Canny Edge Detection

- Most widely used edge detector in computer vision.
- First derivative of the Gaussian closely approximates the operator that optimizes the product of *signal-to-noise* ratio and localization.
- Analysis based on "step-edges" corrupted by "additive Gaussian noise".

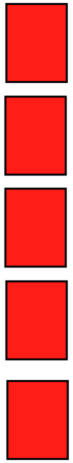
Least squares with binomial weights ~~~ edge detector.

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

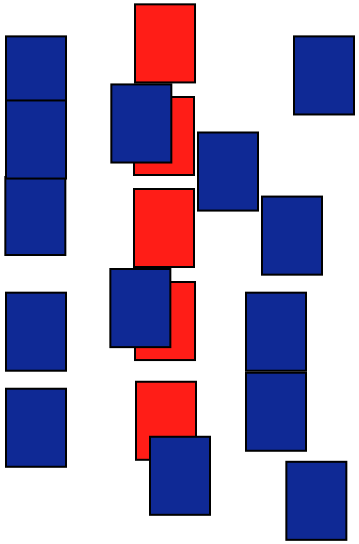
Edge Detection Criteria

- Criteria for optimal edge detection (Canny 86):
  - Good detection accuracy:
    - minimize the probability of false positives (detecting spurious edges caused by noise),
    - false negatives (missing real edges)
  - Good localization:
    - edges must be detected as close as possible to the true edges.
  - Single response constraint:
    - minimize the number of local maxima around the true edge (i.e. detector must return single point for each true edge point)

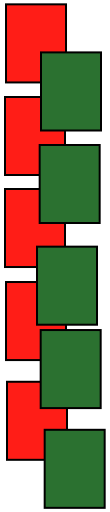
- Examples... valid mostly for straight edges...



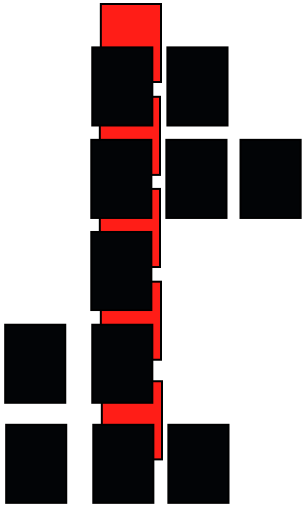
True edge



Poor robustness to noise



Poor localization



Too many responses

Canny Edge Detection

Steps:

1+2. Gaussian smoothing together with derivative of Gaussian (~discrete)

3. Find magnitude and orientation of gradient

4. Extract edge points: Non-maximum suppression

5. Linking and thresholding: Hysteresis

- MATLAB: `edge(I, ‘canny’)`

First Two Steps

- Smoothing

$$I' = g(x,y) * I \qquad g(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Can be done with two one-dimensional filters.

- Derivative

$$S = \nabla(g * I) = (\nabla g) * I =$$

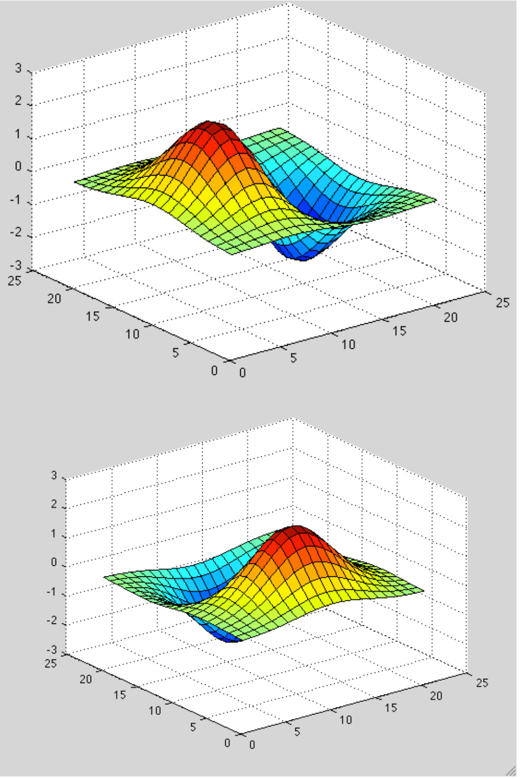
$$= \begin{bmatrix} g_x \\ g_y \end{bmatrix} * I = \begin{bmatrix} g_x * I \\ g_y * I \end{bmatrix} \qquad \nabla g = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

h = g

two dimensional Gaussian

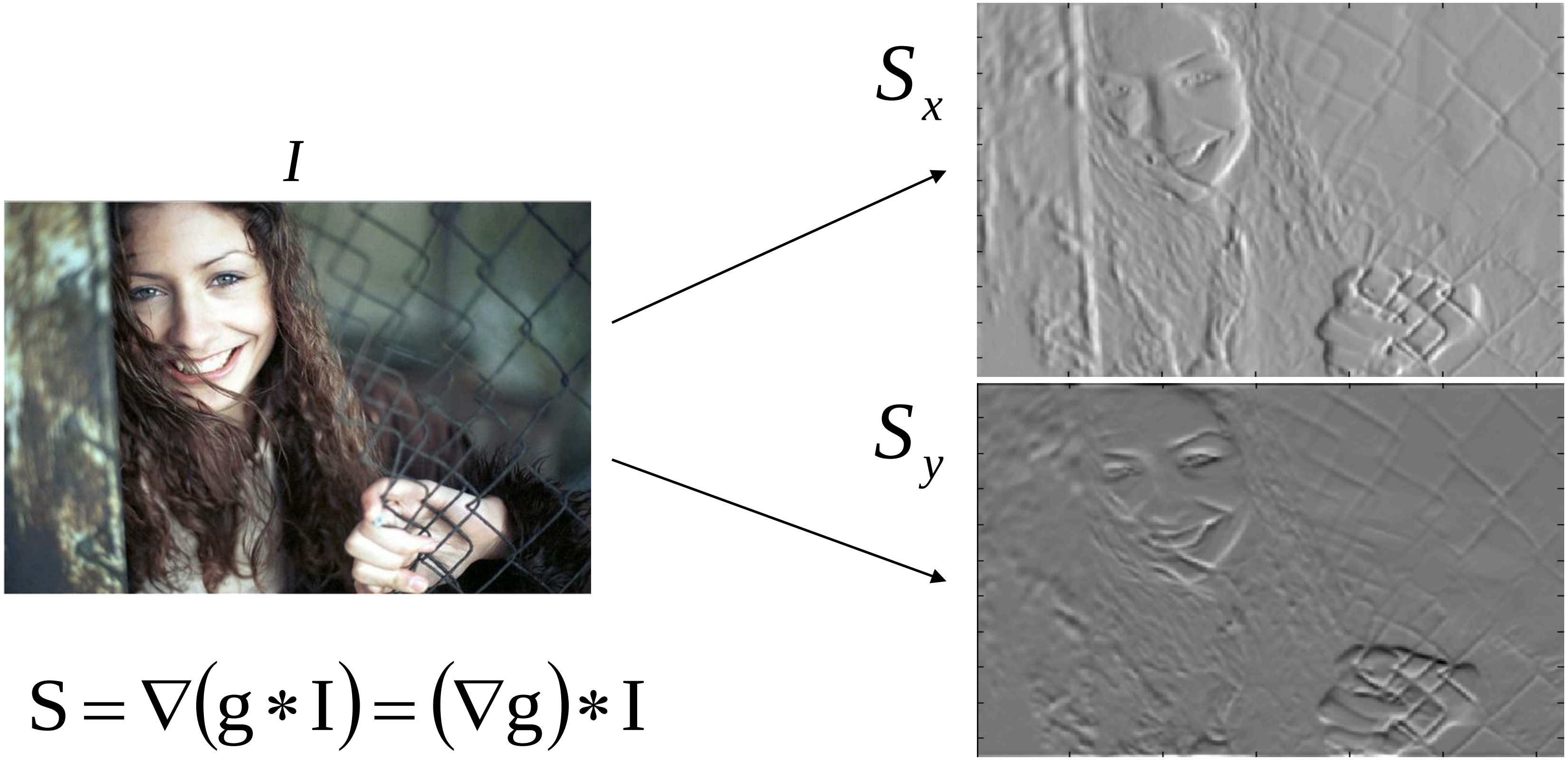
$$h_x(x,y) = \frac{\partial h(x,y)}{\partial x} = \frac{-x}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$h_y(x,y) = \frac{\partial h(x,y)}{\partial y} = \frac{-y}{2\pi\sigma^4} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



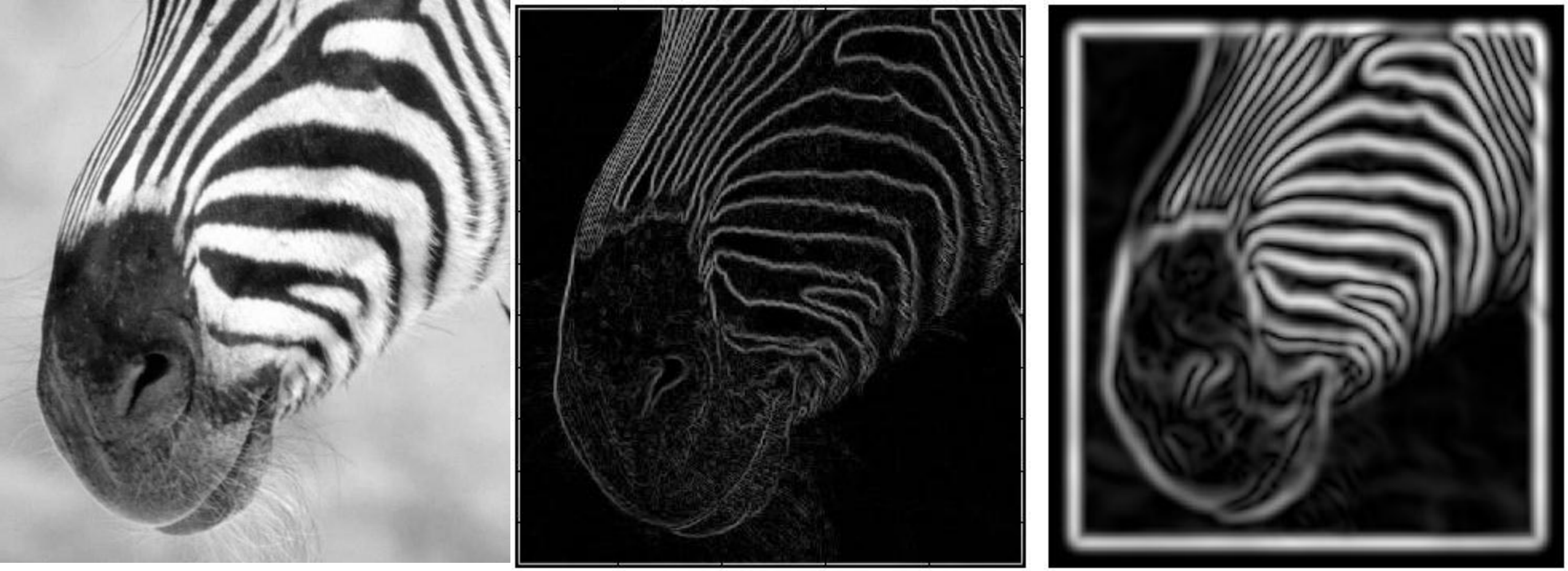
Scale

Example:



$$S = \nabla(g * I) = (\nabla g) * I$$

$$S = \begin{bmatrix} S_x & S_y \end{bmatrix}$$ = gradient vector



sigma 1 pixel 2 pixels

Increased smoothing:

- Eliminates noise edges.
- Makes edges smoother and thicker.
- Removes fine detail.

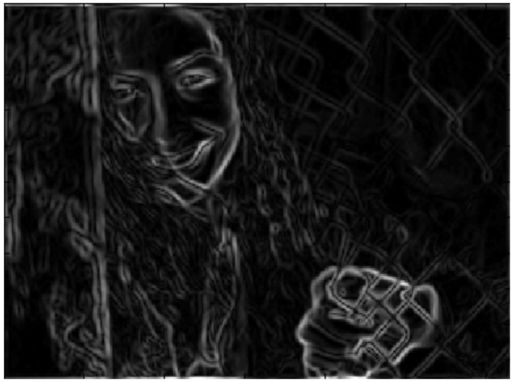
Third Step

- magnitude and direction of $S = \begin{bmatrix} S_x & S_y \end{bmatrix}$

$$\text{magnitude} = \sqrt{(S_x^2 + S_y^2)}$$

$$\text{direction} = \theta = \tan^{-1}\frac{S_y}{S_x}$$

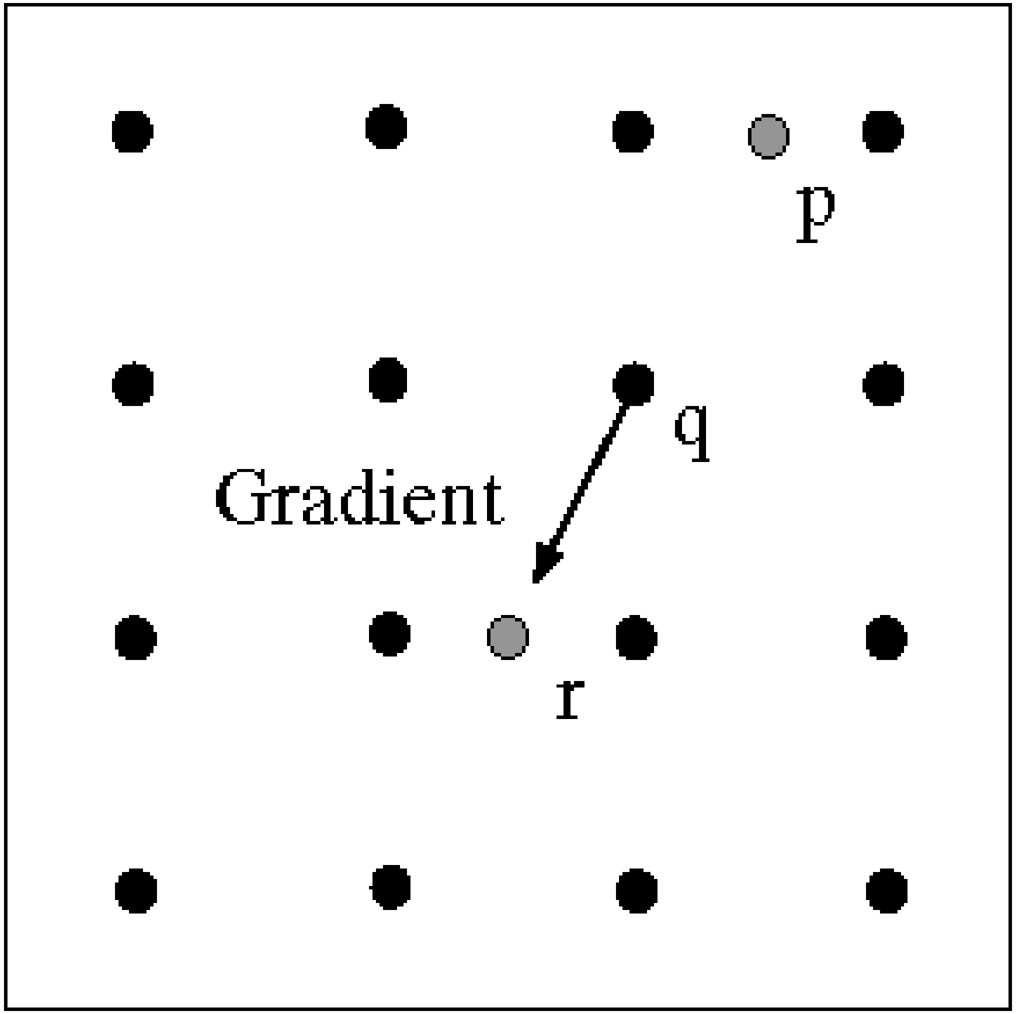
image



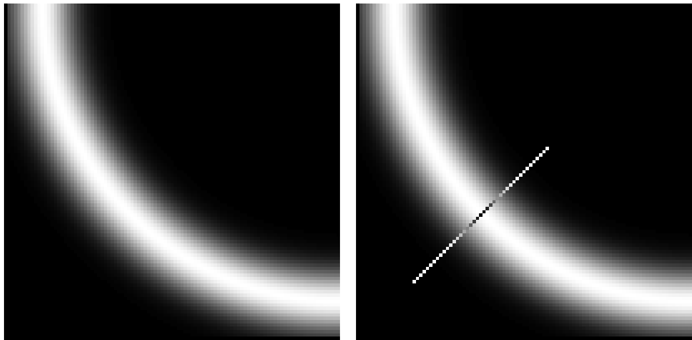
gradient magnitude

Non-maximum suppression along the direction of gradient

Fourth Step

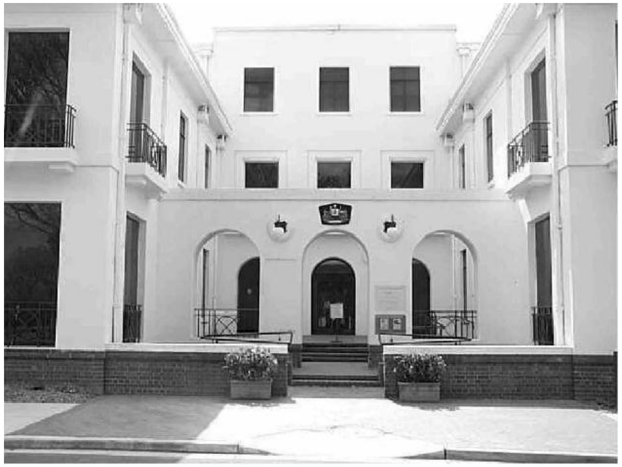


At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

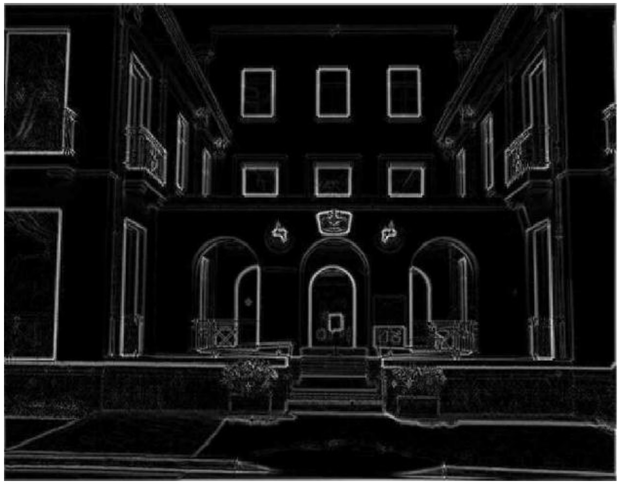


Source: D. Forsyth

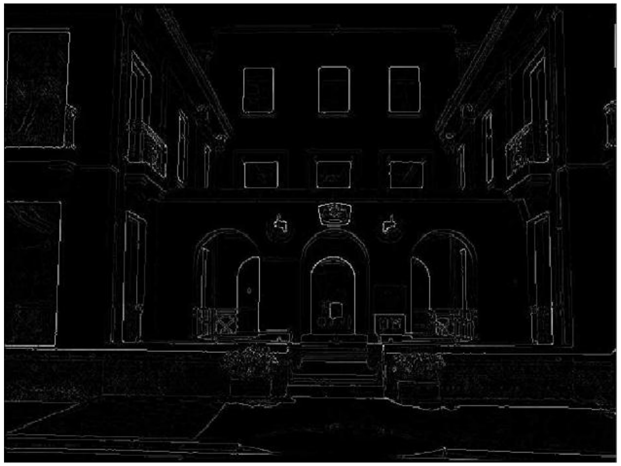
Example:
Non-Maximum Suppression



Original image



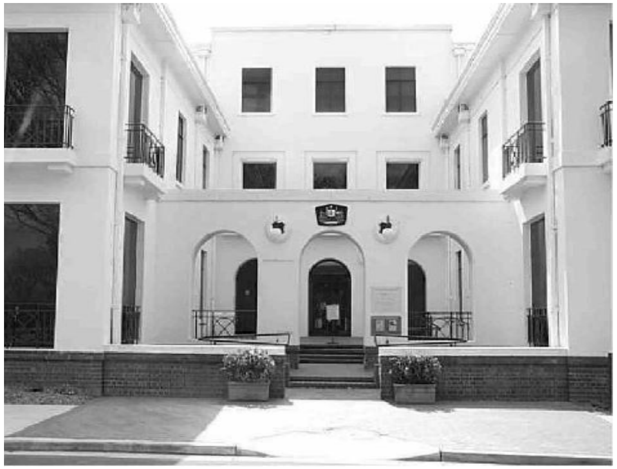
Gradient magnitude

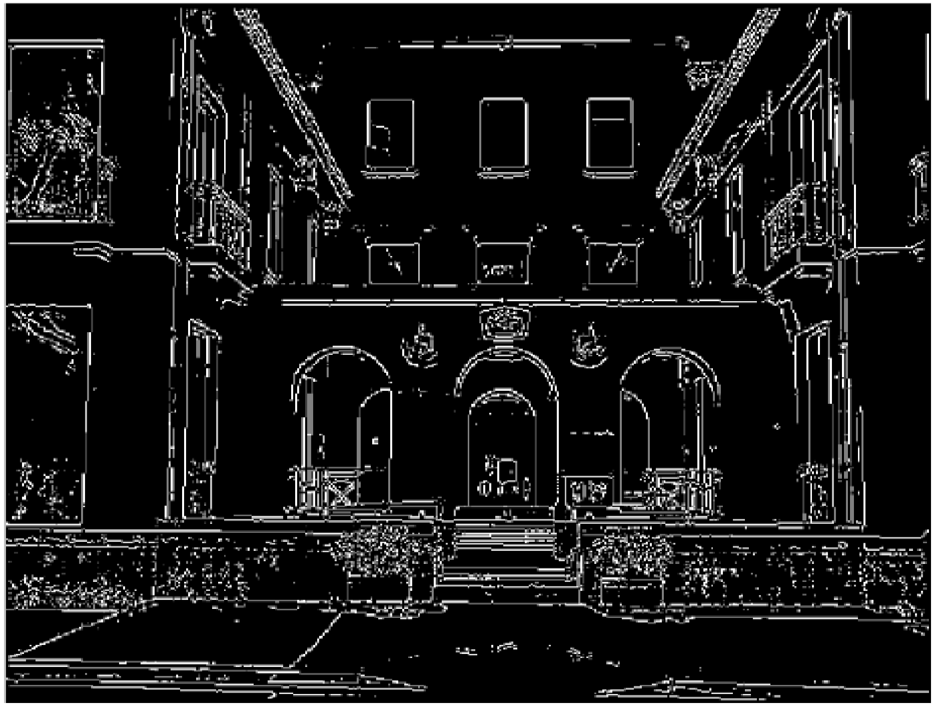


courtesy of G. Loy

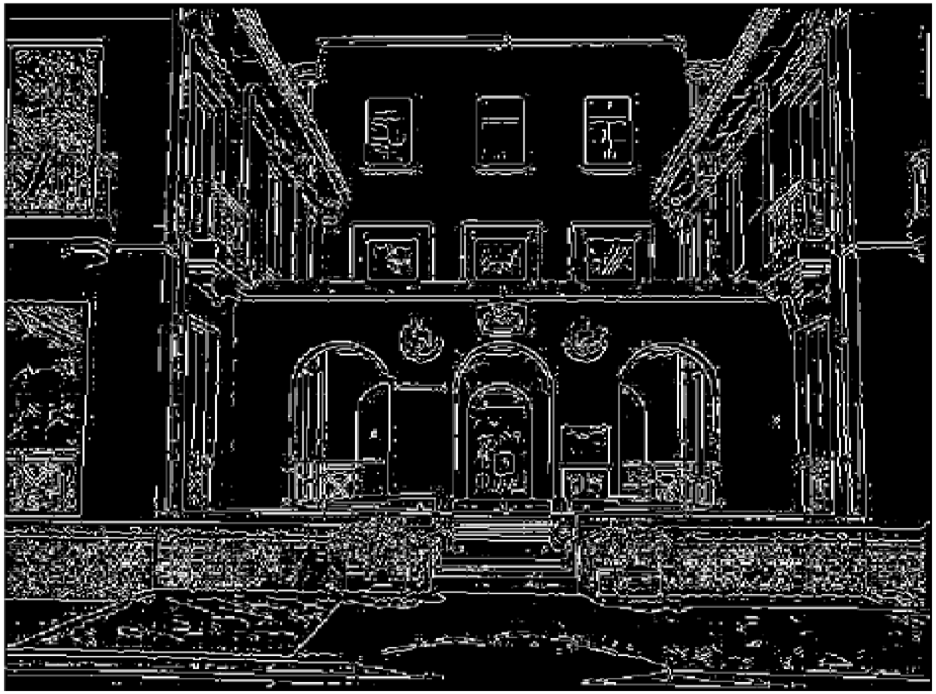
Non-maxima suppressed

Slide credit: Christopher Rasmussen





high threshold
strong edges only



low threshold
weak edges too

Fifth Step: Hysteresis Thresholding

- Hysteresis: no LOW maybe HIGH sure.
- Maintain two thresholds k_{high} and k_{low}
  - Use k_{high} to find strong edges to start edge chain.
  - Use k_{low} to find weak edges along the edge chain.
- Typical ratio of thresholds is roughly

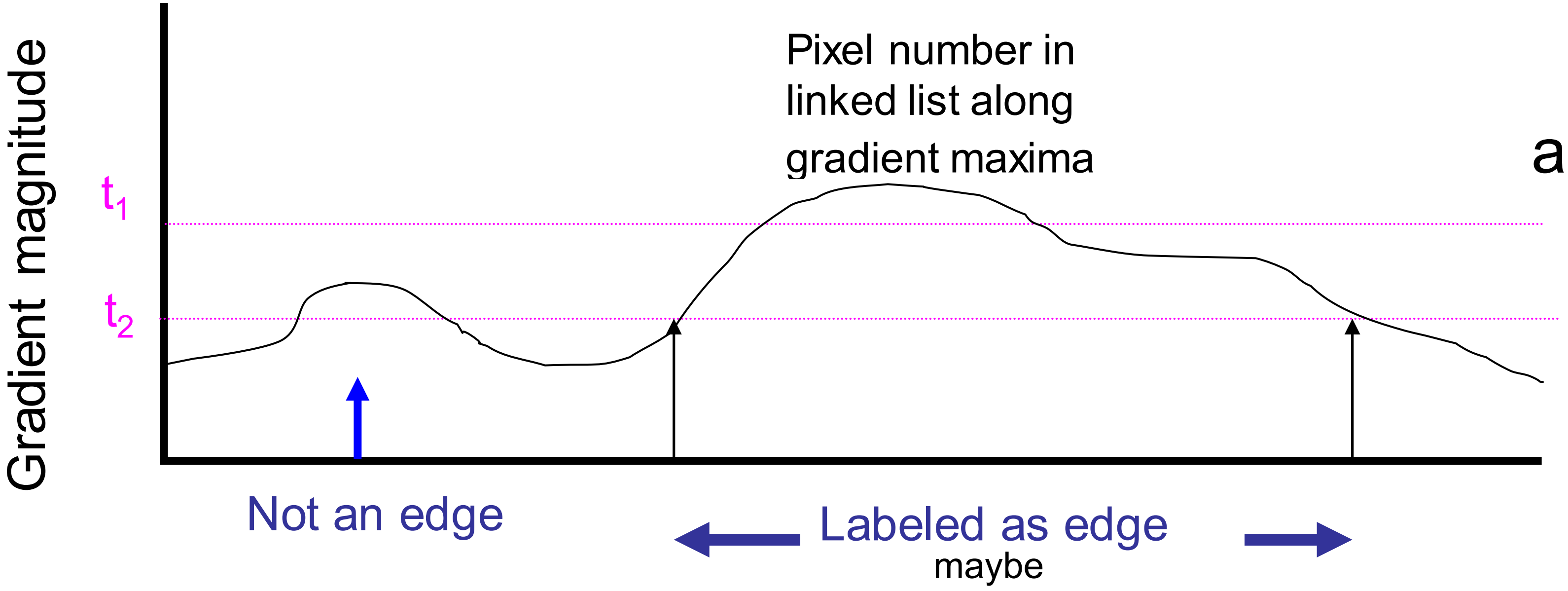
$$k_{high} / k_{low} = 2 - 2.5$$

Closing edge gaps

- Check that maximum value of gradient value is sufficiently large and...

  ... use hysteresis.

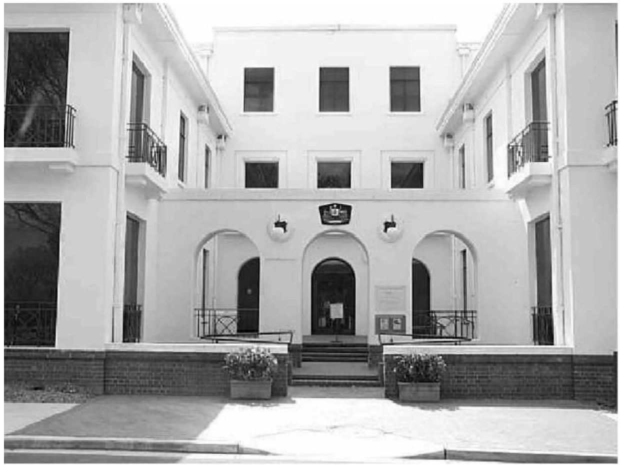
  - use a high threshold to start edge curves and a low threshold to continue them.



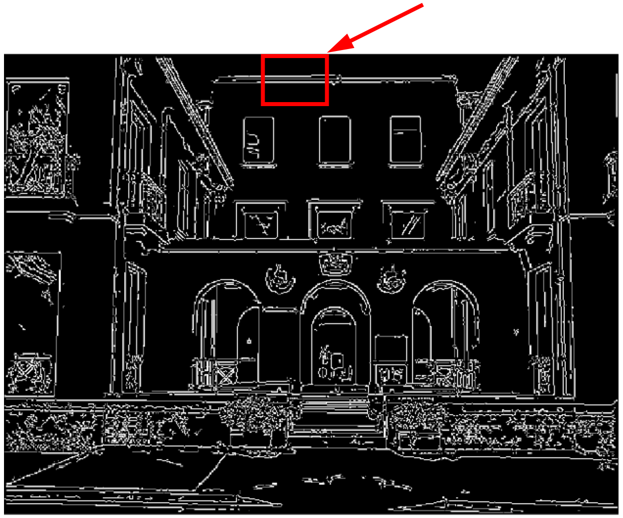
a line

Example

Original image

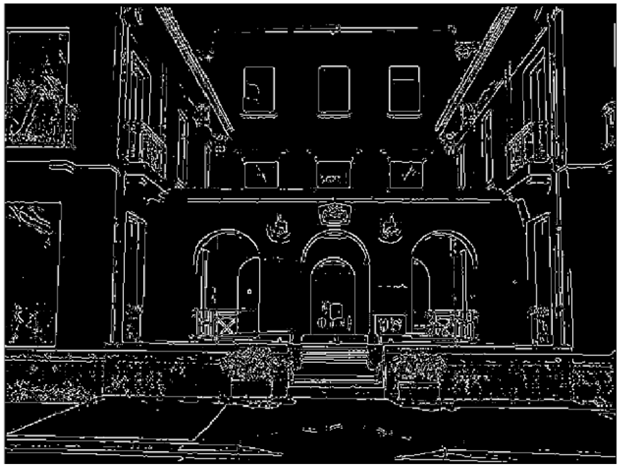


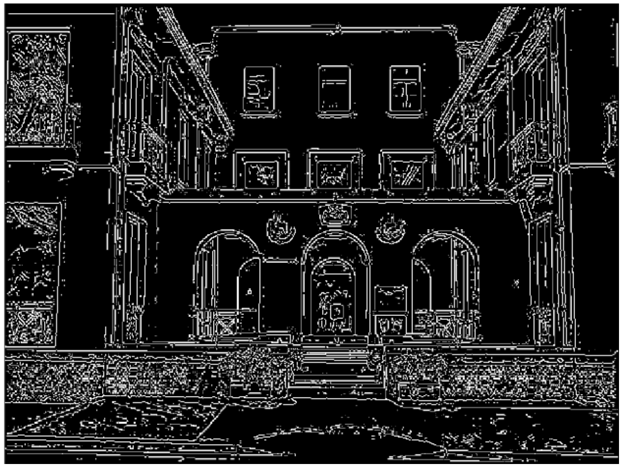
gap is gone



Strong + *connected* weak edges

Strong edges only



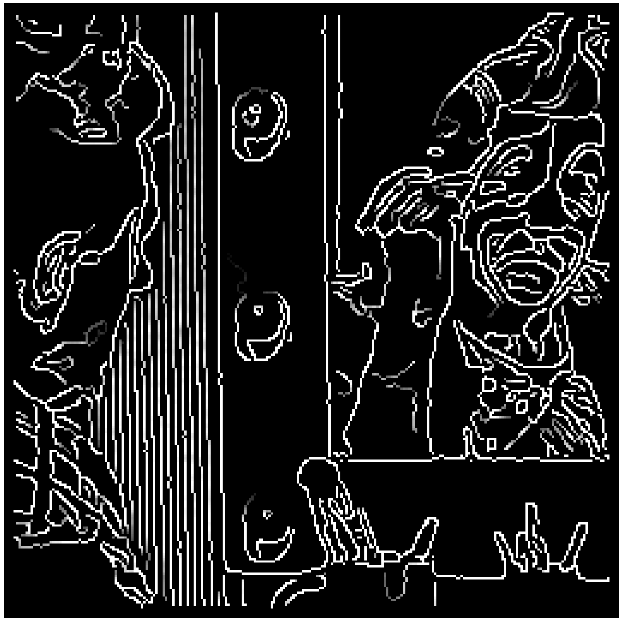


Weak edges too

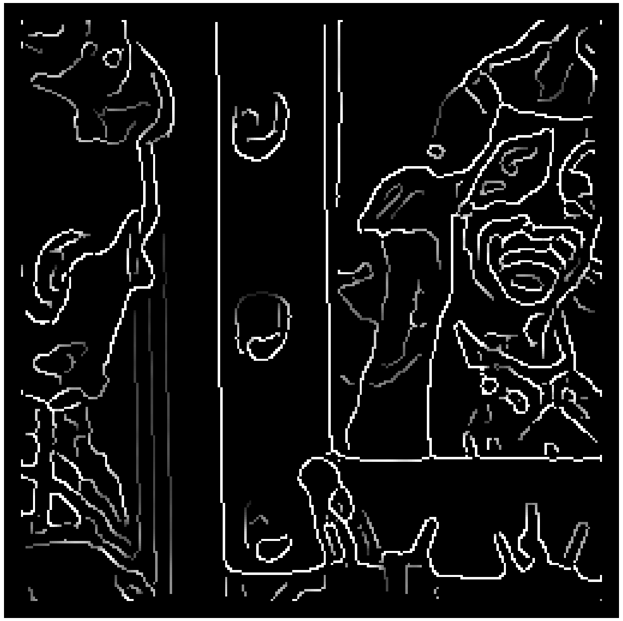
courtesy of G. Loy

Effect of σ (Gaussian kernel spread/size)

original



Canny with $\sigma = 1$

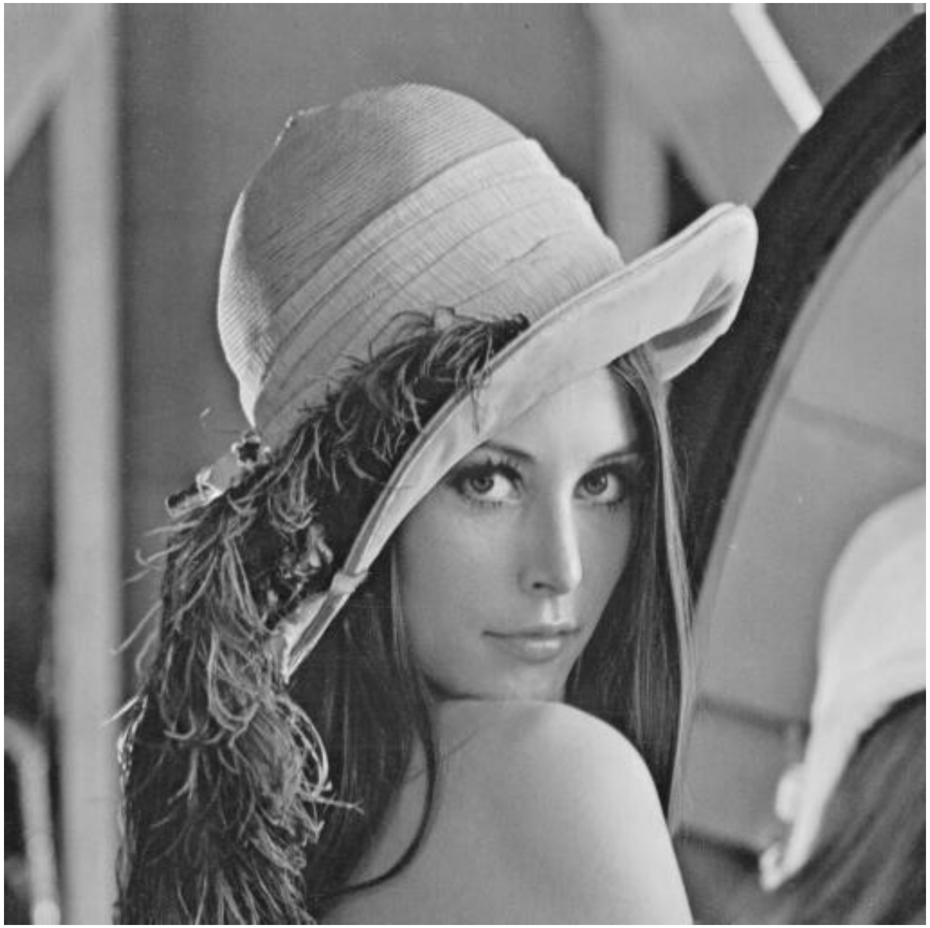


Canny with $\sigma = 2$

- The choice of σ depends on desired behavior
  - large σ detects large scale edges
  - small σ detects fine features

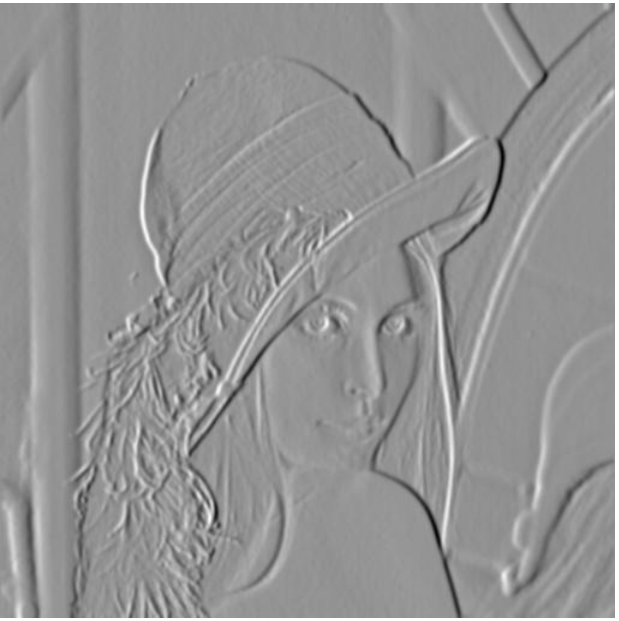
Source: S. Seitz

Example of Canny edge detection

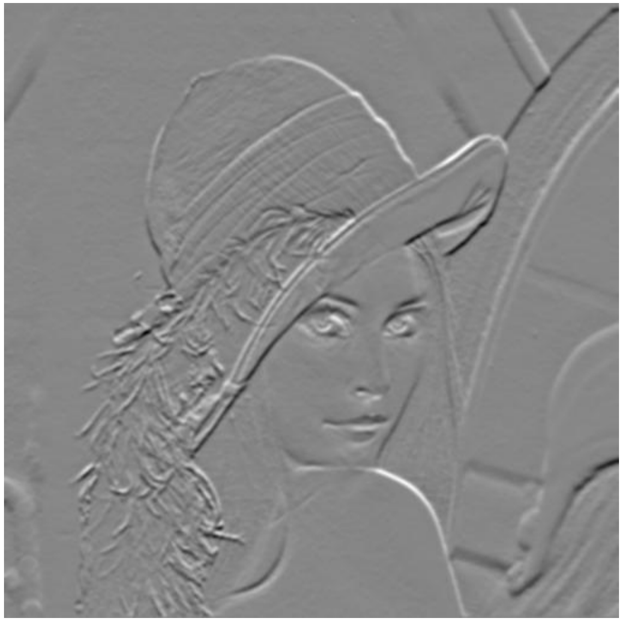


original image (Lena)

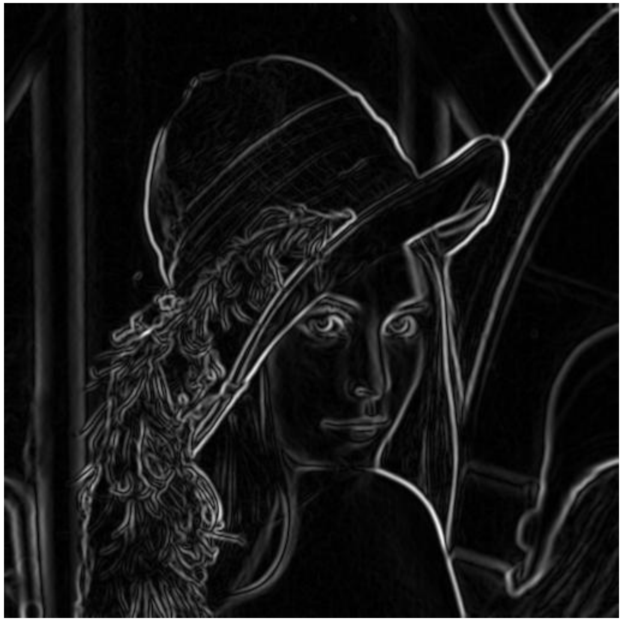
Compute Gradients (DoG)



X-Derivative of Gaussian



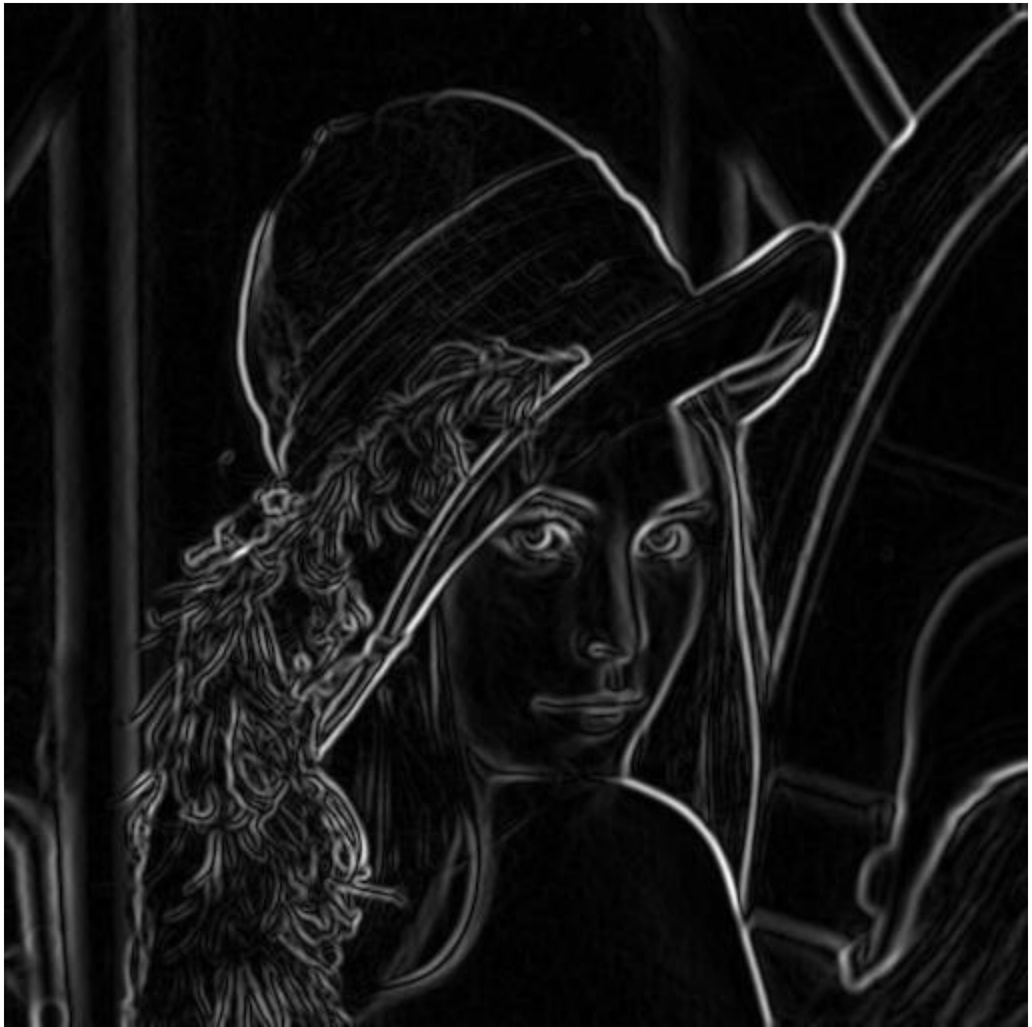
Y-Derivative of Gaussian



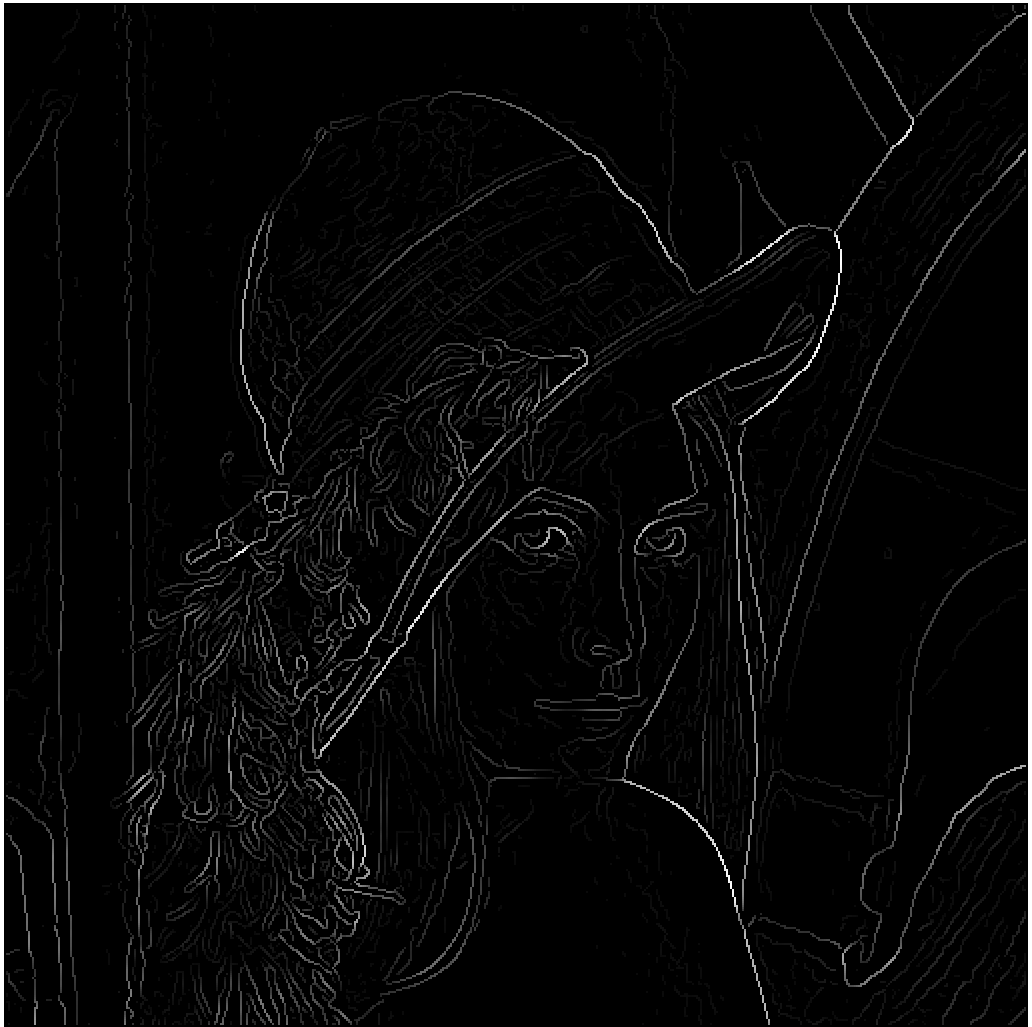
Gradient Magnitude

and orientation

Before non-max suppression...

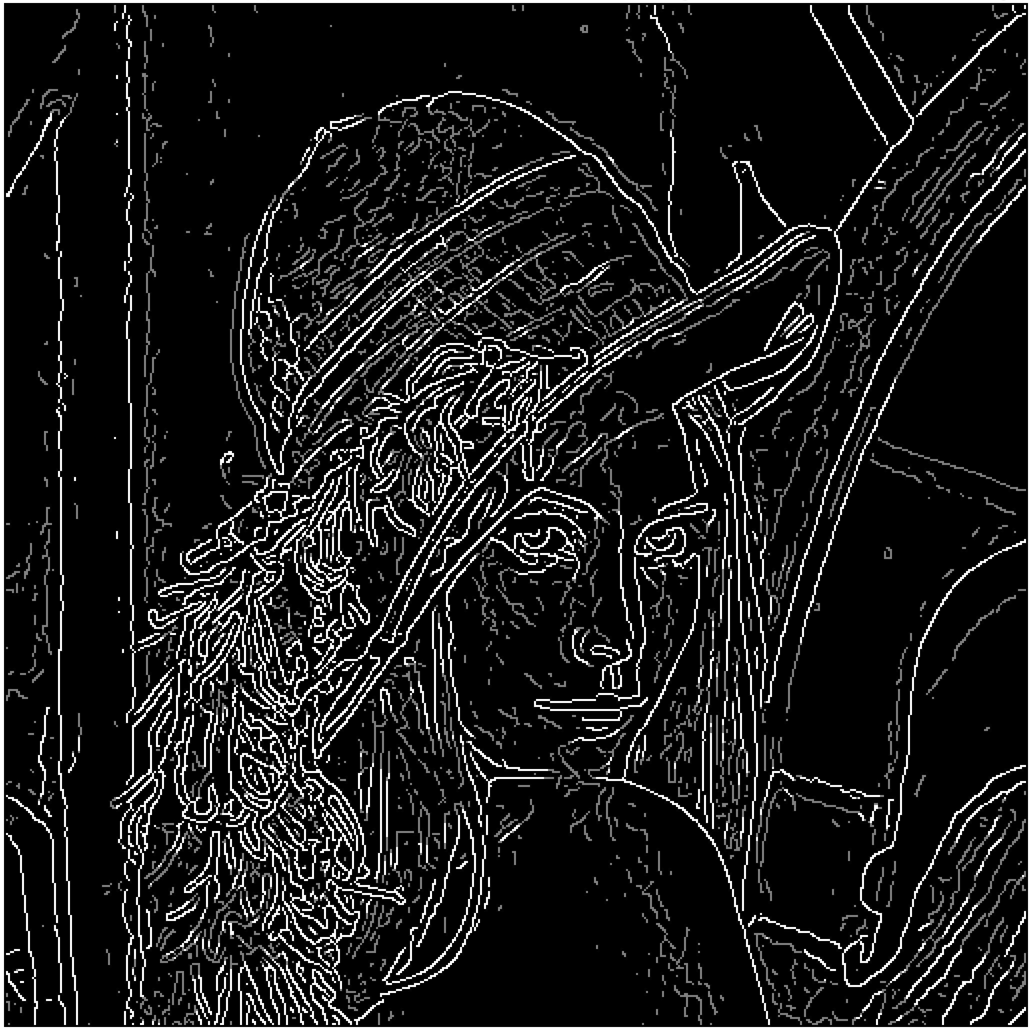


...after non-max suppression

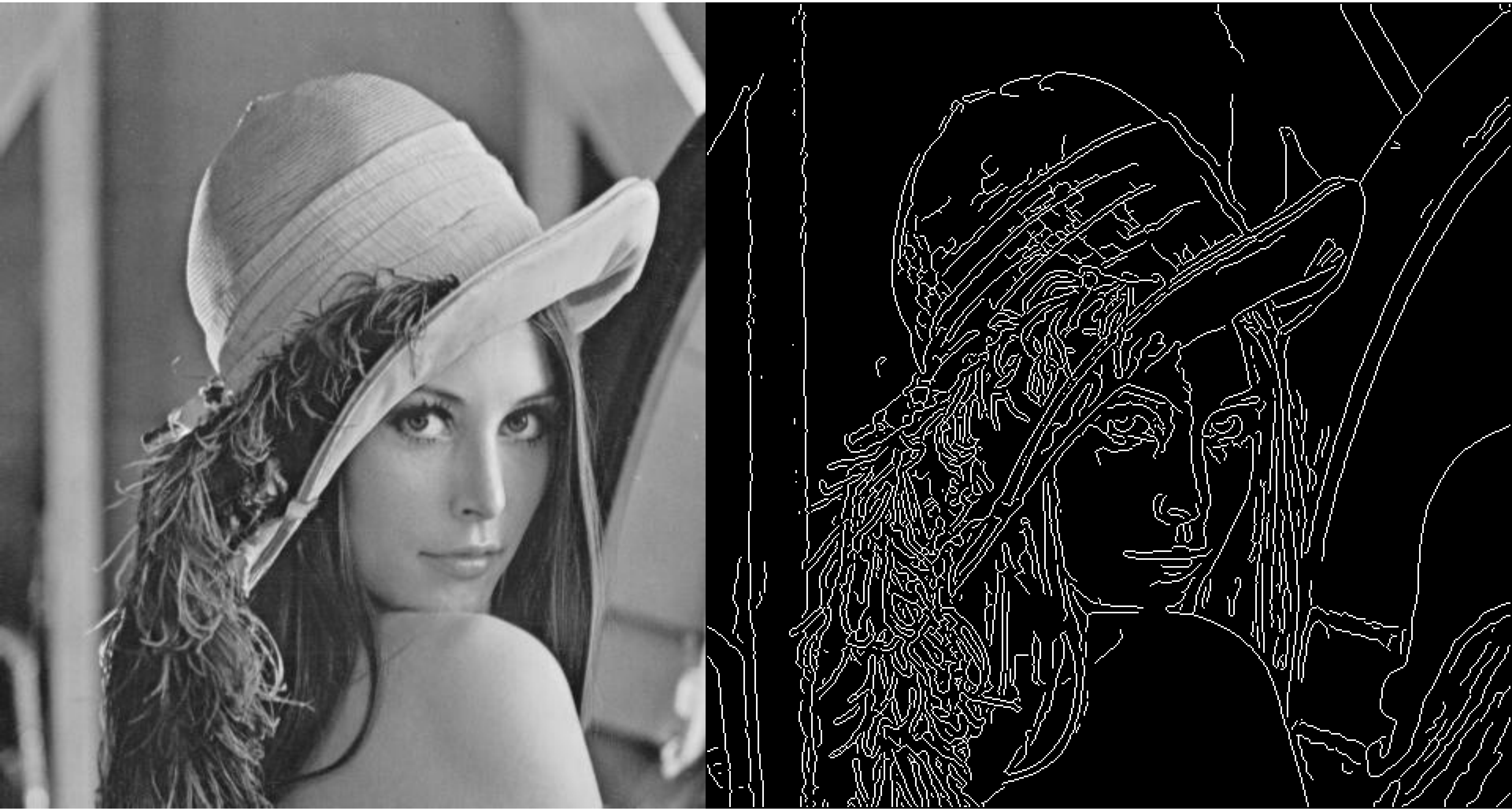


Before the hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels



Final Canny Edges



A *hidden advantage* for the human observer. She/he first see the original image and only after the edges detected.

What happens if she/he cannot see the original image first and therefore can rely on it?

This is how all the computer vision algorithms *has to work... all the time*!