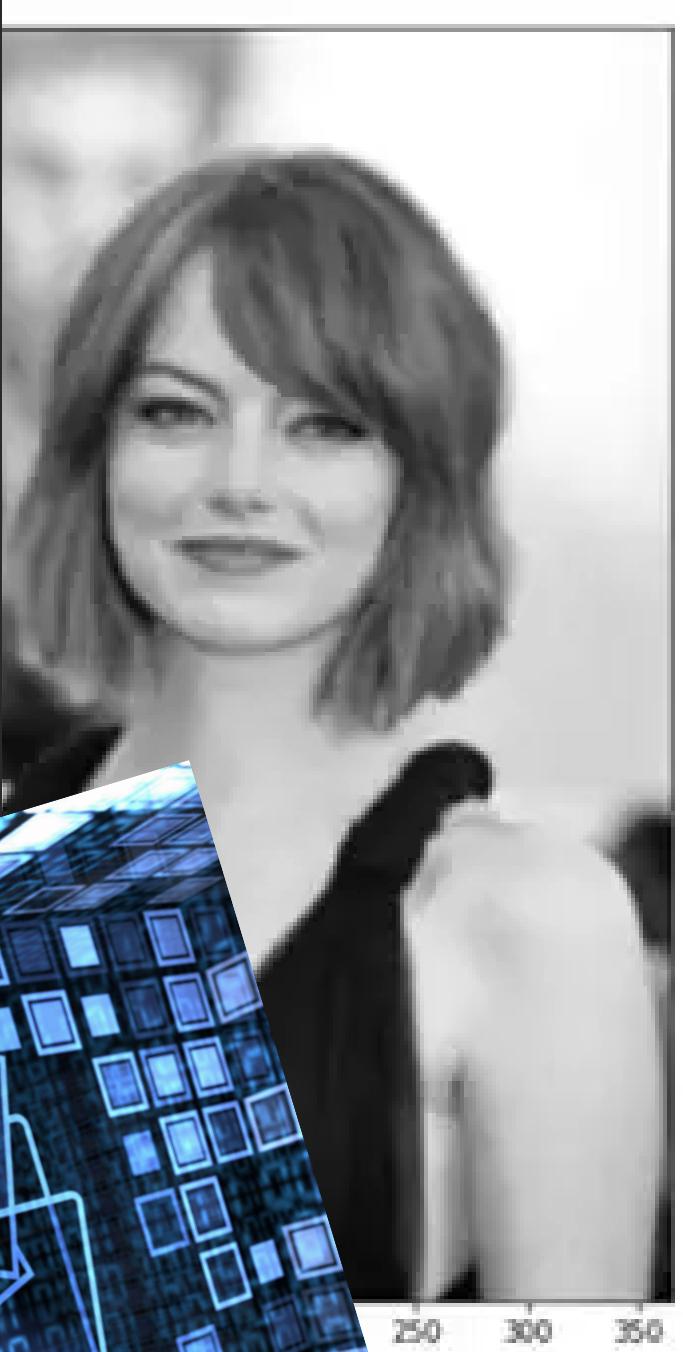
Canny cho ra

- đường biên mảnh
- liên tục
- ít nhiễu
- đạt độ chính xác cao
- phát hiện đúng nhiều biên thật
- hạn chế biên giả

Canny

Sobel

Laplacian



Canny cho ra

- đường biên mảnh,
- liên tục,
- ít nhiễu,
- đạt độ chính xác cao,
- phát hiện đúng nhiều biên thật
- hạn chế biên giả



Đáp ứng tốt các tiêu chí của một bộ phát hiện cạnh lý tưởng.

Canny

Sobel

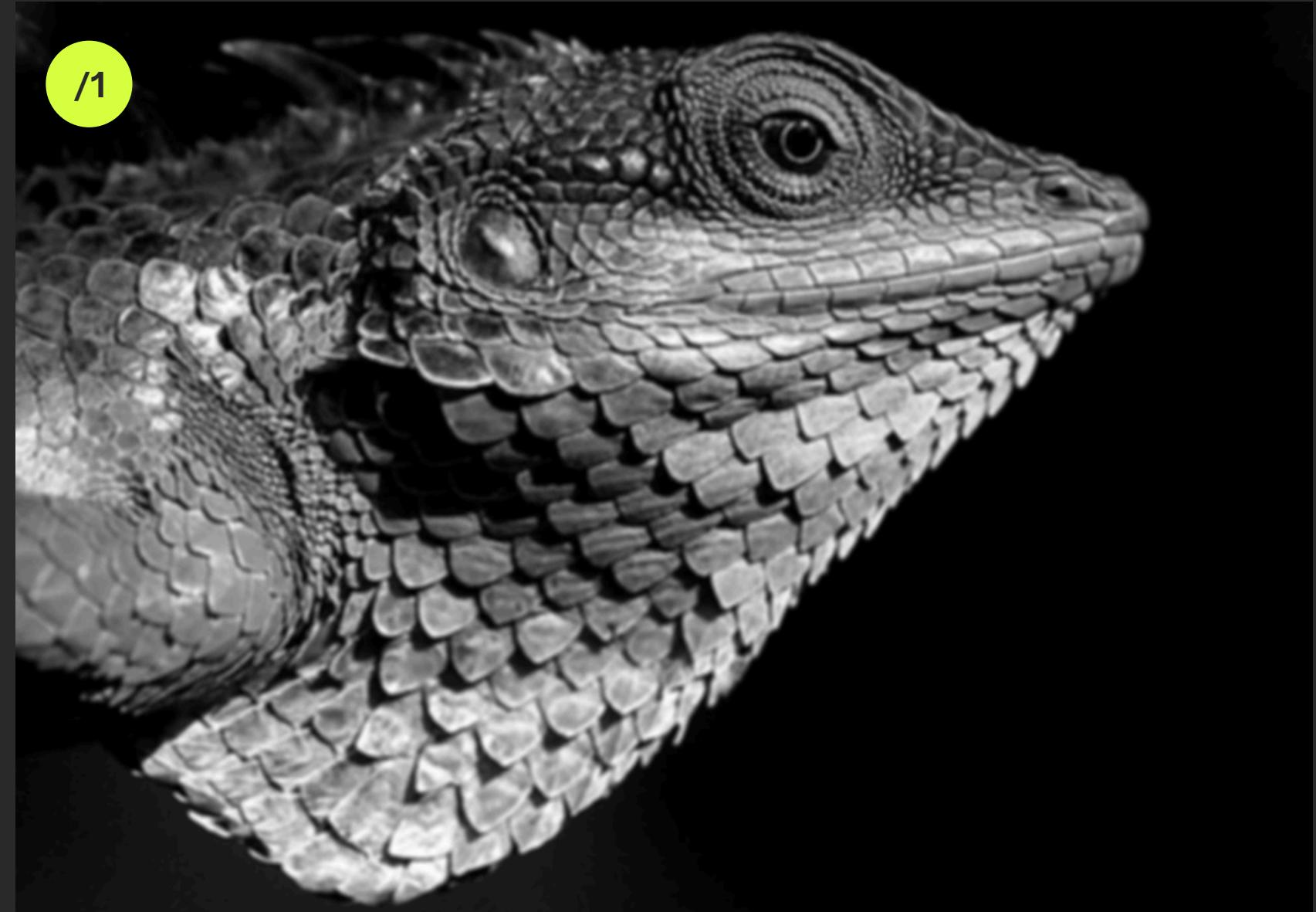
Laplacian

MÔ HÌNH VÀ CÔNG THỨC

- /1 Lọc Gauss: Làm trơn ảnh bằng bộ lọc Gaussian

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

nhằm giảm nhiễu tần số cao, chuẩn bị cho việc tính biên.



Canny

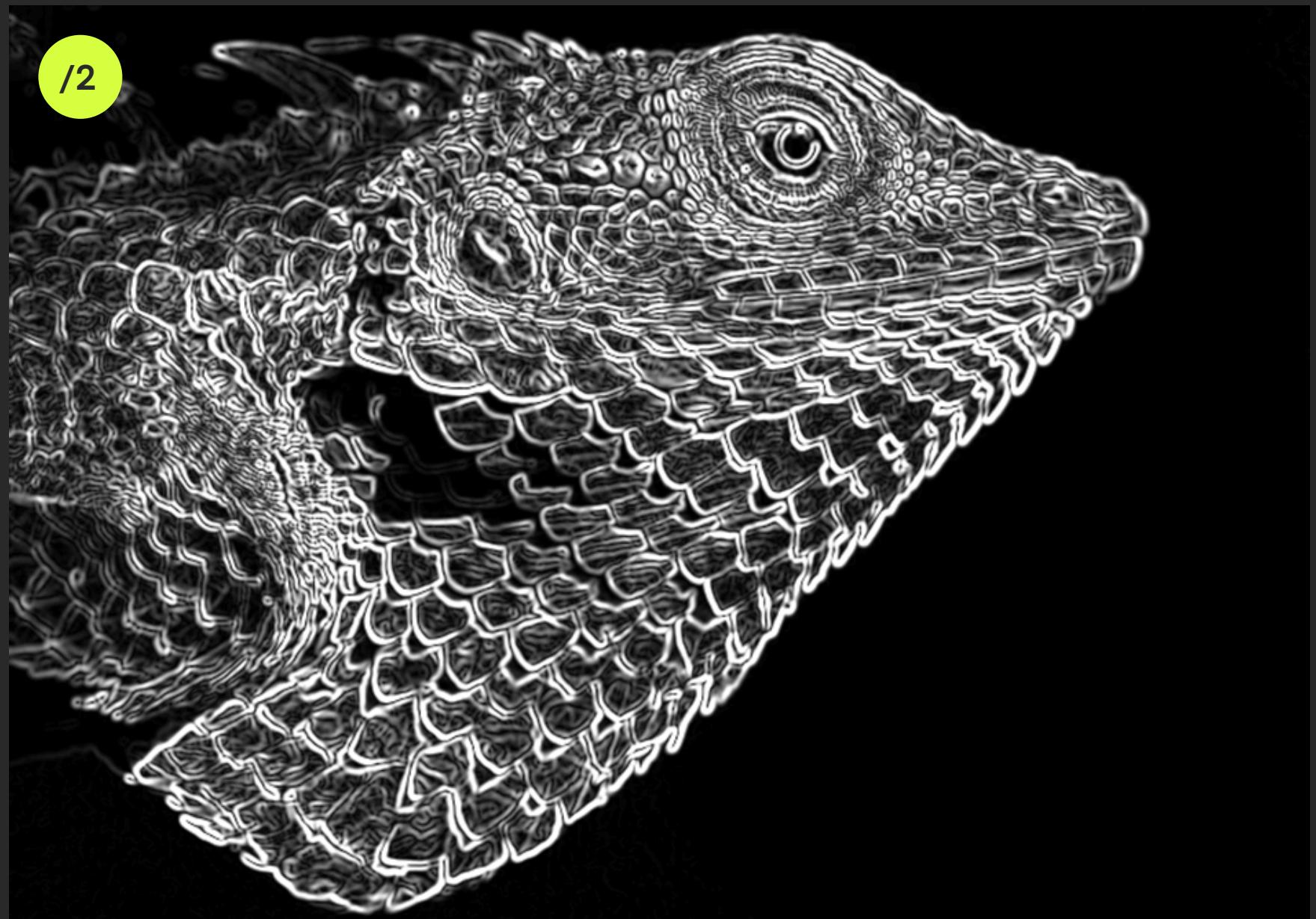
Sobel

Laplacian

MÔ HÌNH VÀ CÔNG THỨC

- /1 Lọc Gauss
- /2 Tính Gradient

Sử dụng toán tử Sobel để xác định đạo hàm theo hai hướng **G_x,G_y**, từ đó tính độ lớn và hướng gradient. Những điểm có gradient lớn là ứng viên biên.



MÔ HÌNH VÀ CÔNG THỨC

- /1 Lọc Gauss: Làm trơn ảnh bằng bộ lọc Gaussian

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

nhằm giảm nhiễu tần số cao, chuẩn bị cho việc tính biên.

- /2 Tính Gradient: Sử dụng toán tử Sobel để xác định đạo hàm theo hai hướng Gx, Gy. Từ đó tính **độ lớn và hướng gradient**. Những điểm có gradient lớn là ứng viên biên.
- /3 Triệt tiêu không cực đại (Non-Maximum Suppression): Làm mảnh biên bằng cách chỉ giữ lại các điểm cực đại dọc theo hướng gradient, loại bỏ điểm không phải đỉnh biên.

Canny

Sobel

Laplacian

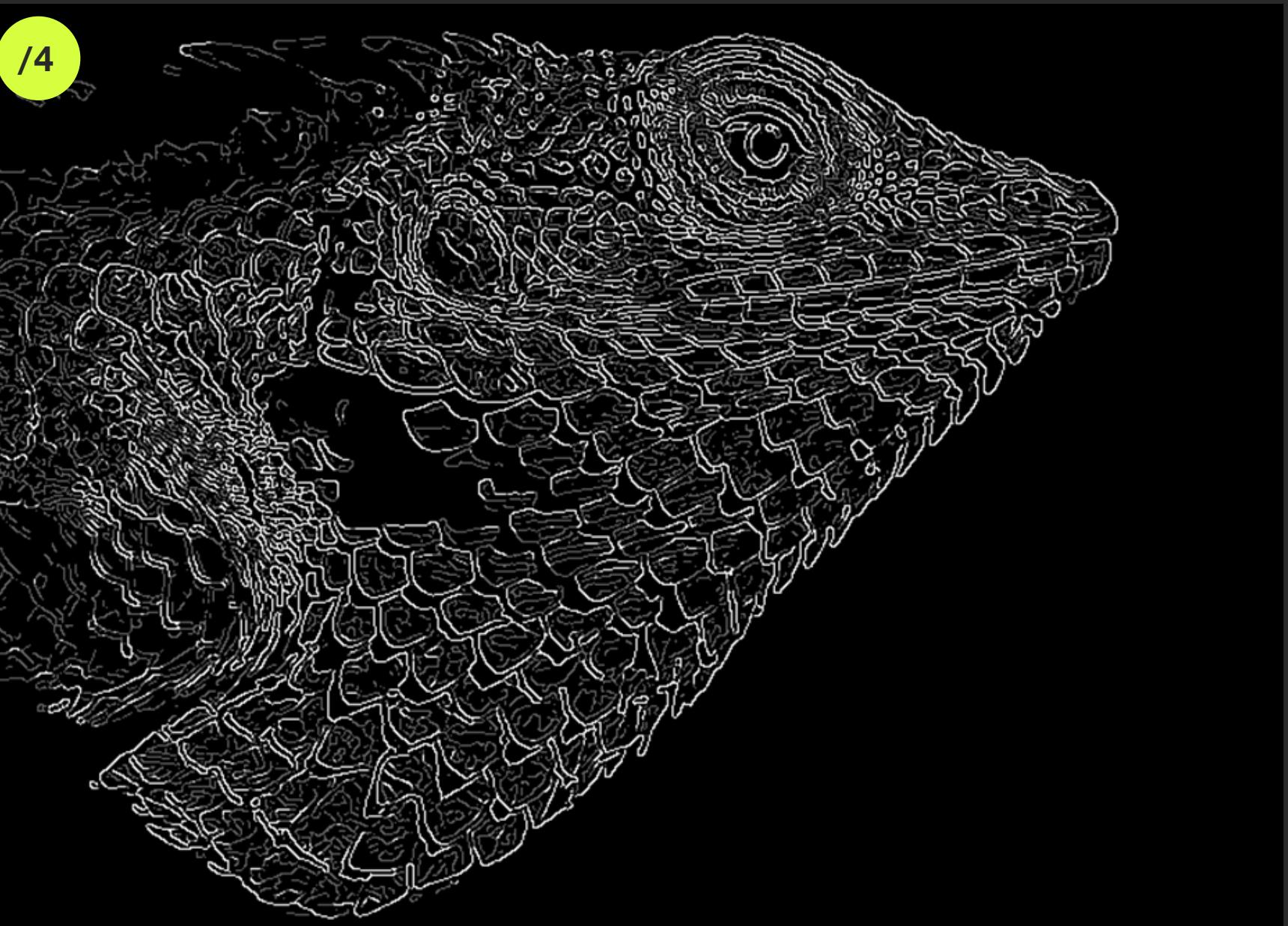
MÔ HÌNH VÀ CÔNG THỨC

/4

Nguõng kép (Double Threshold)

Phân loại điểm biên thành biên mạnh, biên yếu, hoặc không phải biên dựa vào hai ngưỡng

T_{high} và T_{low}



Canny

Sobel

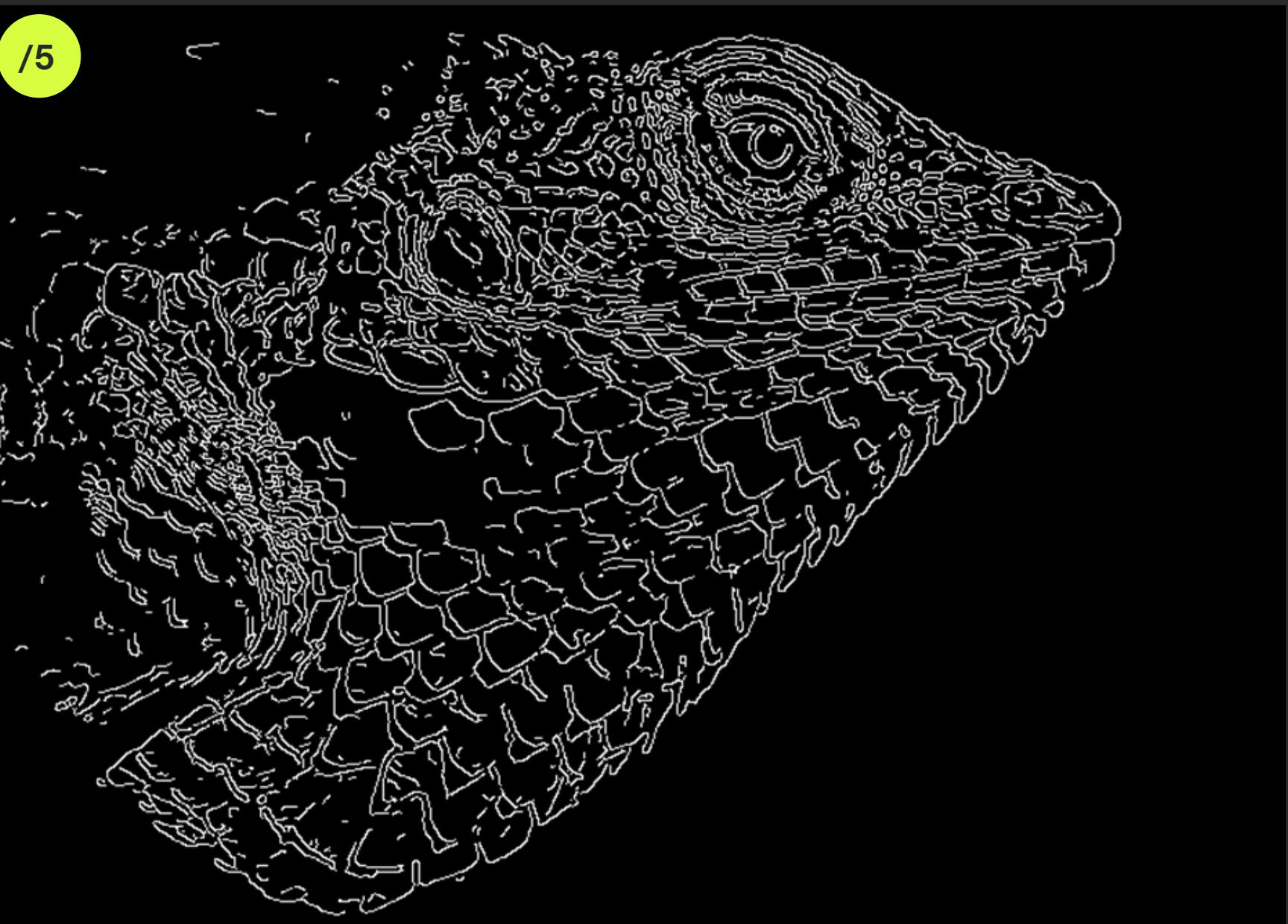
Laplacian

MÔ HÌNH VÀ CÔNG THỨC

/4 Ngưỡng kép (Double Threshold)

/5 Liên kết biên (Hysteresis)

Nối các đoạn biên rời rạc; điểm biên yếu nối với biên mạnh sẽ được giữ lại, ngược lại bị loại bỏ.



Canny

Sobel

Laplacian

MÔ HÌNH VÀ CÔNG THỨC

- /4 Ngưỡng kép (Double Threshold): Phân loại điểm biên thành biên mạnh, biên yếu, hoặc không phải biên dựa vào hai ngưỡng

T_{high} và T_{low}

- /5 Liên kết biên (Hysteresis): Nối các đoạn biên rời rạc; điểm biên yếu nối với biên mạnh sẽ được giữ lại, ngược lại bị loại bỏ.



-----> KẾT QUẢ:

Đảm bảo đầu ra cuối cùng là một ảnh nhị phân chứa các đường biên mạnh, rõ nét, gần đúng với ranh giới thực của các đối tượng trong ảnh và ít bị ảnh hưởng bởi nhiễu ngẫu nhiên

Nội dung

THUẬT TOÁN CANNY

THUẬT TOÁN SOBEL

THUẬT TOÁN LAPLACIAN

ỨNG DỤNG ĐÊM VẬT THỂ

NEXT >>>

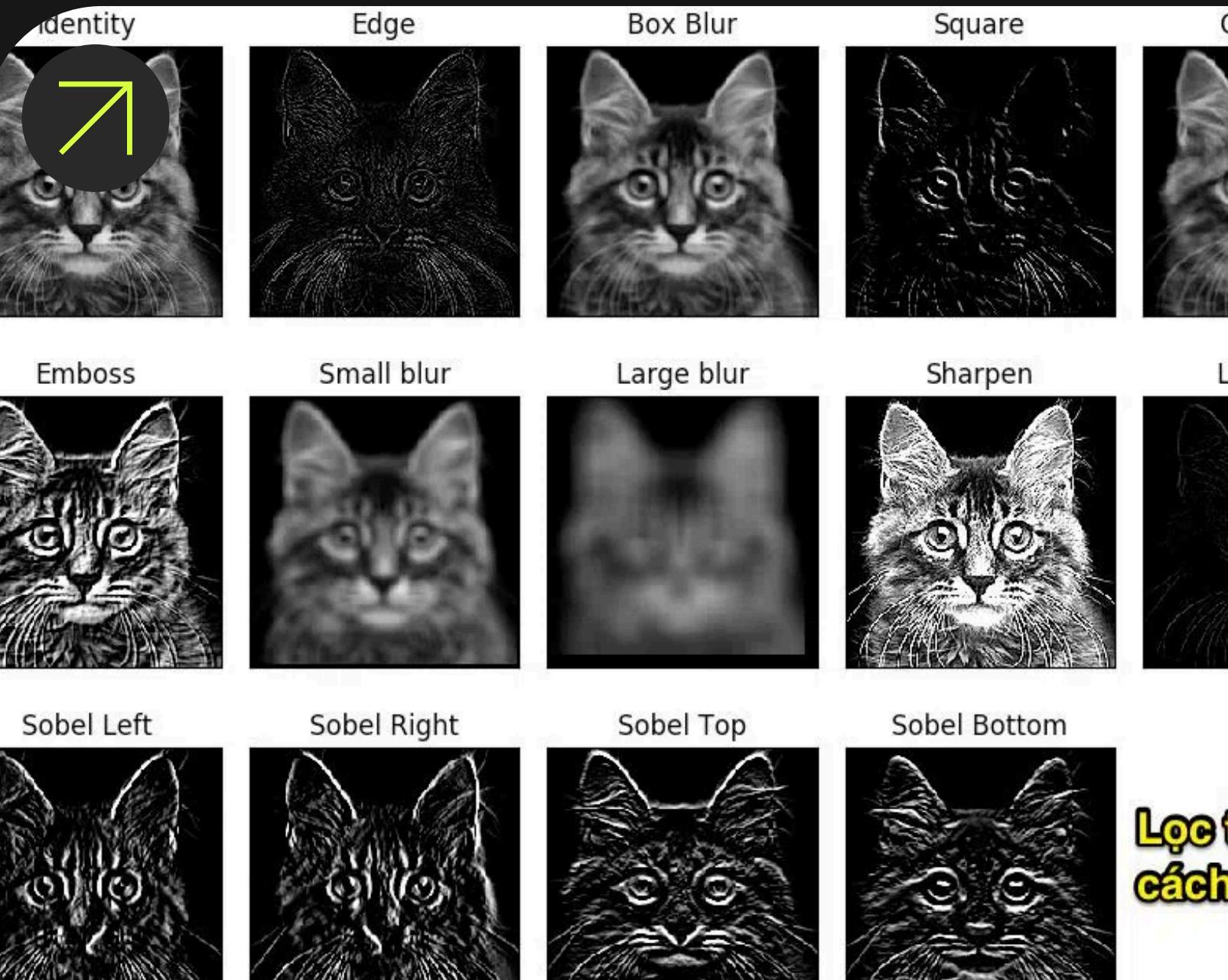
Canny

Sobel

Laplacian

Thuật toán Sobel là phương pháp phát hiện biên dựa trên đạo hàm bậc nhất (**gradient**), dùng phổ biến trong xử lý ảnh và thị giác máy tính.

Dùng phổ biến trong xử lý ảnh và thị giác máy tính.



Lọc
cách

Canny

Sobel

Laplacian

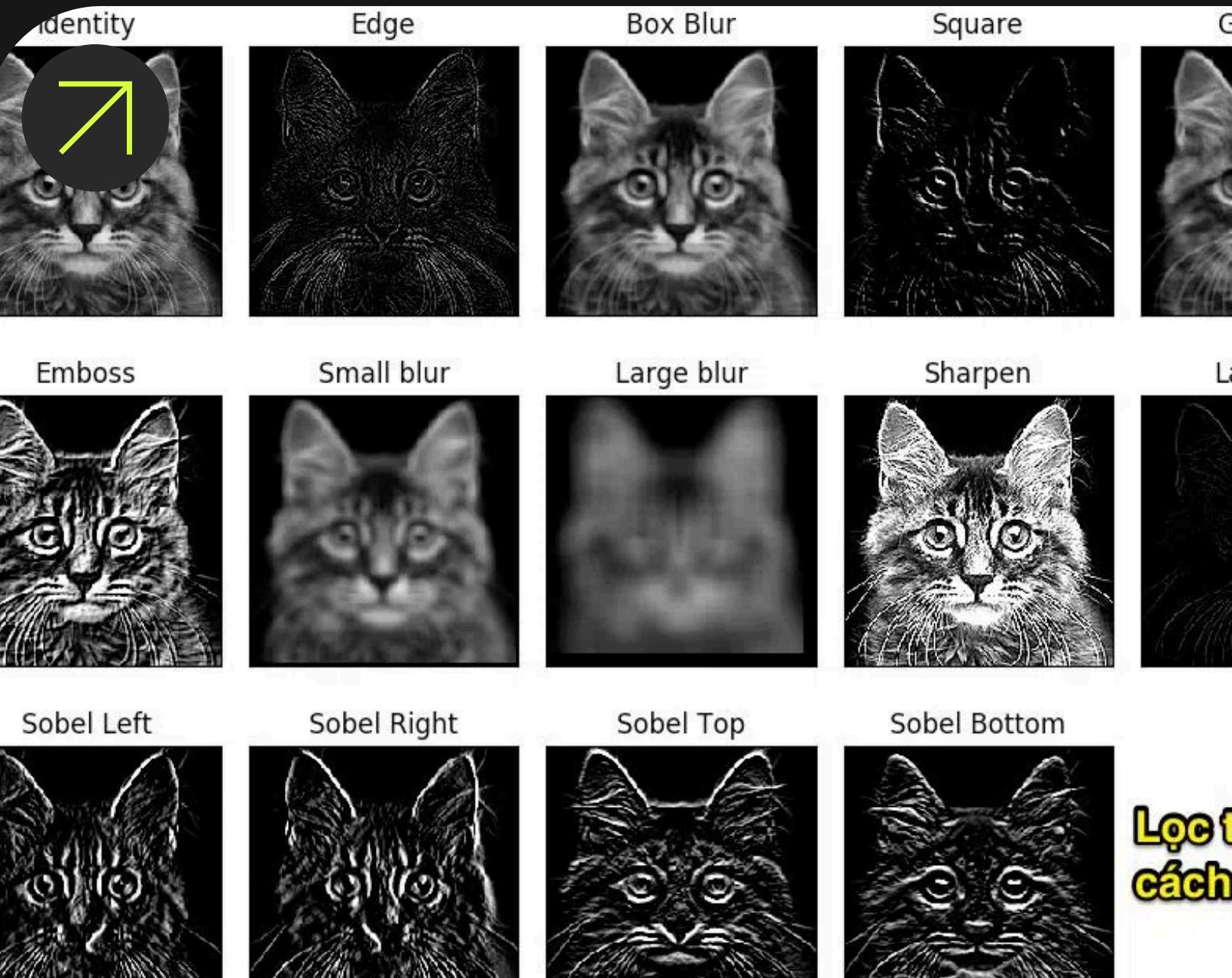
Thuật toán Sobel là phương pháp phát hiện biên dựa trên đạo hàm bậc nhất (**gradient**), dùng phổ biến trong xử lý ảnh và thị giác máy tính.

Dùng phổ biến trong xử lý ảnh và thị giác máy tính.

- Nó tính xấp xỉ **gradient** cường độ sáng bằng cách tích chập ảnh với hai kernel 3×3 theo hướng ngang (G_x) và dọc (G_y).

Từ đó, độ lớn gradient tại mỗi điểm được xác định:

$$M(x, y) = \sqrt{G_x^2(x, y) + G_y^2(x, y)}$$



Canny

Sobel

Laplacian

Thuật toán Sobel là phương pháp phát hiện biên dựa trên đạo hàm bậc nhất (**gradient**), dùng phổ biến trong xử lý ảnh và thị giác máy tính.

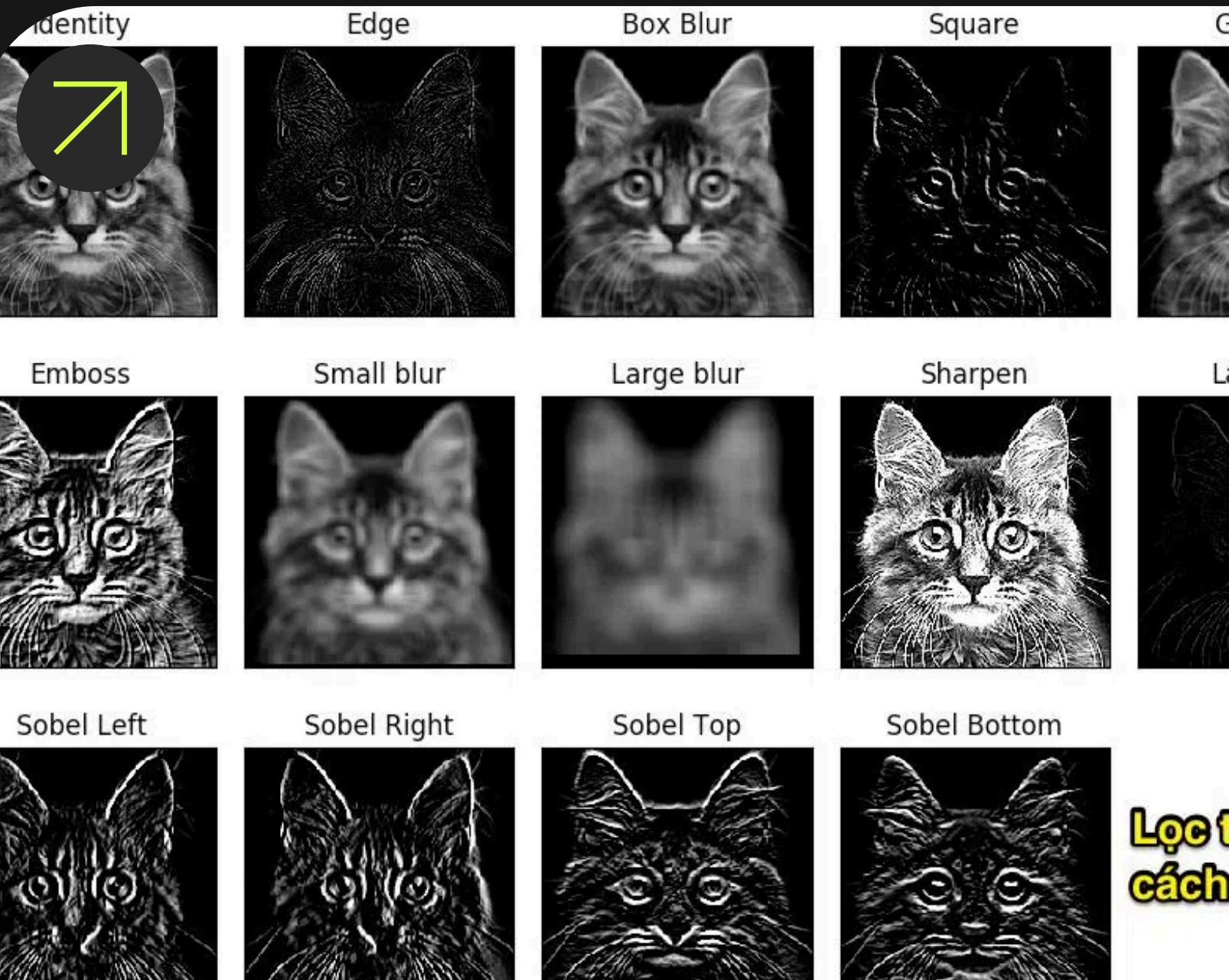
Dùng phổ biến trong xử lý ảnh và thị giác máy tính.

- Nó tính xấp xỉ **gradient** cường độ sáng bằng cách tích chập ảnh với hai kernel 3×3 theo hướng ngang (G_x) và dọc (G_y).

Từ đó, độ lớn gradient tại mỗi điểm được xác định:

$$M(x, y) = \sqrt{G_x^2(x, y) + G_y^2(x, y)}$$

Các điểm có giá trị $M(x, y)$ lớn biểu thị vùng biên mạnh, tức là nơi cường độ sáng thay đổi đột ngột —> **chính là các cạnh (edges) của ảnh**.



Canny

Sobel

Laplacian

• CÔNG THỨC CHÍNH •

Hai kernel Sobel 3×3:

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Từ đó:

$$G_x = A * K_x, \quad G_y = A * K_y$$

$$M(x, y) = \sqrt{G_x^2(x, y) + G_y^2(x, y)}$$

$$\theta(x, y) = \tan^{-1} \left(\frac{G_y}{G_x} \right)$$

Canny

Sobel

Laplacian

• CÔNG THỨC CHÍNH •

Hai kernel Sobel 3×3:

$$K_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad K_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Từ đó:

$$G_x = A * K_x, \quad G_y = A * K_y$$

$$M(x, y) = \sqrt{G_x^2(x, y) + G_y^2(x, y)}$$

$$\theta(x, y) = \tan^{-1} \left(\frac{G_y}{G_x} \right)$$

• CÁC BƯỚC THỰC HIỆN •

- /1 Chuyển ảnh sang mức xám (nếu là ảnh màu)
- /2 Áp dụng tích chập với kernel Sobel Kx và Ky để thu được ảnh gradient Gx, Gy.
- /3 Tính độ lớn gradient M(x,y) để xác định biên mạnh
- /4 Áp dụng ngưỡng (threshold) lên M(x,y) → tạo ảnh nhị phân hiển thị vị trí biên.

Canny

Sobel

Laplacian



**Gradient theo hướng X
(G_x) – các chi tiết thay đổi theo phương ngang được nhấn mạnh (sáng hơn).**



**Gradient theo hướng Y
(G_y) – các chi tiết theo phương dọc nổi bật (sáng).**

Nội dung

THUẬT TOÁN CANNY

THUẬT TOÁN SOBEL

THUẬT TOÁN LAPLACIAN

ỨNG DỤNG ĐÊM VẬT THỂ

NEXT >>>

Canny

Sobel

Laplacian

Thuật toán Laplacian sử dụng đạo hàm bậc hai của ảnh để nhận biết những vùng cường độ thay đổi đột ngột (biên ảnh).

- Toán tử Laplacian:

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Canny

Sobel

Laplacian

Thuật toán Laplacian sử dụng đạo hàm bậc hai của ảnh để nhận biết những vùng cường độ thay đổi đột ngột (biên ảnh).

- Toán tử Laplacian:

$$\nabla^2 f(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

Đặc điểm

- Chỉ cần một kernel duy nhất để phát hiện biên theo mọi hướng.
- Mặt nạ 3×3 phổ biến:

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

- Giá trị âm ở trung tâm giúp phát hiện sự khác biệt cường độ so với láng giềng.

Canny

Sobel

Laplacian

CÔNG THỨC

Phép Laplacian rời rạc được thực hiện bằng tích chập với kernel 3×3 , ví dụ:

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

hoặc biến thể:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sau lọc, thường lấy giá trị tuyệt đối rồi áp dụng ngưỡng để xác định biên.

Canny

Sobel

Laplacian

CÔNG THỨC

Phép Laplacian rời rạc được thực hiện bằng tích chập với kernel 3×3 , ví dụ:

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

hoặc biến thể:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Sau lọc, thường lấy giá trị tuyệt đối rồi áp dụng ngưỡng để xác định biên.

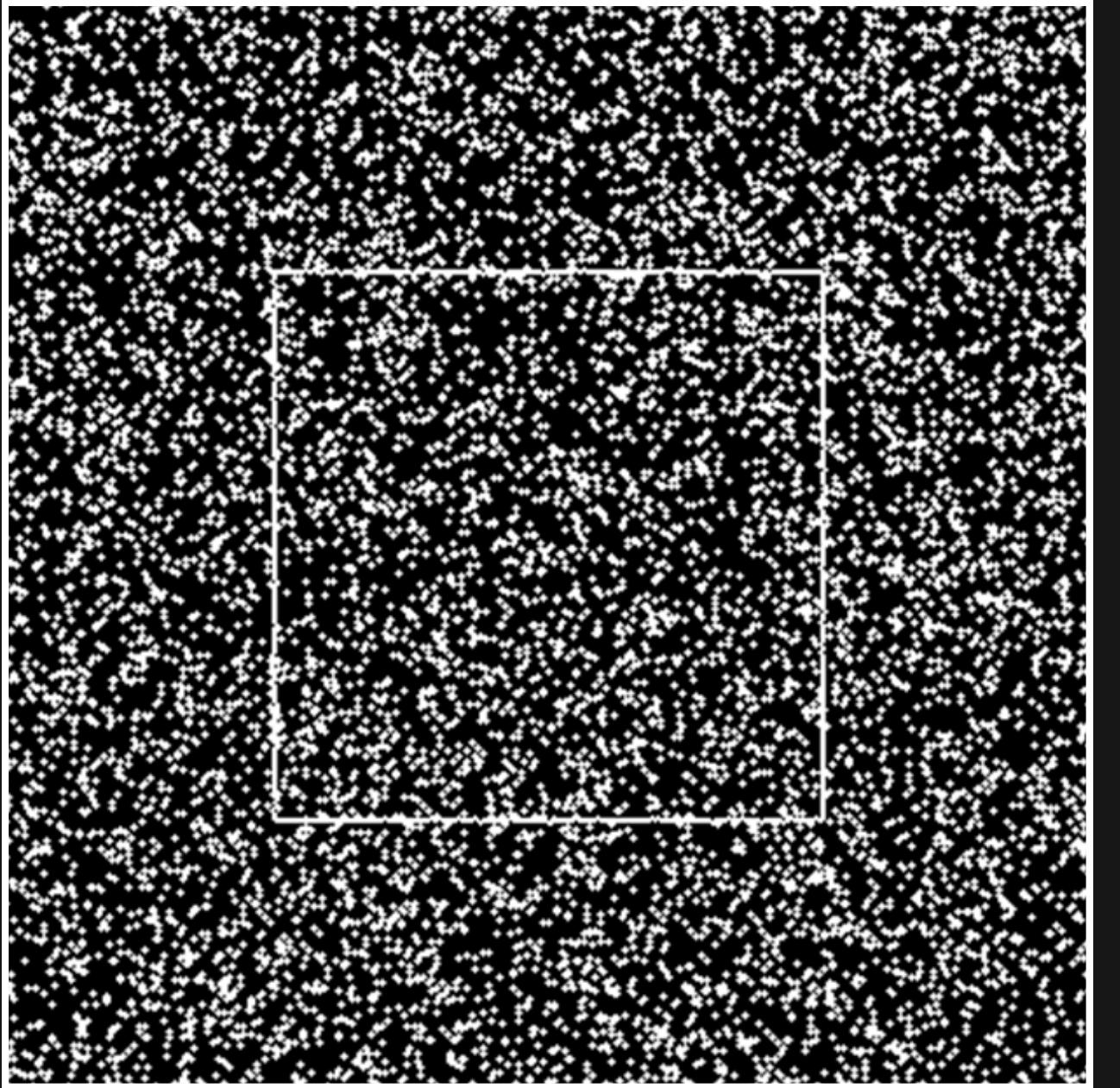
CÁC BƯỚC THỰC HIỆN

- /1 Chuyển ảnh sang xám.
- /2 Làm mượt bằng Gaussian để giảm nhiễu.(tuỳ chọn)
- /3 Tích chập ảnh với kernel Laplacian (hoặc dùng cv2.Laplacian).
- /4 Lấy trị tuyệt đối và ngưỡng hóa để tạo ảnh biên.

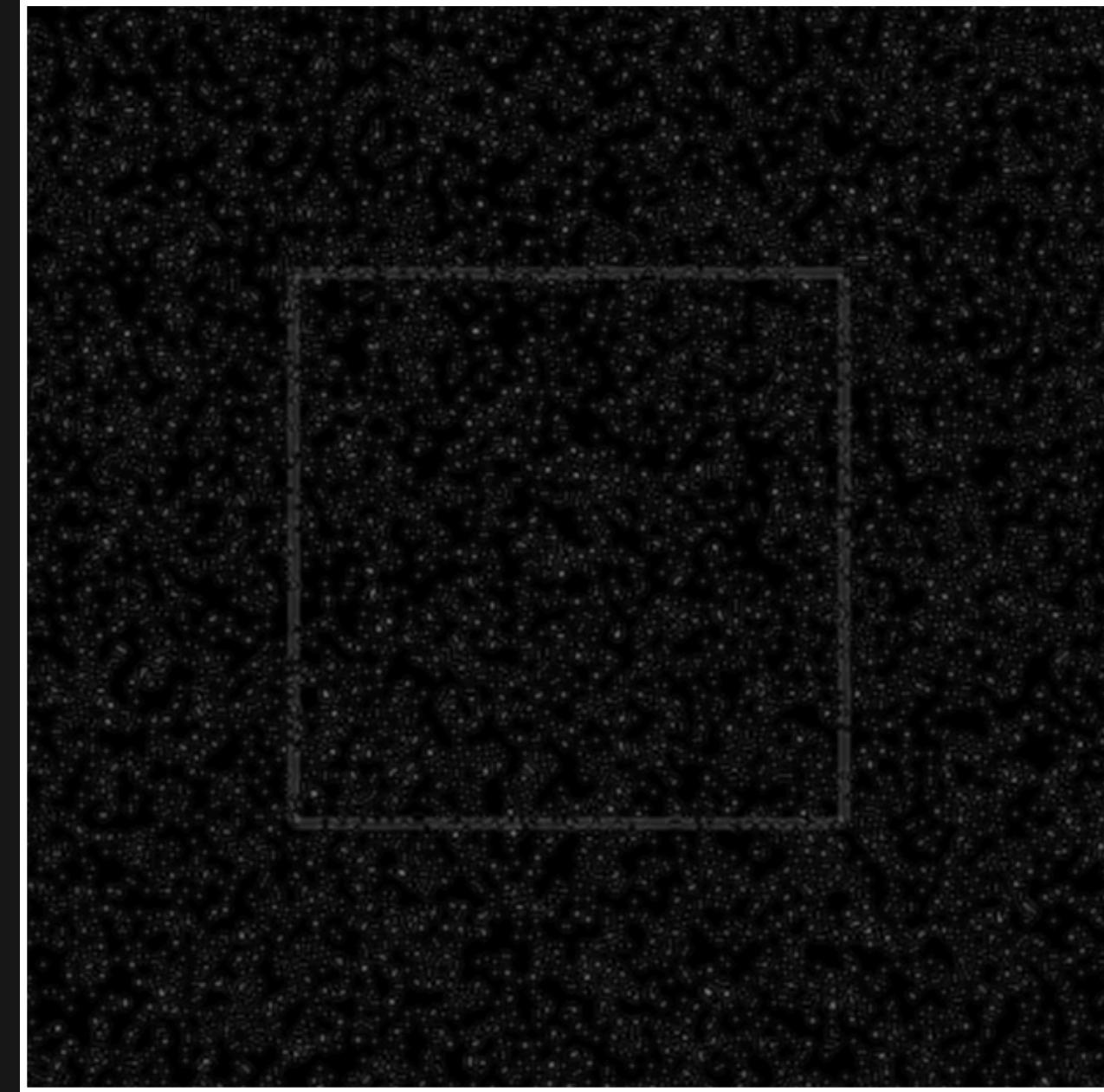
Canny

Sobel

Laplacian



Kết quả áp dụng Laplacian không lọc Gauss trước: nhiễu trong ảnh được khuếch đại thành vô số điểm giả biên (trắng) khắp nơi.



Kết quả áp dụng Laplacian sau khi làm mờ Gauss: hầu như chỉ còn lại đường biên của vật thể, nhiễu đã giảm mạnh

Nội dung

THUẬT TOÁN CANNY

THUẬT TOÁN SOBEL

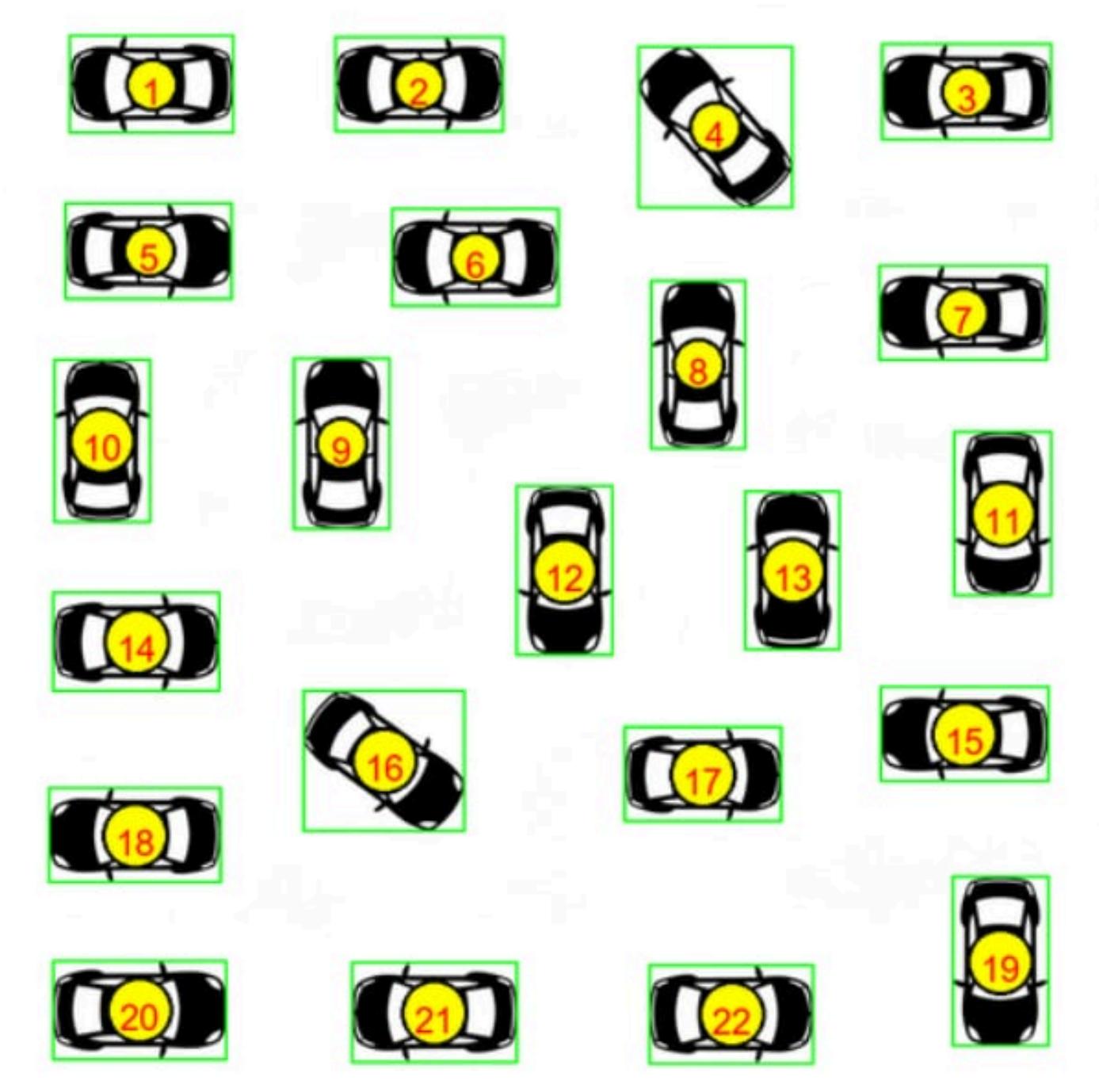
THUẬT TOÁN LAPLACIAN

ỨNG DỤNG ĐÊM VẬT THỂ

NEXT >>>

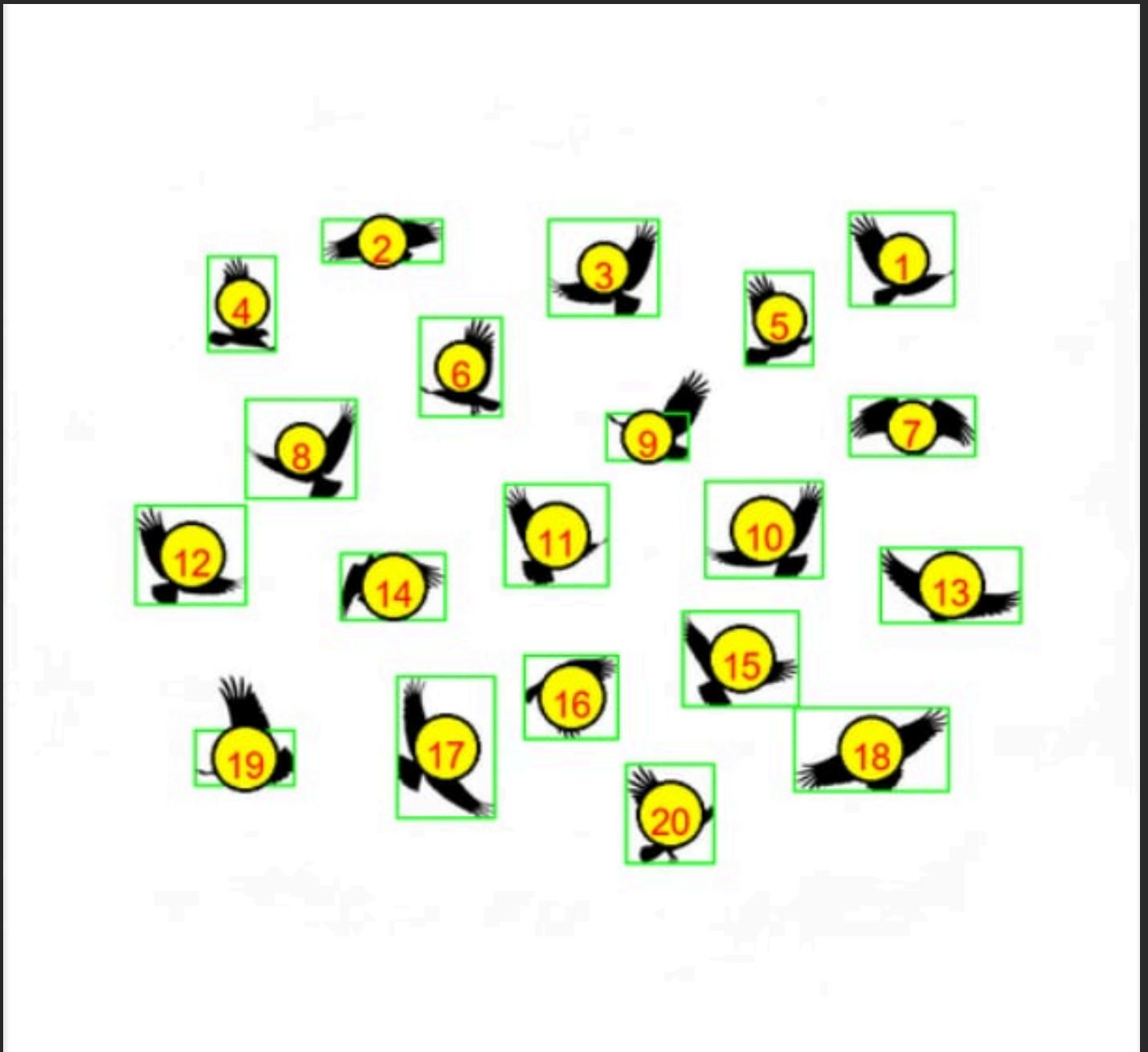
Ứng dụng đếm vật thể

- Trong ứng dụng đếm vật thể sử dụng thuật toán Canny. Vì thuật toán Canny cho biên mỏng, liên tục và ít nhiễu thuận tiện cho việc đếm vật thể



MÔ HÌNH VÀ CÔNG THỨC

- Sử dụng phép đóng ảnh để làm trơn các đường gãy khúc, lấp đầy các khoảng trống nhỏ trên ảnh để thuận tiện cho việc đếm vật thể
- Phép đóng bao gồm phép giãn hình(dilation) theo sau bởi phép co hình(erosion)



PHÉP GIÃN (DILATION)

- Phép giãn hình được thực hiện bằng cách lướt phần tử cấu trúc B qua ảnh gốc A.
 - 1.Đặt tâm của B lên từng pixel của A.
 - 2.Nếu ít nhất một pixel 1 trong B nằm trùng với một pixel 1 của A, thì pixel trung tâm được đặt là pixel 1 trong ảnh kết quả.

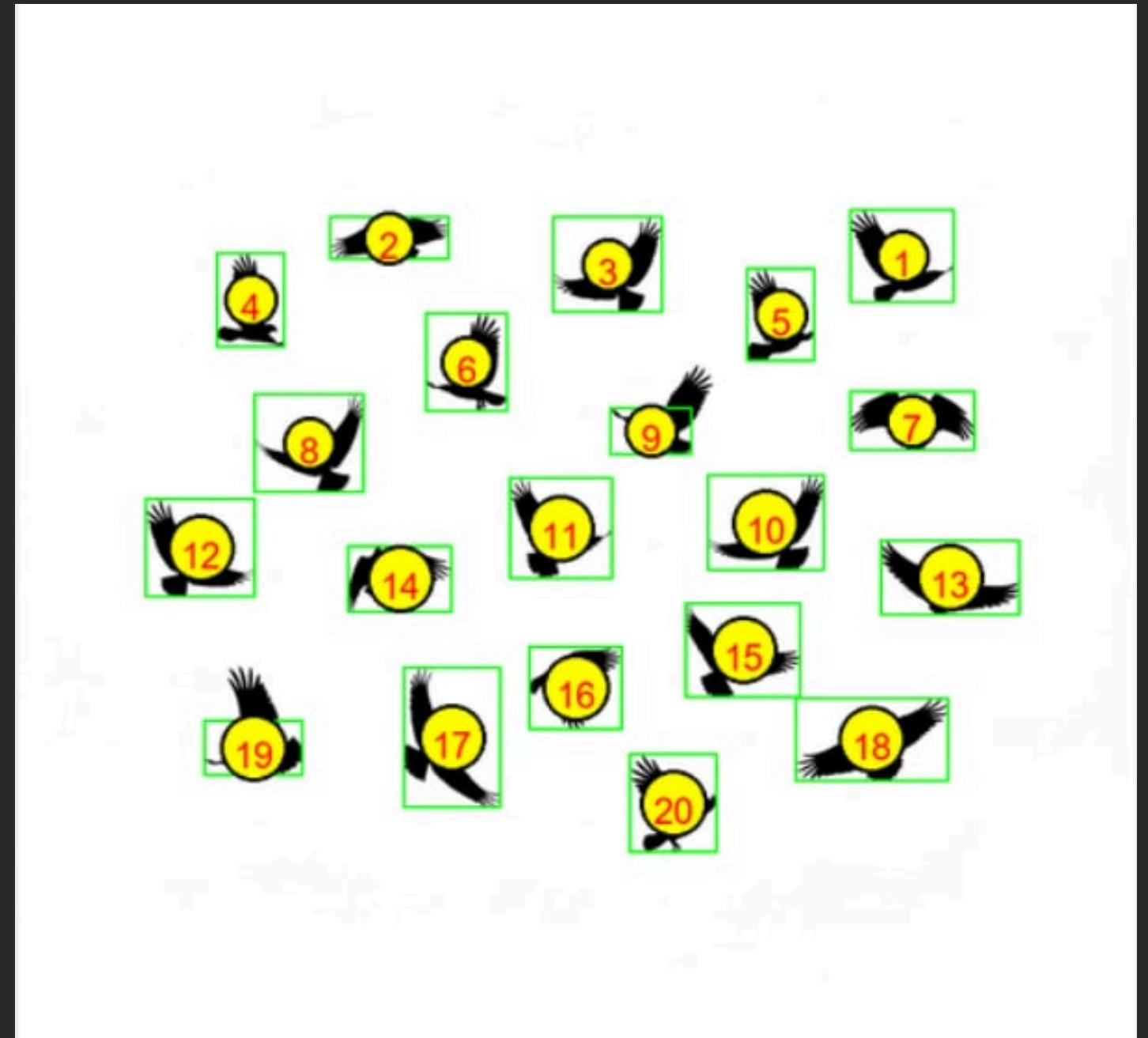
PHÉP CO(EROSION)

- Phép co hình được thực hiện bằng cách lướt phần tử cấu trúc B qua ảnh gốc A theo các bước sau:
 - 1.Đặt tâm của B lên từng pixel của A.
 - 2.Nếu TẤT CẢ các pixel trong B nằm trùng với các pixel 1 của A, thì pixel trung tâm được đặt là pixel 1 trong ảnh kết quả. Ngược lại, nó sẽ bị chuyển thành pixel 0.

CÁC BƯỚC THỰC HIỆN

/1 Chuyển thành binary

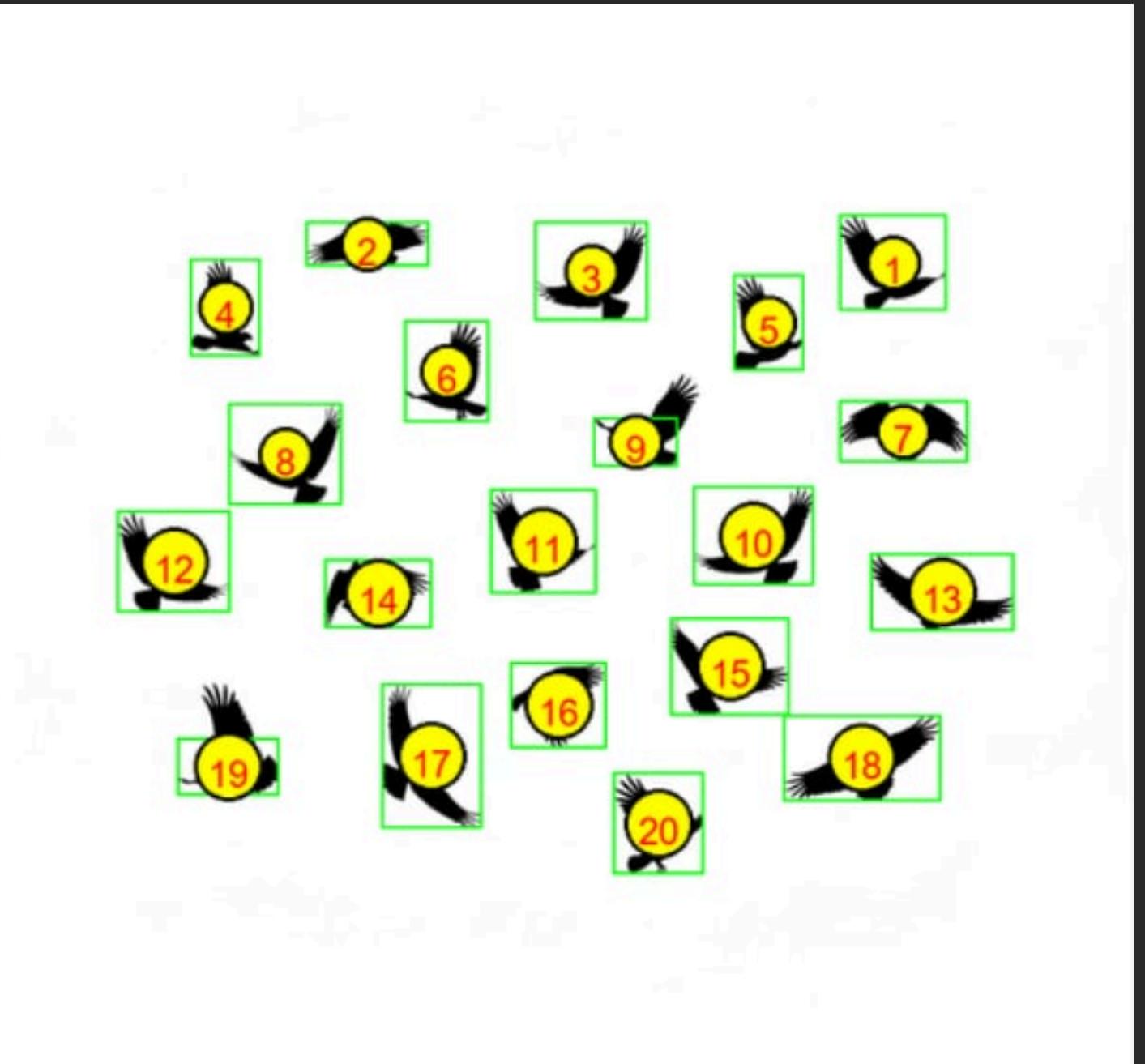
```
def count_objects_from_edges (orig_bgr: np.ndarray, edges: np.ndarray,  
                           min_area: int = 50, morph_iter: int = 4):  
    bin_img = _binary_from_edges (edges)
```



CÁC BƯỚC THỰC HIỆN

/2 Xử lý hình thái học sử dụng phép đóng

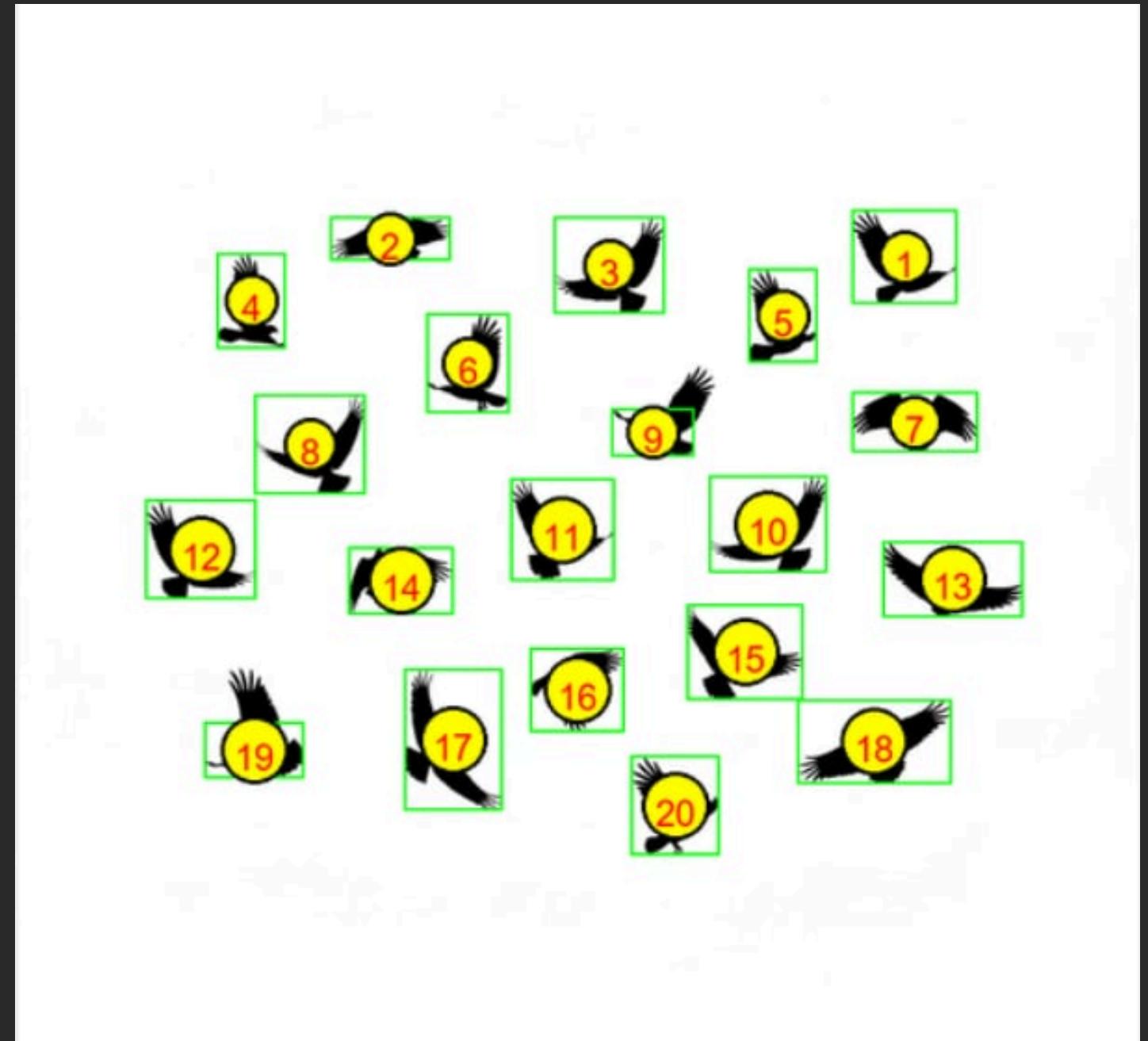
```
def count_objects_from_edges (orig_bgr: np.ndarray, edges: np.ndarray,  
                           min_area: int = 50, morph_iter: int = 4):  
    bin_img = _binary_from_edges (edges)  
    closed = morphology_close (bin_img, kernel_size = 7, iterations = morph_iter)
```



CÁC BƯỚC THỰC HIỆN

/3 Tìm vùng liên thông

```
def count_objects_from_edges (orig_bgr: np.ndarray, edges: np.ndarray,  
                           min_area: int = 50, morph_iter: int = 4):  
    bin_img = _binary_from_edges (edges)  
    closed = morphology_close (bin_img, kernel_size = 7, iterations = morph_iter)  
    contours = find_contours(closed)
```

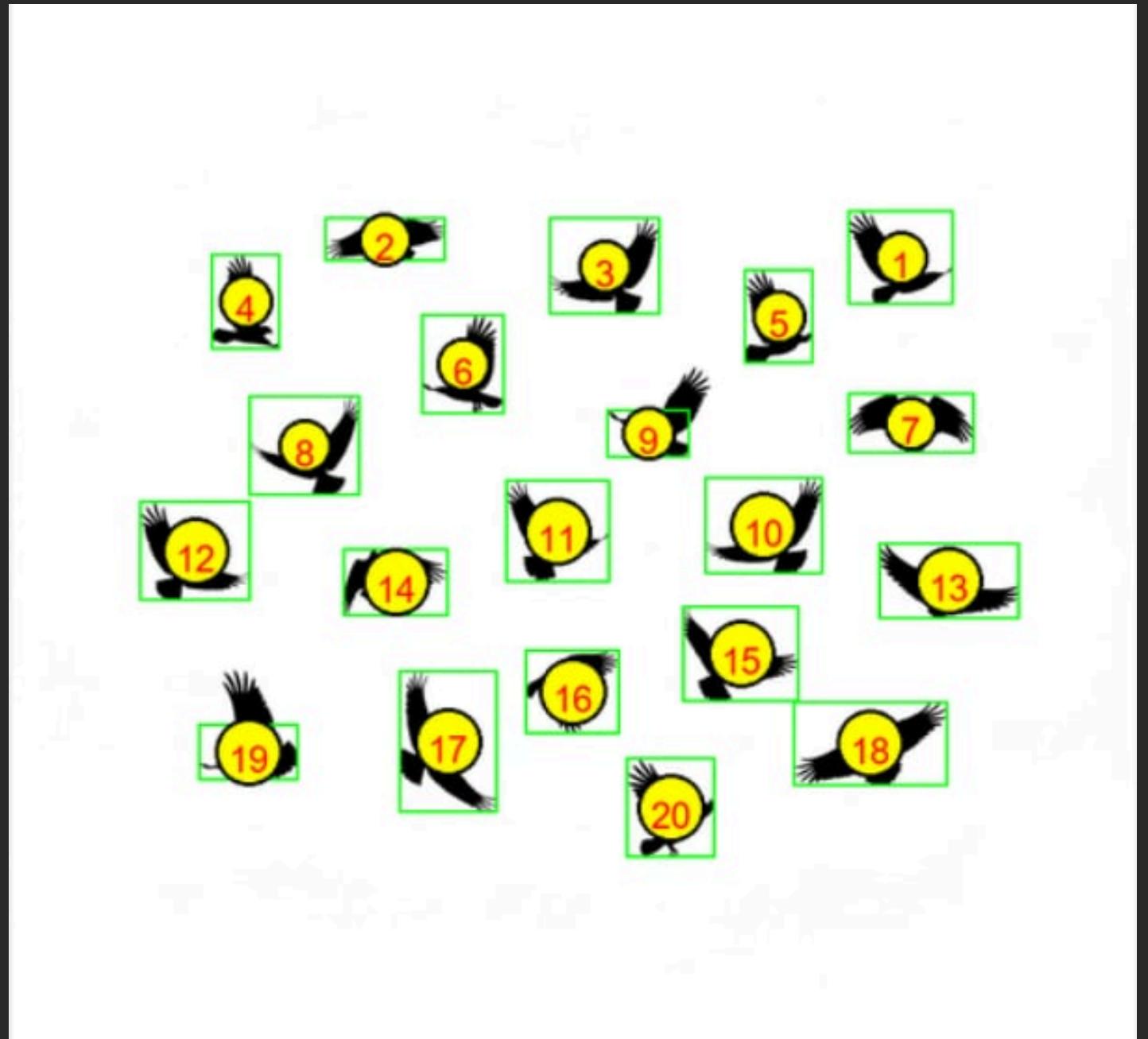


CÁC BƯỚC THỰC HIỆN

/4 Lọc và đếm đối tượng

```
def count_objects_from_edges (orig_bgr: np.ndarray, edges: np.ndarray,
                             min_area: int = 50, morph_iter: int = 4):
    bin_img = _binary_from_edges (edges)
    closed = morphology_close (bin_img, kernel_size = 7, iterations = morph_iter)
    contours = find_contours (closed)
    count = 0
    annotated = orig_bgr.copy()
    contours_data = []
    for idx, cnt in enumerate (contours):
        area = contour_area (cnt)
        if area < min_area:
            print (f " → SKIPPED (too small)")
            continue
        x, y, w, h = bounding_rect (cnt)
        print (f " → KEPT: bbox = ( {x}, {y}, {w}, {h} )")
        annotated = draw_rectangle (annotated, x, y, w, h, (0, 255, 0), 2)
        contour_data.append ((x, y, w, h))
        count += 1

    return count, annotated, contours_data
```



Xử lý ảnh



THANK YOU

...