09/18/13

Finding Boundaries

Computer Vision

CS 482, GMU

Zoran Duric

Many slides from James Hayes, Lana Lazebnik, Steve Seitz, David Forsyth, David Lowe, Fei-Fei Li, and Derek Hoiem

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels

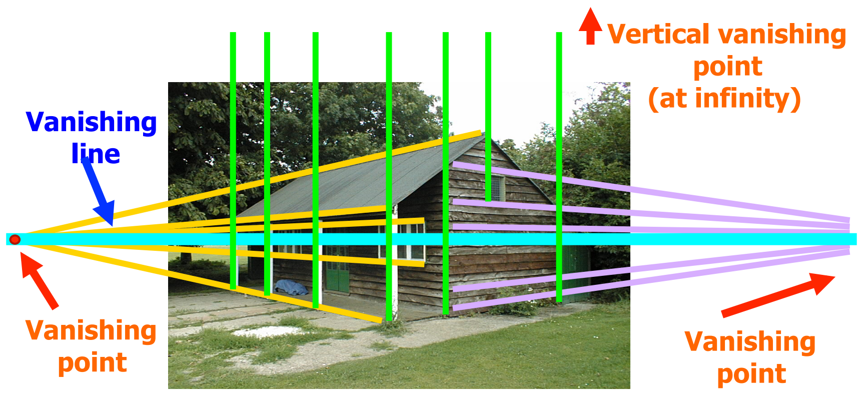
- **Ideal:** artist's line drawing (but artist is also using object-level knowledge)

Source: D. Lowe

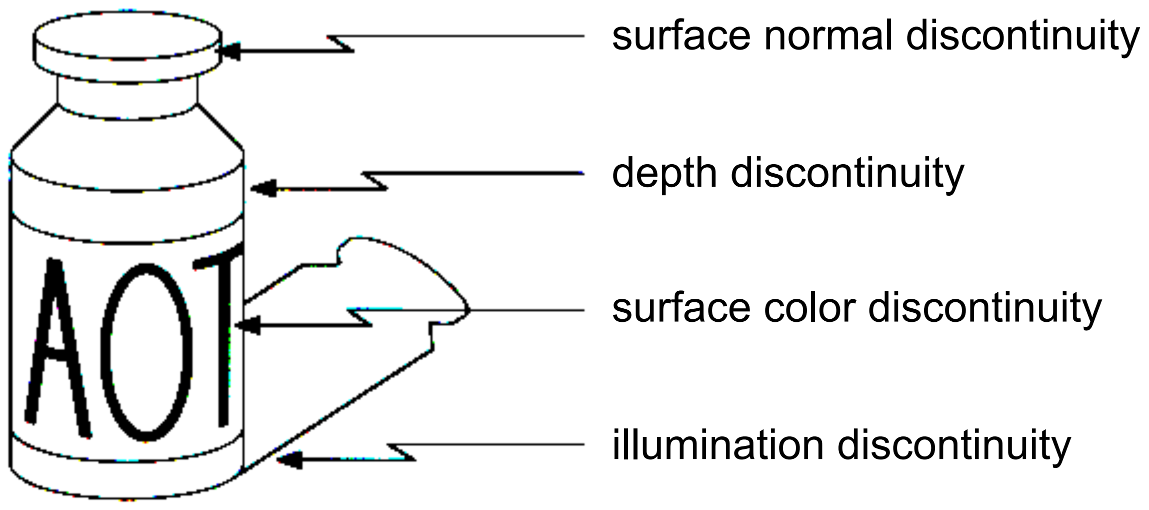
Why do we care about edges?

- Extract information, recognize objects

- Recover geometry and viewpoint



Origin of Edges


AOT
surface normal discontinuity
depth discontinuity
surface color discontinuity
illumination discontinuity

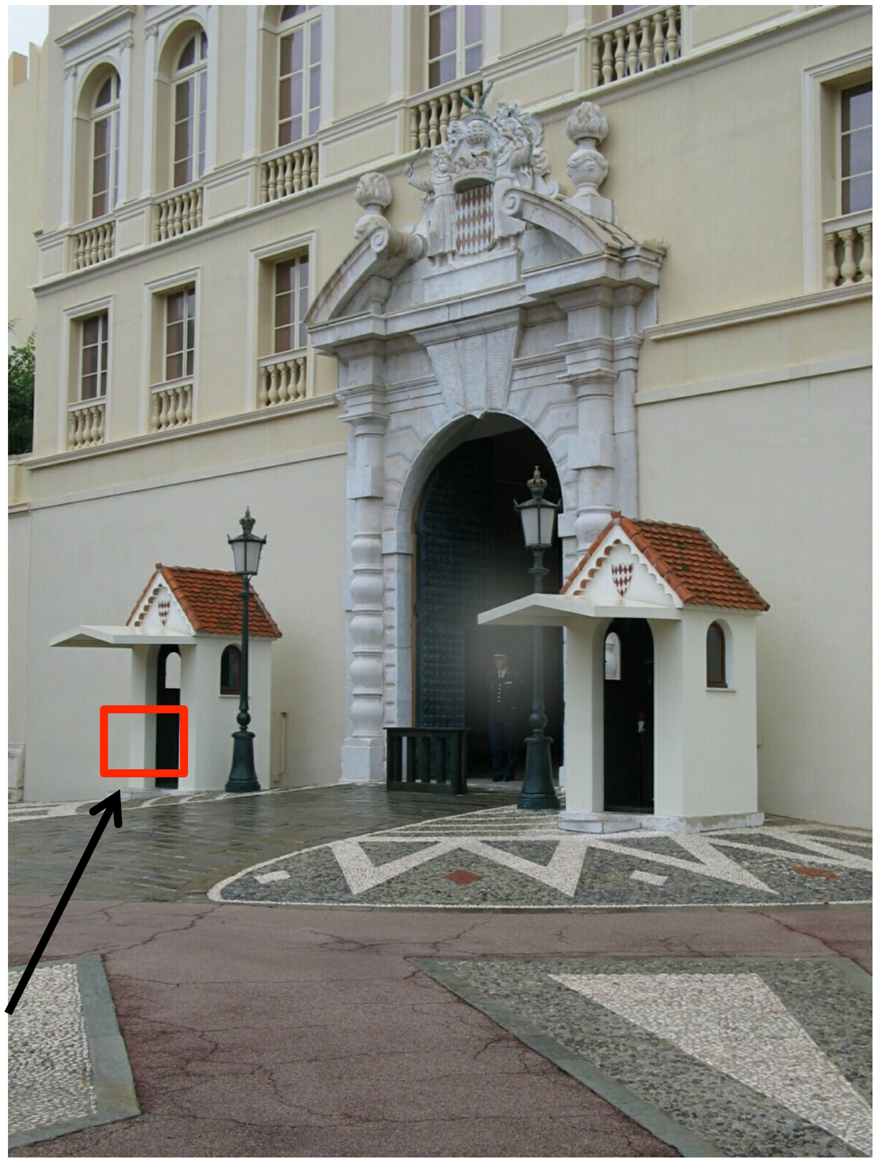
- Edges are caused by a variety of factors

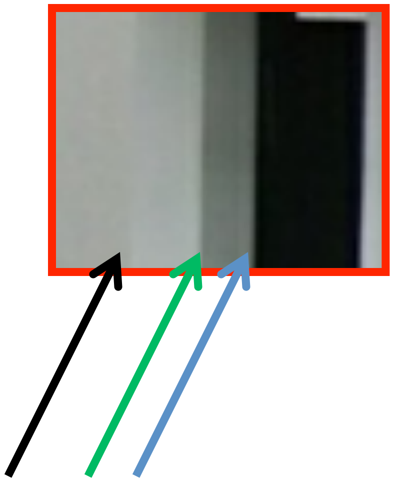
Source: Steve Seitz

Closeup of edges

Source: D. Hoiem

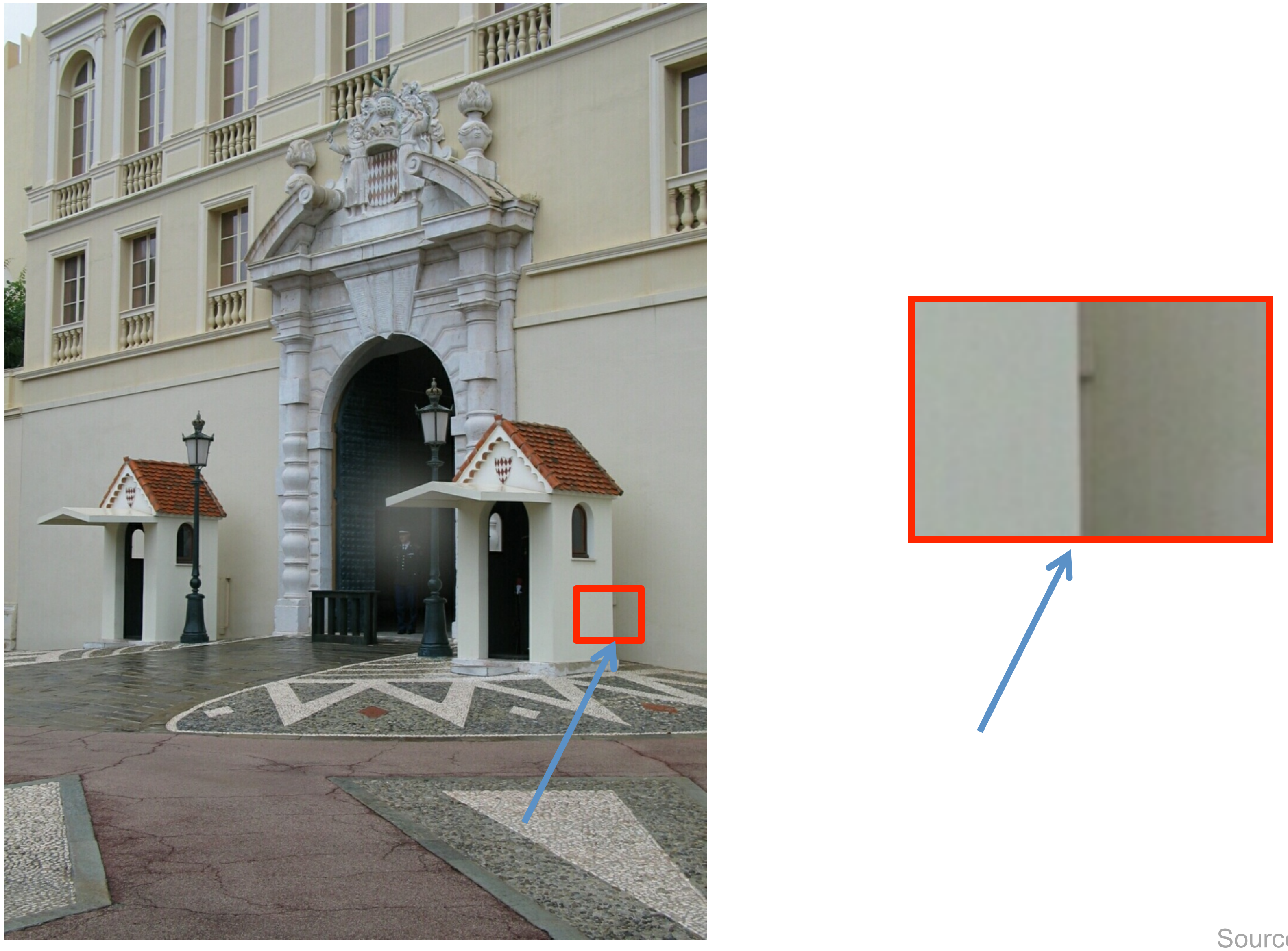
Closeup of edges





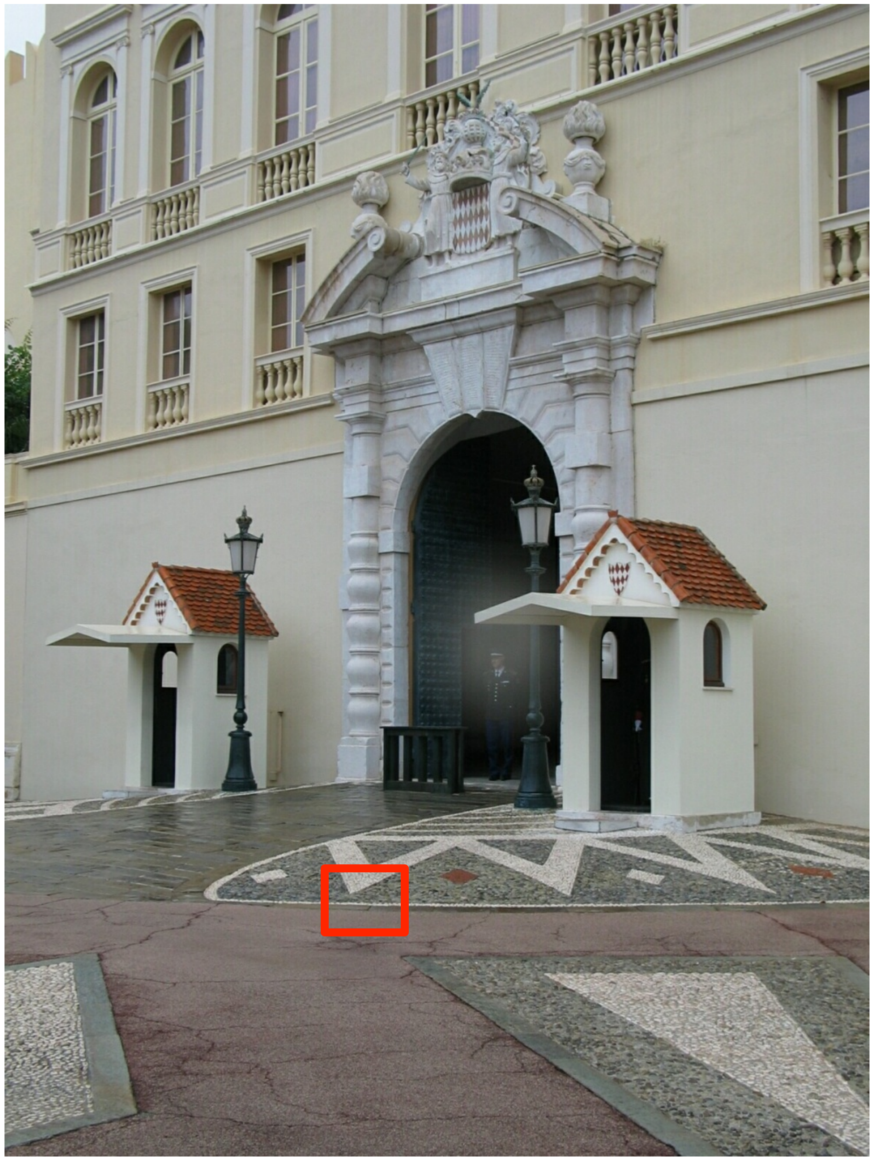
Source: D. Hoiem

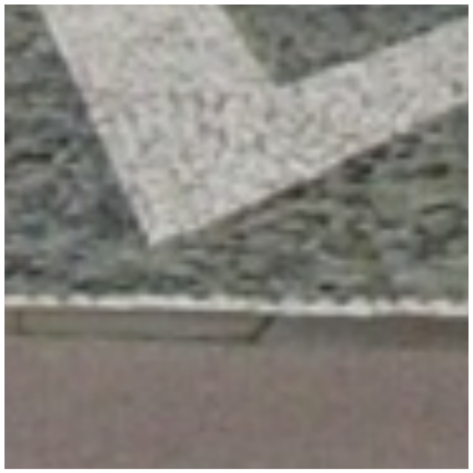
Closeup of edges



Source: D. Hoiem

Closeup of edges

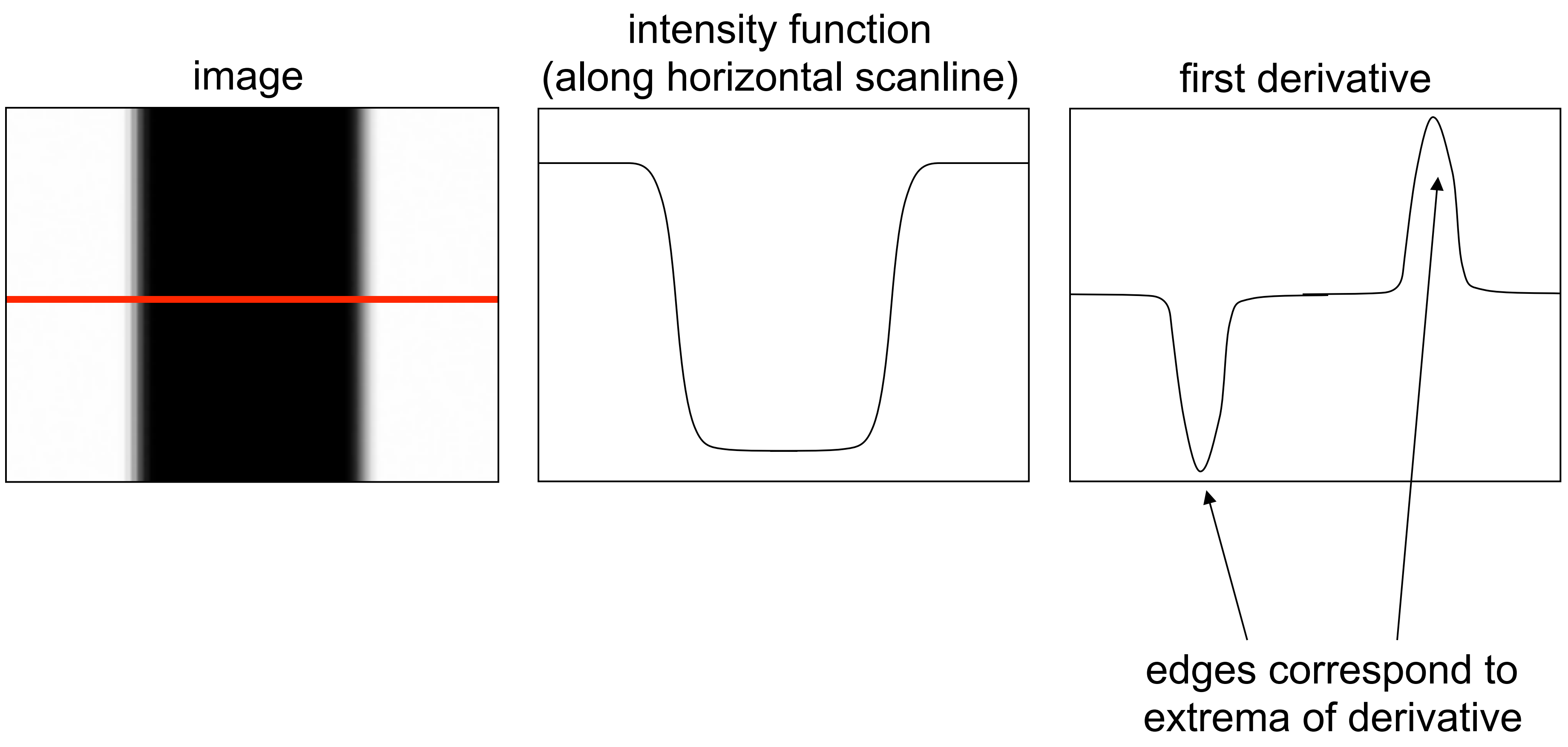




Source: D. Hoiem

Characterizing edges

- An edge is a place of rapid change in the image intensity function



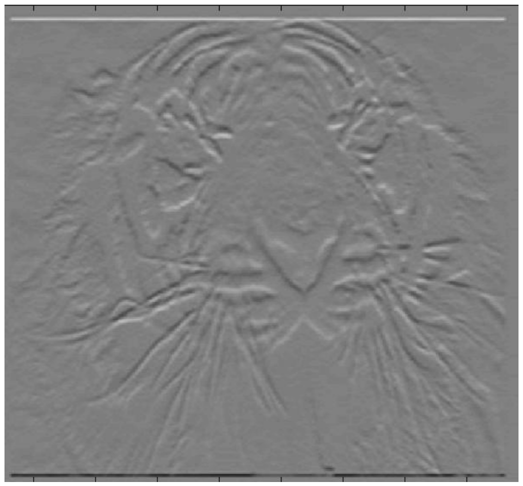
Assorted finite difference filters

Prewitt: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -1 & -1 \\ \hline \end{array}$

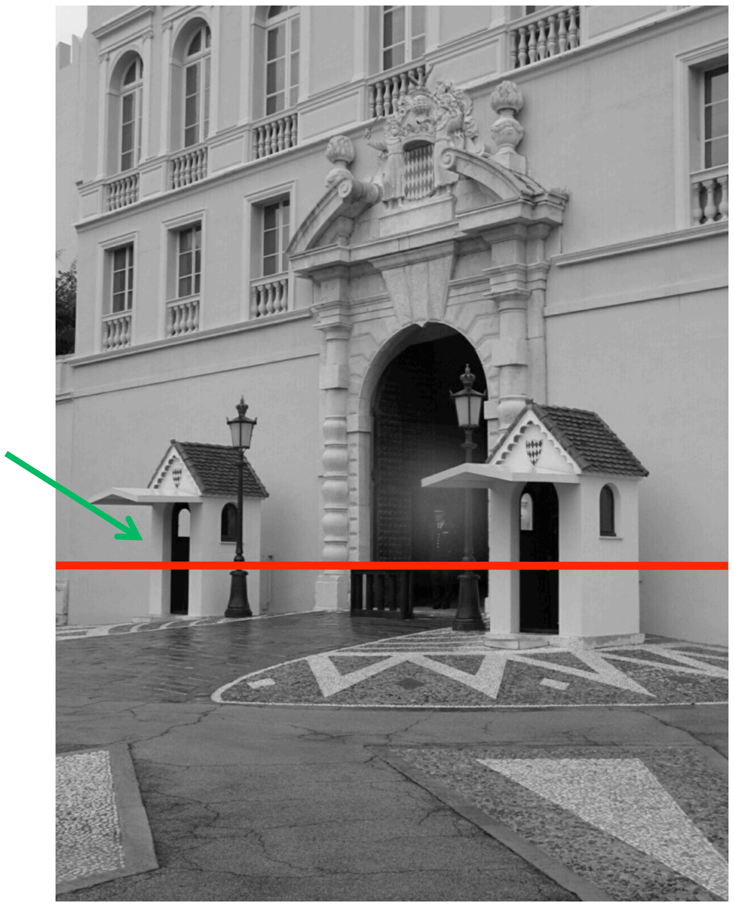
Sobel: $M_x = \begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$

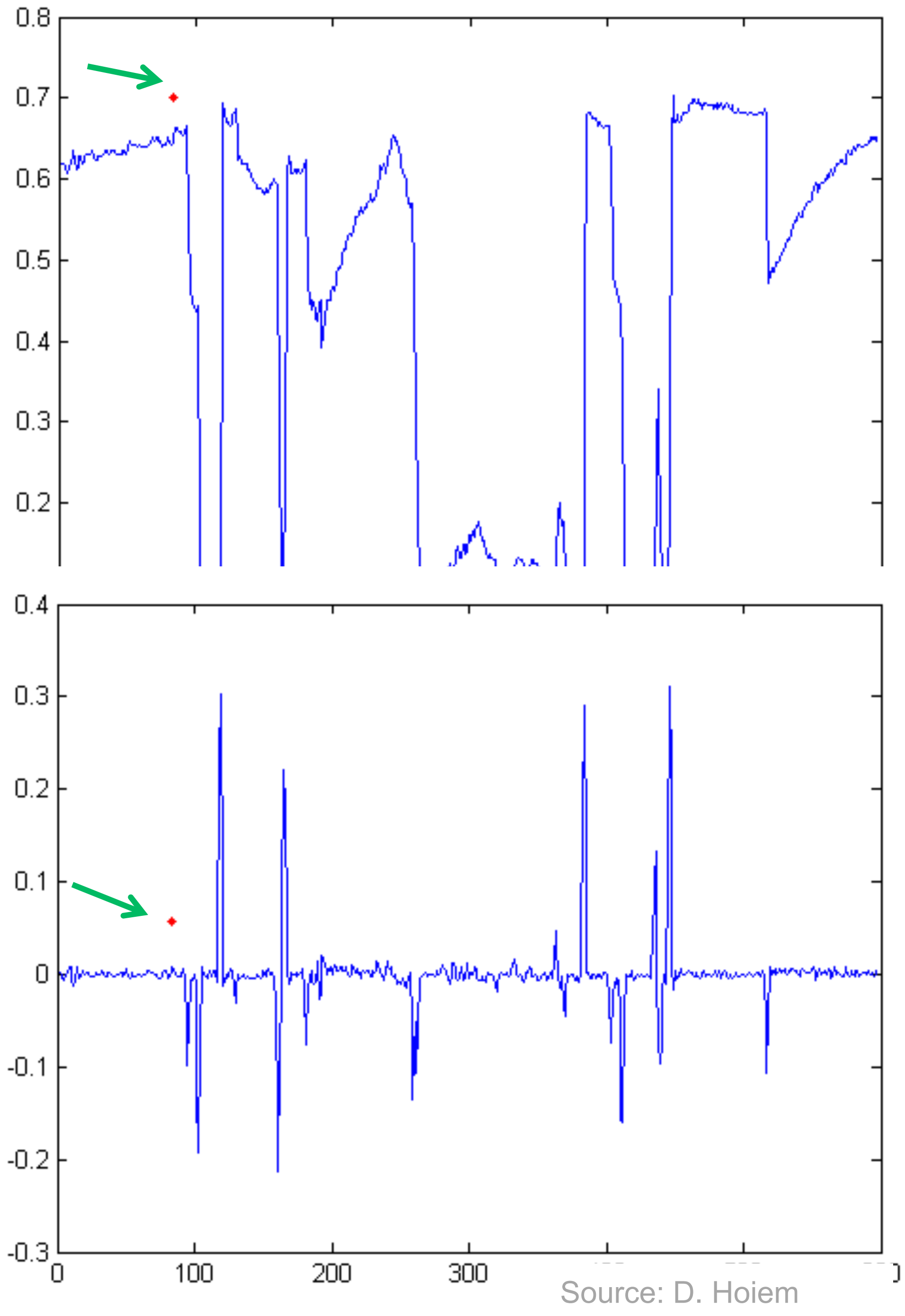
Roberts: $M_x = \begin{array}{|c|c|}\hline 0 & 1 \\ \hline -1 & 0 \\ \hline \end{array}$; $M_y = \begin{array}{|c|c|}\hline 1 & 0 \\ \hline 0 & -1 \\ \hline \end{array}$

```
>> My = fspecial('sobel');
>> outim = imfilter(double(im), My);
>> imagesc(outim);
>> colormap gray;
```



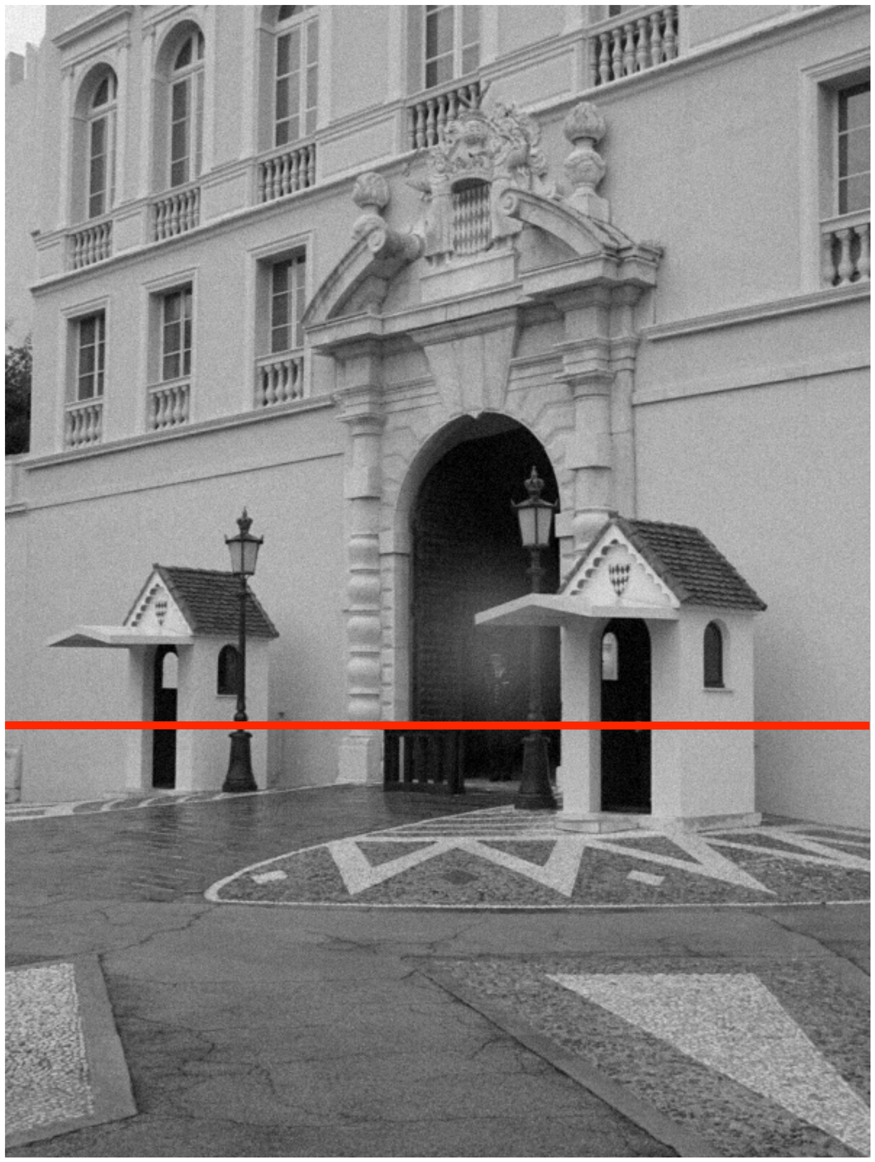
Intensity profile

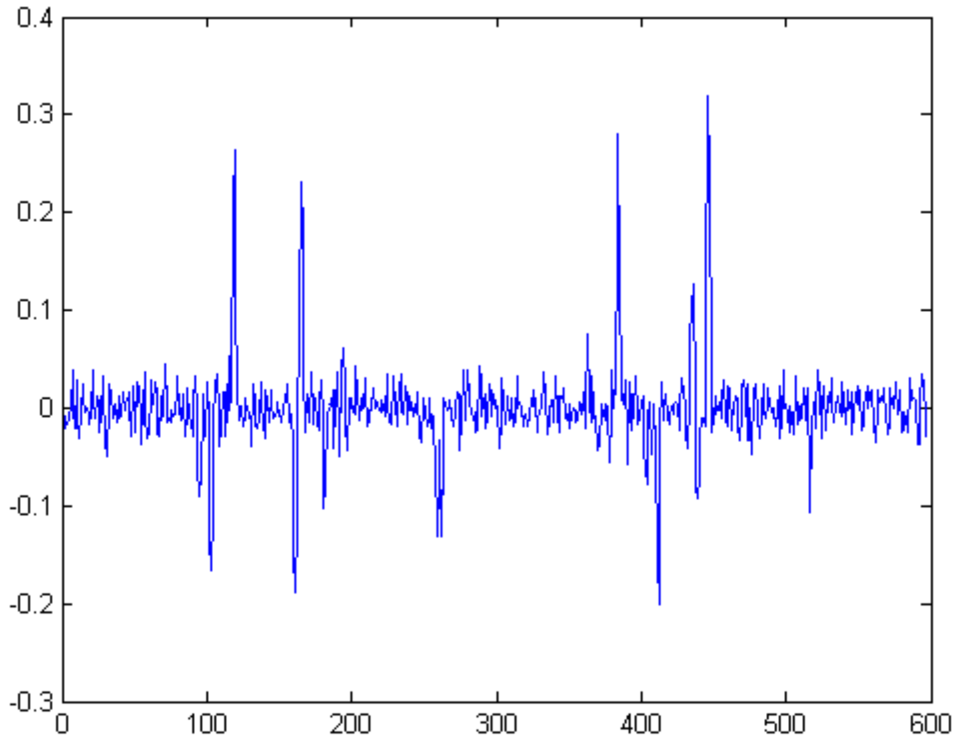




Source: D. Hoiem

With a little Gaussian noise





Gradient

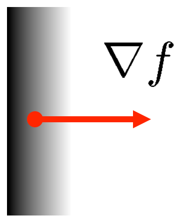
Source: D. Hoiem

Image gradient

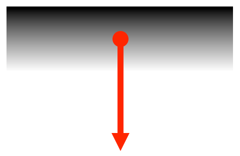
The gradient of an image:

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

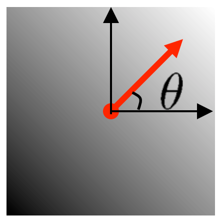
The gradient points in the direction of most rapid change in intensity



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

The gradient direction (orientation of edge normal) is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

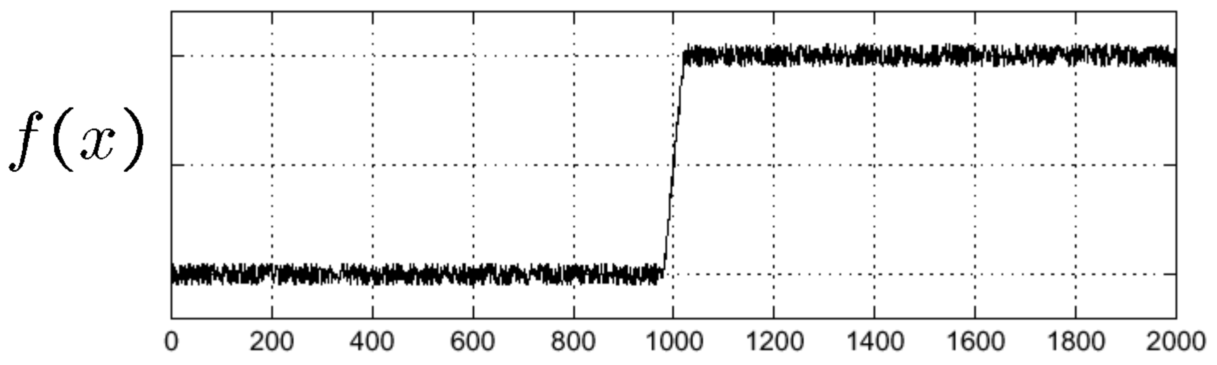
The *edge strength* is given by the gradient magnitude

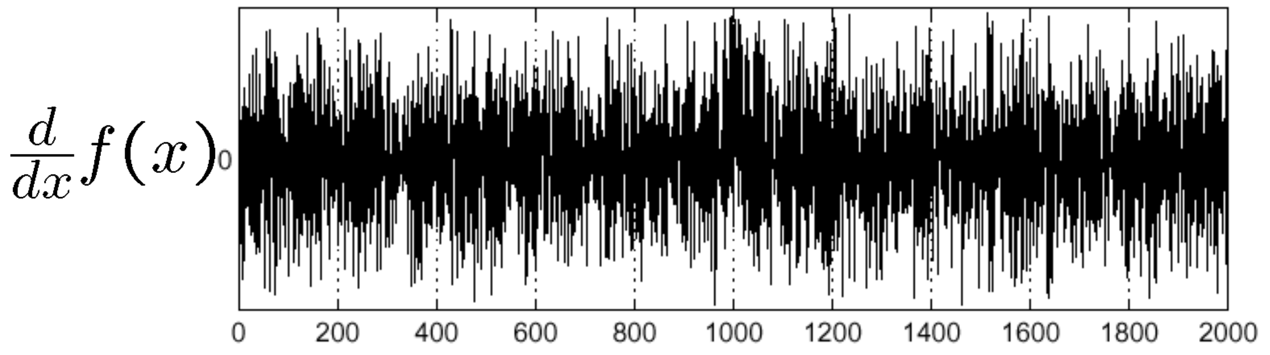
$$||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Slide credit S. Seitz

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal





Where is the edge?

Source: S. Seitz

Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Source: D. Forsyth

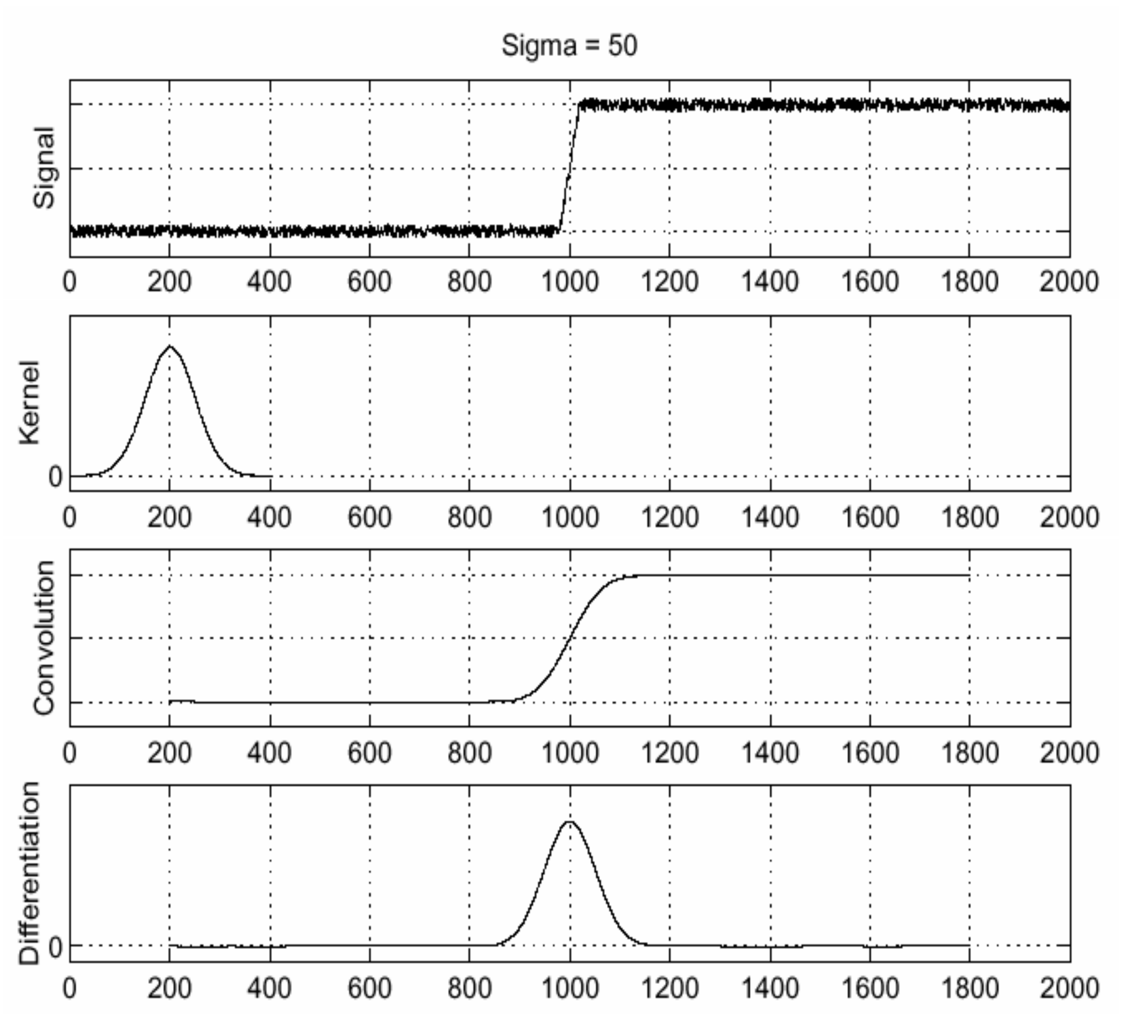
Solution: smooth first

f

g

$f * g$

$\frac{d}{dx}(f * g)$



- To find edges, look for peaks in $\frac{d}{dx}(f * g)$

Source: S. Seitz

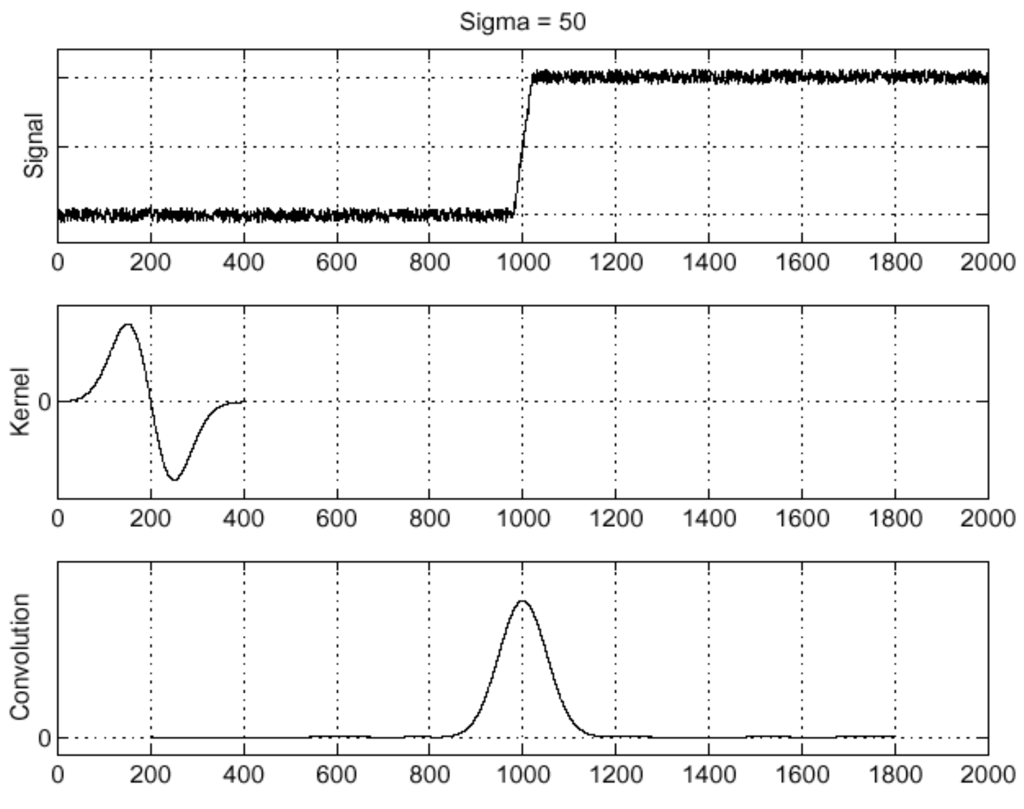
Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative: $$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$
- This saves us one operation:

f

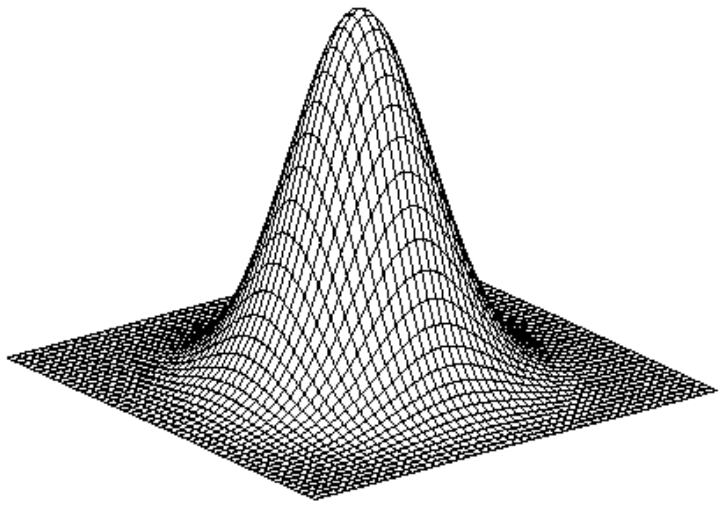
$\frac{d}{dx} g$

$f * \frac{d}{dx} g$

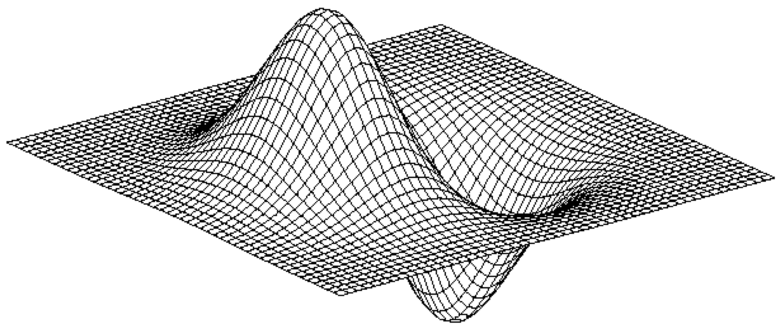


Source: S. Seitz

Derivative of Gaussian filter

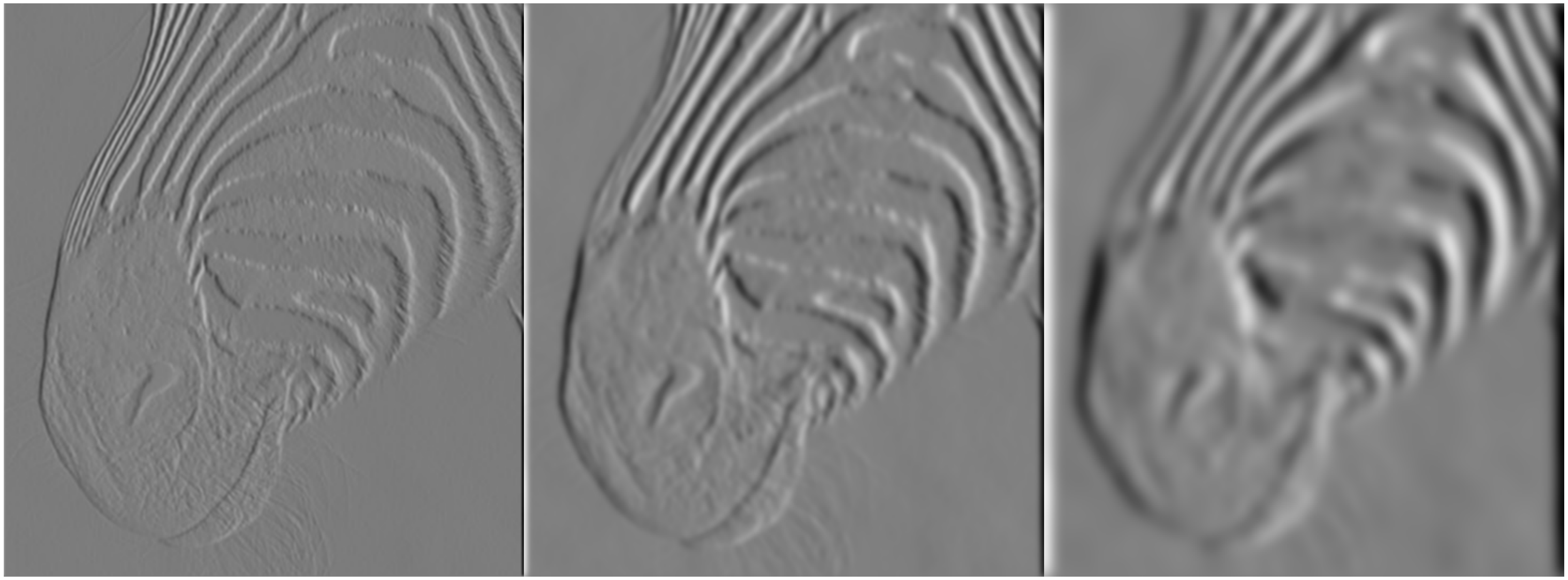


* [1 0 -1]/2 =



Example in Matlab

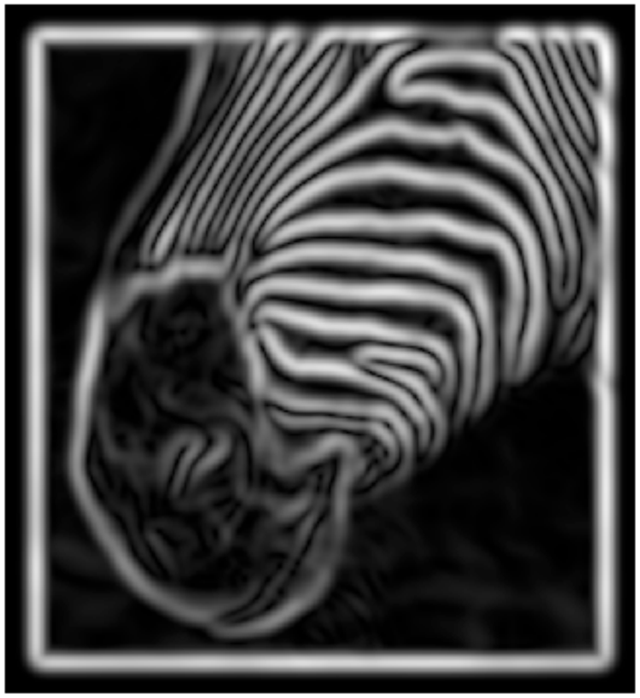
Tradeoff between smoothing and localization



- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth

Implementation issues



- The gradient magnitude is large along a thick "trail" or "ridge," so how do we identify the actual edge points?
- How do we link the edge points to form curves?

Source: D. Forsyth

Designing an edge detector

- Criteria for a good edge detector:
  - **Good detection:** the optimal detector should find all real edges, ignoring noise or other artifacts
  - **Good localization**
    - the edges detected must be as close as possible to the true edges
    - the detector must return one point only for each true edge point
- Cues of edge detection
  - Differences in color, intensity, or texture across the boundary
  - Continuity and closure
  - High-level knowledge

Source: L. Fei-Fei

Canny edge detector

- This is probably the most widely used edge detector in computer vision
- Theoretical model: step-edges corrupted by additive Gaussian noise
- 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

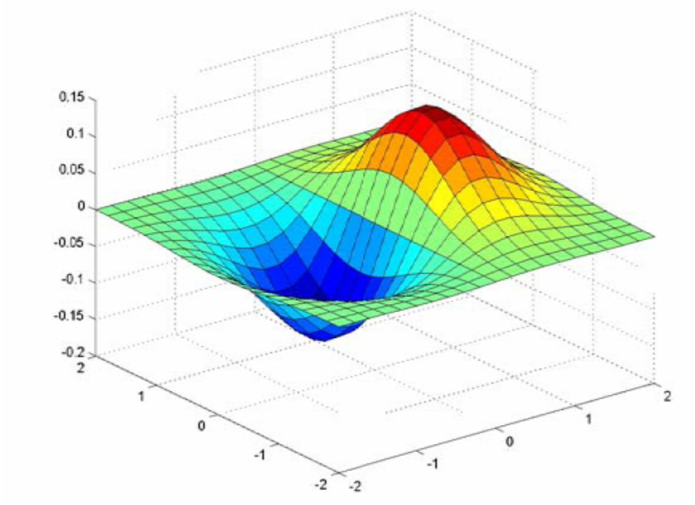
J. Canny, ***A Computational Approach To Edge Detection***, IEEE Trans. Pattern Analysis and Machine Intelligence, 8:679-714, 1986.

Source: L. Fei-Fei

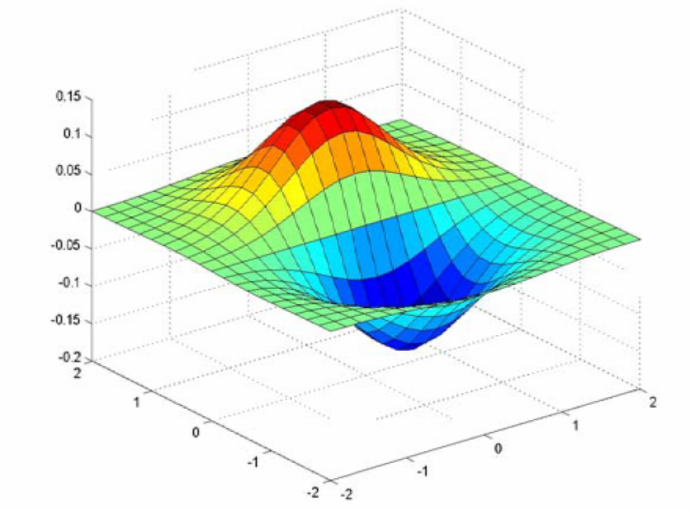
Example

original image (Lena)

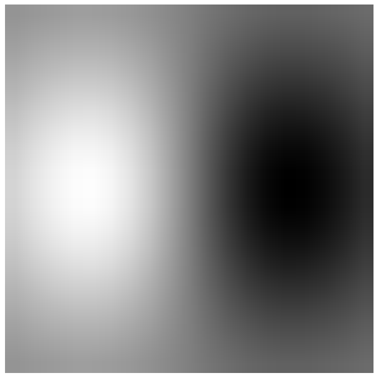
Derivative of Gaussian filter

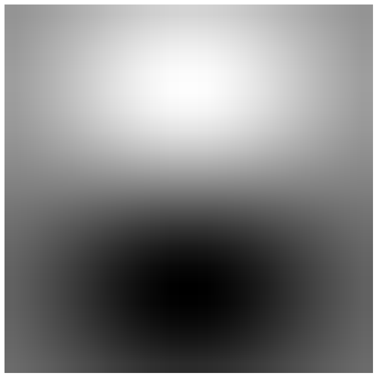


x-direction

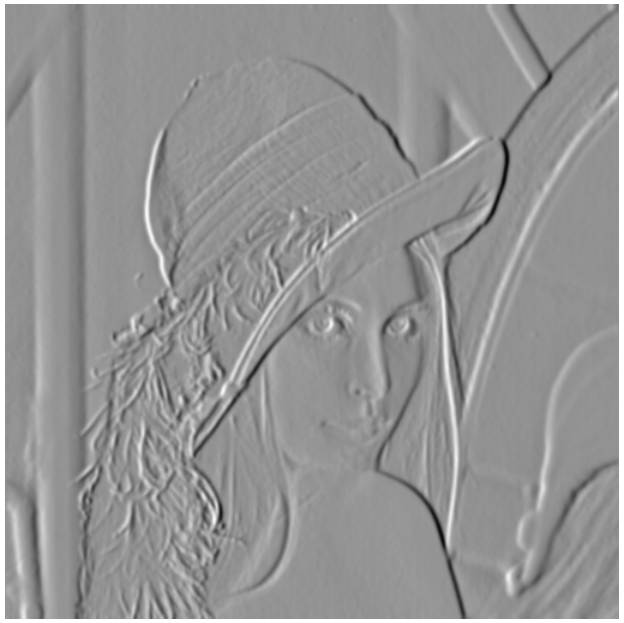


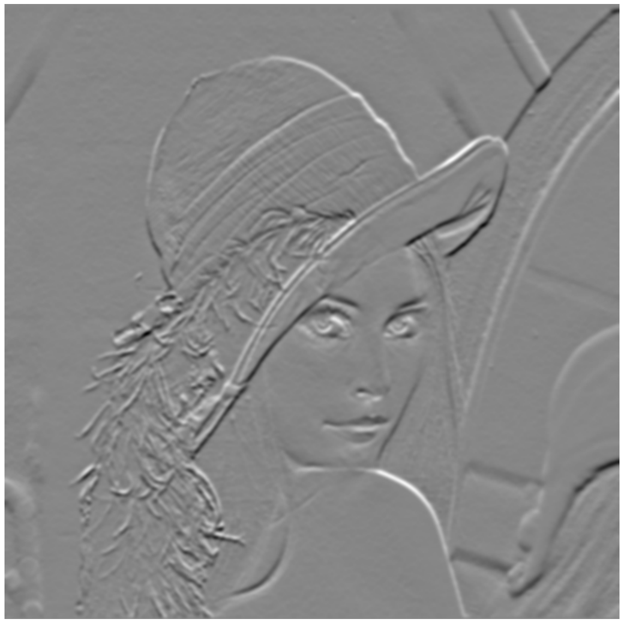
y-direction

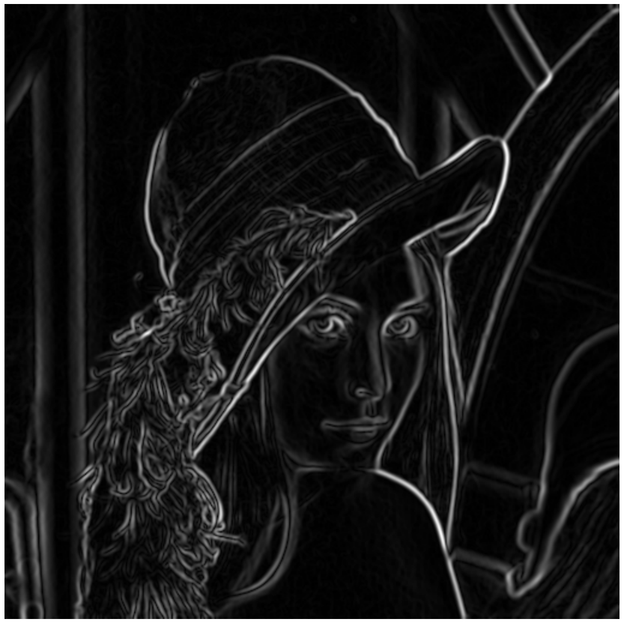




Compute Gradients (DoG)

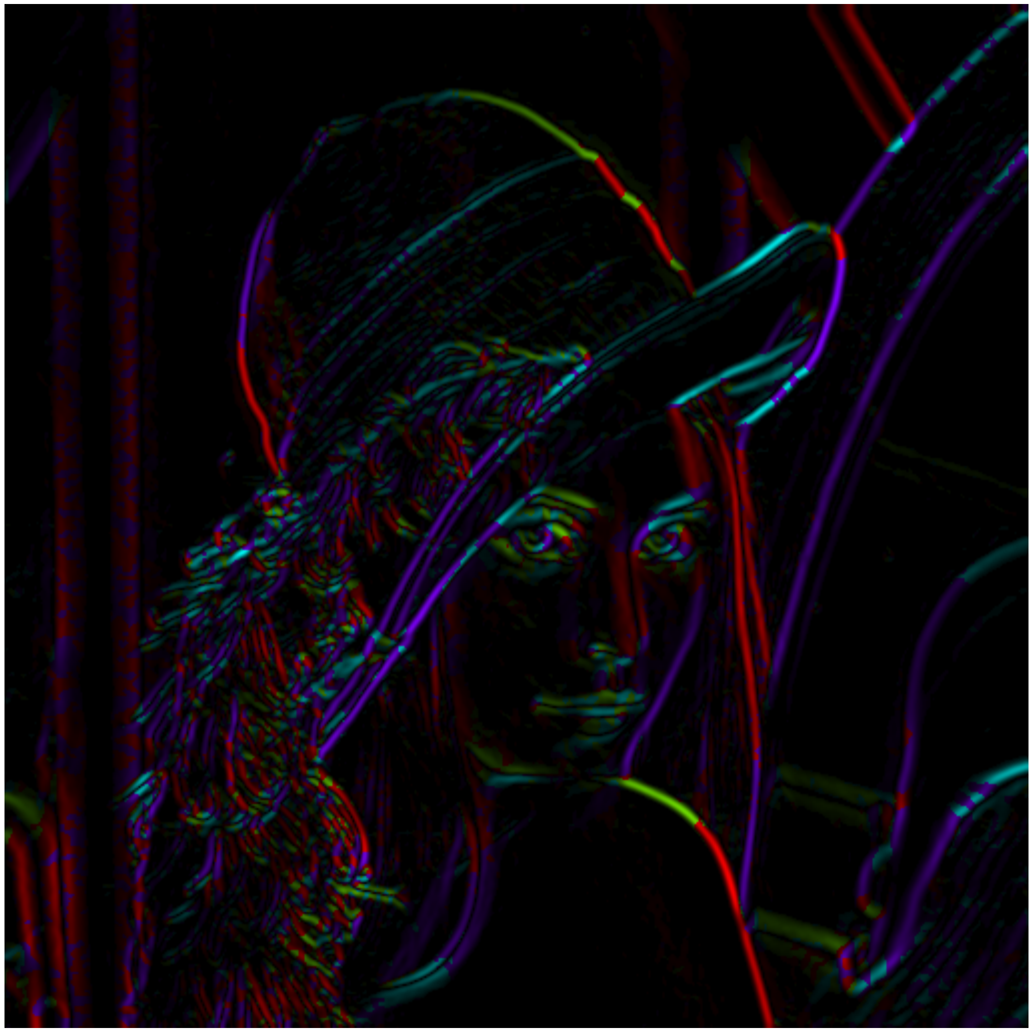

X-Derivative of Gaussian


Y-Derivative of Gaussian


Gradient Magnitude

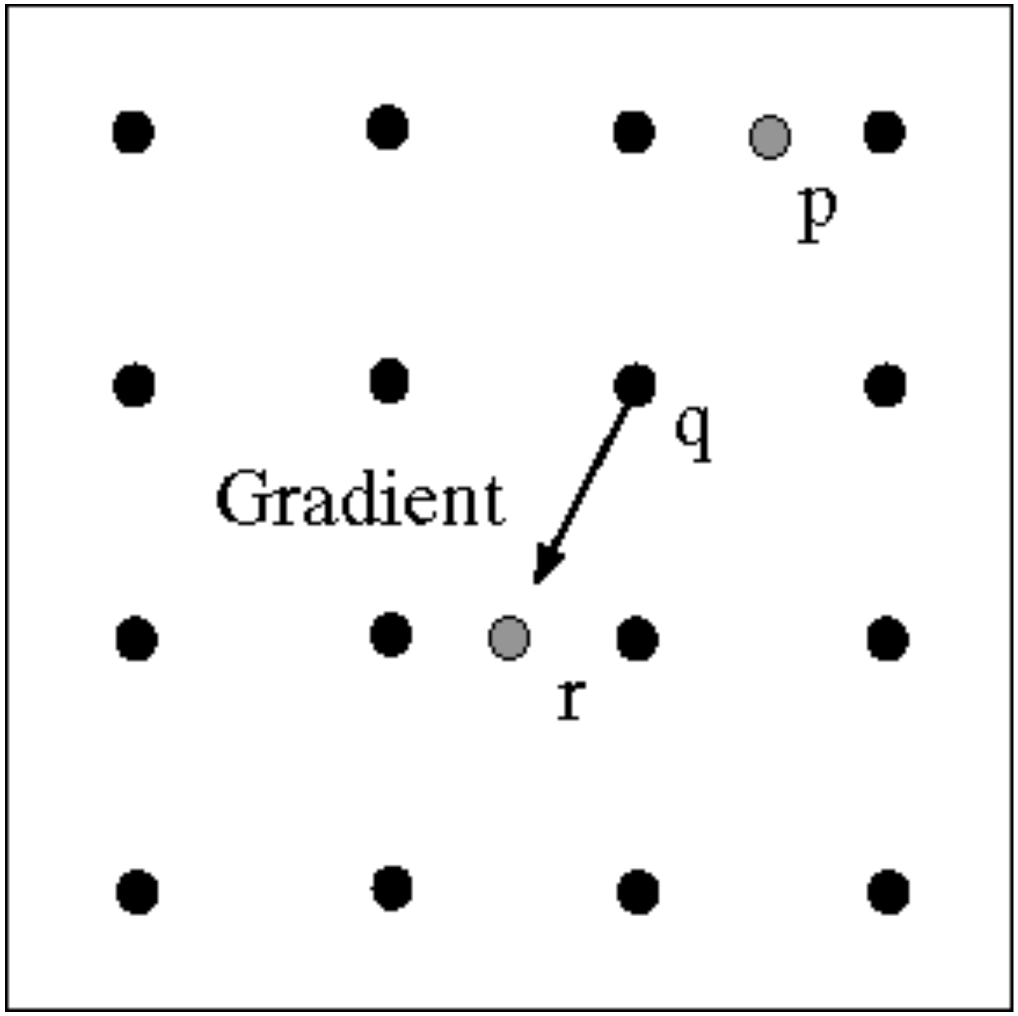
Get Orientation at Each Pixel

- Threshold at minimum level
- Get orientation

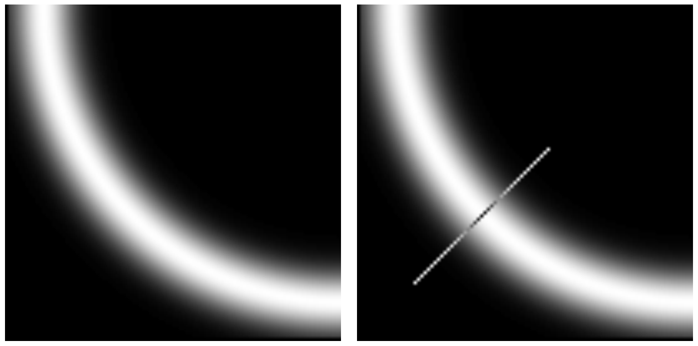


theta = atan2(gy, gx)

Non-maximum suppression for each orientation

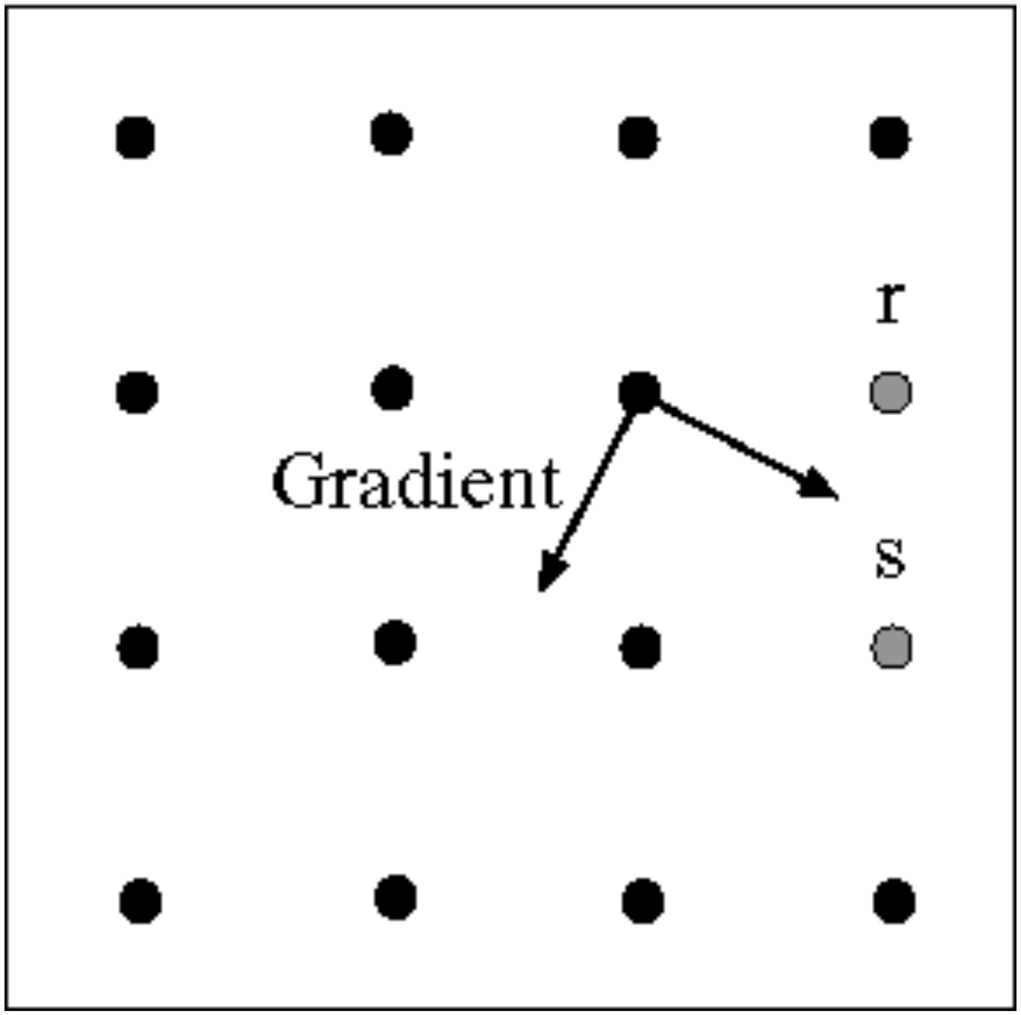


At q, we have a maximum if the value is larger than those at both p and at r. Interpolate to get these values.

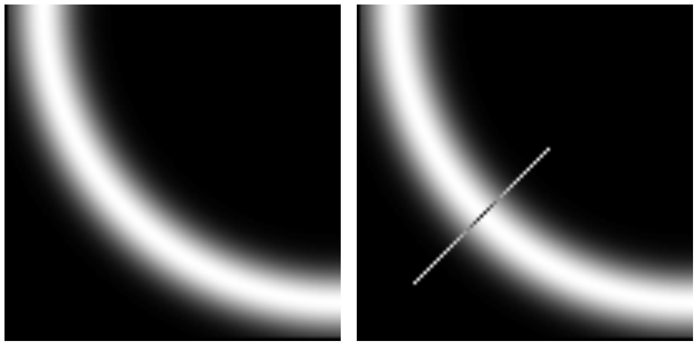


Source: D. Forsyth

Edge linking

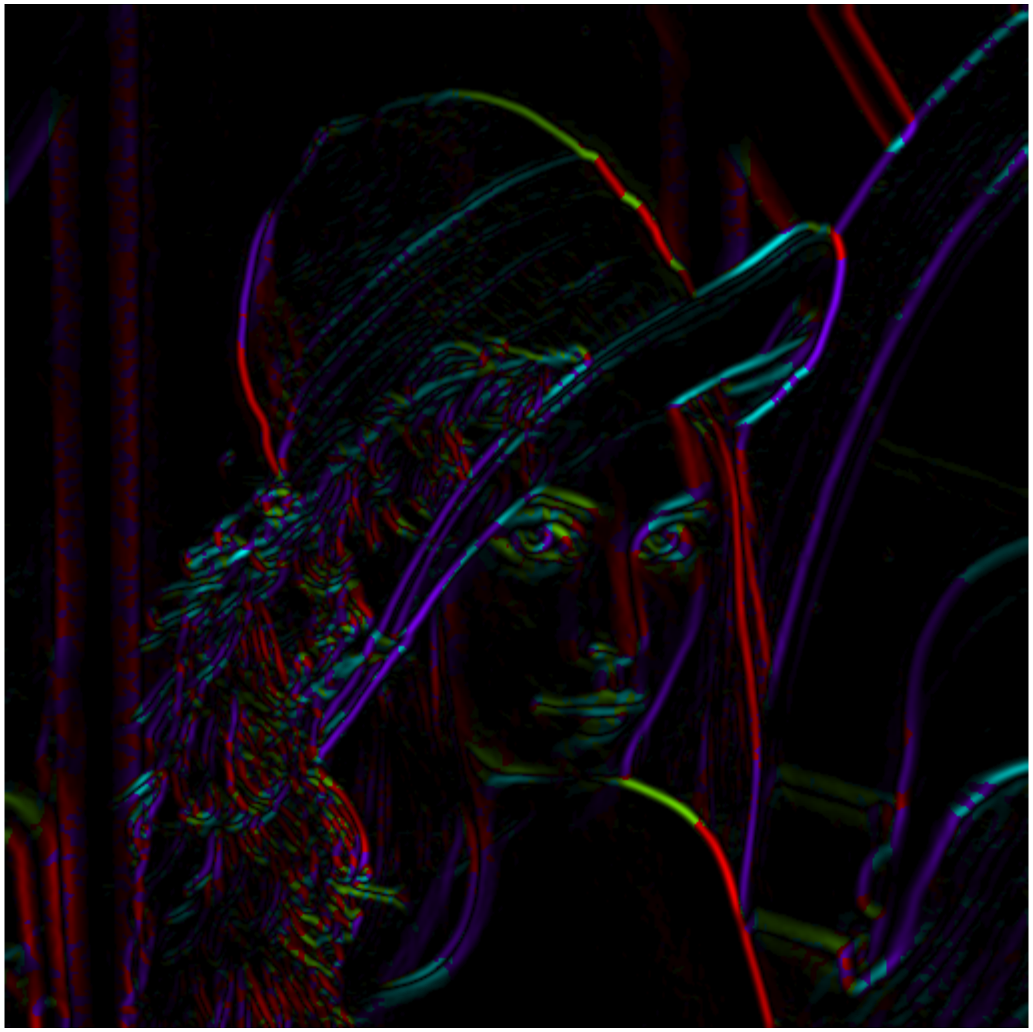


Assume the marked point is an edge point. Then we construct the tangent to the edge curve (which is normal to the gradient at that point) and use this to predict the next points (here either r or s).

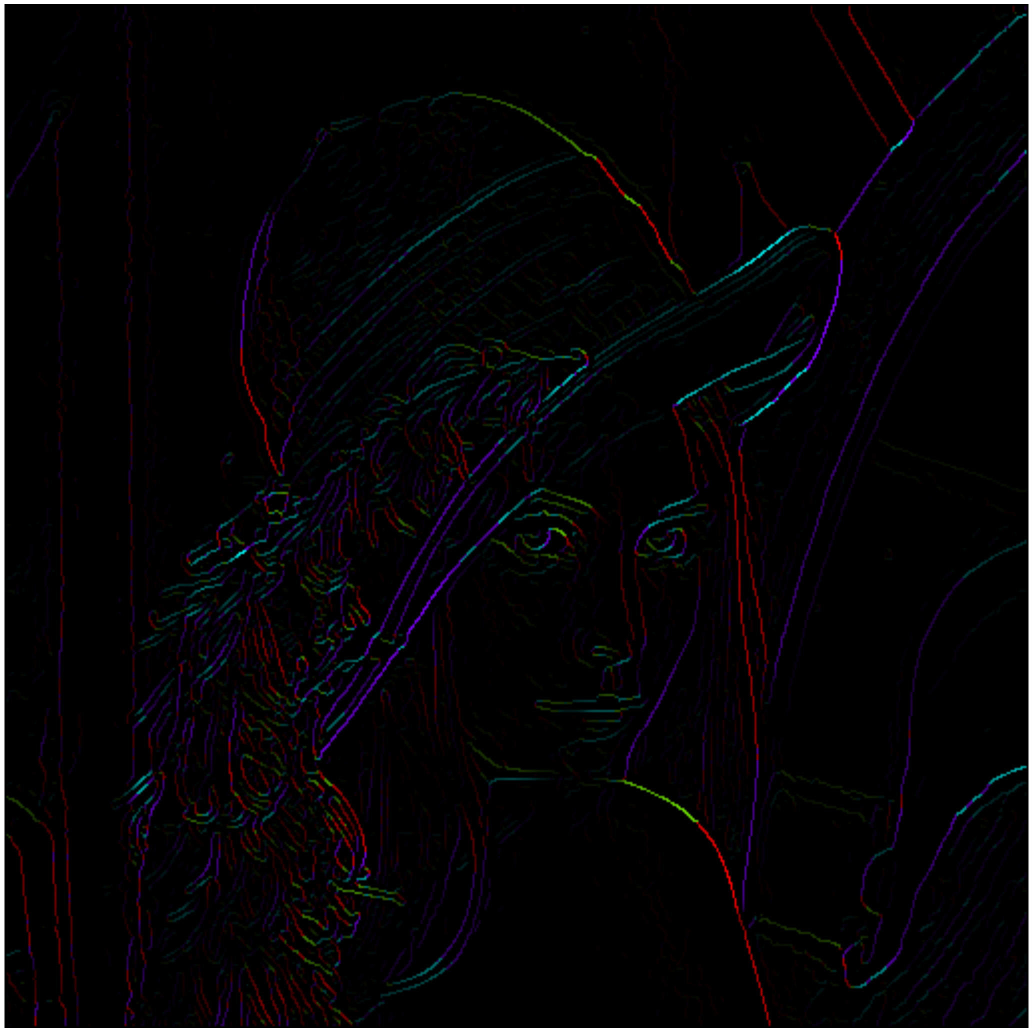


Source: D. Forsyth

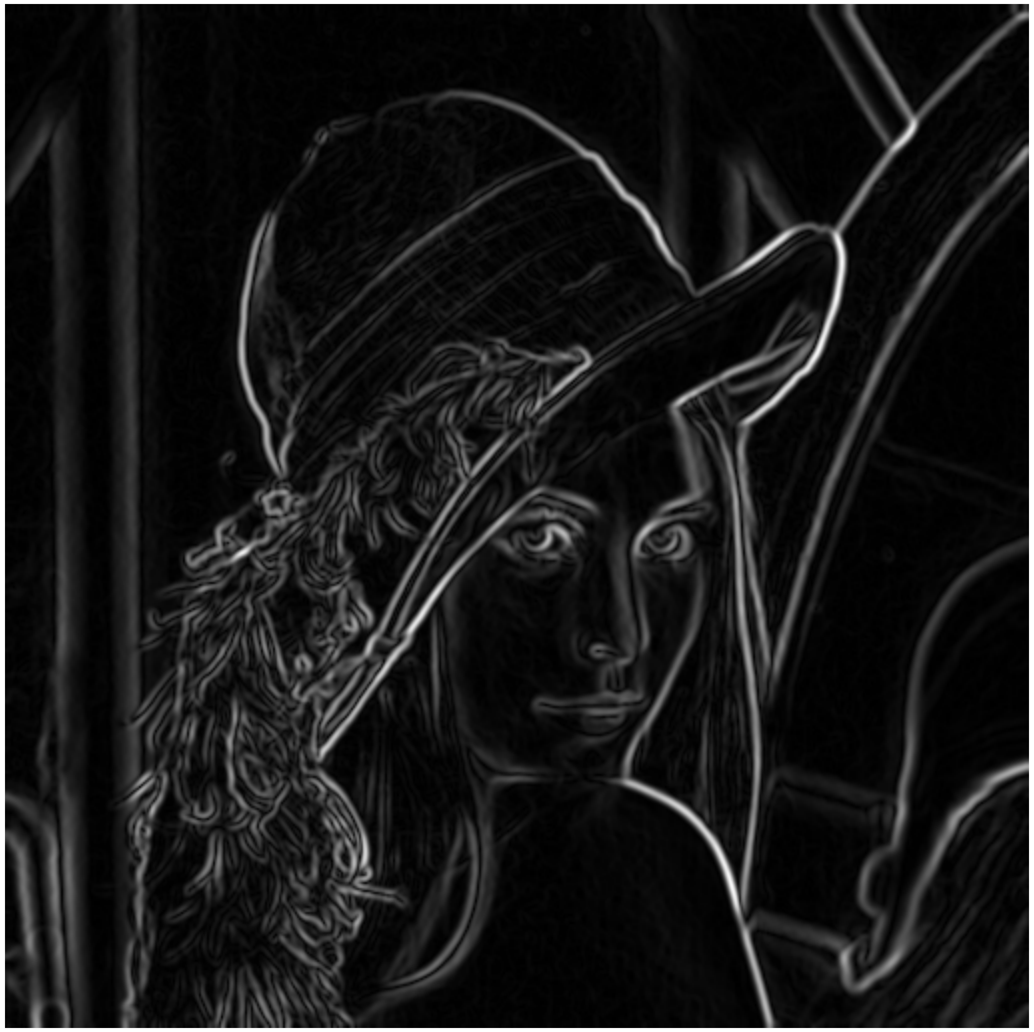
Before Non-max Suppression



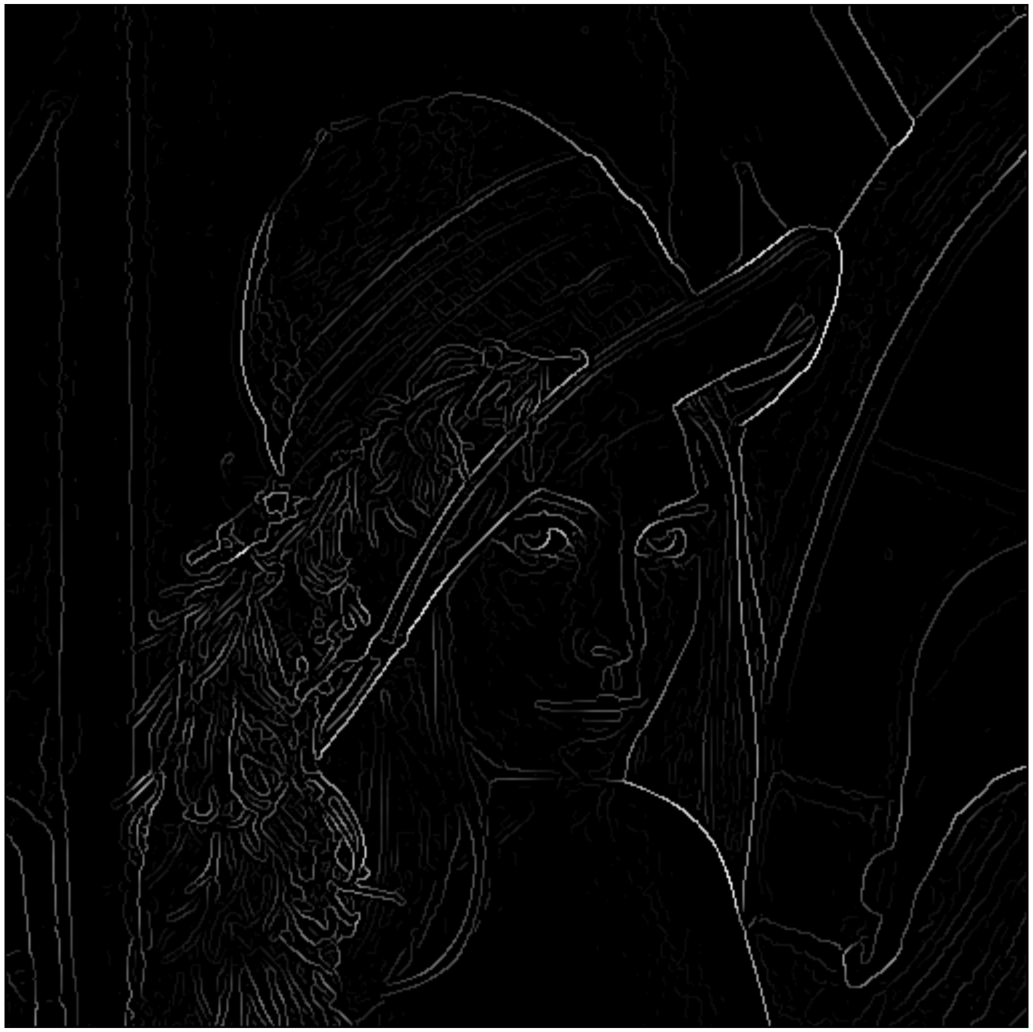
After non-max suppression



Before Non-max Suppression

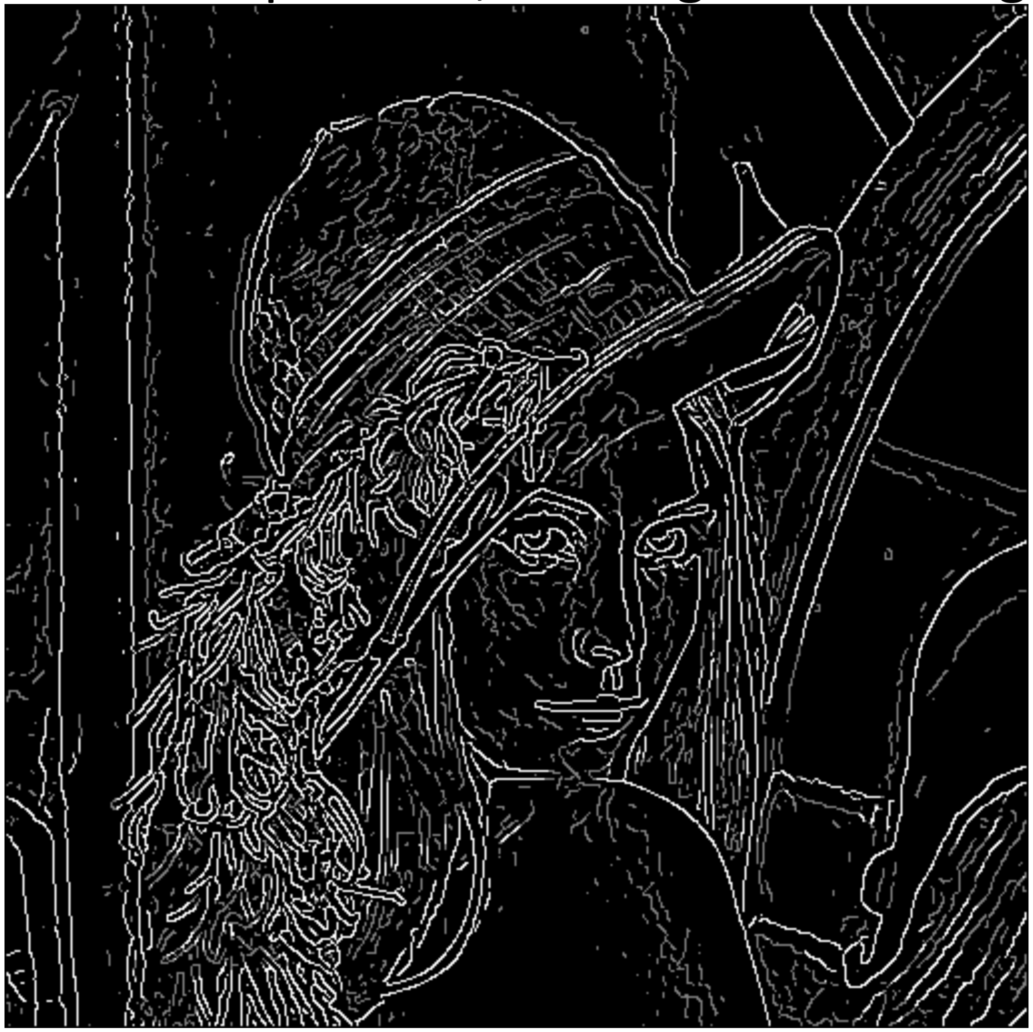


After non-max suppression



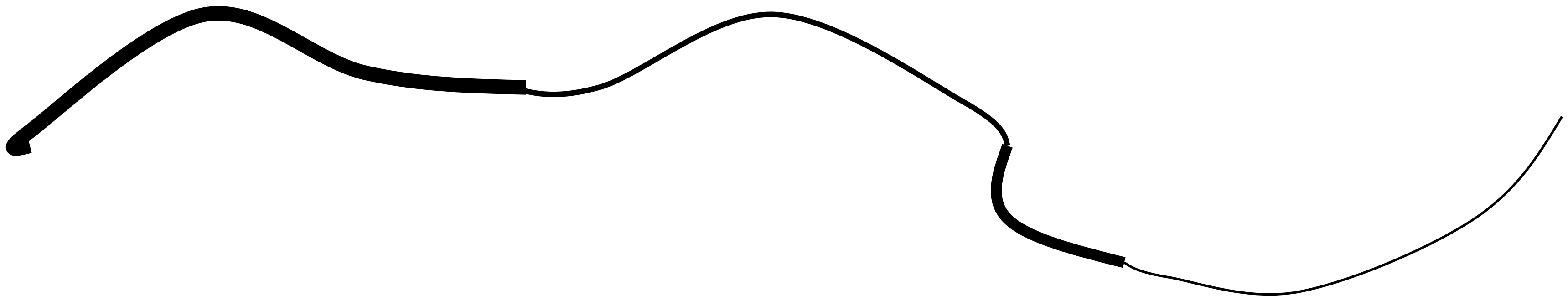
Hysteresis thresholding

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



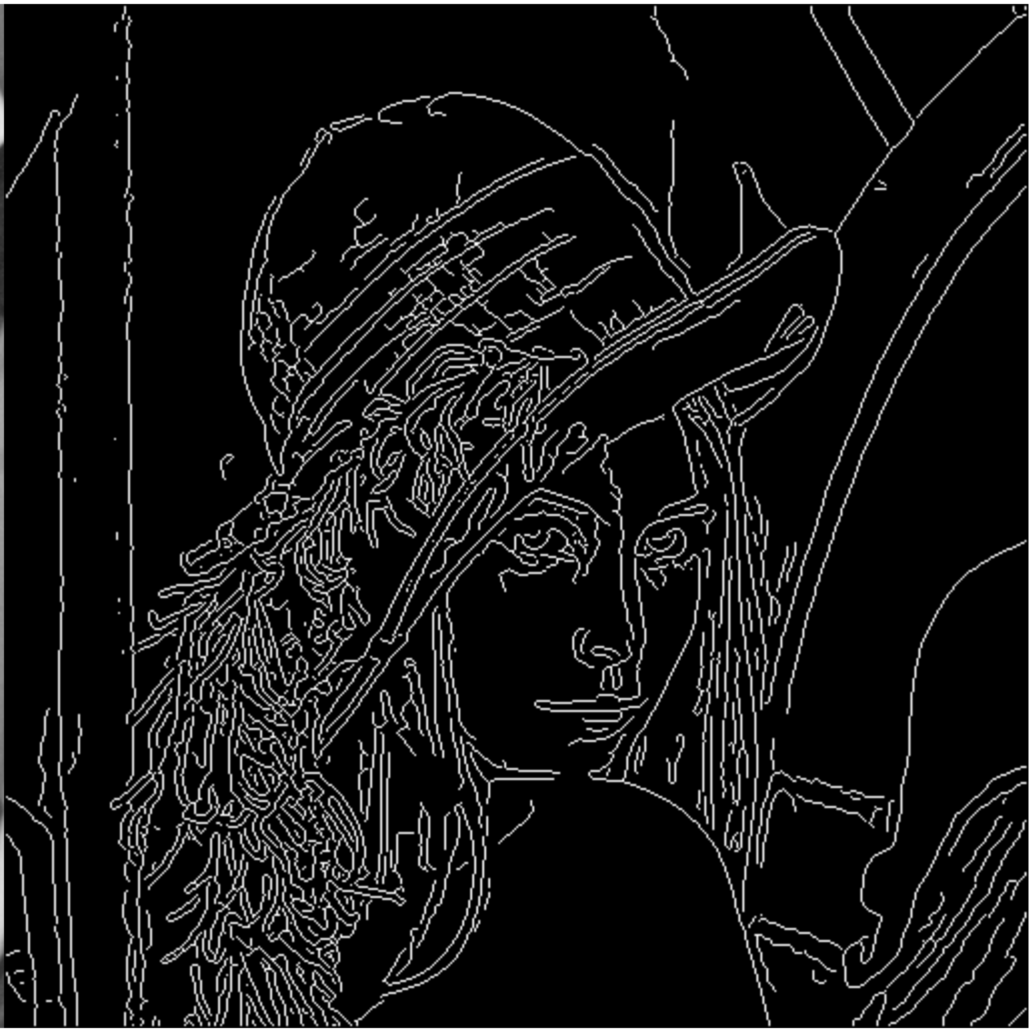
Hysteresis thresholding

- Check that maximum value of gradient value is sufficiently large
  - drop-outs? use **hysteresis**
    - use a high threshold to start edge curves and a low threshold to continue them.



Source: S. Seitz

Final Canny Edges



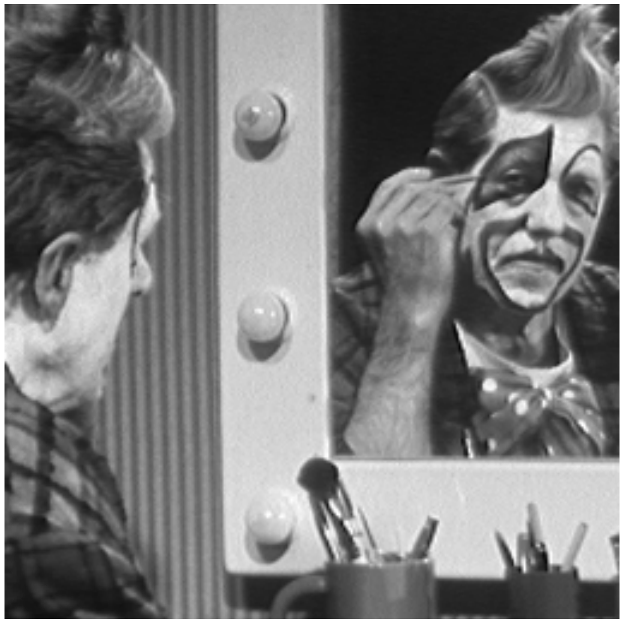
Canny edge detector

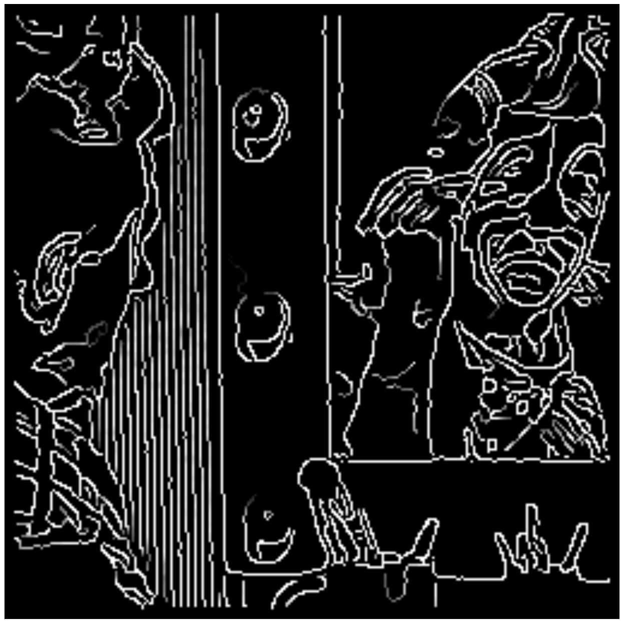
1. Filter image with x, y derivatives of Gaussian
2. Find magnitude and orientation of gradient
3. Non-maximum suppression:
   - Thin multi-pixel wide "ridges" down to single pixel width
4. Thresholding and linking (hysteresis):
   - Define two thresholds: low and high
   - Use the high threshold to start edge curves and the low threshold to continue them

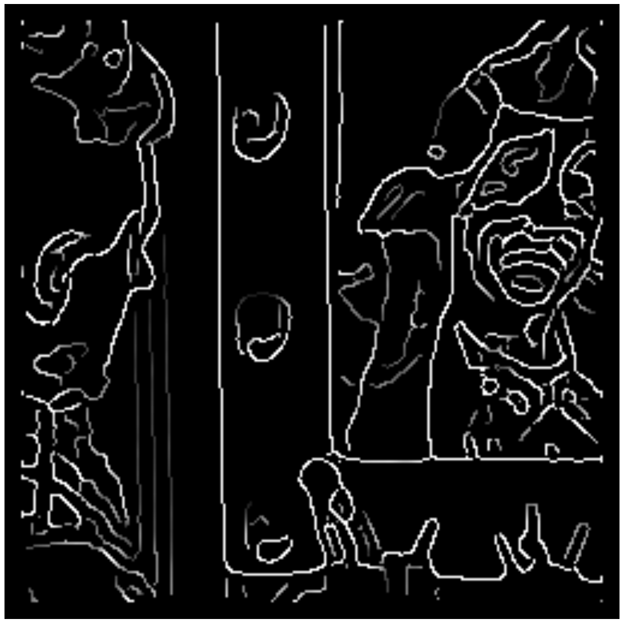
- MATLAB: edge(image, 'canny')
- OpenCV: edges = cv2.Canny(img, th1, th2)

Source: D. Lowe, L. Fei-Fei

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

Displaying Edges

Quiver function in Matlab and SciPy (Matplotlib)

```
> A = imread(MyImage);
> A1 = BiSmooth(A,8);
> [Gx,Gy] = Gradient(A1);
> m1=NMS2(Gx,Gy,3);
> m1 = Hysteresis(m1,3);
> [I,J] = find(m1>0);  % row and column indexes
> In = find(m1>0);     % indexes in a vector form
> [M,N] = size(m1);
> quiver(J,M-I+1,Gx(In),Gy(In),1)
```