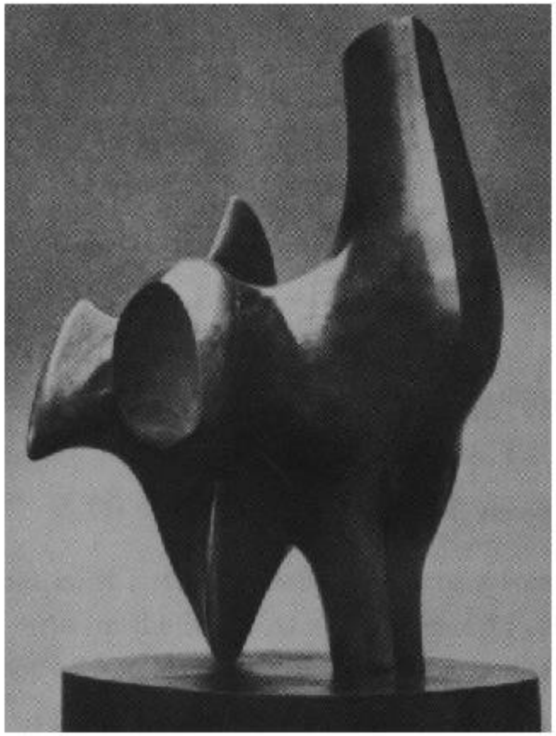
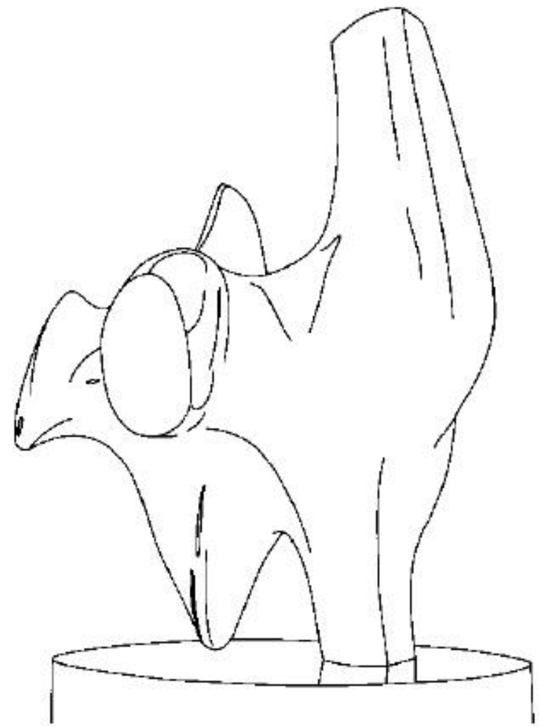
Edge detection

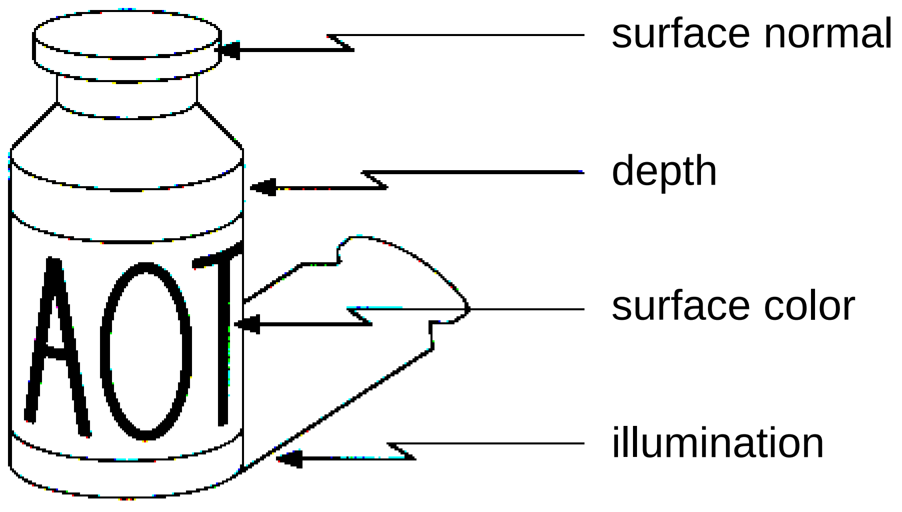
- Convert a 2D image into a set of curves
  - Extracts salient features of the scene
  - More compact than pixels

Edge detection

- **Goal:** Identify visual changes (discontinuities) in an image.
- Intuitively, semantic information is encoded in edges.
- What are some ‘causes’ of visual edges?

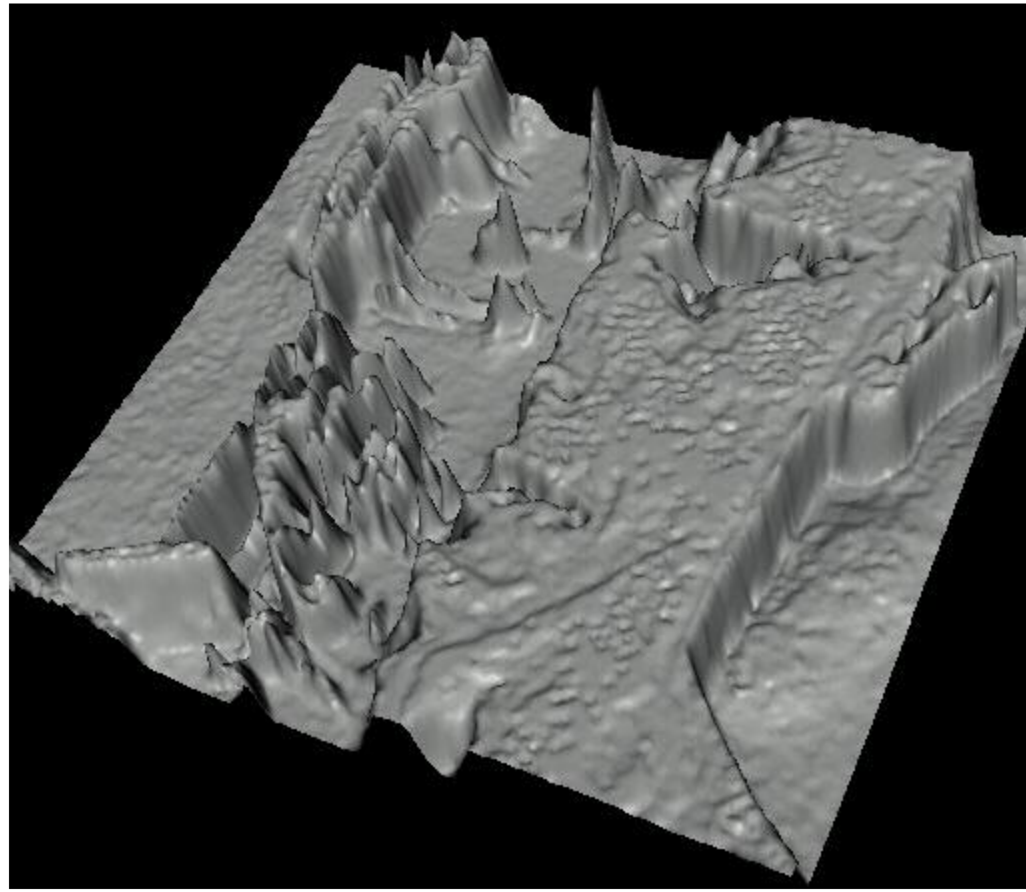
Source: D. Lowe

Origin of Edges



- Edges are caused by a variety of factors

Images as functions…

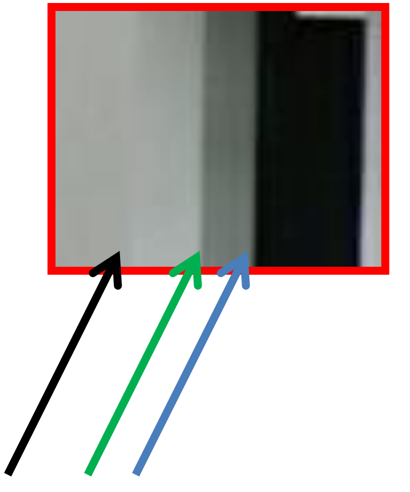


- Edges look like steep cliffs

Closeup of edges

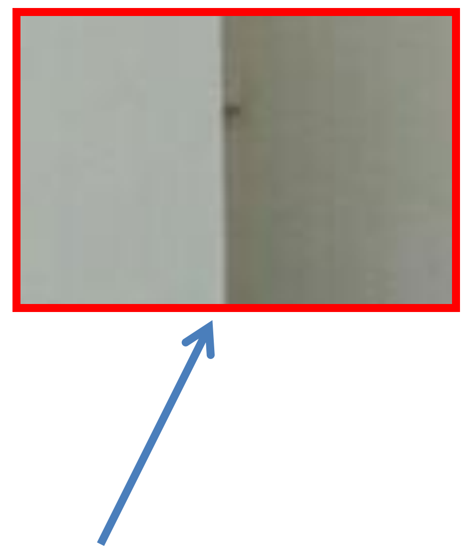
Source: D. Hoiem

Closeup of edges



Source: D. Hoiem

Closeup of edges



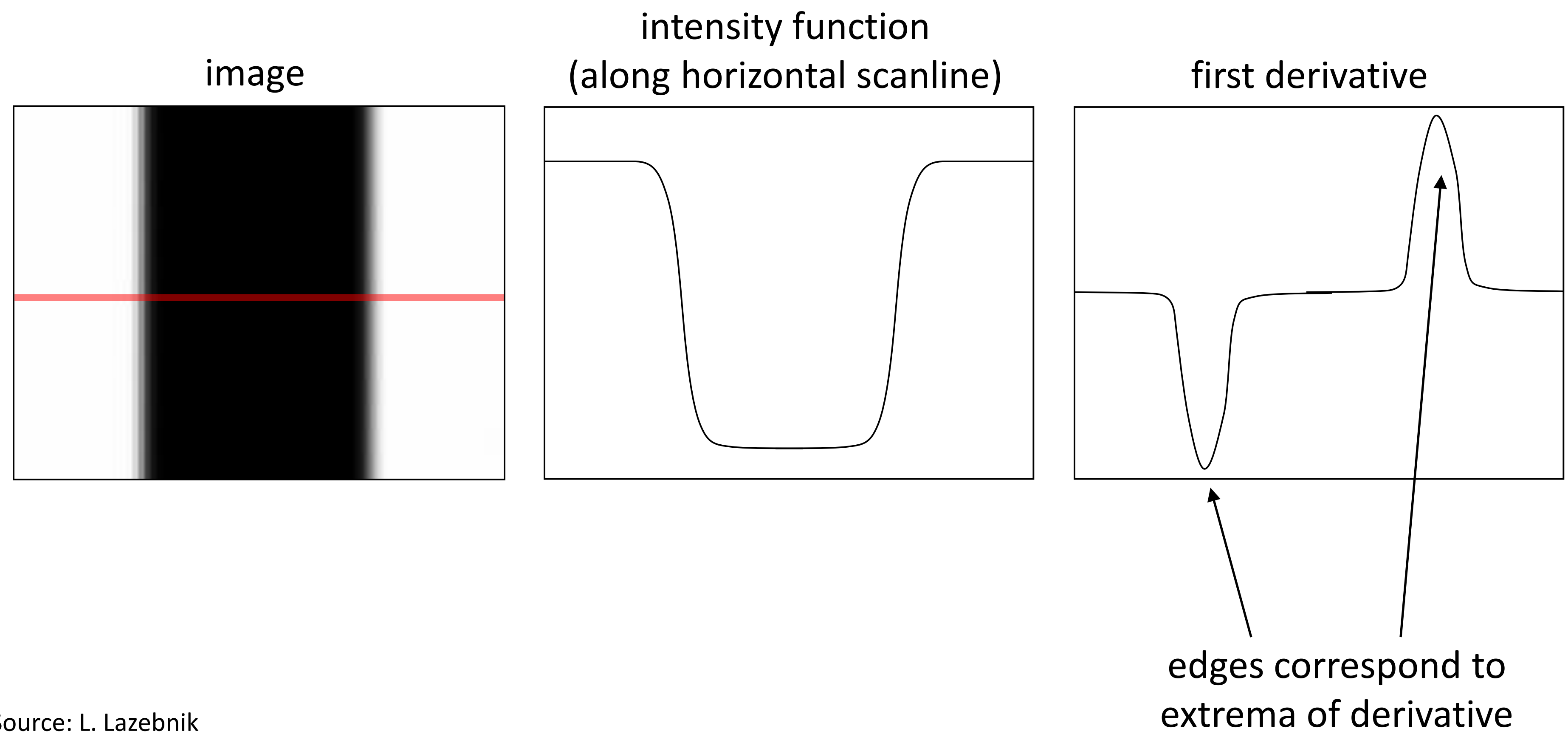
Source: D. Hoiem

Closeup of edges

Source: D. Hoiem

Characterizing edges

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



Source: L. Lazebnik

Image derivatives

- How can we differentiate a *digital* image F[x,y]?
  - Option 1: reconstruct a continuous image, *f,* then compute the derivative
  - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x,y] \approx F[x+1,y] - F[x,y]$$

How would you implement this as a linear filter?

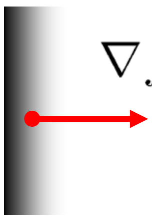
$$\frac{\partial f}{\partial x}: \underset{H_x}{\begin{array}{|c|c|c|}\hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \end{array}} \qquad\qquad \frac{\partial f}{\partial y}: \underset{H_y}{\begin{array}{|c|c|c|}\hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \quad & \quad & \quad \\ \hline \end{array}}$$

Source: S. Seitz

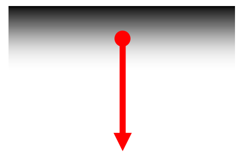
Image gradient

- The *gradient* of an image: $\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$

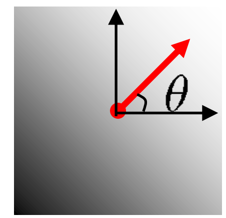
The gradient points in the direction of most rapid increase in intensity



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



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



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

The *edge strength* is given by the gradient magnitude:

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

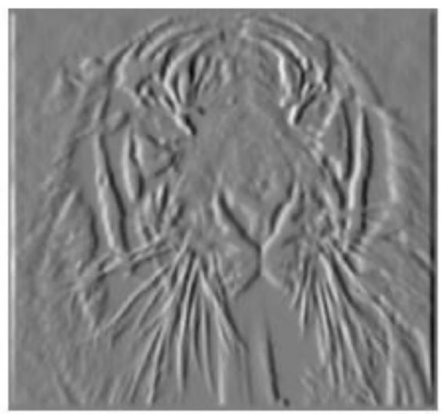
The gradient direction is given by:

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

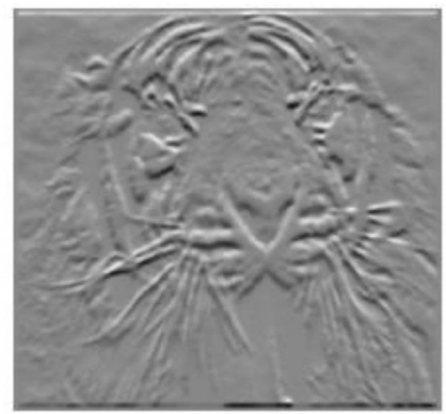
Source: Steve Seitz

Image gradient

Image f



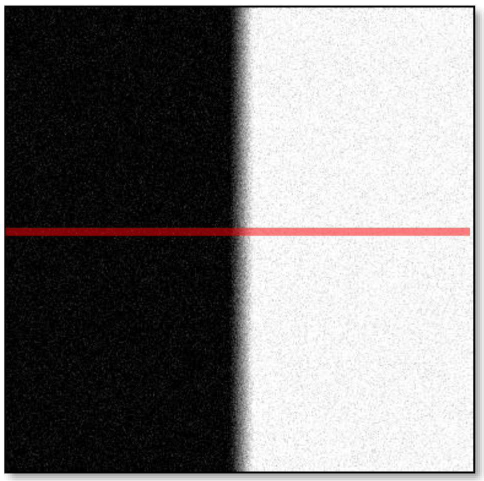
$$\frac{\partial f(x,y)}{\partial x}$$



$$\frac{\partial f(x,y)}{\partial y}$$

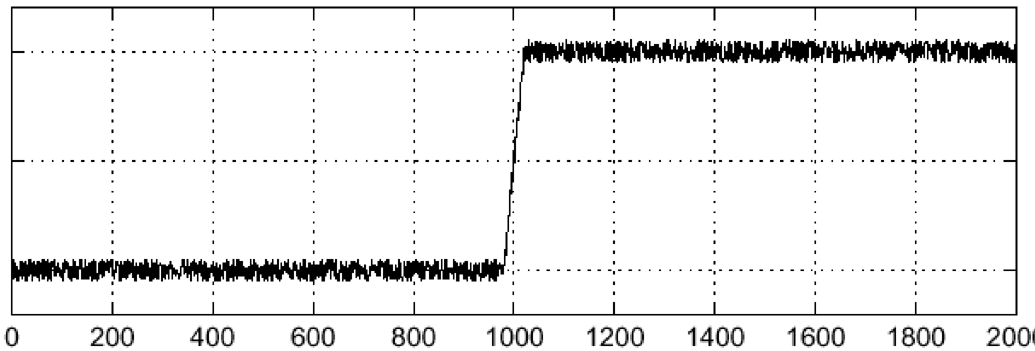
Source: L. Lazebnik

Effects of noise

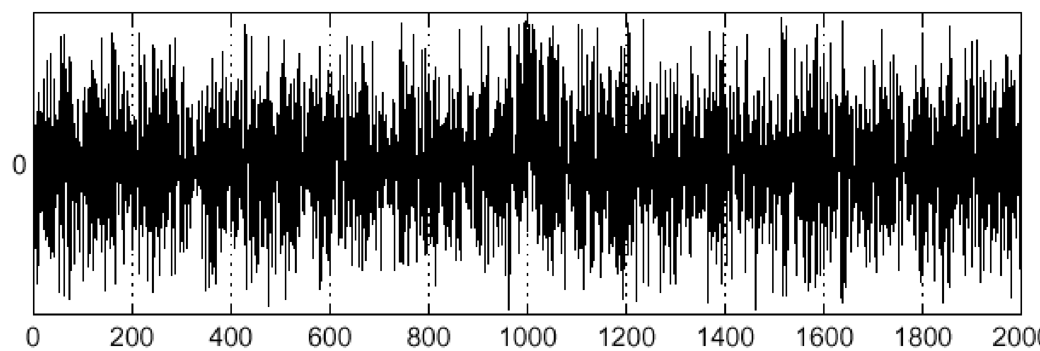


Noisy input image

$f(x)$



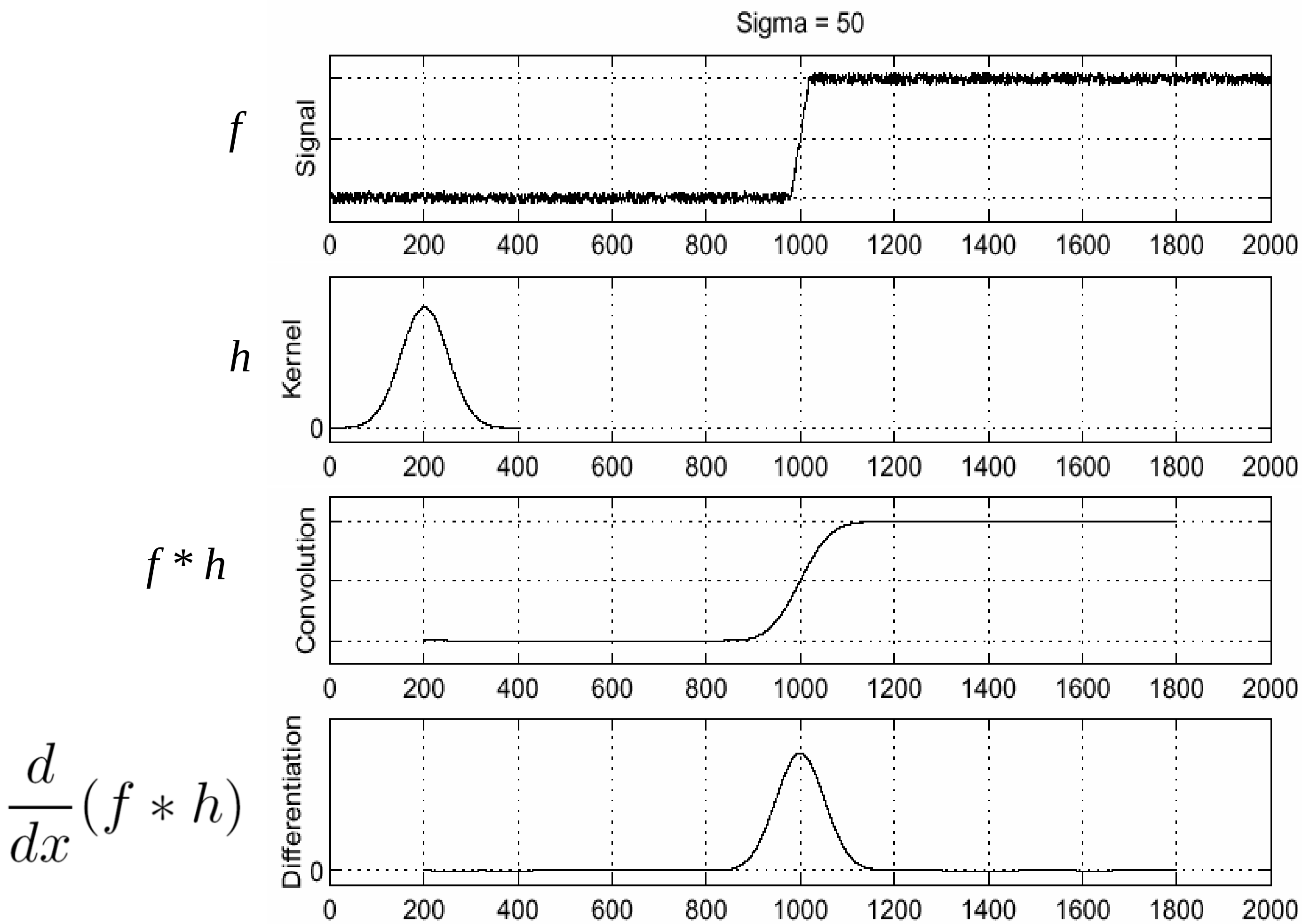
$\frac{d}{dx}f(x)$



Where is the edge?

Source: S. Seitz

Solution: smooth first



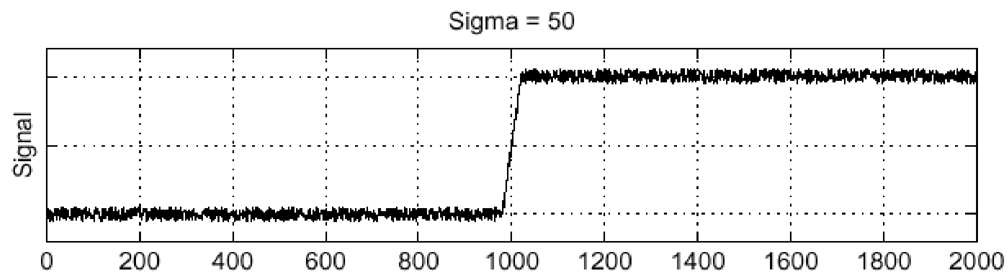
To find edges, look for peaks in $\frac{d}{dx}(f * h)$

Source: S. Seitz

Associative property of convolution

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

f

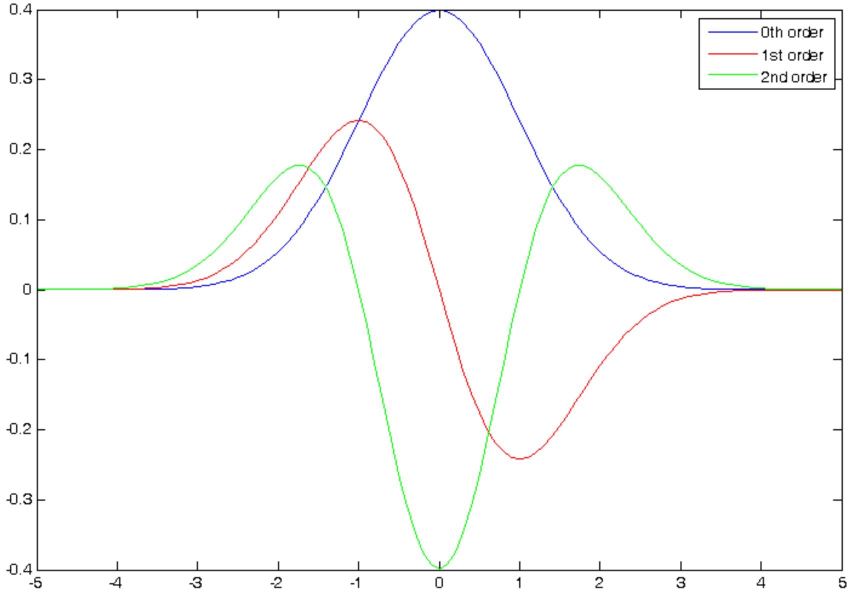


Source: S. Seitz

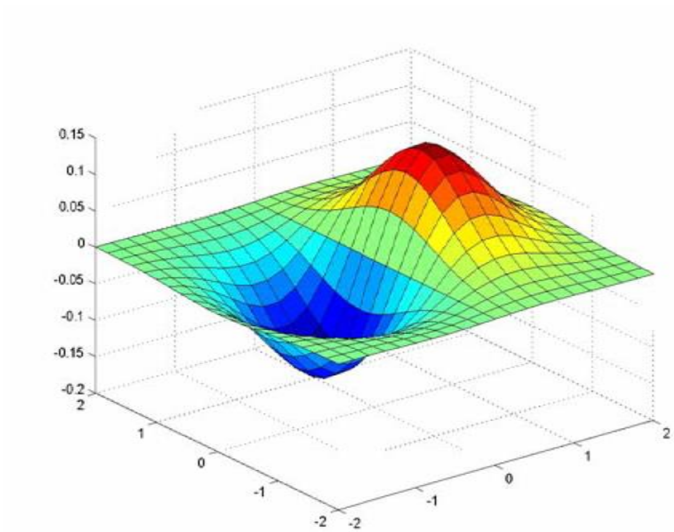
The 1D Gaussian and its derivatives

$$G_\sigma(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

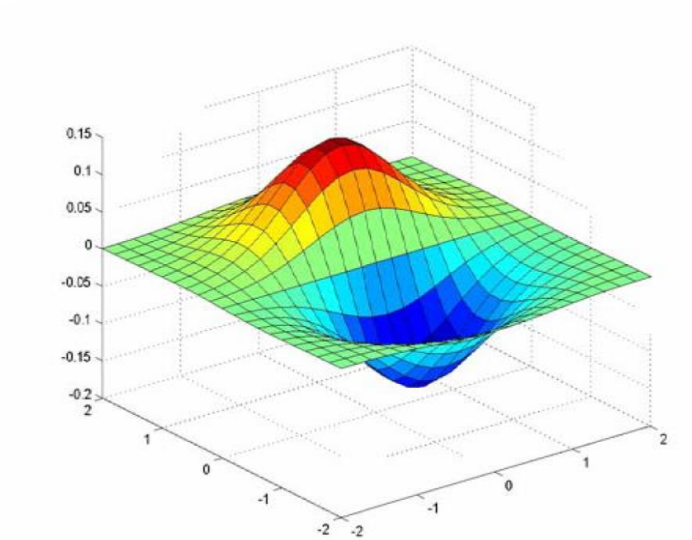
$$G'_\sigma(x) = \frac{d}{dx} G_\sigma(x) = -\frac{1}{\sigma}\left(\frac{x}{\sigma}\right) G_\sigma(x)$$



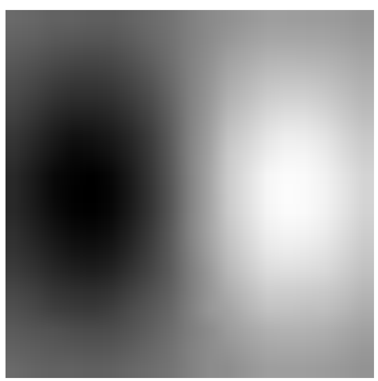
Derivative of Gaussian filter

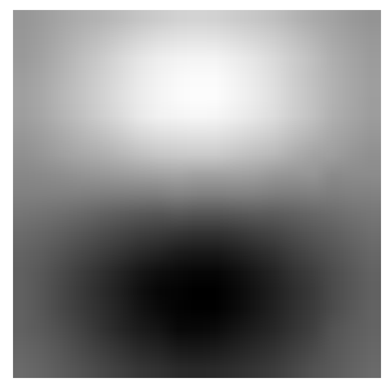


x-direction



y-direction





The Sobel operator

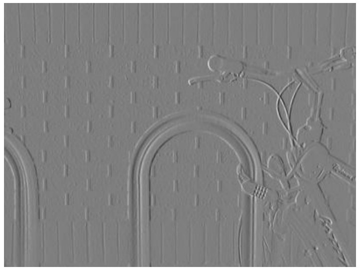
- Common approximation of derivative of Gaussian

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \qquad \frac{1}{8}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

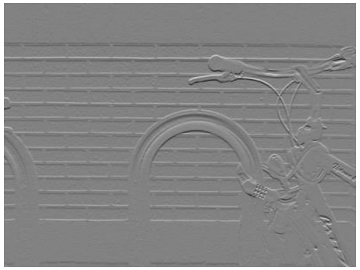
$$s_x \qquad\qquad s_y$$

- The standard defn. of the Sobel operator omits the 1/8 term
  - doesn’t make a difference for edge detection
  - the 1/8 term **is** needed to get the right gradient magnitude

Sobel operator: example



x-gradient



y-gradient

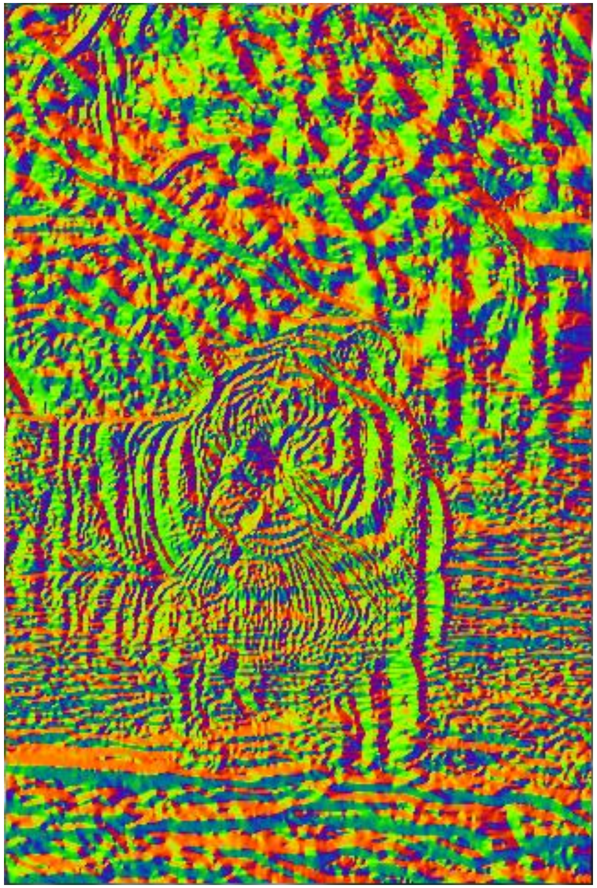
gradient magnitude

Source: Wikipedia

Sobel operator

Original

Magnitude



Orientation

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

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)

Finding edges

gradient magnitude

Finding edges

where is the edge?

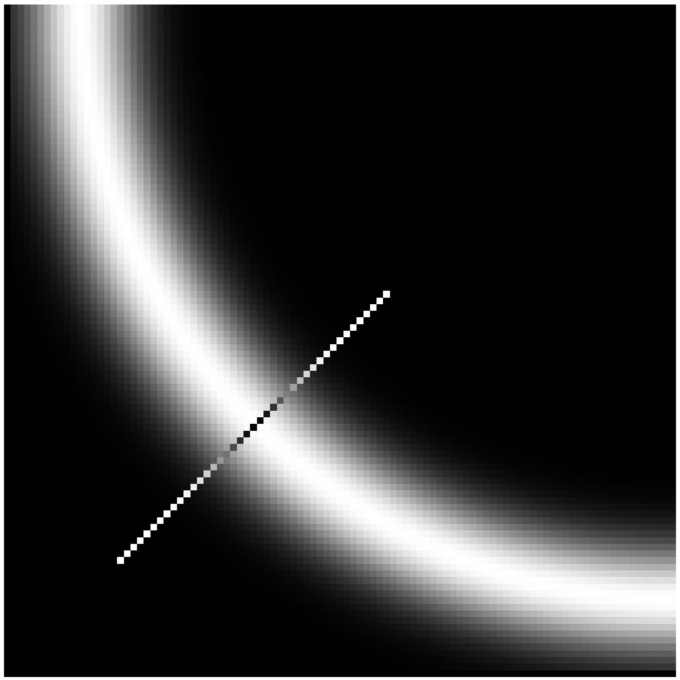
thresholding

Get Orientation at Each Pixel

- Threshold at minimum level
- Get orientation

theta = atan2(gy, gx)

Non-maximum suppression



1. Perform comparisons on the edge strength of the current pixel with the edge strength of the pixel in the positive and negative gradient directions.
2. Preserve the largest and suppress the others.

- Check if pixel is local maximum along gradient direction

Picture from Prem K Kalra

Before Non-max Suppression

After Non-max Suppression

Finding edges

thresholding

Finding edges

thinning

(non-maximum suppression)

Canny edge detector

MATLAB: `edge(image,'canny')`

1. Filter image with derivative of Gaussian

2. Find magnitude and orientation of gradient

3. Non-maximum suppression

4. Linking and thresholding

Source: D. Lowe, L. Fei-Fei