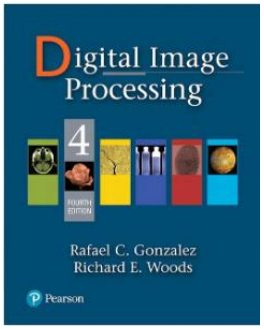


Fall, 2021

Digital Image Processing

Edge Detection

Dr. Tun-Wen Pai

Oct. 18, 2022

Main Steps in Edge Detection

(1) Smoothing: suppress as much noise as possible, without destroying true edges.

(2) Enhancement: apply differentiation to enhance the quality of edges (i.e., sharpening).

(3) Thresholding: determine which edge pixels should be discarded as noise and which should be retained (i.e., threshold edge magnitude).

(4) Localization: determine the exact edge location.

Gradient Representation

The gradient is a vector which has magnitude and direction:

$$magnitude(grad(f)) = \sqrt{\frac{\partial f}{\partial x}^2 + \frac{\partial f}{\partial y}^2}$$

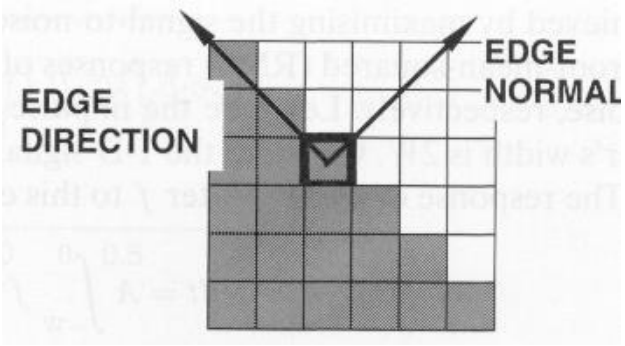
(approximation by absolute sum)

$$direction(grad(f)) = \tan^{-1}(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x})$$

Magnitude: indicates edge strength.

Direction: indicates edge direction.

- i.e., perpendicular to edge direction



https://www.khanacademy.org/math/multivariable-calculus/multivariable-derivatives/gradient-and-directional-derivatives/v/gradient-and-contour-maps

Approximate Gradient

Approximate gradient using finite differences:

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h} \qquad \frac{\partial f}{\partial y} = \lim_{h \to 0} \frac{f(x, y+h) - f(x, y)}{h}$$

$$\frac{\partial f}{\partial x} = \frac{f(x+h_x, y) - f(x, y)}{h_y} = f(x+1, y) - f(x, y), \quad (h_x = 1)$$

$$\frac{\partial f}{\partial y} = \frac{f(x, y+h_y) - f(x, y)}{h_y} = f(x, y+1) - f(x, y), \quad (h_y = 1)$$

Edge Detection Steps Using Gradient

(1) Smooth the input image ($\hat{f}(x,y) = f(x,y) * G(x,y)$)

(2) $\hat{f}_x = \hat{f}(x,y) * M_x(x,y)$ $\longrightarrow$ $\frac{\partial f}{\partial x}$

(3) $\hat{f}_y = \hat{f}(x,y) * M_y(x,y)$ $\longrightarrow$ $\frac{\partial f}{\partial y}$

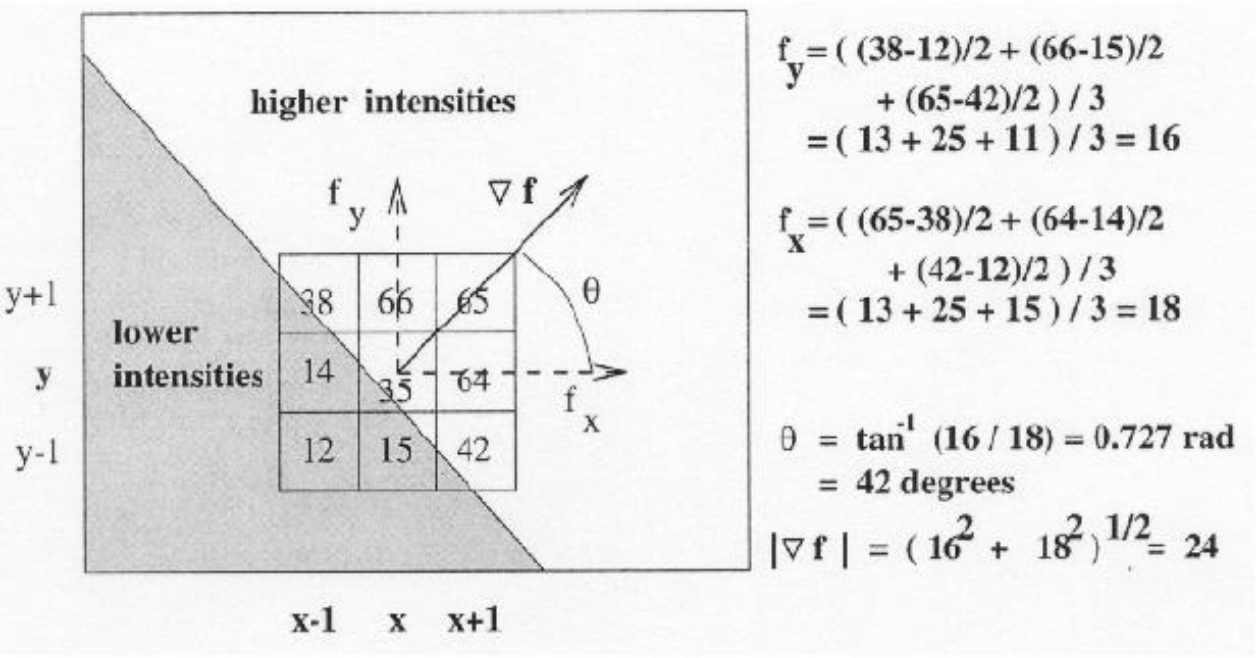
(4) $magn(x,y) = |\hat{f}_x| + |\hat{f}_y|$ (i.e., sqrt is costly!)

(5) $dir(x,y) = \tan^{-1}(\hat{f}_y / \hat{f}_x)$

(6) If $magn(x,y) > T$, then possible edge point

Example (using Prewitt operator)

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$



$f_y = ((38-12)/2 + (66-15)/2 + (65-42)/2) / 3 = (13 + 25 + 11) / 3 = 16$

$f_x = ((65-38)/2 + (64-14)/2 + (42-12)/2) / 3 = (13 + 25 + 15) / 3 = 18$

$\theta = \tan^{-1} (16 / 18) = 0.727 \text{ rad} = 42 \text{ degrees}$

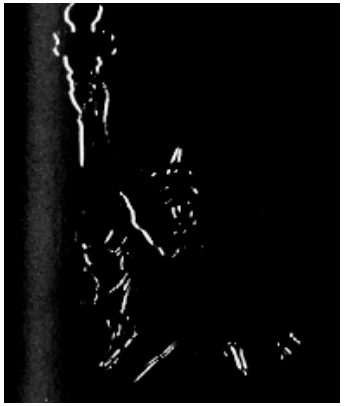
$|\nabla f| = (16^2 + 18^2)^{1/2} = 24$

Note: in this example, the divisions by 2 and 3 in the computation of f_x and f_y are done for normalization purposes only

Isotropic property of gradient magnitude

- The magnitude of the gradient detects edges in all directions.

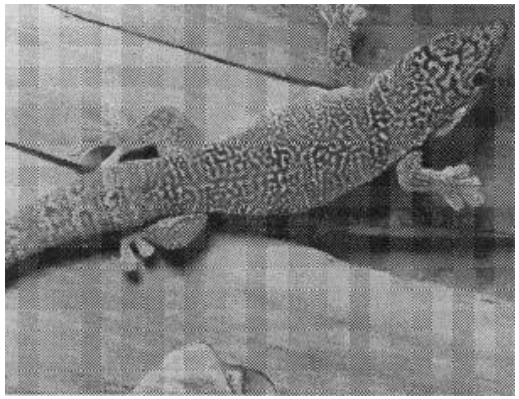
$\frac{d}{dx}I$ $\frac{d}{dy}I$ $\sqrt{(\frac{d}{dx}I)^2+(\frac{d}{dy}I)^2}$

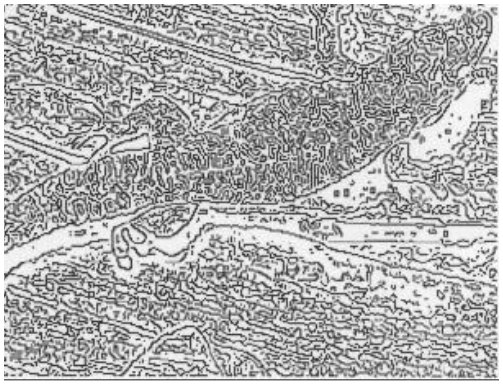
Practical Issues

Noise suppression-localization tradeoff.

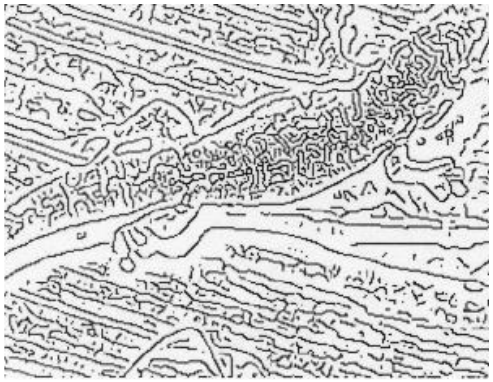
- Smoothing depends on mask size (e.g., depends on σ for Gaussian filters).
- Larger mask sizes reduce noise, but worsen localization (i.e., add uncertainty to the location of the edge) and vice versa.



smaller mask



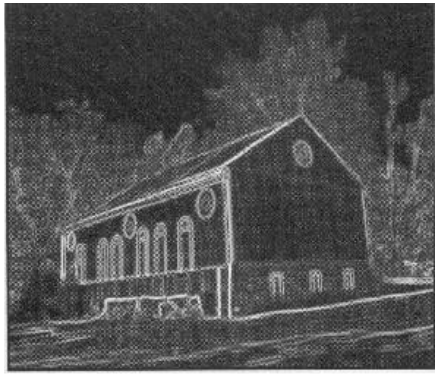
larger mask



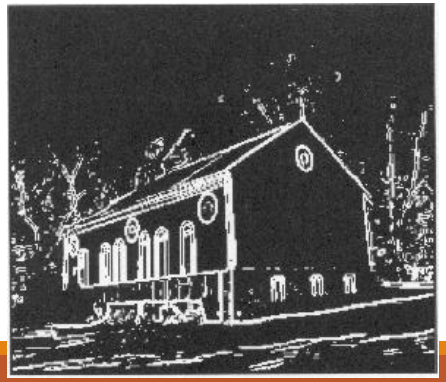
Practical Issues (cont'd)

Choice of threshold.

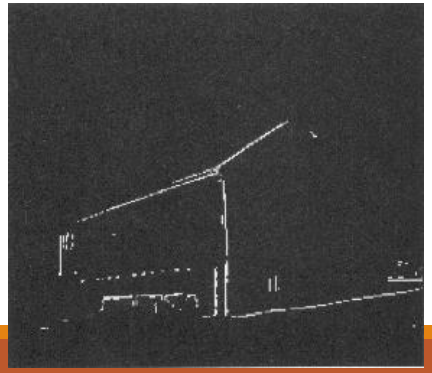
gradient magnitude



low threshold

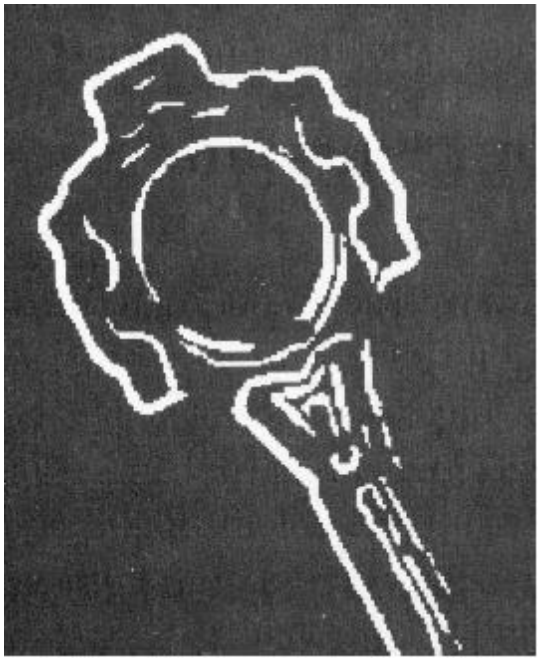


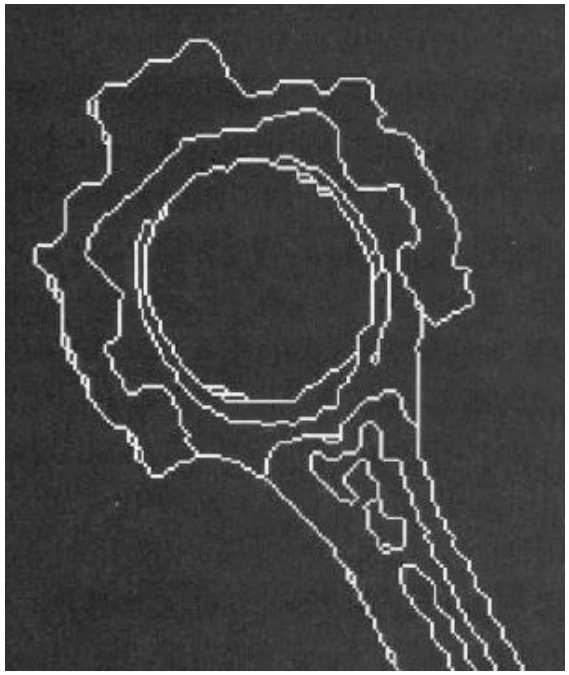
high threshold



Practical Issues (cont'd)

Edge thinning and linking.





Criteria for Optimal Edge Detection

(1) Good detection

- Minimize the probability of false positives (i.e., spurious edges).
- Minimize the probability of false negatives (i.e., missing real edges).

(2) Good localization

- Detected edges must be as close as possible to the true edges.

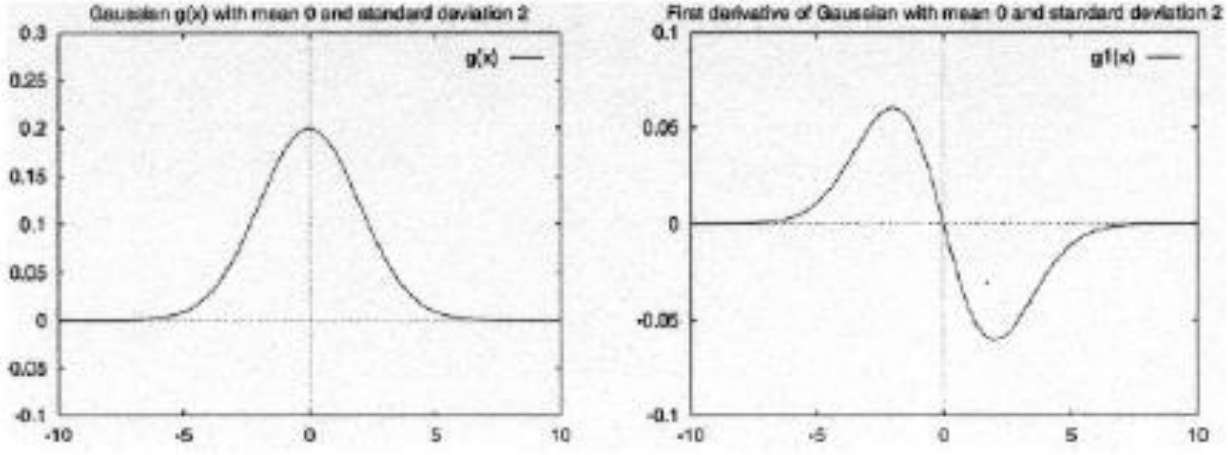
(3) Single response

- Minimize the number of local maxima around the true edge.

Canny edge detector

Canny has shown that the **first derivative of the Gaussian** closely approximates the operator that optimizes the product of *signal-to-noise* ratio and localization.

(i.e., analysis based on "step-edges" corrupted by "Gaussian noise“)



J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Steps of Canny edge detector

Algorithm

1. Compute f_x and f_y

$$f_x = \frac{\partial}{\partial x}(f * G) = f * \frac{\partial}{\partial x} G = f * G_x$$

$$f_y = \frac{\partial}{\partial y}(f * G) = f * \frac{\partial}{\partial y} G = f * G_y$$

$G(x, y)$ is the Gaussian function

$G_x(x, y)$ is the derivate of $G(x, y)$ with respect to x: $G_x(x, y) = \frac{-x}{\sigma^2} G(x, y)$

$G_y(x, y)$ is the derivate of $G(x, y)$ with respect to y: $G_y(x, y) = \frac{-y}{\sigma^2} G(x, y)$

Steps of Canny edge detector (cont'd)

2. Compute the gradient magnitude and direction

$$magn(x,y) = |\hat{f}_x| + |\hat{f}_y| \quad dir(x,y) = \tan^{-1}(\hat{f}_y / \hat{f}_x)$$

3. Apply non-maxima suppression.

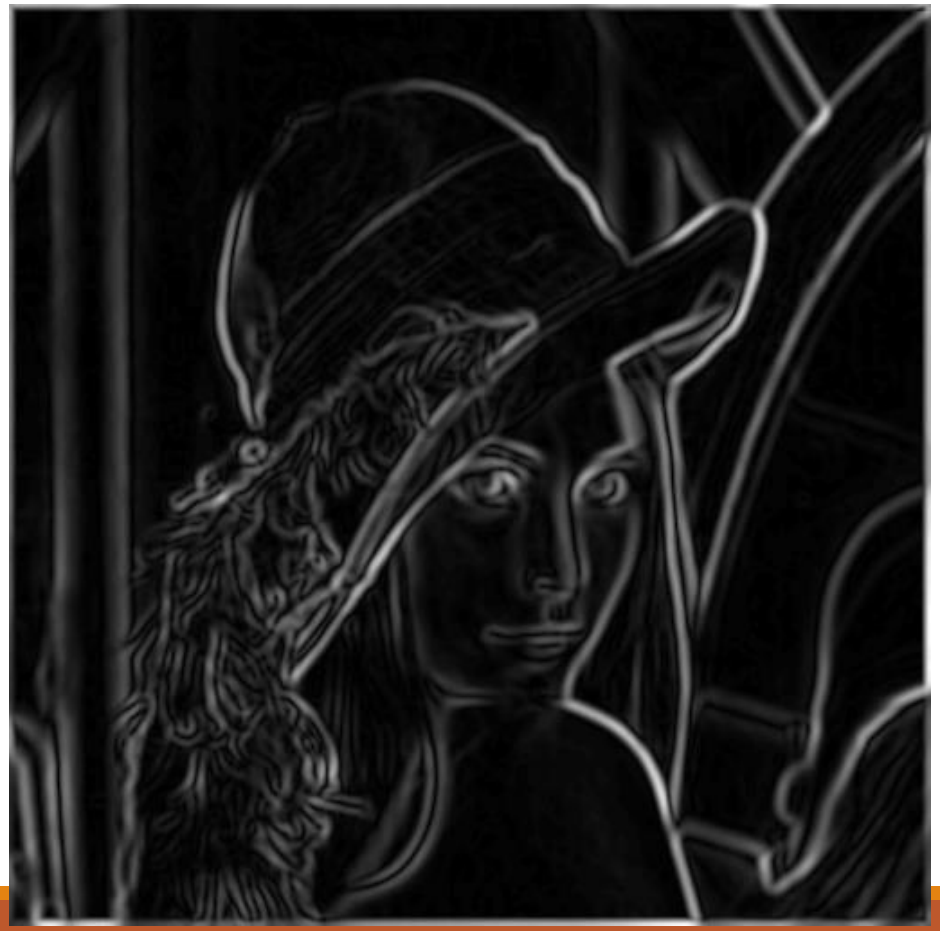
4. Apply hysteresis thresholding/edge linking.

Canny edge detector - example

original image

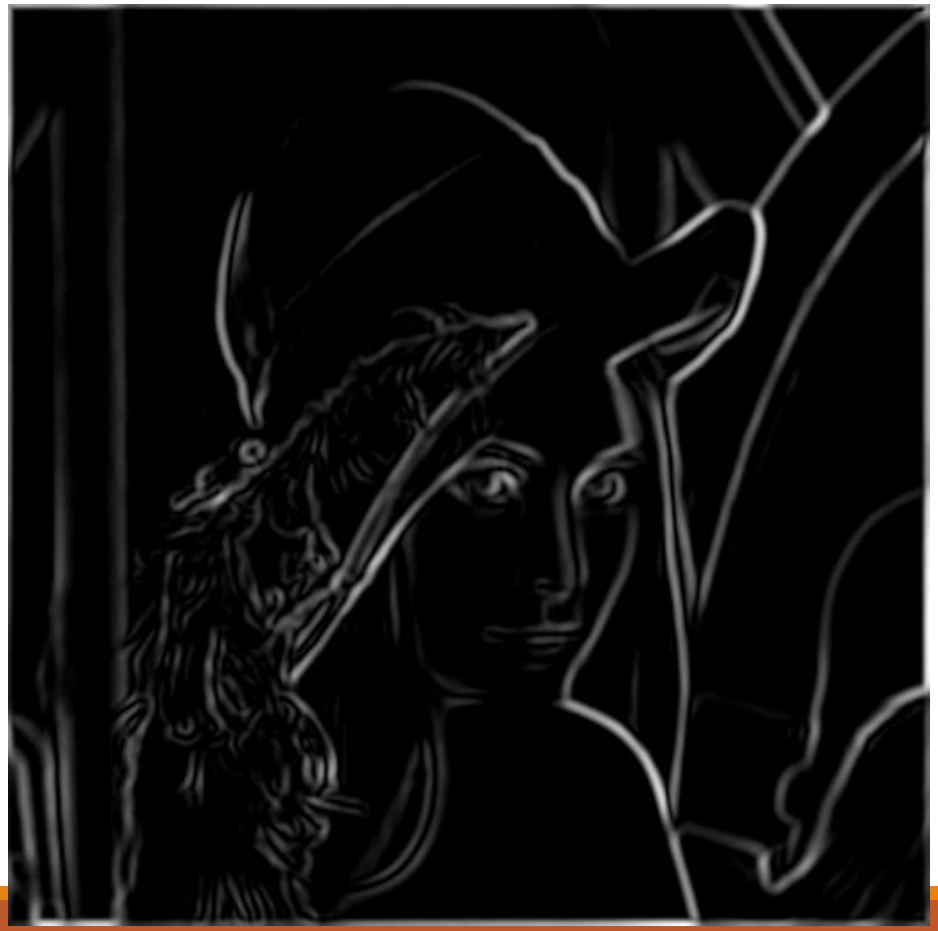
Canny edge detector – example (cont'd)

Gradient magnitude



Canny edge detector – example (cont'd)

Thresholded gradient magnitude

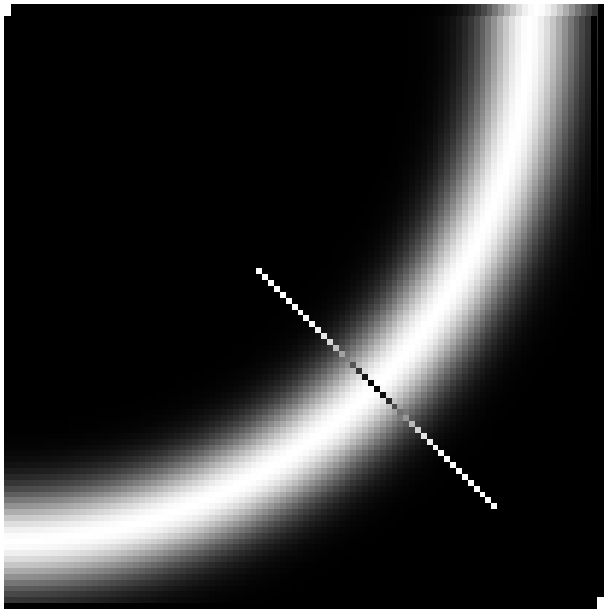


Canny edge detector – example (cont'd)

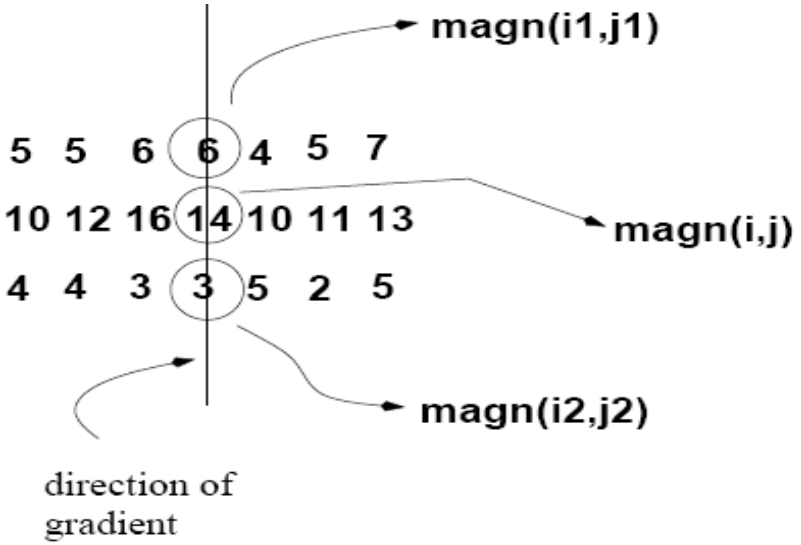
Thinning (non-maxima suppression)

Non-maxima suppression

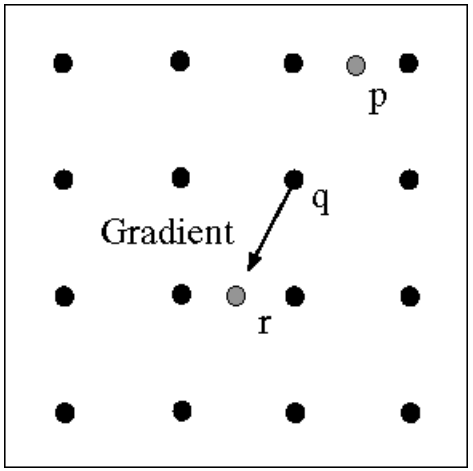
- Check if gradient magnitude at pixel location (i,j) is local maximum along gradient direction



Non-maxima suppression (cont'd)



Warning: requires checking interpolated pixels p and r



Algorithm

For each pixel (i,j) do:

if $magn(i, j) < magn(i_1, j_1)$ or $magn(i, j) < magn(i_2, j_2)$
then $I_N(i, j) = 0$
else $I_N(i, j) = magn(i, j)$

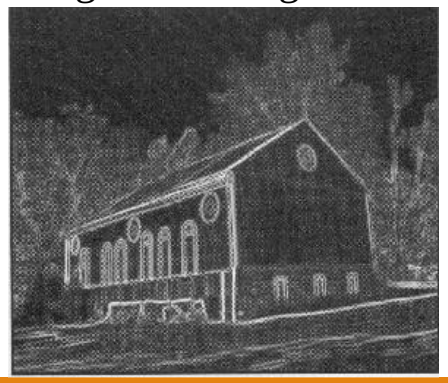
Hysteresis thresholding

- Standard thresholding:

$$E(x,y) = \begin{cases} 1 & \text{if } \|\nabla f(x,y)\| > T \text{ for some threshold } T \\ 0 & \text{otherwise} \end{cases}$$

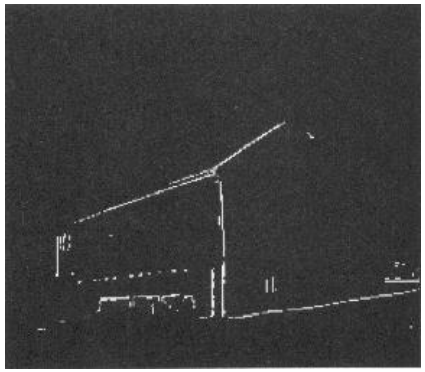
- Can only select “strong” edges.
- Does not guarantee “continuity”.

gradient magnitude



low threshold

high threshold



Hysteresis thresholding (cont'd)

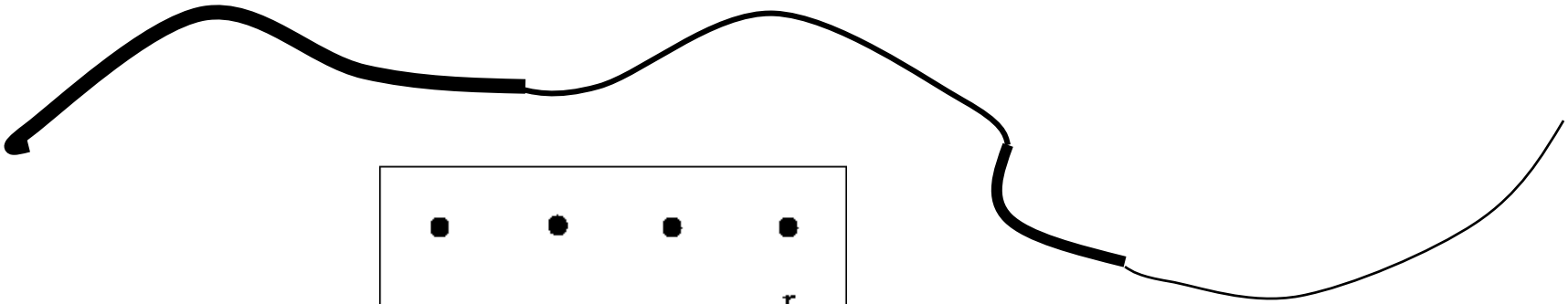
- Hysteresis thresholding uses two thresholds:
  - low threshold t_{low}
  - high threshold t_{high} (usually, $t_{high} = 2t_{low}$)

$$t_{high} \begin{cases} \|\nabla f(x,y)\| \geq t_{high} & \text{definitely an edge} \\ t_{high} \geq \|\nabla f(x,y)\| \geq t_{low} & \text{maybe an edge, depends on context} \\ \|\nabla f(x,y)\| < t_{low} & \text{definitely not an edge} \end{cases}$$

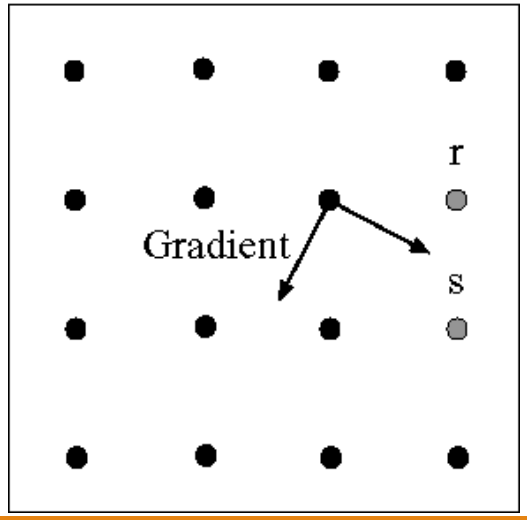
- For "maybe" edges, decide on the edge if neighboring pixel is a strong edge (2 pixels).

Hysteresis thresholding/Edge Linking

Idea: use a **high** threshold to start edge curves and a **low** threshold to continue them.



Use edge "direction" for linking edges



Hysteresis Thresholding/Edge Linking (cont'd)

Algorithm

1. Produce two thresholded images $I_1(i, j)$ and $I_2(i, j)$. (using t_l and t_h)

(note: since $I_2(i, j)$ was formed with a high threshold, it will contain fewer false edges but there might be gaps in the contours)

2. Link the edges in $I_2(i, j)$ into contours

    2.1 Look in $I_1(i, j)$ when a gap is found.

    2.2 By examining the 8 neighbors in $I_1(i, j)$, gather edge points from $I_1(i, j)$ until the gap has been bridged to an edge in $I_2(i, j)$.

Note: large gaps are still difficult to bridge.
(i.e., more sophisticated algorithms are required)

The original image

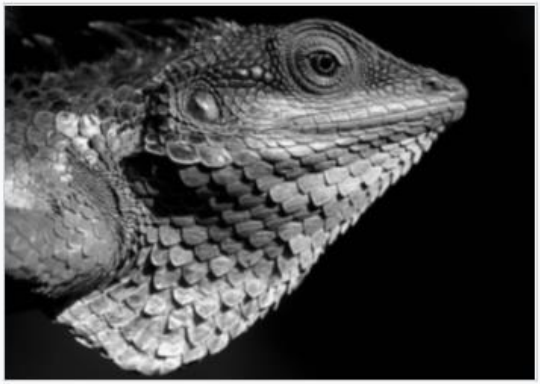
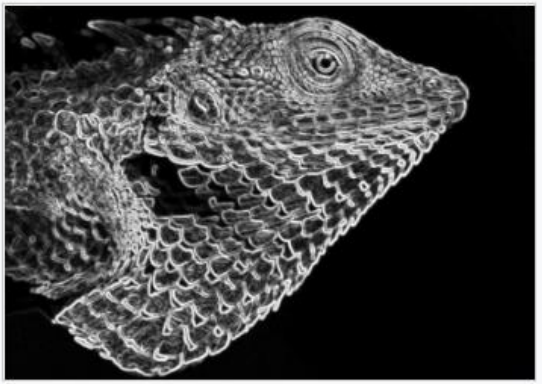
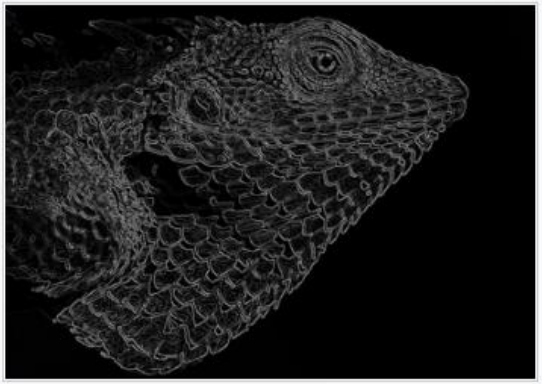


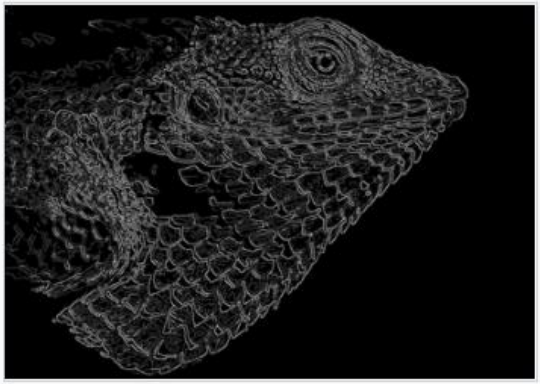
Image has been reduced to grayscale, and a 5x5 Gaussian filter with σ=1.4 has been applied



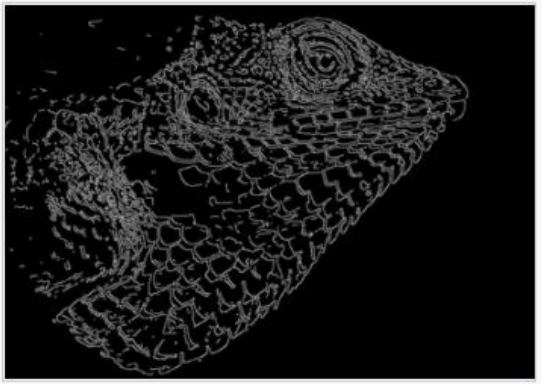
The intensity gradient of the previous image. The edges of the image have been handled by replicating.



Non-maximum suppression applied to the previous image.



Double thresholding applied to the previous image. Weak pixels are those with a gradient value between 0.1 and 0.3. Strong pixels have a gradient value greater than 0.3



Hysteresis applied to the previous image

https://en.wikipedia.org/wiki/Canny_edge_detector